Chapter 2
Foundations of and Recent Advances in Artificial Life Modeling with Repast 3 and Repast Simphony

Michael J. North and Charles M. Macal

2.1 Introduction

Artificial life focuses on synthesizing "life-like behaviors from scratch in computers, machines, molecules, and other alternative media" [24]. Artificial life expands the "horizons of empirical research in biology beyond the territory currently circumscribed by life-as-we-know-it" to provide "access to the domain of life-as-it-could-be" [24]. Agent-based modeling and simulation (ABMS) are used to create computational laboratories that replicate real or potential behaviors of actual or possible complex adaptive systems (CAS). The goal of agent modeling is to allow experimentation with simulated complex systems. To achieve this, agent-based modeling uses sets of agents and frameworks for simulating the agent's decisions and interactions. Agent models show how complex adaptive systems may evolve through time in a way that is difficult to predict from knowledge of the behaviors of the individual agents alone. Agent-based modeling thus provides a natural framework in which to perform artificial life experiments. The free and open source Recursive Porous Agent Simulation Toolkit (Repast) family of tools consists of several advanced agent-based modeling toolkits.

2.1.1 Artificial Life

The discipline of artificial life studies the synthesis of forms and functions that appear alive. Artificial life allows scientific studies of biological systems outside the currently observable accidents of history. According to Langton [24]:

> Biology is the scientific study of life – in principle, anyway. In practice, biology is the scientific study of life on Earth based on carbon-chain chemistry. There is nothing in its charter that restricts biology to carbon-based life; it is simply that this is the only kind of life that has been available to study. Thus, theoretical biology has long

> faced the fundamental obstacle that it is impossible to derive general principles from single examples.
>
> Without other examples, it is difficult to distinguish essential properties of life – properties that would be shared by any living system – from properties that may be incidental to life in principle, but which happen to be universal to life on Earth due solely to a combination of local historical accident and common genetic descent.
>
> In order to derive general theories about life, we need an ensemble of instances to generalize over. Since it is quite unlikely that alien life-forms will present themselves to us for study in the near future, our only option is to try to create alternative life-forms ourselves – artificial life – literally "life made by Man rather than by Nature."

Langton's description of artificial life indicates the depth but belies the age of the discipline. According to Di Paolo [12]:

> To say that artificial life is a young discipline in name only is to exaggerate, but it would be mistaken to think that its goals are new. The marriage of synthetic scientific aims with computational techniques makes artificial life a product of the last fifteen years, but its motivations have much deeper roots in cybernetics, theoretical biology, and the age-old drive to comprehend the mysteries of life and mind. Little wonder that a good part of the work in this field has been one of rediscovery and renewal of hard questions. Other disciplines have sidestepped such questions, often for very valid reasons, or have put them out of the focus of everyday research; yet these questions are particularly amenable to be treated with novel techniques such as computational modeling and other synthetic methodologies. What is an organism? What is cognition? Where do purposes come from?

2.1.2 Agent-Based Modeling for Artificial Life

Agent-based modeling and simulation are used to create computational laboratories that replicate selected real or potential behaviors of actual or possible complex adaptive systems. A complex adaptive system is made up of agents that interact, mutate, replicate, and die while adapting to a changing environment. Holland has identified the three properties and four mechanisms that are common to all complex adaptive systems [19]:

1. The nonlinearity property occurs when components or agents exchange resources or information in ways that are not simply additive. An example is a photosynthetic cell agent that returns 1 calorie of energy when 1 calorie is requested, 2 calories of energy when 2 calories are requested, and 3 calories of energy when 10 calories are requested.
2. The diversity property is observed when agents or groups of agents differentiate from one another over time. An example is the evolutionary emergence of new species.
3. The aggregation property occurs when a group of agents is treated as a single agent at a higher level. An example is the ants in an ant colony.
4. The flows mechanism involves exchange of resources or information between agents such that the resources or information can be repeatedly

forwarded from agent to agent. An example is the flow of energy between agents in an ecosystem.
5. The tagging mechanism involves the presence of identifiable flags that let agents identify the traits of other agents. An example is the use of formal titles such as "Dr." in a social system.
6. The internal models mechanism involves formal, informal, or implicit representations of the world embedded within agents. An example is a predator's evolving view of the directions prey are likely to flee during pursuit.
7. The building blocks mechanism is used when an agent participates in more than one kind of interaction. An example is a predator agent that can also be prey for larger predators.

Of course, these properties and mechanisms are interrelated. For example, with aggregation, many agents can act as one. With building blocks, one agent in some sense can act as many. Agent-based models normally incorporate some or all of the properties and mechanisms of complex adaptive systems.

The goal of agent modeling is to allow experimentation with simulated complex systems. To achieve this, agent-based modeling uses sets of agents and frameworks for simulating the agent's decisions and interactions. Agent models can show how complex adaptive systems can evolve through time in a way that is difficult to predict from knowledge of the behaviors of the individual agents alone. Agent modeling focuses on individual behavior. The agent rules are often based on theories of the individual such as rational individual behavior, bounded rationality, or satisficing [39]. Based on these simple types of rules, agent models can be used to study how patterns emerge. Agent modeling may reveal behavioral patterns at the macro or system level that are not obvious from an examination of the underlying agent rules alone: These patterns are called emergent behavior. Agent-based modeling and simulation thus provide a natural framework in which to perform artificial life experiments.[1]

Agent-based modeling and simulation are closely related to the field of Multi-agent Systems (MAS). Both fields concentrate on the creation of computational complex adaptive systems. However, agent simulation models the real or potential behaviors of complex adaptive systems and MAS often focuses on applications of artificial intelligence to robotic systems, interactive systems, and proxy systems.

[1] ABMS, agent-based modeling (ABM), agent-based simulation (ABS), and individual-based modeling (IBM) are all synonymous. ABMS is used here since ABM can be confused with anti-ballistic missile, ABS can be confused with anti-lock brakes, and IBM can be confused with International Business Machines Corporation.

2.1.3 Chapter Organization

This chapter provides an overview of the Repast agent modeling toolkit from the perspective of artificial life. This chapter is organized into four parts. The introduction describes artificial life and agent-based modeling and simulation. The second section discusses the Repast agent modeling toolkit's development ecosystem and underlying concepts. The third section reviews a series of Repast artificial life models of artificial evolution and ecosystems, artificial societies, and artificial biological systems. The final section presents a summary and conclusions.

2.2 REPAST

The Recursive Porous Agent Simulation Toolkit (Repast) family of tools is one of several agent modeling toolkits available. Repast borrows many concepts from the Swarm agent-based modeling toolkit [40]. Repast is differentiated from Swarm in several respects that are discussed later.

Two generations of the Repast toolkit family are in widespread use. The first, Repast 3, is available in pure Java and pure Microsoft.Net forms and uses a library-oriented approach to model development [33]. The second, Repast Simphony, is available in pure Java. It can be used as a library, but the preferred model of development is to use its point-and-click wizard-based system to create models, execute them, and analyze the results [21, 36].

Repast is a free open-source toolkit that was originally developed by Sallach, Collier, Howe, North, and others [9]. Repast was created at the University of Chicago. Subsequently, it has been maintained by organizations such as Argonne National Laboratory. Repast is now managed by the nonprofit volunteer Repast Organization for Architecture and Design (ROAD). ROAD is led by a board of directors that includes members from a wide range of government, academic, and industrial organizations. The Repast system, including the source code, is available directly from the web [38].

Repast seeks to support the development of extremely flexible models of living social agents, but it is not limited to modeling living social entities alone. From the ROAD home page [38]:

> Our goal with Repast is to move beyond the representation of agents as discrete, self-contained entities in favor of a view of social actors as permeable, interleaved, and mutually defining; with cascading and recombinant motives. . . We intend to support the modeling of belief systems, agents, organizations, and institutions as recursive social constructions.

2.2.1 Repast 3

At its heart, Repast toolkit version 3 can be thought of as a specification for agent-based modeling services or functions. There are three concrete implementations of this conceptual specification. Naturally, all of these versions have the same core services that constitute the Repast system. The implementations differ in their underlying platform and model development languages. The three implementations are Repast for Java (RepastJ), Repast for the Microsoft.Net framework (Repast.Net), and Repast for Python Scripting (RepastPy). RepastJ is the reference implementation that defines the core services. The fourth version of Repast, namely Repast for Oz/Mozart (RepastOz), is an experimental system that partially implements the Repast conceptual specification while adding advanced new features [30, 42]. An example Repast model user interface is shown in Fig. 2.1.

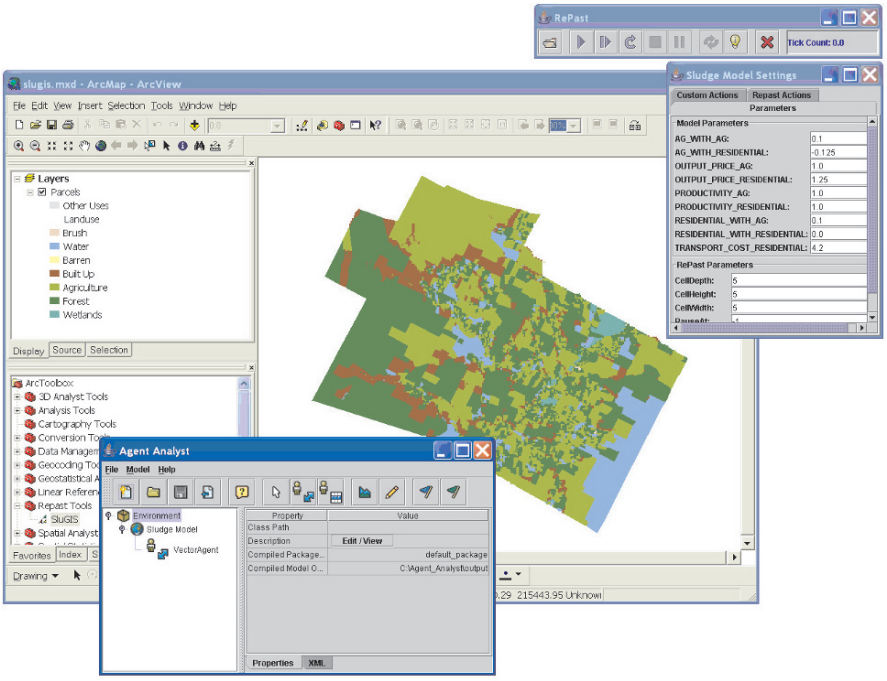

Fig. 2.1 A Repast 3 model user interface using the ESRI ArcGIS agent analyst extension.

Repast 3 has a variety of features, including the following:

- Repast 3 includes a variety of agent templates and examples. However, the toolkit gives users complete flexibility as to how they specify the properties and behaviors of agents.

- Repast 3 is fully object-oriented.
- Repast 3 includes a fully concurrent discrete event scheduler. This scheduler supports both sequential and parallel discrete event operations.
- Repast 3 offers built-in simulation results logging and graphing tools.
- Repast 3 has an automated Monte Carlo simulation framework.
- Repast 3 provides a range of two-dimensional agent environments and visualizations.
- Repast 3 allows users to dynamically access and modify agent properties, agent behavioral equations, and model properties at run time.
- Repast 3 includes libraries for genetic algorithms, neural networks, random number generation, and specialized mathematics.
- Repast 3 includes built-in Systems Dynamics modeling.
- Repast 3 has social network modeling support tools.
- Repast 3 has integrated geographical information systems (GIS) support.
- Repast 3 is fully implemented in a variety of languages including Java and C#.
- Repast 3 models can be developed in many languages including Java, C#, Managed C++, Visual Basic.Net, Managed Lisp, Managed Prolog, and Python scripting.
- Repast 3 is available on virtually all modern computing platforms, including Windows, Mac OS, and Linux. The platform support includes both personal computers and large-scale scientific computing clusters.

Repast 3's features directly support the implementation of models with Holland's three properties and four mechanisms [19]:

1. Repast 3 allows nonlinearity in agents since their behaviors are completely designed by users. Repast's Systems Dynamics, genetic algorithms, neural networks, random number generation, and social networks libraries make this process easy.
2. Repast 3 supports diversity by giving users complete control over the way their agents are defined and initialized. Again, the Repast libraries simplify the specification of diversity.
3. Repast 3 allows the aggregation property by allowing users to specify and maintain groups of agents.
4. Repast 3 supports the flows mechanism with features such as its Systems Dynamics tools and social network library.
5. Repast 3 provides for the tagging mechanism by allowing agents to present arbitrary markers.
6. Repast 3 makes the internal models mechanism available through both its flexible definition of agents and its many behavioral libraries.
7. Repast 3 supports the building blocks mechanism through its object-oriented polymorphism.

Repast 3 for Python Scripting (RepastPy) enables visual model construction with agent behaviors defined in Python [27]. RepastPy models can be automatically converted to RepastJ models using RepastPy's export option.

RepastPy users work with the interface shown in the upper left-hand window of Fig. 2.2 to add the components to their models. RepastPy users then employ Python to script the behaviors of their agents, as shown in the lower right-hand window of Fig. 2.2.

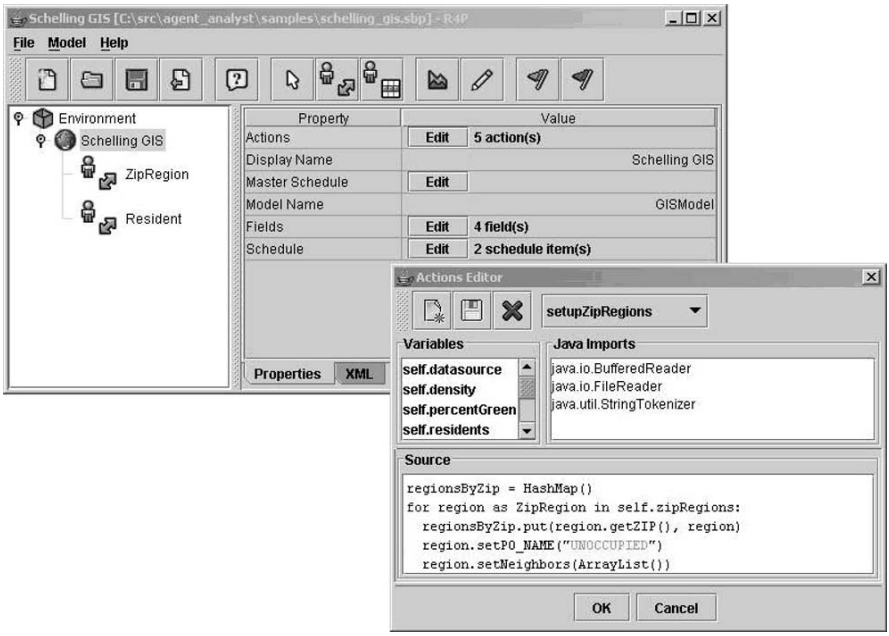

Fig. 2.2 The RepastPy interface.

The components in the example model are shown on the left-hand side of the upper window of Fig. 2.2. These components include the simulation environment specification, the model specification ("Schelling GIS"), the ZIP code region agent specification ("ZipRegion"), and the residential agent specification ("Resident"). Properties for the model specification such as the "Actions," "Display Name," and "Master Schedule" are shown on the right-hand side of the upper window in the figure. The Actions "Edit" button is used to access the Python scripting for the agent behaviors.

The Python scripting window for the example model is shown in the lower window of Fig. 2.2. The agent properties ("Variables"), the agent behavior libraries ("Java Imports"), and behavior code ("Source") can be seen in this window.

There is a special version of RepastPy known as the Agent Analyst that is an extension to the ESRI ArcGIS geographical information systems platform. ESRI ArcGIS is the leading commercial GIS, with well over one million users. Agent Analyst is a fully integrated ArcGIS Model Tool. This means that Agent Analyst has drag-and-drop integration with ArcGIS. Agent

Analyst users can create RepastPy models from within ArcGIS with a few mouse clicks. Fig. 2.1 shows the SLUDGE Geographical Information System (SluGIS) Agent Analyst model running within ArcGIS. SluGIS is described in the section on artificial societies.

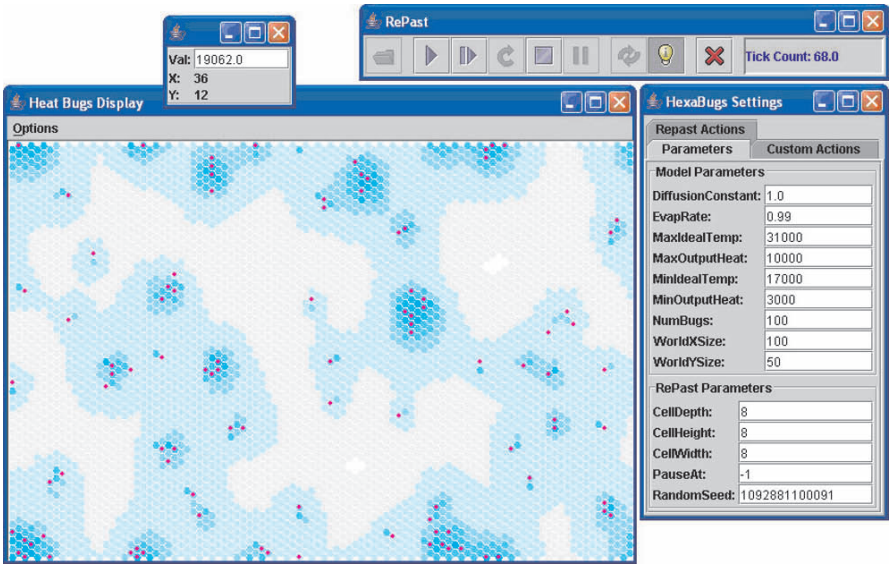

Fig. 2.3 The RepastJ Hexabugs model.

RepastJ is written entirely in Java [15]. An example RepastJ model, Hexabugs, is shown in Fig. 2.3. The Hexabugs model is discussed in the section on artificial biological systems. Since RepastJ is pure Java, any development environment that supports Java can be used. The free and open-source Eclipse development environment is recommended [13]. Eclipse provides a variety of powerful editing and development features, including code authoring wizards, automatic code restructuring tools, design analysis tools, Unified Modeling Language (UML) tools, extensible markup language (XML) tools, and integration with version control systems. Fig. 2.4 shows part of the RepastJ AgentCell model in Eclipse. The AgentCell modules are shown in the upper left "Package Explorer" tab. The cell agent component is highlighted in this tab. Part of the cell agent code is shown in the upper middle "Cell.java" tab. Some of the cell agent properties and methods can be seen in the "Outline" tab on the far right. Part of the cell agent documentation is shown in the button right tab. Additionally, a code module dependency graph can be seen in the lower left "Dependency Graph View" tab. This graph shows the connections between some of the main AgentCell modules.

Both RepastJ and RepastPy models can be developed and executed on nearly any modern computing platform. This is particularly beneficial for

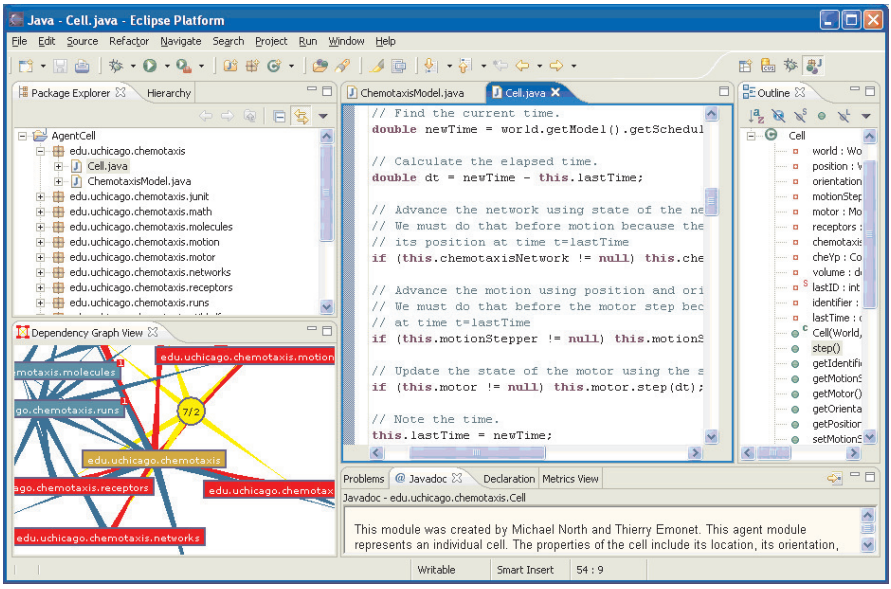

Fig. 2.4 RepastJ in the Eclipse development environment.

artificial life researchers since models can be constructed on readily available workstations and then executed on large-scale clusters without changing code. An example of this will be provided along with the description of the AgentCell model.

Repast for the Microsoft.Net framework (Repast.Net) is written entirely in C# [2]. An example Repast.Net named Rocket Bugs is shown in Fig. 2.5. The Rocket Bugs model is a Cartesian elaboration of the Hexabugs model in which some of the agents herd the other agents in the system. Some of the code for this model is displayed in the Visual Studio Environment in Fig. 2.6. It can be seen in the three windows on the left in the figure that the Rocket Bugs simulation uses a combination of Managed C++, C#, and Visual Basic.Net, all in a single seamless model. Additionally, note in the lower right that Repast.Net comes with a full set of specialized Visual Studio templates. These templates automate the initial creation of both Repast.Net models and model components such as agents.

All three versions of Repast are designed to work well with other software development tools. For example, all three versions are integrated with GISs such as the RepastPy Agent Analyst example shown in Fig. 2.1. However, since RepastJ is the most widely used version of Repast, the integration examples will focus on Java. See ROAD for many other examples [38].

RepastJ easily permits aspect-oriented software development. Aspects implement cross-cutting concerns that allow software idioms repeated throughout a model to be factored to reduce redundancy [14]. See Walker, Baniassad,

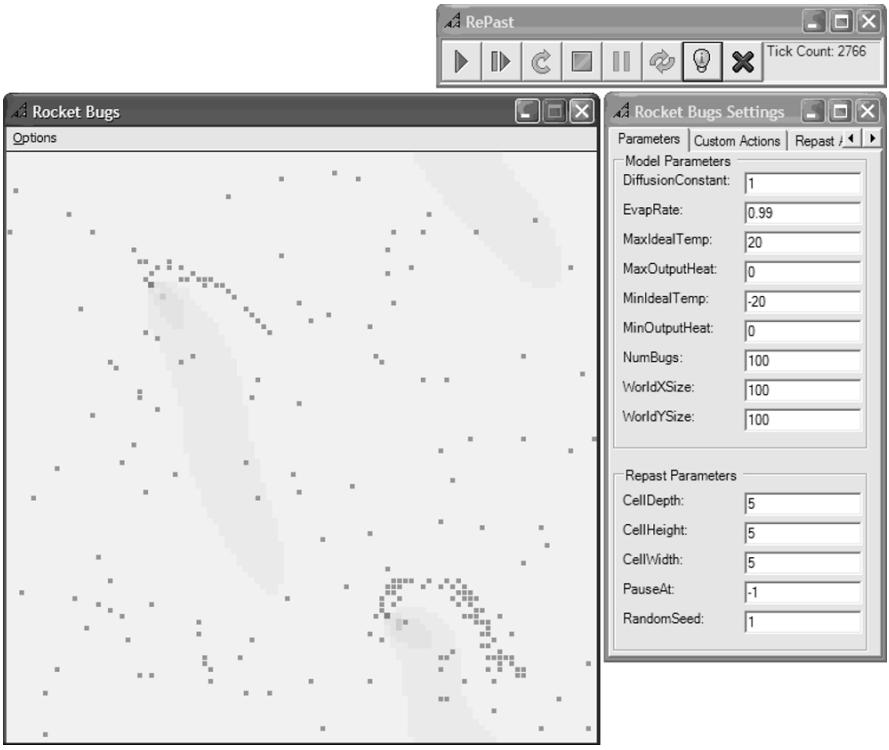

Fig. 2.5 Repast.Net Rocket Bugs model.

and Murphy for a discussion of the use of Aspects for software development [43].

RepastJ includes its own built-in logging facilities but also works with the high-performance Log4j system and also with the National Center for Supercomputing Applications' (NCSA) Hierarchical Data Format 5 (HDF5) data storage system [18, 32]. The use of Log4j, among other logging tools, in conjunction with AspectJ is discussed briefly by Cloyer et al. [8].

RepastJ unit testing is performed with JUnit as outlined in Beck and Gamma [3]. Unit testing allows software to be tested on an incremental modular level. The combination of these and other tools with RepastJ allows sophisticated models to be constructed reliably and efficiently.

The Repast 3 system has two layers. The core layer runs general-purpose simulation code written in Java or C#. This component handles most of the behind-the-scenes details. Repast users do not normally need to work with this layer directly. The external layer runs user-specific simulation code written in Java, C#, Python, Managed C++, Managed Lisp, Managed Prolog, Visual Basic.Net, or other languages. This component handles most of the center stage work. Repast users regularly work with this layer.

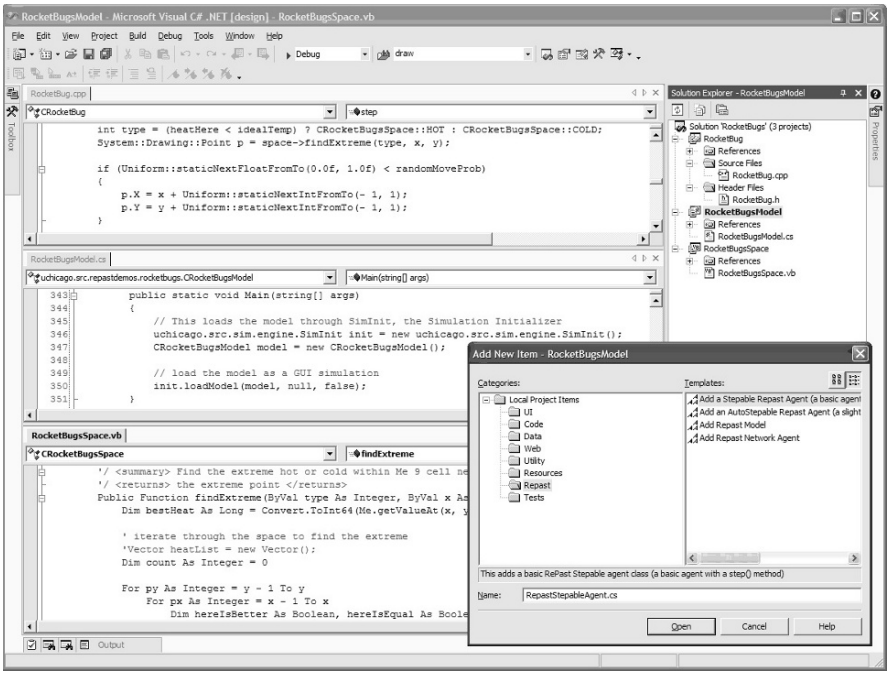

Fig. 2.6 Repast.Net in Microsoft Visual Studio.Net.

The Repast 3 system has four fundamental components, as shown in Fig. 2.7. The components are the simulation engine, the input/output (I/O) system, the user interface, and the support libraries. Each of these components is implemented in the core layer and is accessed by the user in the external layer. A Unified Modeling Language (UML) diagram showing the relationships between these components is presented in Fig. 2.8. Information on UML notation can be found in Booch [5].

The Repast 3 simulation engine is responsible for executing simulations. The Repast engine has four main parts, namely the scheduler, the model, the controller, and the agents. The relationship among these components is indicated in Figs. 2.7 and 2.8 and is discussed later in this section.

The Repast 3 scheduler is a full-featured discrete event scheduler. Simulations proceed by popping events or "actions," as they are called in Repast, off an event queue and executing them. These actions are such things as "move all agents one cell to the left," "form a link with your neighbor's neighbor," or "update the display window." The model developer determines the order in which these actions execute relative to each other using ticks. As such, each tick acts as a temporal index for the execution of actions. For example, if event X is scheduled for tick 3, event Y for tick 4, and event Z for tick 5, then event Y will execute after event X and before event Z. Actions scheduled for execution at the same tick will be executed with a simulated concurrency. In

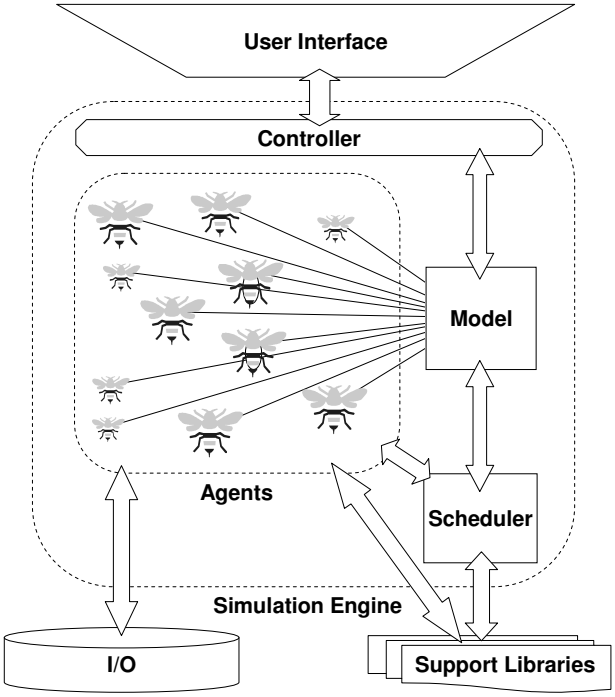

Fig. 2.7 Repast overview diagram.

this way, the progression of time in a simulation can be seen as an increase in the tick count.

The Repast 3 scheduler includes full support for concurrent task execution. Tasks become concurrent when actions are given both a starting time and duration. When durations are specified, actions that can be started in the background are run concurrently. Actions with nonzero durations will run concurrently with other actions with compatible tick counts as well as block the execution of other actions with higher tick counts until the current action is completed. For example, consider a process that contains some long-running and complicated behavior that can be started at time t with results needed at $t + 5$. Imagine that there are actions that can be run concurrently over time t to $t+5$. This behavior can be modeled as an action with a five-tick duration. In terms of implementation, this action will run in its own thread that is amenable to being run on a separate processor or even on another computer. This allows the natural introduction of complex concurrent and parallel task execution into Repast simulations. Since durations are optional, modelers can begin by creating sequential simulations and then introduce concurrency as needed.

Repast 3 schedulers are themselves actions that can be recursively nested following the composite design pattern [16]. This allows a Repast action to be

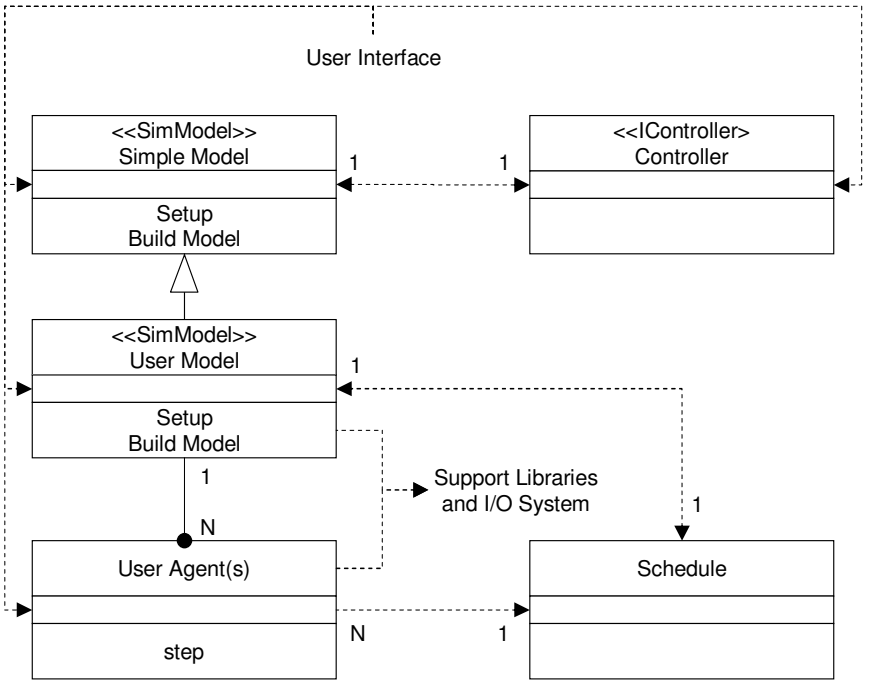

Fig. 2.8 Core Repast UML diagram.

as complex as needed for a given application. It even allows advanced multi-scale simulations to be constructed by combining existing models such that the full schedules of lower-level models run as simple actions in higher-level models.

Repast 3 models contain the definition of the simulation to be run by the scheduler. Repast models include the list of agents to be executed, the simulation initialization instructions, and the user interface specification.

Repast 3 controllers connect models and schedulers. They activate the selected model and then manage the interactions between the user or batch execution system and the model.

Repast 3 agents are created by users from components within Repast. A variety of options are available, including geographically situated agents and network-aware agents. Agents receive data from, and provide results to, the Repast I/O system.

The Repast 3 I/O system allows agents to be created based on input properties. It also can store data from both agents and overall models. Repast includes a set of results loggers that support a range of storage formats.

The Repast 3 user interface supports the display of model results and allows user to interact with running models. Repast user interface examples are shown in Figs. 2.1, 2.3, 2.5 and 2.9. Model user interfaces can include

graphical outputs or maps of the agent states as well as interactive probes that allow users to view and modify agent states. An agent map is shown in the lower left of Fig. 2.3 and an agent probe is shown in the upper left. Users have full control over what is available through both maps and probes.

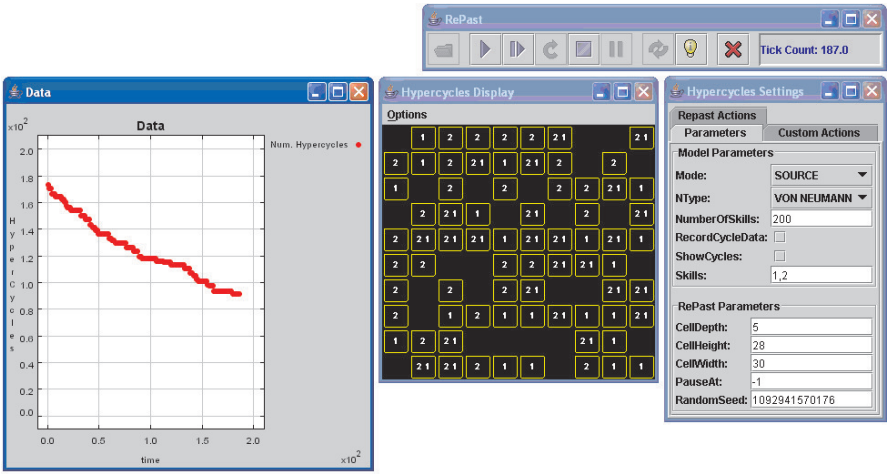

Fig. 2.9 Padgett, Lee, and Collier's RepastJ Hypercycle model.

The Repast 3 support libraries include a variety of tools for both mathematics and modeling. The mathematics support includes a range of random distribution generators and statistical aggregation tools commonly found in all kinds of simulation toolkits [25]. The modeling support includes genetic algorithms and neural network tools among other features [22].

2.2.2 Repast Simphony

Repast Simphony (Repast S) represents a substantial advance in artificial life modeling, and agent-based modeling in general, compared to previous technologies. As described in [21], Repast S builds upon the Repast 3 model development approach by introducing the following model creation process:

- The modeler creates model pieces, as needed, in the form of plain old Java objects (POJOs), often using automated tools or scripting languages such as Groovy [23]. An example agent behavior flowchart is shown in Fig. 2.10. The contents of the flowchart are automatically compiled to Groovy source code and then to Java bytecode.
- The modeler uses declarative configuration settings to pass the model pieces and legacy software connections to the Repast S runtime system.

- The modeler uses the Repast S runtime system to declaratively tell Repast S how to instantiate and connect model components.
- Repast S automatically manages the model pieces based on both interactive user input and declarative or imperative requests from the components themselves.

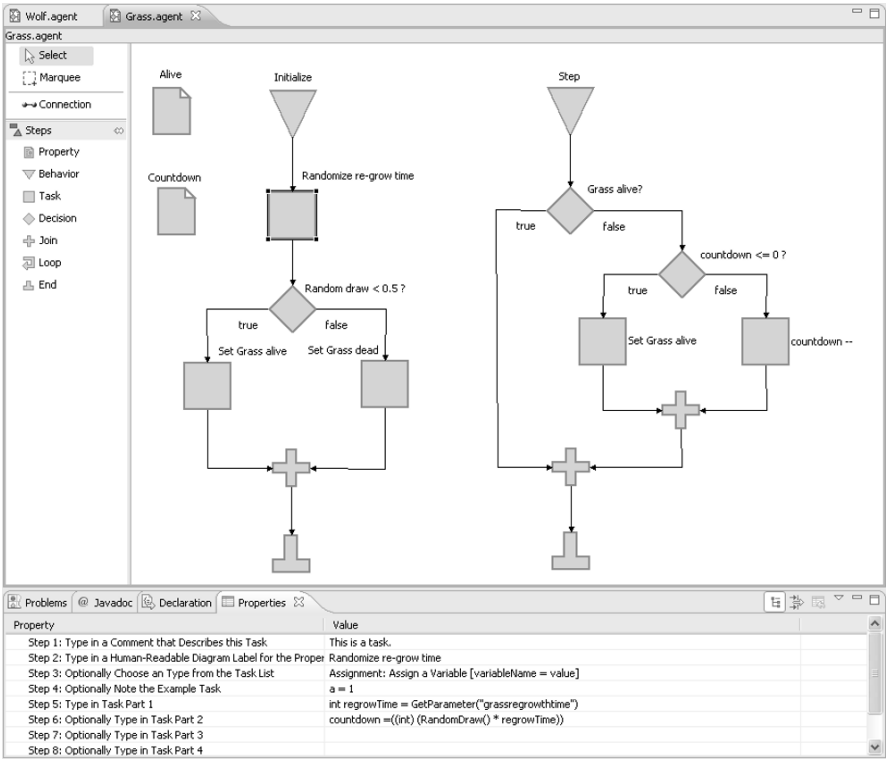

Fig. 2.10 An example agent behavior flowchart.

The POJO model components can represent anything but are most commonly used to represent the agents in the model. Although the POJOs can be created by using any method, this chapter discusses one powerful way to create POJOs for Repast S: the Repast S development environment. However, modelers can use any method – from hand coding to wrapping binary legacy models to connecting into enterprise information systems – to create the Repast S POJO model components.

Regardless of the source of the POJOs, the Repast S runtime system is used to configure and execute Repast S models. The Repast S runtime system includes the following:

- Point-and-click model configuration and operation;

- Integrated two-dimensional, three-dimensional (3D), and other views (see Fig. 2.11 for an example 3D GIS view);
- Automated connections to enterprise data sources;
- Automated connections to powerful external programs for conducting statistical analysis and visualizing model results.

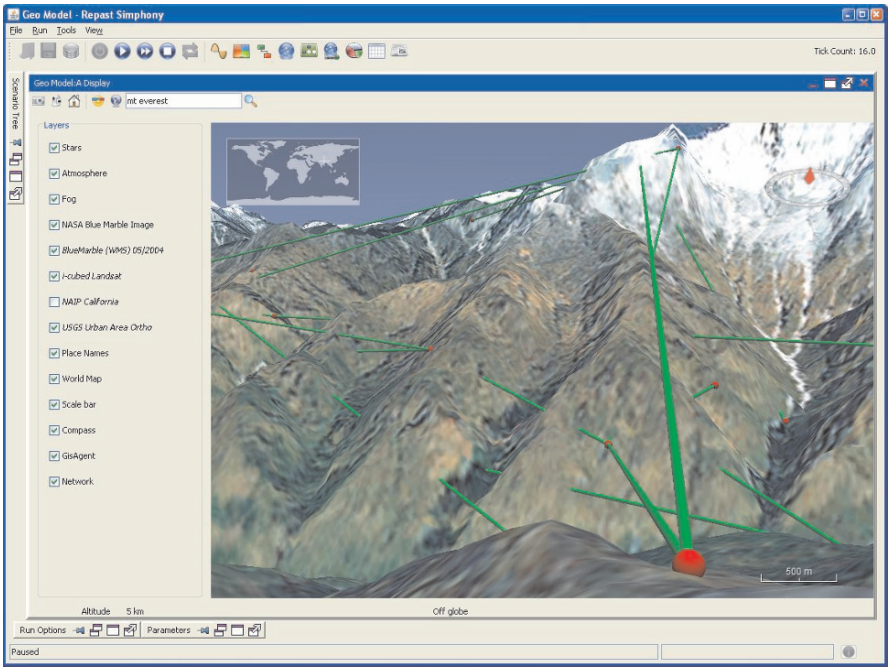

Fig. 2.11 An example 3D GIS view.

2.2.3 Using Repast 3 and Repast Simphony

As previously mentioned, all versions of Repast are distributed under a variation of the BSD license [38]. This license states that Repast can be used for virtually any purpose without fees and without a requirement to release propriety model source code. See ROAD for details [38]. This license allows Repast to be freely used in education, research, and entertainment by nonprofit, government, and commercial organizations.

Many educational institutions are now using or have used Repast for either education or research. These institutions include the University of Chicago, the University of Michigan, Iowa State University, the Swiss Federal Institute

of Technology Zurich, the Illinois Institute of Technology, and Harvard University. In particular, the University of Chicago is the birthplace of Repast. The educational uses generally focus on providing students with a laboratory environment for experiments with complex systems and for instructing students on agent-based modeling concepts. The research work includes the development of models in a variety of domains as well as model-theoretic studies. Several of the models are discussed in the following sections. The model-theoretic work mostly involves additions to and extensions of the Repast framework itself. This list of educational institutions using Repast is rapidly growing.

A significant number of U.S. federal government agencies and other organizations are using or have used various versions of Repast. These users have Repast models that focus on a range of mission-critical applications such as infrastructure security and network communications planning.

Several commercial organizations are working with Repast. These organizations include software developers such as ESRI and other private organizations. These corporations are using Repast for several purposes, including strategic planning and commercial software enhancement.

2.3 Repast Artificial Life Models

Agent-based modeling has been used in an enormous variety of applications to social systems, covering human as well as nonhuman systems. Applications range from modeling ancient civilizations that have been gone for hundreds of years, to modeling how to design new markets that do not currently exist, such as space tourism and digital news. Selected applications are listed in Table 2.1 for the period 2004 to 2008. Earlier models can be found in the first edition of this book [35]. All of the applications listed acknowledge using Repast as the underlying agent-based modeling toolkit.

Several of the works contend that using agent-based modeling versus other modeling techniques is necessary because agent-based models can explicitly model the complexity arising from individual actions and interactions that exist in the real world. All of the works support the use of Repast as the tool of choice based on reasons having to do with usability, ease of learning, cross-platform compatibility, and its sophisticated capabilities to connect to databases, graphical user interfaces, and GISs.

Griffin and Stanish [17] developed an agent-based model, using Repast, for the Lake Titicaca basin of Peru and Bolivia covering the late prehistoric period, 2500 BC to AD 1000. The model was used to study hypotheses for the causal variables affecting prehistoric settlement patterns and political consolidations. The model's geo-spatial structure consists of a 50,000 km^2 grid composed of 1.5-km square cells. Each cell modeled the geography, hydrology, and agricultural potential. Agents consist of settlements, peoples,

Application area	Model description and reference
Air Traffic Control	An agent-based model of air traffic control to analyze control policies and performance of a capacity constrained air traffic management facility (Conway [10]).
Anthropology	An agent-based model of prehistoric settlement patterns and political consolidation in the Lake Titicaca basin of Peru and Bolivia (Griffin and Stanish [17]). An agent-based model of linguistic diversity (de Bie and de Boer [4]).
Ecology	Agent-based model of predator–prey relationships between transient killer whales and other marine mammals (Mock and Testa [29]). Aphid population dynamics in agricultural landscapes using agent-based modeling (Parry, et al. [37]).
Energy Analysis	An agent-based model for scenario development of offshore wind energy (Mast et al. [28]). Agent-based model of residential energy generation (Houwing and Bouwmans [20]).
Epidemics	An agent-based computer model to retrospectively simulate the spread of the 1918–1919 influenza epidemic through the small fur-trapping community of Norway House in Manitoba, Canada (Carpenter [6]).
Marketing	An agent-based simulation to model the possibilities for a future market in sub-orbital space tourism (Charania et al. [7]). A multi-agent based simulation of news digital markets (López-Sánchez et al. [26]). An agent-based model of Rocky Mountain tourism (Yin [45]). An agent-based computational economics model using Repast to study market mechanisms for the secondary use of the radio spectrum (Tonmukayakul [41]).
Organizational Decision Making	Agent based modeling approach to allow negotiations in order to achieve a global objective, specifically for planning the location of intermodal freight hubs (van Dam et al. [11]). An evaluation framework for supply chains based on corporate culture compatibility using agent-based modeling (Al-Mutawah and Lee [1]). Emergency response (Narzisi et al. [31]).
Social Science	Simulating the process of social influence within a population, using dynamic social impact theory (Wragg [44]).

Table 2.1 Selected recent Repast applications

political entities, and leaders that interact with each other and with the environment (grid). Agent behavior is modeled as a set of condition-action rules that are based on hypothesized causal factors affecting agriculture, migration, competition, and trade processes. The authors report that through a series of simulation runs, the model produced a range of alternative political prehistories and the emergence of macro-level patterns that corresponded to observed patterns in the archaeological record. Simulation results provided insights into region-wide political consolidation.

De Bie and de Boer [4] developed an agent-based model of linguistic diversity implemented in Repast. The authors model language diversity as the

result of language mutation. Agents adopt mutations from other agents based on social impact theory, in which less common varieties of language have a relatively greater influence per individual speaking the variety. Using the model, they demonstrate how different language patterns can exist at the same time.

Carpenter [6] constructed an agent-based model with Repast to retrospectively simulate the spread of influenza in the 1918–1919 pandemic through the small fur-trapping community of Norway House in Manitoba, Canada. According to the author, historical information about the influenza pandemic was used to create a computer simulation that could be manipulated in ways that would not be possible, or ethical, in real life. Carpenter contends that by using agent-based modeling, an artificial landscape can be populated with heterogeneous agents who move and interact in ways that more closely resemble human behavior than is possible to do using other modeling techniques. The model was used to address research questions on the influence of changes in population movement patterns on the transmission of disease, specifically whether the seasonal population movements could influence the spread of the influenza through the community and whether a winter epidemic could differ from a summer epidemic predominantly due to changes in seasonal population movement.

Charania et al. [7] used agent-based simulation and Repast to model possible futures for a market in sub-orbital space tourism. Each agent is a representation of an entity within the space industry, such as consumers, producers, the government, etc., that provides or demands different products and services. Tourism companies seek to maximize profits while they compete with other companies for sales. Individual companies decide the price they will charge for a flight aboard their vehicle. Customers evaluate the products offered by the companies according to their individual tastes and preferences.

López-Sánchez et al. [26] developed a multi-agent-based simulation of news digital markets called SimwebAB, using Repast. Their approach is to adapt traditional business models to the new market. They use the model to investigate the dynamics of the new market and gain insights into how to exploit the impending paradigm shift in news contents, marketing, and distribution. The authors contend their model is useful for informing business strategy decisions.

Yin [45] used Repast to develop an agent-based model of Rocky Mountain tourism and applied it to the town of Breckenridge, Colorado. The model was used to explore how homeowners' investment and reinvestment decisions are influenced by the level of investment and amenities available in their neighborhoods. The dynamics and indirect spatial impacts of amenity-led mountain tourism on development were explored. The author found that individual levels of appreciation of amenities and continuing investment in a neighborhood attracted investment and reinvestment and created pressure for high-density resort housing development at the aggregate level.

Tonmukayakul [41] developed an agent-based computational economics model using Repast to study market mechanisms for the secondary use of the radio spectrum. Secondary use is the temporary access of the existing licensed spectrum by users who do not own a spectrum license. Using transaction cost economics as the theoretical framework, the objective of the model was to identify the preconditions for when and why the secondary use market could emerge from the repeated interactions of agents in the simulated market and what form it might take. Understanding the dynamics for this hypothetical market could lead to policy instruments to more effectively manage the spectrum.

Al-Mutawah and Lee [1] developed an agent-based supply chain model for evaluating supply chain corporate culture compatibility, using Repast. The model focuses on a three-level supply chain. The model integrates the framework for cultural learning to evaluate the management performance of the supply chain under different scenarios and assumptions.

Wragg [44] used agent-based modeling and Repast to simulate the process of social influence within a population, using dynamic social impact theory. The motivation was to understand the social processes and power dynamics of local populations in response to recent military operations in Afghanistan and Iraq. The author contends that the simulations reproduced the expected characteristics of social influence, such as opinion clustering, opinion polarization, minority opinion decay, and, more generally, the nonlinearity of public opinion change. The author highlights the need for accurate data concerning a population's social hierarchy, social networks, behavior patterns, and human geography. These data are essential for determining the impacts of word-of-mouth and mass-media-driven information campaigns on the population.

Narzisi et al. [31] developed a agent-based disaster simulation framework, using Repast, to simulate catastrophic, emergency scenarios that required optimized emergency response. Incidents included the release of chemical agents, bomb explosions, food poisonings, and small pox outbreaks. The model includes large numbers of agents in several categories, including individuals in the population, hospitals, ambulances and on-site responders.

Conway [10] used Repast to build an agent-based model of air traffic control. The model was used to analyze the effectiveness of control policies for a capacity-constrained air traffic management facility. The model is used to address situations where capacity is overburdened and there is a potential for resultant delays to propagate throughout the flight schedule. The model includes representations of air traffic system attributes such as system capacity, demand, airline schedules and strategy, and aircraft capability.

Van Dam et al. [11] applied an agent-based modeling approach to modeling the negotiation process. The objective was to investigate the conditions under which a global objective could be achieved by individual decision makers acting in their own interests. The model incorporates the agent's decision-making process for planning the location of intermodal freight hubs.

Mast et al. [28] developed an agent-based model using Repast for investigating scenarios for developing offshore wind energy resources in the Netherlands. A simple model was developed that includes the actors involved in the supply chain, represented at a relatively high level of aggregation. The model was used to investigate opportunities and threats to this emerging industry.

Houwing and Bouwmans [20] developed an agent-based model of residential energy generation to investigate how distributed energy resources will contribute to the European Union's stated policy goals of (1) energy market liberalization and (2) decreasing environmental impacts from energy use. The modeling study focused on residential power and heat generation through the use of micro-combined heat and power units; options for heat storage were modeled. The authors estimated the impact of residential power units on household energy flows, energy costs, and CO_2 emissions. The model results show that the operational impacts are highly dependent on the control mode adopted for heating and power units. Agent-based modeling proved useful in modeling both the individual technology units and the individual decision making behaviors of the consumers to operate the units.

Mock and Testa [29] use Repast to develop an agent-based model of predator–prey relationships between transient killer whales and threatened marine mammal species in Alaska. Threatened species include sea lions and sea otters. The authors state that previously only simplistic, static models of killer whale consumption had been constructed due in part to the fact that the interactions between transient killer whales and their marine mammal prey are poorly suited to classical predator–prey modeling approaches such as those based on the Lotka–Volterra differential equation framework. This killer whale model is an agent-based model at both the individual and hunting group levels. Individual agents eat, grow, reproduce, and die. Hunting groups change in size and composition while encountering other marine mammals.

Parry et al. [37] modeled the dynamics of aphid populations in agricultural landscapes using a spatially explicit agent-based simulation model and Repast. Aphid agents interact with one another and with the landscape environment over time. The model is heavily parameterized and coupled to a GIS for geo-spatial realism. The authors demonstrate that a spatial model that explicitly considers environmental factors (e.g., landscape properties, wind speed and direction, etc.) provides greater insight into aphid population dynamics over spatial and temporal dimensions than other modeling approaches.

2.4 Conclusions

Artificial life focuses on synthesizing "life-like behaviors from scratch in computers, machines, molecules, and other alternative media" [24]. Artificial life expands the "horizons of empirical research in biology beyond the territory

currently circumscribed by life-as-we-know-it" to provide "access to the domain of life-as-it-could-be" [24]. Agent-based modeling and simulation are used to create computational laboratories that replicate selected real or potential behaviors of actual or possible complex adaptive systems. Agent-based models can be used to escape the accident of history in the form of "life-as-we-know-it" by revealing alterative forms of "life-as-it-could-be."

Repast is a family of free and open-source agent modeling toolkits. Repast's features directly support the implementation of models with Holland's three properties and four mechanisms of complex adaptive systems [19]. As such, Repast 3 and Repast Simphony are natural frameworks in which to perform artificial life experiments. More information on the use of Repast and other modeling tools can be found in North and Macal [34].

Repast has many academic, government, and industry users. These users are involved in a variety of application areas, including educational, research, and commercial uses. In particular, there are many examples in which Repast has been used extensively for artificial life applications in topical areas such as artificial evolution and ecosystems, artificial societies, and artificial biological systems.

References

1. Al-Mutawah K and Lee V (2008) An Evaluation Framework for Supply Chains Based on Corporate Culture Compatibility. In: Supply Chain, Theory and Applications, Kordic V (ed.) pp. 59–72, I-Tech Education and Publishing, Vienna, Austria.
2. Archer T (2001) Inside C#. Microsoft Press, Redmond, Washington.
3. Beck K and Gamma E (1998) Test infected: Programmers love writing tests. Java Report 3:37–50.
4. de Bie P and de Boer B (2007) An Agent-Based Model of Linguistic Diversity. In: Proc. ESSLLI 2007 Workshop on Language, Games, and Evolution, Benz A, Ebert C and van Rooij R (eds.), pp. 1–8, Available online at `http://frim.frim.nl/Dublin.pdf`.
5. Booch G (1993) Object-oriented Design with Applications. Addison-Wesley, Reading, MA.
6. Carpenter, C., 2004, Agent-Based Modeling of Seasonal Population Movement and the Spread of the 1918–1919 Flu: The Effect on a Small Community, University of Missouri-Columbia, Master's Thesis, Department of Anthropology.
7. Charania AC, Olds JR, and DePasquale D (2006) Sub-Orbital Space Tourism Market: Predictions of the Future Marketplace Using Agent-Based Modeling, SpaceWorks Engineering, Inc., Atlanta, GA, Available online at `http://www.sei.aero/uploads/archive/IAC-06-E3.4.pdf`.
8. Cloyer A, Clement A, Bodkin R, and Hugunin J (2003) Practitioners report: Using aspectJ for component integration in middleware. In: Companion of the 18th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications. R. Crocker and G. Steele Jr. (eds.) ACM, New York.
9. Collier N, Howe T, and North M (2003) Onward and upward: The transition to Repast 2.0. In: Proc. of the 1st Annual North American Association for Computational Social and Organizational Science Conference, Electronic Proceedings. Pittsburgh, PA.
10. Conway SR (2006) An Agent-Based Model for Analyzing Control Policies and the Dynamic Service-Time Performance of a Capacity-Constrained Air Traffic

Management Facility, ICAS 2006 – 25th Congress of the International Council of the Aeronautical Sciences Hamburg, Germany, 3–8 September 2006. Available online at http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20060048296_2006250468.pdf.
11. van Dam KH, Lukszo Z, Ferreira L, and Sirikijpanichkul A (2007) Planning the Location of Intermodal Freight Hubs: An Agent Based Approach, In: Proceedings of the 2007 IEEE International Conference on Networking, Sensing and Control, pp. 187–192, London, UK, 15–17 April 2007.
12. Di Paolo E (2004) Unbinding biological autonomy: Francisco Varela's contributions to artificial Life. Journal of Artificial Life, Vol. 10, Issue 3, 231–234.
13. Eclipse Home Page (2008) http://www.eclipse.org/.
14. Elrad T, Filman R, and Bader A (2001) Aspect-oriented programming: Introduction. Communications of ACM 44:29–32.
15. Foxwell H (1999) Java 2 Software Development Kit. Linux Journal. Specialized Systems Consultants, Seattle, Washington.
16. Gamma E, Helm R, Johnson R, and Vlissides J (1995) Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, Reading, MA.
17. Griffin AF and Stanish C (2007) An agent-based model of prehistoric settlement patterns and political consolidation in the Lake Titicaca Basin of Peru and Bolivia, structure and dynamics: eJournal of Anthropological and Related Sciences, 2(2). Available online at http://repositories.cdlib.org/imbs/socdyn/sdeas/vol2/iss2/art2.
18. Gülcü C (2003) The Complete Log4j Manual: The Reliable, Fast, and Flexible Logging Framework for Java. QOS.ch, Lausanne, Switzerland
19. Holland J (1996) Hidden Order: How Adaptation Builds Complexity. Addison-Wesley, Reading, MA.
20. Houwing M and Bouwmans I (2007) Agent-Based Modelling of Residential Energy Generation with Micro-CHP, Delft University of Technology, Available online at http://wiki.smartpowersystem.nl/images/d/dc/M_Houwing&I_Bouwmans_Napa2006_FIN.pdf.
21. Howe TR, Collier NT, North MJ, Parker BT, and Vos JR (2006) Containing Agents: Contexts, Projections, and Agents. In: Proceedings of the Agent 2006 Conference on Social Agents: Results and Prospects, Argonne National Laboratory, Argonne, IL.
22. Java Object Oriented Neural Engine (Joone) Home Page (2004) http://www.jooneworld.com/
23. König D, Glover A, King P, Laforge G, and Skeet J (2007) Groovy in Action. Manning Publications, Greenwich, CT.
24. Langton C (1994) What is Artificial Life?, The Digital Biology Project. Available at http://www.biota.org/papers/cglalife.html.
25. Law AA (2007) Simulation Modeling and Analysis. 4th ed. McGraw-Hill, New York.
26. López-Sánchez M, Noria X, Rodriguez JA, and Gilbert N (2005) Multi-agent based simulation of news digital markets. International Journal of Computer Science & Applications, II(I). Available online at http://www.tmrfindia.org/ijcsa/v21.html.
27. Lutz M and Ascher D (1999) Learning Python. O'Reilly Press, Sebastopol, CA.
28. Mast EH, van Kuik GAM, and van Bussel GJW (2007) Agent-Based Modelling for Scenario Development of Offshore Wind Energy, T. Chaviaropoulos (ed.), Proceedings of the 2007 European Wind Energy Conference & Exhibition in Milan, pp. 1–4, Brussels, EWEA.
29. Mock KJ and Testa JW (2007) An Agent-Based Model of Predator–Prey Relationships between Transient Killer Whales and Other Marine Mammals, University of Alaska Anchorage, Anchorage, AK, May 31, 2007. Available online at http://www.math.uaa.alaska.edu/~orca/.
30. Mozart Consortium: Mozart Programming System 1.3.1 (2004). Available online at http://www.mozart-oz.org/.

31. Narzisi GV, Mishra B (2006) Multi-Objective Evolutionary Optimization of Agent-Based Models: An Application to Emergency Response Planning, New York University, Available online at http://www.cs.nyu.edu/mishra/PUBLICATIONS/06.ci06PlanC.pdf.
32. NCSA, HDF 5 Home Page (2004) http://hdf.ncsa.uiuc.edu/HDF5/.
33. North M, Collier N, and Vos R (2006) Experiences creating three implementations of the Repast agent modeling toolkit. ACM Transactions on Modeling and Computer Simulation, 16(1):1–25.
34. North M and Macal C (2007) Managing Business Complexity: Discovering Strategic Solutions with Agent-Based Modeling and Simulation, Oxford University Press, New York.
35. North MJ and Macal CM (2005) Escaping the Accidents of History: An Overview of Artificial Life Modeling with Repast, In: Adamatzky A and Komosinski M (eds.). Artificial Life Models in Software, 1st ed., pp. 115–141, Springer, Heidelberg.
36. North MJ, Tatara E, Collier NT, Ozik J (2007) Visual Agent-based Model Development with Repast Simphony. In: Proceedings of the Agent 2007 Conference on Complex Interaction and Social Emergence, Argonne National Laboratory, Argonne, IL.
37. Parry H, Evans AJ and Morgan D (2004) Aphid Population Dynamics in Agricultural Landscapes: An Agent-Based Simulation Model, International Environmental Modelling and Software Society iEMSs 2004 International Conference University of Osnabrück, Germany, 14–17 June 2004. Available online at http://www.iemss.org/iemss2004/pdf/landscape/parraphi.pdf.
38. ROAD: Repast 3.0 (2004) http://repast.sourceforge.net/.
39. Sandler T (2001) Economic Concepts for the Social Sciences. Cambridge University Press, Cambridge.
40. Swarm Development Group: Swarm 2.2 (2004) http://wiki.swarm.org/.
41. Tonmukayakul A (2007) An Agent-Based Model for Secondary Use of Radio Spectrum. Ph.D. thesis, University of Pittsburgh, School of Information Sciences.
42. Van Roy P and Haridi S (2004) Concepts, Techniques, and Models of Computer Programming. MIT Press, Boston, MA.
43. Walker R, Baniassad E, and Murphy G (1999) An initial assessment of aspect-oriented programming. In: Proc. 1999 Int. Conf. Software Engineering. IEEE, Piscataway, NJ, pp. 120–135.
44. Wragg T (2006) Modelling the Effects of Information Campaigns Using Agent-Based Simulation, DSTO Defence Science and Technology Organisation, Edinburgh South Australia, DSTO-TR-1853.
45. Yin L (2007) Assessing indirect spatial effects of mountain tourism development: An application of agent-based spatial modeling. Journal of Regional Analysis & Policy 37(3):257–265. Available online at http://www.jrap-journal.org/pastvolumes/2000/v37/F37-3-8.pdf.