TOWARD A GRAPHICAL ABM TOOLKIT WITH GIS INTEGRATION

W. RAND,* Northwestern Institute on Complex Systems,
Northwestern University, Evanston, IL
D. BROWN, R. RIOLO, and D. ROBINSON,
University of Michigan, Ann Arbor, MI

ABSTRACT

Agent-based modeling (ABM) has proved useful in a number of fields. Many of the early successes of ABM were due to its ability to represent the processes of a phenomenon. However, there is less emphasis within ABM on developing its ability to replicate spatial patterns of phenomena. In order to accomplish that, more powerful spatial modeling techniques, like those within geographical information systems (GISs), are necessary. The integration of ABM and GIS into a cohesive package would allow for elegant modeling of both process and pattern. One problem with an integrated toolkit is that most GIS users are not programmers. However, most GIS users are familiar with the use of detailed graphical user interfaces (GUIs) in order to create complex visualizations of data. Thus, providing a GUI to access an integrated ABM-GIS toolkit would vastly expand the number of users for such a toolkit. This paper is a first step toward describing such a toolkit. It first outlines several design principles for an ABM-GIS toolkit and then describes a survey of extant toolkits (Repast Py, NetLogo, and MobiDyc) that were selected on the basis of design principles. The toolkits were surveyed to see how well they fulfill some of the design principles. This survey was not meant to be a comparative review of these toolkits; rather, it was conducted to determine what useful design principles could be gathered from them that might inform a new “ideal” ABM-GIS toolkit. Finally, the paper concludes with some design recommendations for such a toolkit.

Keywords: Agent-based modeling, toolkit, GUI, GIS, design

INTRODUCTION

Agent-based modeling (ABM) has proved useful in a number of fields, from population biology, ecology, and epidemiology to international relations, economics, and urban planning (Epstein and Axtell 1996; Axelrod 1997). However, as this modeling technique continues to mature, it will often be useful to integrate it with more powerful data-handling methods like geographical information systems (GIS) (Gimblett 2002; Parker 2005). To date, typical agent-based models of spatially embedded systems use very simplistic representations of space, spatial patterns, and spatial processes. Where ABM has excelled is in its ability to represent the processes underlying a particular phenomenon, but it does not have a rich representation of the patterns of phenomena. On the other hand, while GISs are regularly used to build complex and interesting spatial models that clearly represent the patterns of a phenomenon, these models tend either to be static models of pattern or to be statistical (e.g., Markovian) models of process and

---

* *Corresponding author address*: William Rand, Northwestern Institute on Complex Systems, Northwestern University, 600 Foster Street, Evanston, IL 60208-4057; e-mail: wrand@northwestern.edu.

therefore do not contain algorithmic processes to generate the phenomenon. Thus, easy access to ABM techniques would enhance the range of models that GIS users could employ, by making it possible to combine individual (bottom-up) models of processes with sophisticated spatial models of pattern.

However, to make it possible to define arbitrarily complex agent behaviors, general-purpose agent-based modeling packages rely, more or less, on universal computer programming languages like Java, NetLogo, Python, Objective-C, and so on. But most GIS users are not programmers by training; instead, they have learned to use the powerful graphical user interfaces (GUIs) now available on most GIS systems. Thus the motivation for our project is to explore how to make it easier for GIS users to employ ABM techniques in combination with standard GIS tools by using standard GUI interfaces and frameworks. We believe one way to move toward that goal is to design a conceptual architecture for ABM toolkits that specifically facilitates the definition of combined spatial and agent-based process models within a GUI framework. We feel that by doing this, we can greatly expand the range of ABM applications and bring this technology to a new group of users.

Since a number of existing systems have already been designed to make it easier for nonprogrammers to create agent-based models, we began by reviewing these systems and their intended scope. In this paper, we examine three ABM GUI toolkits and evaluate their capabilities on several dimensions related to their functionality, interface, and primary intended audience. We chose these systems on the basis of their explicit use of a GUI, their capability to support spatially explicit ABMs, and their ability to minimize programming requirements. The first is NetLogo (Wilensky 1999), which has an easy-to-use GUI for developing the interface of the ABM. The second is Repast Py (Collier and North 2004), which has a GUI for model development as well as strong GIS integration. The final toolkit we examined was MobiDyc (Ginot et al. 2002), which has one of the most comprehensive GUIs for model development and also has an ecological focus that aligns well with the interests of many GIS users.

We carried out a systematic characterization of the functionality of all three platforms. In this paper, we answer a list of questions devised to categorize and describe the capabilities of each platform. After describing the results of our review of these systems, we discuss what was learned about each toolkit's contribution to the development of ABM architecture design, and then we distill these lessons into a list of desired capabilities for a GUI-based ABM-GIS toolkit. In short, we found that each toolkit had its own strengths and weaknesses, and we summarize these in order to create a picture of a more ideal toolkit. We conclude this paper with a presentation on the desired capabilities of our "ideal" toolkit and on general lessons gleaned from experience with existing systems.

DESIGN PRINCIPLES FOR AN ABM-GIS-GUI TOOLKIT

Having established that there are reasons why a combined ABM-GIS toolkit with a GUI would be useful (Brown et al. 2005), we felt it would be useful to systematically characterize what we would want in such a toolkit. By creating a list of desired capabilities, we can start to understand how such a toolkit could be put together. Since there are three main elements to the combined toolkit (ABM, GIS, and GUI), we created separate lists of desired characteristics for each of those areas; these characteristics are summarized in Table 1.

TABLE 1 Desirable characteristics

ABM	Event scheduling, heterogeneous entities, environmental processes
GIS	Multiple layers of spatial data, rapid spatial queries, GIS to ABM correspondence
GUI	Simple to use, drag and drop interaction, ability to create complex queries

To begin with, from the world of ABM, we want the full power of scheduling and heterogeneous entities that are normally available in ABM. Thus, we want the ability to schedule an event at any time in the future, and when it occurs, to allow it to trigger other events. This aspect of ABM comes out of work in discrete event simulation (DES) (Cassandras and Lafortune 1999). It is one of the main components in the creation of rich models of process and events. One of the hallmarks of ABM is the ability to create large numbers of heterogeneous agents and combine those agents into arbitrary groups. This gives the modeler the ability to describe agent heterogeneity as (1) variation in properties and methods and (2) the categorization of different agent types (Brown and Robinson in review). Moreover. the modeler can combine these agents and ask all of them to carry out an action simultaneously. Another feature of ABM that has proven useful is the ability to use multiple representations of the environment within the same model. In the case of this toolkit, different GIS maps could be utilized to situate the model in heterogeneous locations and assess model performance under different environmental conditions or landscape patterns. Similarly, ABM also has a powerful representation of environment processes. The environment has the ability to carry out its own processes and interact with the agents in autonomous ways. For instance, the ecological processes of photosynthesis, nutrient cycling, water use efficiency, and succession may interact locally and autonomously with climate conditions within a predator-prey or grazing model (Wilensky and Reisman 2005).

Of course there are also some capabilities from GIS that would be desirable. First of all, the ability to store multiple layers of data in one collection, which is tied together by the physical location of those layers in the world, is a powerful data model tool that would be useful within an ABM. For instance, residents moving around in a residential location model should be able to access information — such as the amount of open space, distance to a central business district, and proximity to schools — for one location in an easy and effective manner. Moreover, the ability to do rapid spatial queries would be useful. For instance, in the residential location model, developers should be able to quickly determine which lots are available within a hundred meters of a main arterial road. Another desired capability would be the transformation of GIS objects into ABM agents. For instance, a store in a GIS database could be reified as an ABM agent that buys and sells products with its neighbors. Of course the ability to export the GIS data about the environment to the ABM is also very important.

Finally, a GUI for constructing integrated ABM and GIS models, such as the GUI that is available within ArcGIS for building spatial data models (i.e., model builder), would be very useful. Model builders should be able to create agents, processes, and data reporters by doing nothing but pointing, clicking, and typing a few names. However, just because the GUI would be simple does not mean that it would necessarily only involve the creation of simple methods. Traditional GIS systems (like ArcGIS) use drop-down menus to construct detailed and rich structured query language (SQL) queries into the GIS database. These query systems are easy to use in part because they are graphical and in part because they require little (if any) formal knowledge of programming.

SURVEY

On the basis of these design guidelines (see Table 1), we undertook a qualitative survey of toolkits that have been built with one or more of these guidelines in mind. Our overall goal was to understand better whether or not extant toolkits had already integrated the aspects of a toolkit that we desired, and, if so, how they accomplished this integration. The specific objectives of this survey were therefore twofold: (1) evaluate the toolkit in terms of how well it accomplished the task we had set before us and (2) examine the basic ideas of the toolkit and see if there was anything useful we could incorporate into our design of an ideal toolkit. To accomplish this task, we created a list of questions about the capabilities of each toolkit and sought to answer those questions by examining the toolkits. However, in order to carry out this survey, we first had to determine which toolkits to examine and then determine what questions we would answer about each of the toolkits. Finally, we had to actually answer the questions and summarize the results.

Selection of Toolkits

There exists a myriad of ABM toolkits: Repast, Swarm, MAML, Ascape, AnyLogic, MASON, CORMAS, NetLogo, and MobiDyc, among others. As a result, narrowing down the toolkits to a reasonable number that we could survey was daunting. However, since a number of existing systems have already been designed to make it easier for nonprogrammers to create agent-based models, we began by reviewing these systems and their intended scope. We developed a list of criteria for determining which toolkits we would examine. The toolkit had to have a strong GUI, powerful ABM tools, strong support for the toolkit, and be provided for free. In addition, we thought it would be very useful if the toolkit already had some GIS integration and ability to model ecological systems (since that is one of the major uses of GIS data).

Given the above criteria, we selected three ABM GUI toolkits and evaluated their capabilities on several dimensions related to their functionality, interface, and primary intended audience. The first toolkit we examined was NetLogo, which was developed by Wilensky as a pedagogical and research tool (Wilensky 1999). NetLogo has an easy-to-use GUI for developing the interface of the ABM. It also has an interesting programming paradigm (everything happens in parallel) and was built with a “low-threshold, high-ceiling” language paradigm (Tisue and Wilensky 2004). The second was Repast Py (Collier and North 2004), which was developed at Argonne National Laboratory in order to make Repast easier to use (Collier et al. 2003). Repast Py has a GUI for model development that utilizes a drag-and-drop interface, and Repast Py also has strong GIS integration. The final toolkit we examined was MobiDyc (Ginot et al. 2002), which was developed at the National Research Center in Avignon, France, and was primarily built for ecological modeling. The basic concept of MobiDyc is that everything is an agent, including tasks and the environment. MobiDyc has one of the most comprehensive GUIs for model development and requires the use of only drop down menus to build a model. It also has an ecological focus that aligns well with the interests of many GIS users.

Design of the Survey

We carried out a systematic characterization of the functionality of all three platforms. In this paper, we answer a list of questions we devised to categorize and describe the capabilities of each platform. These questions are of the form “Can the system...?,” referring to specific capabilities. Besides providing detailed responses to these questions (Appendix 1), we also graded the ability of each system to carry out the particular function by using a simplified scale (Appendix 1 and Table 2). The system receives a ‘G’ (color-coded as green) if it was possible to carry out the entire task using the (G)raphical interface, a ‘P’ (color-coded as yellow) if there were specific (P)rimitives in the toolkit for carrying out the task, a ‘C’ (color-coded as red) if (C)oding was required to carry out the task, and an ‘N’ (color-coded as black) if it was (N)ot possible (without extreme measures) to carry out the task.

In order to clarify our thinking about capabilities that we desired in the integrated toolkit, we distinguished the following six modeling topics crucial to any ABM development used for rigorous scientific purposes and publication: (1) agents, (2) agent groups, (3) environment, (4) experiments, (5) reports, and (6) interoperability. We determined these topics were relevant on the basis of our experience with building and utilizing ABMs in the past. Some of these topics are not specific to use for GIS users. However, we believe that the design of experiments, software interoperability, and model output through reports and graphs are important topics relevant to the design, use, and interpretation of any ABM, and we therefore included them in the overall survey.

Once we had the six major groups established, we developed a list of questions within each group that detailed the functionality we desired in any integrated toolkit. Within each group of questions, we also found it useful to create subcategories that helped to classify the question. Finally within each of these subcategories, we listed the questions in approximate order of difficulty, moving from the least difficult to the most difficult goals to accomplish.

One word of warning: Many of these questions were very difficult to answer in any objective sense. However, we did attempt to create standards within the grading so that even if the answers are not absolute grades in any sense, they are at least a decent relative comparison of the three toolkits. In the end, because of the subjective nature of these results, they may not be as applicable for a particular project as for another one.

It is also important to remember that surveys like this one only make sense within the context of the questions being asked. Our questions and answers were designed specifically to inquire about the construction of an integrated ABM-GIS toolkit with a strong GUI. There are many criteria that we could have utilized that we did not. For instance, we did not ask “Are the primitives easy to use? Is the architecture of the toolkit intuitive? Is there a wide base of support for the toolkit?” It may very well be impossible to carry out a truly comprehensive survey of toolkits that would be appropriate for all users; hence, all such surveys are going to be subjective and thus at least partially controversial. There have been several other surveys of toolkits that have had other goals; some of these are more general surveys (Gilbert and Bankes 2002; Tobias and Hoffman 2004; Fedrizzi 2005; Wiedmann and Girardin 2005; Railsback et al. in review).

RESULTS

We present the results of our survey in two different formats. In the more extensive format (Appendix 1), we present all of the questions and the exact answers that we gave to those questions. Both are presented in terms of a quick description of an answer and the grading system described above. In addition, for quicker reference and to provide a higher-level summary of our results, Table 2 presents the letter grades that we gave to each toolkit for each answer (G, P, C, N) and is color-coded to reflect these grades (green, yellow, red, black). In addition, the questions are not listed in full in Table 2, but the categories, subcategories, and keywords referencing the question are listed.

DISCUSSION

The results of our survey were mixed. It seems obvious that none of these packages measure up to our ideal toolkit in terms of ABM-GIS integration with a strong GUI. However, we were able to update our design principles by looking over these results.

For instance, NetLogo has a programming paradigm (enforced parallelism) that causes the programmer to write code for the model in a specific way. MobiDyc also makes use of a particular paradigm (everything is an agent). As we went through the questions in the survey, we realized that this paradigm had a dramatic effect on the answers to some of the questions for these toolkits, but it did not necessarily have a negative effect. In some cases, it probably had a positive effect. In the end, it was clear that the programming paradigm utilized by a toolkit will force trade-offs to be made in the toolkit; thus, choosing a paradigm requires careful thought before designing a new toolkit.

NetLogo probably has one of the best GUIs for designing the look of the ABM, but it has little to no GUI for actually creating the model. This was an interesting result, and it convinced us that being able to design the look of the ABM enhances the model development experience for novices. Having to specify screen coordinates and sizes within code is very daunting; being able to drag and drop graphs and sliders around the world is much more natural.

Instead of providing the ability to design many (if any) of the model components graphically, NetLogo relies on a long list of primitives that can be used to carry out most of the basic operations that an ABM developer would desire. This emphasis on primitives, as opposed to visual programming, may not specifically address the goals we had in this survey but it does seem to aid novice programmers in learning how to program. In fact, NetLogo and MobiDyc together caused us to reassess our desire for a strictly graphically based language. It may, in fact, be easier to use a large graphical component with some simple coding than to design a fully functional GUI-only system.

NetLogo has also made recent strides in being able to run experiments from the GUI (i.e., BehaviorSpace) without ever having to control the model from the command line. This is a feature that will likely be appreciated by novice model users who simply want to see what the effect of a particular range of values is on the overall model performance. Part of the “ease of use” of NetLogo is a result of the fact that it has good support and the development team has included new features requested by users on a regular basis. Though support was not an explicit part of our survey, it does have a positive impact on many of the questions that we asked in our survey.

Repast Py has, by far, the best GIS integration of any of the toolkits we examined. It allows the model developer to read GIS data within the drag-and-drop environment and the click of a button. In addition, since it works with both OpenMap and ESRI products, it is usable by a

TABLE 2 Summarized and color-coded results

Agents			
Question?	**NetLogo**	**RepastPy**	**MobiDyc**
Creation			
basic	P	G	G
types	P	G	G
Properties			
basic	P	G	G
values	P	P	G
type-based	P	G	G
Methods			
basic	P	C	G
Initialization			
external	G	C	G
GIS	C	G	N
Scheduling			
parallel	P	N	G
agents	N	N	G
schedule	P	G	G
properties	C	C	G
Sensors			
other agents	P	C	G
environment	P	C	G
Effectors			
other agents	P	C	G
environment	P	C	G
Termination			
die	P	N	G
kill	C	N	G

Groups			
Question?	**NetLogo**	**RepastPy**	**MobiDyc**
Creation			
groups	P	P	G
het. groups	C	C	G
Scheduling			
schedule	P	C	G

Reports			
Question?	**NetLogo**	**RepastPy**	**MobiDyc**
world display	G	G	G
agent stats	G	C	G
envt. Stats	G	C	G
Graphs	G	G	G
output files	G	G	G
GIS	N	C	N

Environment			
Question?	**NetLogo**	**RepastPy**	**MobiDyc**
Initialization			
Values	P	C	G
External	P	C	G
GIS	C	G	N
statistical	C	C	G
non-Euclidean	C	G	N
Properties			
Global	P	G	G
Raster	G	G	G
Vector	C	G	N
GIS methods	N	C	N
layers	N	C	N
Methods			
basic	P	G	G
independent	P	G	G
topology	N	N	N
Scheduling			
schedule	P	G	G
independent	P	G	N

Experiments			
Question?	**NetLogo**	**RepastPy**	**MobiDyc**
batch	G	C	G
monte carlo	G	C	G
sweep par.	G	G	G

Interoperability			
Question?	**NetLogo**	**RepastPy**	**MobiDyc**
called from	C	C	N
calls to	C	C	N
analysis	G	G	G
experimental	C	C	N

Legend	
N = No	N
C = Code	C
P = Primitive	P
G = Graphical	G

wide variety of GIS practitioners. There is still work that needs to be done in terms of incorporating topological vector data and multiple layers and being able to easily carry out spatial queries, but, in general, Repast Py is a good first step toward GIS integration into an ABM toolkit.

Repast Py also used different GUIs for different types of models. For instance when work was being done with a vector-based model, a different GUI was required from the one used when work was being done with a raster-based model. In fact, these two worlds are so different that it may be impossible to reconcile them within one GUI.

MobiDyc seemed to be the toolkit closest toward achieving our goal of having a truly GUI-driven ABM toolkit. It had selectable menus for everything. However, the interface seemed a little confusing at times, and sometimes it was inefficient to select three or four menu items just to write a simple equation like "z = x + y." In addition, MobiDyc lacks GIS integration and, because it is written in SmallTalk, is not easily extensible.

However, in MobiDyc, it is possible to write very complicated expressions with just a few primitives. The entire MobiDyc "language" can fit on one sheet of paper with brief descriptions and yet has been used to build some fairly complicated and complex ecological models. Therefore, it seems clear that designing a good system of primitives is critical to the development of a good toolkit.

CONCLUSION

In the future, we hope to make use of this survey to design a toolkit that would meet the goal of integrating ABM and GIS while still being usable by a novice model builder. A large component of this design will involve the identification and description of the primitives of the language. A "primitive" is a basic command that is easily identifiable and can be used by a model builder without an explicit knowledge of the internal implementation of that primitive. In particular, one group of primitives that would be useful for us would be those related to the modeling of land use dynamics (e.g., land-use modeling primitives [LUMPs]), which would be tailored to allow GIS users who are interested in land-use and land-cover change to build models of real systems.

The design of primitives is very important to the eventual realization of such a toolkit. If the primitives of the toolkit are chosen carefully, then it is possible for novice users to build complicated models. NetLogo provides a clear example of that, having been used, for example, by elementary school students to build models of traffic simulation. However, the primitives also dictate what is hard and what is easy in a given language. For instance, because of the particular parallel paradigm chosen in NetLogo, it can be difficult to build a true discrete-event simulator within that toolkit.

In order to move forward toward the design of such a toolkit, we plan to refine and reconsider our goals. As mentioned above, maybe it is not necessary to have every aspect of the toolkit be built around visual programming aspects. Of course, one of the major components of this design process will be the development of a set of ideal LUMPs. Making a concise but effective list of primitives will facilitate the development of a prototype of an ideal ABM-GIS.

Ultimately, there does appear to be a trade-off between the ease of use and power of the modeling environment, but based on our analysis of these three toolkits, we believe that we have not yet hit the pareto-optimal front of that trade-off and that it is possible to continue to make improvements in both areas.

REFERENCES

Axelrod, R. 1997. "Advancing the Art of Simulation in the Social Sciences," pp. 21–40 in R. Conte, R. Hegelsmann, and P. Terna, *Simulating Social Phenomena,* Berlin, Germany: Springer-Verlag.

Brown, D.G., et al. 2005. "Spatial Process and Data Models: Toward Integration of Agent-based Models and GIS," *Journal of Geographic Systems, Special Issue on Space-Time Information Systems* **7**(1): 25–47.

Brown, D.G., and D.T. Robinson. In review. "Effects of Heterogeneity in Residential Preferences on an Agent-based Model of Urban Sprawl," *Ecology and Society*.

Cassandras, C.G., and S. Lafortune. 1999. *Introduction to Discrete Event Systems,* Springer.

Collier, N., et al. 2003. "Onward and Upward: The Transition to Repast 2.0," presented at First Annual North American Association for Computational Social and Organizational Science Conference, Pittsburgh, PA.

Collier, N., and M. North. 2004. "Repast for Python Scripting," in C.M. Macal, D. Sallach, and M.J. North (editors), *Proceedings of the Agent 2004 Conference on Social Dynamics: Interaction, Reflexivity and Emergence,* ANL/DIS-05-6, co-sponsored by The University of Chicago and Argonne National Laboratory, Oct. 7–9.

Epstein, J., and R. Axtell. 1996. *Growing Artificial Societies: Social Science from the Bottom up,* Cambridge, MA: MIT Press.

Fedrizzi, A. 2005. *Evaluation of Tools for Agent-based Simulation,* Technische Universität München, p. 218.

Gilbert, N., and S. Bankes. 2002. "Platforms and Methods for Agent-based Modeling," in *Proceedings of the National Academy of Sciences* **99**(7), Supplement 3, pp. 197–198.

Gimblett, R.H. 2002. *Integrating Geographic Information Systems and Agent-based Modeling Techniques for Simulating Social and Ecological Processes,* New York, NY: Oxford University Press.

Ginot, V., et al. 2002. "A Multi-agents Architecture to Enhance End-user Individual-based Modelling," *Ecological Modelling* (157):23–41.

Parker, D.C. 2005. "Challenges and Prospects for Integration of Geographic Information Systems and Agent-based Models of Land Use," in D.J. Maguire, M.F. Goodchild, and M. Batty, *GIS, Spatial Analysis and Modeling,* ESRI Press.

Railsback, S.F., et al. In review. “Agent-based Simulation Platforms: Review and Development Recommendations,” *Simulation;* available at http://www.humboldt.edu/~ecomodel/documents/ABMPlatformReview.pdf.

Tisue, S., and U. Wilensky. 2004. “NetLogo: Design and Implementation of a Multi-agent Modeling Environment,” in C.M. Macal, D. Sallach, and M.J. North (editors), *Proceedings of the Agent 2004 Conference on Social Dynamics: Interaction, Reflexivity and Emergence*, ANL/DIS-05-6, co-sponsored by The University of Chicago and Argonne National Laboratory, Oct. 7–9.

Tobias, R., and C. Hoffman. 2004. “Evaluation of Free Java-libraries for Social-scientific Agent-based Simulation,” *Journal of Artificial Societies and Social Simulation* **7**(1).

Wiedmann, N.B., and L. Girardin. 2005. “Technical Note: Evaluating Java Development Kits for Agent-based Modeling,” *Journal of Artificial Societies and Social Simulation* **8**(2).

Wilensky, U. 1999. *NetLogo*, Center for Connected Learning and Computer-based Modeling, Northwestern University, Evanston, IL.

Wilensky, U., and K. Reisman. 2005. “Thinking Like a Wolf, a Sheep, or a Firefly: Learning Biology through Constructing and Testing Computational Theories — An Embodied Modeling Approach,” *Cognition & Instruction*.

APPENDIX 1: FULL SURVEY RESULTS

Agents			
Question? Can it...	**NetLogo**	**Repast Py**	**MobiDyc**
Creation			
create agents?	yes P	yes G	yes G
create different types of agents?	yes, by creating different breeds P	yes, you create different agent classes graphically and then instantiate them G	yes, called entities, and there can be different "stages" within entities G
Properties			
create agent properties?	yes, using -own predicate P	yes, agents have fields G	yes, agents have attributes G
set agent properties to heterogeneous values?	yes, you can specify all agents properties uniquely P	yes, you can specify all agents properties uniquely P	yes, attributes can be initialized via a "series" G
create different properties for each agent type?	yes, different breeds own different properties P	yes, all agents must have their fields specified G	yes, each entity has different properties, though some are automatic ("age," "location") G
Methods			
create agent methods?	yes, but all methods are available to all entities P	yes, agents have individual methods C	yes, agents have tasks; some are built in, and others can be created G
Initialization			
initialize agents from external sources?	yes, using import-world and from text files G	yes, you can read in data using standard file I/O C	yes, there is a standard initialization file format and you can write Smalltalk I/O code G
initialize agents from GIS data?	yes, using standard GridASCII and file I/O C	yes, agents can be read directly from shapefiles via the GUI G	no, though you could convert GIS data into the proper MobiDyc format N
Scheduling			
have agents take actions in a distributed, parallel fashion?	yes, this is the standard method of executing actions P	no, agents take actions in asynchronous fashion N	yes, you can switch the scheduler between synchronous and sequential modes G
create events as agents?	no, events are methods that are requested of agents N	no, methods are something that agents and the environment have and are not agents themselves N	yes, all tasks and events are actually considered agents and thus treated in the same way as other agents G
schedule agents to take actions?	yes, though there is no discrete event system, you can ask a turtle to do anything at any time P	yes, there is a full dynamic event system scheduler G	yes, though there is no schedule, you can ask an agent to perform a task conditionally G
schedule agents to take actions on the basis of their properties?	yes, though there is no schedule, you can ask them to take actions on the basis of properties C	yes, you can determine in the code if the agent should actually take the action C	yes, though there is no schedule, you can ask them to take actions on the basis of properties G

Agents			
Question? Can it...	**NetLogo**	**Repast Py**	**MobiDyc**
Sensors			
have agents learn about other agents?	yes, all agents can access anything about all other agents, though it can be difficult to single out agents that do not have particular properties or spatial nearness P	yes, agents can access information about other agents C	yes, all agents have access to all other agents G
have agents learn about their environment?	yes, agents can ask questions of the patches P	yes, agents can access information about the environment C	yes, all agents of any entity type can access all other agents, and since MobiDyc uses an agent to represent the environment, that includes the environment G
Effectors			
have agents which affect other agents?	yes, any agent can ask any other agent to set a particular value P	yes, agents can force other agents to change fields or execute methods given the right permissions C	yes, "modify an attribute" is one of the most common tasks G
have agents which affect the environment?	yes, agents can set attributes of patches and can "stamp" their environment P	yes, agents can change environmental values C	yes, agents can modify attributes of the environment G
Termination			
destroy agents?	yes, die is a primitive P	no, Python's **del or "die"?** is not supported, though you can create workarounds that "imitate death" usually N	yes, "die" is a built-in task G
have agents destroy each other?	yes, agents can be asked by other agents to die C	no, see above N	yes, "kill" is a built-in primitive G
Creation			
create groups of agents?	yes, breeds are a great way to do this P	yes, there are even primitives to get many basic groups like neighbors P	yes, you have entities, stages, and some basic queries like neighbors G
create groups made of heterogeneous agent types?	yes, but very constrained; you can create a property shared by two different groups and then create an agent class using a filter based on that property C	yes, you can create lists of different types of agents fairly easily C	yes, but these groups are calculated each time you perform a task and are not preserved over time G
Scheduling			
schedule agents to take actions on the basis of a group that is independent of the type?	yes, though there is no general scheduling mechanism, you can ask different breeds to take different actions P	yes, agents in a group can be asked to perform an action, but it cannot be explicitly scheduled C	yes, though there is no general scheduling mechanism, you can ask different groups to take different actions G

Agents			
Question? Can it...	**NetLogo**	**Repast Py**	**MobiDyc**
Initialization			
create environmental values?	yes, patches have a **-own ok???** predicate P	yes, you can create underlying grids like in regular Repast, but there is no way to just set up a grid with values from the GUI C	yes, cells have attributes since they are also agents G
initialize the environment from external sources?	yes, you can read in values from files using standard I/O and then set patch values based on that; import-**pcolor ok??** lets you do this from the menu G	yes, you can read in values from files using standard I/O C	yes, there is a standard initialization file format G
initialize the environment from GIS data?	yes, but there are no specific GIS I/O primitives C	yes, a GIS environment is a specific type that allows you to read in shapefiles to define the environment G	no, there is no standard way to read in GIS data, though you could write a script to turn GIS data into the MobiDyc file format C
initialize the environment from statistical distributions?	yes, most standard distributions can be generated C	yes, most standard distributions can be generated C	yes, but the initialization only happens once and thus is always the same every time you start the model G
create non-Euclidean environments?	yes, standard techniques exist to create network-based topologies C	yes, networked environments are another built-in environment type G	no, it is all grid based N
Properties			
create global properties for the entire model?	yes, the globals command defines properties for the whole world P	yes, the environment has fields that can be set for the whole world G	yes, these are considered nonlocated agents G
create properties in the environment on a raster basis?	yes, rasters/grids are the basic environment G	yes, the normal grid data can be a raster, but there is no way to import GIS raster data G	yes, rasters/grids are the basic environment G
create properties in the environment on a vector basis?	yes, by using the network-like agents to demarcate areas; moreover, you can describe a raster with a vector to a very fine level C	yes, either through the use of a network or through GIS data G	no, though agents are smaller than the grid N
create properties based on GIS methods (i.e., buffering, intersections)?	no, though there are some things like neighbors and the like that could be used to generate similar results N	yes, you can use either OpenMap or ArcObjects to manipulate GIS objects C	no, though there are some things like neighbors and the like that could be used to generate similar results N

Agents			
Question? Can it...	**NetLogo**	**Repast Py**	**MobiDyc**
create properties of the environment in multiple layers?	no, but patches can have multiple properties that might be equivalent to multiple layers N	yes, you can create multiple layers in a grid model, but not in a network model or GIS model C	no, but cells can have multiple properties that might be equivalent to multiple layers N
Methods			
have the environment take action?	yes, patches can determine that certain things should be done, and diffuse is a basic command P	yes, the environment has its own actions; in GIS models, the environment is even identified with agents G	yes, the cells can perform tasks just like any other agent G
have the environment act independently of the agents?	yes, diffuse is one clear example of this; more important, patches are agents independent of the turtles P	yes, though sometimes the environment is an agent as described above G	yes, and you can even modify whether the grid or the agents act first G
have the environment enforce topological rules?	no, agents are responsible for checking that they are not violating any topological rules N	no, agents are responsible for checking that they are not violating any topological rules N	no, agents are responsible for checking that they are not violating any topological rules N
Scheduling			
schedule the environment to take actions?	yes, though there is no schedule, any patch can be asked to do anything at any time P	yes, the environment can use the same scheduler as the agents G	yes, though there is no schedule, any cell can be asked to do anything at any time G
schedule the environment to take action independently of the agents?	yes, patches/the environment are independent agents P	yes, the environment can use the same scheduler as the agents G	no, though the whole world can be asked to perform actions at a different time than the agents G
generate graphical output of the world?	yes, this is all manipulated from the interface G	yes, you drag and drop a viewer into the model G	yes, you can define your own visualization options G
calculate statistics about agents?	yes, BehaviorSpace makes this easy, G	yes, you can write code to calculate just about any statistic C	yes, you can perform many standard statistical calculations G
calculate statistics about the environment?	yes, BehaviorSpace makes this easy G	yes, you can write code to calculate just about any statistic C	yes, cells are just like agents in this environment G
output statistics to graphical displays?	yes, the graphs themselves are designed graphically and are linked to report values in the code G	yes, you can select from a drop-down menu what variables you want to graph G	yes, they have line graphs and histograms; unfortunately, these are not real time; they can be examined only after the experiment G
output statistics to an output file?	yes, BehaviorSpace makes this easy G	yes, this is part of each graph you **care or "create"??** and **is?** specified in the GUI G	yes, you can save the text used to generate any display G

Agents			
Question? Can it...	**NetLogo**	**Repast Py**	**MobiDyc**
output data to a GIS server?	no, but the data can be written to a text file and then imported N	yes, you can write back to ShapeFiles C	no, there is no way to write to a GIS file, though you could use the output text file as a GIS input N
run the model in batch mode (i.e. run the model one or more times without the GUI)?	yes, by turning off the update display and using BehaviorSpace, or it can be called from another Java program, or it can be run from the command line by using the headless version G	yes, you can turn off the GUI output and just have the controller come up C	yes, you define it through the GUI and run it from there, but you can turn off visualization G
run the model automatically with different random number seeds in batch mode?	yes, by using BehaviorSpace G	yes, though you have to create a random number seed input that varies as one of the parameters in multi-run C	yes, part of the standard batch mode is to select the number of times to replicate the experiment G
sweep parameters while running the model multiple times?	yes, by using BehaviorSpace G	yes, by using multi-run, though it has never worked in our installation G	yes, and there are even multiple ways that MobiDyc will sweep the parameters for you G
be called from Java or C?	yes, there is a Java API that allows you to call a NetLogo model C	yes, you can export to Java and then compile it in any way you want C	no, since the code is in SmallTalk, it would be hard to access from anything but SmallTalk N
call Java or C standard programming libraries?	yes, you can use the **extension's apostrophe ok?** API to call out to other Java libraries and even create new primitives in the language C	yes, it supports all Python and Java objects C	no, it could read other SmallTalk libraries, but that is all N
generate data for use with other analysis tools?	yes, you can output data to text files and then analyze them, or you can use the CSV files generated by BehaviorSpace G	yes, you can output data to text files, and supposedly multi-run will output data to XML files C/G	yes, in fact, they are working on an interface with R G
be run using third-party experimental tools (e.g., execute a model via a shell process from a third-party software platform)?	yes, you can run NetLogo with the command line and pass in an arbitrary parameter list via the BehaviorSpace files C	yes, since you can create a standard Repast model, but this takes work C	no, since there is no way to run it from the command line N