TOWARD A GRAPHICAL ABM TOOLKIT WITH GIS INTEGRATION

W. RAND*, Northwestern University, Evanston, IL
D. BROWN, University of Michigan, Ann Arbor, MI
R. RIOLO, University of Michigan, Ann Arbor, MI
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, less emphasis has been placed in ABM on developing its ability to replicate spatial patterns of phenomena. In order to do that, more powerful spatial modeling techniques, like those within geographical information systems (GIS), are necessary. The integration of these two tool sets 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, but 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 detailed 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 that goal. We first outline several design principles for an ABM-GIS toolkit and then describe a survey of extant toolkits (RepastPy, NetLogo, and MobiDyc) that were selected based on the design principles. The toolkits were surveyed to see how well they fulfill some of the design principles. This survey is not meant to be a comparative review of these toolkits but rather it was conducted to determine what useful design principles can 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. 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). 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 process of a particular phenomenon, but it does not have a rich conception of the pattern of phenomena. On the other hand, while GIS are regularly used to build complex and interesting spatial models that clearly represent the pattern of a phenomenon, these models tend to be either static models of pattern or to be statistical (e.g., Markovian) models of process, and thus do not contain a rich understanding of the process of the phenomenon. Thus easy access to ABM techniques would enhance the range of models GIS users could employ, by making it

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

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 and 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 non-programmers 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, which has an easy to use GUI for developing the interface of the ABM. The second is RepastPy, which has a GUI for model development as well as strong GIS integration. The final toolkit we examined was MobiDyc, 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 we devised to categorize and describe the capabilities of each platform. After describing the results of our review of these systems, we discuss what we learned about each toolkit's contribution to the development of ABM architecture design, and then distill these lessons into a list of desiderata 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 of desired capabilities of our “ideal” toolkit as well as on general lessons gleaned from experience with existing systems.

DESIGN PRINCIPLES FOR AN ABM-GIS-GUI TOOLKIT

Having established that there are at least a few reasons why a combined ABM-GIS toolkit with a GUI would be useful, we needed to figure out what we would want in such a toolkit. By creating a list of desiderata we can start to understand how such a toolkit could be put together. Since there are three main elements to this toolkit (ABM, GIS, and GUI) we will break down what our desired characteristics within each of these three would be.

To begin with, from the world of ABM we would want the full power of scheduling and heterogeneous entities that are normally available in ABM. Thus we would want the ability to schedule an event at any time in the future, and when it occurs allow it to trigger other events. This aspect of agent-based modeling comes out of work in discrete event simulation (DES) (Cassandras and Lafortune 1999). This also allows 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 allows the modeler the ability to describe different properties and methods for different agents in the world. Moreover the modeler can combine these agents, and ask them to all carry out an action simultaneously. Another feature of ABM that has proven useful is the ability to use multiple different kinds of environments with the same model. In the case of this toolkit this might mean being able to arbitrarily switch between different GIS maps while still utilizing the same model. ABM also has a powerful representation of the environment. The environment has the ability to carry out its own processes and interact with the agents in autonomous ways. For instance, a wolf-sheep ABM may interact with grass that is growing on the environment independently of the wolf and sheep agents.

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 data set that is tied together by the physical location of those layers in the world, is a powerful model that would be useful within an ABM. For instance, residents moving around in a residential location model should be able to access information like the amount of open space, distance to 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, GUI model building would be very useful. Model builders should be able to create agents, processes and data reporters with nothing but point and clicking and the typing of a few names. However, just because the GUI is simple does not mean that it would necessarily only involve the creation of simple methods. Traditional GIS systems (like ArcView) use drop down menus to construct detailed and rich SQL (structured query language) 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, 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 how they had accomplished this integration. The specific objectives of this survey were therefore twofold: (1) to evaluate the toolkit in terms of how well it accomplished the task we had set before us, and (2) to 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 what toolkits we would examine, then we had to 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 agent-based modeling toolkits (e.g. 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 non-programmers 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 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).

We chose three ABM GUI toolkits and evaluated 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 the stated intention of their developers to provide a system that minimizes programming requirements. 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 RepastPy (Collier and North 2004), which was developed at Argonne National Lab in order to make Repast easier to use (Collier, Howe et al. 2003). RepastPy has a GUI for model development that utilizes a drag and drop interface, and RepastPy also has strong GIS integration. The final toolkit we examined was MobiDyc (Vincent, Christophe 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 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 detailed responses to these questions (Appendix 1) we also graded the ability of each system to carry out the particular function, using a simplified scale (Appendix 1

and Table 1). 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 based on our experience with building and utilizing ABMs in the past. Some of these topics are not specific to use for GIS users, but design of experiments, software interoperability, and model output through reports and graphs are important topics to the design, use, and interpretation of an ABM in general. Therefore it was deemed necessary to include 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 list the questions in approximate order of difficulty, moving from the least difficult goals to accomplish to the most difficult.

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, due to the subjective nature of these results, they may not be as applicable for your particular project.

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 into 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, 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.

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 those questions. Both 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 1 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 presented but the categories and subcategories are as well as a keyword referencing the question.

Table 1: 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	C	C	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	N	G	N
Heterogeneous	N	G	N
het. Times	N	G	N
Properties	N	C	N
Sensors			
other agents	C	C	G
Environment	C	C	G
Effectors			
other agents	C	C	G
Environment	P	C	G
Termination			
Die	P	N	G
Kill	C	N	G

Groups			
Question?	NetLogo	RepastPy	MobiDyc
Creation			
Groups	C	P	G
het. Groups	C	C	G
Scheduling			
Schedule	N	N	N

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

Environment			
Question?	NetLogo	RepastPy	MobiDyc
Initialization			
values	P	C	G
external	C	C	G
GIS	C	G	N
statistical	C	C	G
non-Euclidean	N	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	N	G	N
independent	N	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	N	C	N
analysis	C	G	G
experimental	C	C	N

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

DISCUSSION

The results of our survey were mixed. It seems obvious that no toolkit yet measures 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 forces 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 had a dramatic effect on the answer for these toolkits to some of the questions, but it was not necessarily 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 in the toolkit to be made, and thus choosing that 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 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 graphical 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 this work is a result of the fact that NetLogo has good support and includes 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.

RepastPy 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 of the environment and the click of a button. In addition, since it works with both OpenMap and ESRI products, it is useable by a wide variety of GIS practitioners. There is still work that needs to be done in terms of incorporating topological vector data, multiple layers, and being able to easily carry out spatial queries, but in general RepastPy is a good first step toward GIS integration into and ABM toolkit.

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

MobiDyc seemed to be the closest toolkit toward 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 useable 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 allowing 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 paradigm chosen in NetLogo it can be difficult to build a true discrete event simulator.

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." This may allow us to develop a prototype of an ideal ABM-GIS.

Ultimately, there does appear to be a trade-off between 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 yet and that it is possible to continue to make improvements.

REFERENCES

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

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

Collier, N. and M. North (2004). Repast for Python Scripting. Agent 2004, Chicago, IL.

Tisue, S. and U. Wilensky (2004). NetLogo: Design and Implementation of a Multi-Agent Modeling Environment. Agent 2004, Chicago, IL.

Vincent, G., L. P. Christophe, et al. (2002). "A multi-agents architecture to enhance end-user individual-based modelling." Ecological Modelling(157): 23-41.

Wilensky, U. (1999). NetLogo, Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

APPENDIX 1: FULL SURVEY RESULTS

Agents			
Question? Can it...	**NetLogo**	**RepastPy**	**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, iteratively or randomly, but not an arbitrary nth agent C	yes, iteratively, randomly and an arbitrary nth agent C	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 though right now I can't seem to add new methods on the CSCS version C	yes, agents have tasks, some of which 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 C
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	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	no, methods are something that agents and the environment have and are not agents themselves	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?	no, in the sense that there is no predefined scheduling mechanism in NetLogo. Yes, in the sense that you can ask a turtle to do anything at any time C	yes, there is a full Dynamic Event System scheduler, G	no, in the sense that there is no predefined scheduling mechanism. Yes, in the sense that you can ask an agent to perform a task conditionally G

schedule different agent classes to take heterogeneous actions?	no, but you can filter agentsets and ask different agentsets to take different actions C	yes, the different agent types have completely separate schedules G	no, but entities are completely different and carry out different actions G
schedule the same agent type to take heterogeneous actions at different times?	no, since there is no schedule but you can filter agentsets and ask them to take different actions C	yes, you can schedule events at a tick, at a pause, at an interval, at the start and at the end G	no, since there is no schedule but you can have them take different actions conditionally G
schedule agents to take actions on the basis of their properties?	no, since there is no schedule them but 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	no, since there is no schedule them but you can ask them to take actions on the basis of properties G
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 don't have particular properties or spatial nearness C	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 C	yes, agents can access information about the environment C	yes, all agents of any entity type can access all other agents G
Effectors			
have agents which affect other agents?	yes, any agent can ask any other agent to set a particular value C	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 is not supported, though you can create workarounds that "imitate death" usually C	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 C	yes, "kill" is a built in primitive G

Groups			
Question? Can it...	**NetLogo**	**RepastPy**	**MobiDyc**
Creation			
create groups of agents?	yes, through the use of agentsets C	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?	no, as mentioned above there is no general scheduling mechanism, but you can ask different groups to take different actions C	no, agents in a group can be asked to perform an action but it cannot be scheduled	no, as mentioned above there is no general scheduling mechanism, but you can ask different groups to take different actions G

Environment			
Question? Can it...	**NetLogo**	**RepastPy**	**MobiDyc**
Initialization			
create environmental values?	yes, patches have a -own 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 C	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?	no, but you can simulate them using agents to represent connections between different agents C	yes, networked environments are another built-in environment type G	no, it is all grid based
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 non-located 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?	no, but the rasters are large and agents are vector points C	yes, either through the use of a network, or through GIS data G	no, though agents are smaller than the grid

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 C	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 G
create properties of the environment in multiple layers?	no, but patches can have multiple properties which might be equivalent to multiple layers C	yes, you can create multiple layers in a GridModel, but not in a Network Model or GIS model C	no, but cells can have multiple properties which might be equivalent to multiple layers G
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 GISModels 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 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 C	no, agents are responsible for checking that they are not violating any topological rules C	no, agents are responsible for checking that they are not violating any topological rules G
Scheduling			
schedule the environment to take actions?	no, as mentioned above, but any patch can be asked to do anything at any time C	yes, the environment can use the same scheduler as agents G	no, as mentioned above, but any cell can be asked to do anything at any time G
schedule the environment to take action independently of the agents?	no, though the whole world can be asked to diffuse at any time P	yes, the environment can use the same scheduler as agents G	no, though the whole world can be asked to perform actions at a different time than the agents G

Reports			
Question? Can it...	**NetLogo**	**RepastPy**	**MobiDyc**
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, but due to the scoping rules sometimes these results can be hard to collect correctly, some commands to deal with reporting are built-in, C	yes, you can write code to calculate just about any stat C	yes, you can perform many standard statistical calculations G
calculate statistics about the environment?	yes, but see above, patches can be asked to-report values as well, C	yes, you can write code to calculate just about any stat C	yes, cells are just like agents in this environment G
output statistics to graphical displays?	yes, the graphs themselves are designed graphically but they are linked to report values in the code C	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, but can only be examined after the experiment G

output statistics to an output file?	yes, some standard I/O procedures exist C	yes, this is part of each graph you care and specified in the GUI G	yes, you can save the text used to generate any display G
output data to a GIS server?	no, but the data can be written to a text file and then imported C	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 G

Experiments			
Question? Can it...	**NetLogo**	**RepastPy**	**MobiDyc**
run the model in batch mode?	yes, turning off the update display, it can even be called from another java program C/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, 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 replication the experiment G
sweep parameters while running the model multiple times?	yes, using BehaviorSpace G	yes, using multi-run though I have never gotten it to work in our installation G	yes, and there are even multiple ways that MobiDyc will sweep the parameters for you G

Interoperability			
Question? Can it...	**NetLogo**	**RepastPy**	**MobiDyc**
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 anyway you want C	no, since the code is in SmallTalk it would be hard to access from anything but SmallTalk
call Java or C standard programming libraries?	no, NetLogo is interpreted and hence there is no way to call other code	yes, it supports all python and java objects C	no, it could read other SmallTalk libraries but that's it
generate data for use with other analysis tools?	yes, you can output data to text files and then analyze it, 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?	yes, but in a tricky fashion since you would have to work through the provided Java API instead of just using a command line processor 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