Review of Spatial Simulation Tools for Geographic Information Systems

Luís Moreira de Sousa
Instituto Superior Técnico
Lisbon, Portugal
luis.moreira.de.sousa@ist.utl.pt

Alberto Rodrigues da Silva
INESC-ID
Instituto Superior Técnico
Lisbon, Portugal
alberto.silva@acm.org

Abstract—**Spatial simulation has been largely absent from traditional Geographic Information Systems (GIS) software packages. Both the advanced skills needed to use this technique and the relative specificity of its application has resulted in a myriad of independent tools, each with different features. The choice of a proper tool for disclosing the dynamics of change in a GIS context is anything but obvious. This work presents a comparative review of different types of tools available for the development of Spatial Dynamics models. These tools are compared along three different vectors: application domain, ease of use by non-programmers (the typical GIS technician) and interoperability with geo-referenced data. Unlike for other disciplines (e.g. systems engineering) a simulation tool for GIS with a wide variety of application domains but accessible to non-programmers seems largely lacking.**

Keywords: Spatial Simulation; Cellular Automata; Agent Based Modelling;

I. INTRODUCTION

The data stored in an information system usually portraits the world as it is now, or was at a specific point or interval in time. This is especially true for spatial data but in this case with the added certainty that it will also evolve. The patterns of land cover and land use, of social, economic, and demographic variables in general, constantly change with time. Objectively, any piece of spatial data is valid only within a specific time frame, just as if any cartographic composition was a still picture taken to the elements represented.

In order to deal with this reality, entire organizations exist with the sole purpose of collecting and updating spatial data, by field campaigns with on site visits, by air borne or space borne data acquisition [1]. Nonetheless, regular data collection provides at best a periodic picture of the changing reality, which for some applications might not be enough [2]. Stakeholders of an information system may need to know not only how the data changed in the past, but in order to plan ahead or otherwise reason upon the data, they may also need to understand why it changed the way it did and how it might continue to evolve in the future.

This need is met recurring to two processes that are part of the same scientific domain: Spatial Modelling and Spatial Simulation. *Modelling* is the process by which the fundamental drivers of change - the *Spatial Dynamics* - are captured into mathematical, logic or functional constructs. *Simulation* is a process through which a model is applied to a set of data during a certain period of time. Modelling and Simulation can be seen as a single technology, for the process of Modelling is chiefly a trial version of Simulation. Spatial Dynamics is captured by applying heuristic or hypothetical models to periods of time for which the data evolution is known, thus allowing for validation and/or calibration. When the model reaches a satisfactory level of success against known data it can then be applied to periods of time for which knowledge is scarce (usually the future) producing new sets of data, pictures of time epochs missing from the base data [2].

The oldest of the techniques used in Spatial Simulation are Cellular Automata (CA) [3] in which the world is discretized in a grid of equal sized cells that evolve in accordance to a fixed set of rules. More recently, Agent-based Simulations have become a popular paradigm that has also been applied to spatial simulation. An agent can be defined as an autonomous object that perceives and reacts to its environment [4,5], a concept that largely benefited from the emergence of Object Oriented (OO) programming. Agent-based Simulations and CA are two concepts that superimpose to some extent in the GIS context, though the former brought new planes of processing where geographic entities not only react to stimuli but also store knowledge and reason before acting. Beyond that, agents can be used to model phenomena that do not have precise geographic meaning, such as social or economic interactions.

On the GIS related sciences, Spatial Simulations have been used extensively, of which the following fields can de highlighted

- *Urban Planning* - understanding and forecasting changes in the urban landscape to allocate new infrastructure [2];
- *Land Use* - studying the dynamics of land use, e.g. changes between agricultural, urban and forest areas [6];
- *Forestry/Wild Fire* - understanding forest growth, studying and anticipating fire spread [7];
- *Biology* - modelling habitat evolution and studying population dynamics [8].

Of the several spatial analysis techniques, Spatial Simulation is the most complex; a simple statistical or mathematical trend analysis, predictive enough for most data recorded in regular information systems, is insufficient in GIS due to the multi-dimensional and heterogeneous character of spatial data. Furthermore, the augmented degrees of freedom of spatial data result in highly specific models, only usable within the particular application in focus. Thus, most spatial dynamics models are developed *ad hoc* by the end user organization, developing its own software libraries. Modern GIS packages continue largely lacking tools dedicated to this technology.

Using a general purpose programming language to build a Spatial Simulation the majority of the instructions coded are extraneous to the concepts in the underlying model. Besides implementing the model, the program has to control the flow of execution, manage system resources, and manipulate data structures. Burdening the model with these tasks can lead to several problems [9]: (i) difficulties verifying the correct model implementation by the program; (ii) limited model generality due to difficult modification and/or adaptation; (iii) difficulty comparing computer models, usually restricted to their inputs and outputs [10]; (iv) problematic integrating with other models or tools (e.g. GIS or visualization packages), often limited to the exchange of output files.

Beyond general purpose programming languages, presently a spectrum of Spatial Simulation tools can be devised, ranging from those that present support at Program-level, closer to the programming language, to those that operate at Model-level, closer to the conceptual model that represents reality [9]. Somewhere in the middle of this spectrum lay Domain Specific Languages (DSL). For each of these categories there is a set of advantages and drawbacks that must be carefully weighted before choosing a particular tool.

This article reviews a series of spatial modelling/simulation tools in the GIS context, in which of the categories presented: Program-level tools (Section II), Model-level tools (Section III) and Domain Specific Languages (Section IV). Section V compares the set of tools reviewed along three vectors: application domain, ease of use and interoperability with geo-referenced data. Section VI sums up the article and its conclusions.

II. Program-level Tools

Program-level support tools extend the facilities available in general-purpose programming languages, usually providing useful software libraries for building specific classes of models. This approach substantially reduces coding time and can increase program reliability. Higher-level code, usually in a general-purpose OO programming language, specifies how objects are used to produce the desired model behaviour. These tools can be called code packages, code libraries or toolkits.

The main advantage of this type of tools is the encapsulation of the model from functionality not directly related to spatial dynamics. These include, graphical display, data input and output, statistical data collection, etc, for which a plethora of functions is provided in the form of a code library. The improvements are two fold: (i) it relieves the modeller from banal programming tasks, allowing a higher focus on dynamics; (ii) it produces leaner and easier to read code, for much complexity is isolated and standardized by the code library.

On the downside these tools require an extra learning effort for their proper use. Beyond having relevant knowledge on the base programming language, a modeller wishing to use on of these tools must learn to some detail the behaviour of at least part of the functions/objects/methods provided by the tool-kit. The more the functionality it has to offer, the longer will it take to fully learn its usage. Besides that, Bennenson and Torrens [11] suggest that with denser libraries, programmers can eventually run into some discomfort with conflicting or incompatible functionality that is only found at later development stages. These disadvantages have been mitigated to some extent with the emergence of user communities that share experience and assistance and by opening and sharing the tool-kit's source code.

De Smith et. al. [12] reported that by 2007 more than 100 of these toolkits were available worldwide. A selection of the most popular is described below.

Swarm was the first of these tools, developed during the 1990s at the Santa Fé Institute, delivering a set of objects and methods for the development of spatial simulations and results presentation [13]. It yearned great popularity in its early days, but integration with GIS is weak, limited to raster data.

The Recursive Porous Agent Simulation Tool-kit (**RePAST**) is a newer Java library that evolved from an eclectic package at the Chicago University, supporting different techniques that go well beyond spatial simulation, which have made it very popular [14]. Perhaps the most useful of these tools today, it also provides good integration with GIS.

The Multi-Agent Simulator Of Neighbourhoods (**MASON**) is also a Java library but conceived with the aim of being light, fast and portable. Conceived at the George Mason University, it is a modern tool, highly compact, that although providing less functionality than RePAST [15], already supports interaction with both vector and raster data sets.

III. Model-level Tools

Model-level support tools allow the usage of spatial simulation models without requiring programming. These are pre-programmed models, designed for specific application fields that can be parametrized by the user. The larger the number of parameters the user can set and update, the larger the tool's flexibility. They allow faster model development and provide fairly straightforward mechanisms for implementation, though invariably constraint the modeller to a specific application and dynamics framework.

The Object-Based Environment for Urban Simulation (**OBEUS**) is a tool dedicated to Urban Planning and Management, developed at the Tel Aviv University, as an implementation of the theory of Geographic Automata Systems [11]. The tool allows the development of models through a graphical interface that then generates a C# coded simulation which maybe further refined by coding in a commercial C# workbench.

AnyLogic is an eclectic commercial tool supporting various areas of simulation, with pre built models on specific areas. It provides several graphical languages to develop model behaviour through state charts and flow diagrams, plus a code library to be used with Eclipse for model refining. It also ships with a GIS API that allows the input of spatial data.

The Tool for Exploratory Landscape Scenario Analyses (**TELSA**) is a program specialized in ecosystems, the typical commercial tool for spatial simulation, allowing the study of different management scenarios. It completely dispenses programming and is parametrizable through a diagrammatic language (VDDT) developed by the vendor, ESSA Technologies [16]. It is dependent on several third party commercial software, included those that provide GIS interoperability.

LANDIS is the result of a joint project of the US Forest Service with several universities of that country, is a simulation for the forest land cover at large scale. The user provides a set of input spatial variables in the form of raster layers for which a number of pre-defined behaviours is available [17].

SLEUTH is the oldest of the tools of this genre, created back in the mid 1990s at the University of California and dedicated to urban development. It uses only six spacial raster layers as input, for which a set of behaviours can be adjusted. It became a popular tools in its domain, being successfully applied to different parts of the world [18].

IV. Domain Specific Languages

Midway between Program-level and Model-level support tools are domain specific tools, usually providing Model-level support for a range of application domains. They make fewer assumptions about the underlying model structure than do pre-programmed models, often providing ways of developing new behaviours. Programming is often required but in a restricted environment where behaviour is described using simple constructs, encapsulating most of the traditional coding activities. A pure DSL provides a programming language, either textual or graphical, as the sole developing infrastructure.

StarLogo/NetLogo is the last of a generation of languages that evolved at the MIT from the functional language *Logo*, specialized on agent-based simulation. Closed source, it has been used as teaching tool due to the simplicity of the code it produces. Nevertheless, it may also be a useful option for prototyping in real life problems [19]. Interaction with GIS is supported, but only for input data sets.

AgentSheets is a simulation tool funded by the National Science Foundation in the United States , developed for teaching purposes whereby models are built in a drag-and-drop interface using graphical stereotypes [20]. It is being used as the basis of several educational courses mostly aimed at high school. Though simulations with a spatial meaning can be developed, no integration with GIS data is available.

The Spatially Explicit Landscape Event Simulator (**SELES**) is a declarative language for landscape dynamics modelling, resulting froma research project at the Simon Fraser University. It tries to balance the flexibility of programming with the ease of use of pre-programmed models [9]. A dedicated development environment is provided, that though closed source, is freely distributable. GIS interoperability is guaranteed by the input and output of raster datasets.

Financed by the Institut National de la Recherche Agronomique (INRA) in France, ,the Modelling Based on Individuals for the Dynamics of Communities (**MOBYDIC**) project produced a programming language dedicated to population dynamics. It allows the development of complex models from simple primitives, close to natural language [21], in reality being a code library for the OO language Smalltalk. It provides no direct interoperability with spatial information.

V. Comparison and Present Difficulties

In this section a comparative classification is performed of each tool according to three vectors of analysis: applications domain, ease of use and GIS interoperability. A three grade system is used: good, medium and weak, denoted respectively by three, two and one stars. In cases where a particular tool doesn't provide support no grade is attributed (represented with the “-” character).

Table I compares the application domain of each tool. In this comparison not only are taken into account the native application areas, but also the tools' underlying platform and distribution flavour. While a certain tool may present itself as a one-size-fits-all solution for spatial simulation, it is important to assess other constraints to its application, such as platform dependency, extensibility or portability. What can be observed from this comparison is that Program-level tools are much more broad reaching in this regard than other tools. Model-level tools or DSL not only narrow their scope in their native application field but also invariably introduce dependencies on third parties, either be it on other software or operating systems. Only two of these tools stand out in this regard: AnyLogic and NetLogo, which attempt at wider portability by adopting Java as platform; nonetheless, being closed source tools, are always at a disadvantage against Program-level tools, especially in scientific applications where verifiability is paramount.

In Table II is compared the ease of use of each tool. Without surprise Model-level tools appear as those easier to learn and use without programming training. These are also almost exclusively those tools that provide graphical interfaces for model development. RePAST provides a diagrammatic interface for behaviour description, but it is somewhat restricted to a single aspect of development. Of the DSL, only AgentSheets provides a graphical development interface, an aspect that casts these tools at visible disadvantage against Model-level solutions.

The last comparison vector, GIS integration, is presented in Table III. Each tool was assessed in terms of its capability to interact with spatial datasets in common formats (Shapefile, TIFF, etc) both as inputs to models and as outputs to visualize results in GIS software. This assessment also distinguished between vector and raster formats, for some data may only be available in one of them (e.g. satellite imagery). The first point to make is that specialization seems also to impose a loss of GIS interoperability. Of the twelve tools surveyed, only five can both read and write some form of geo-referenced data, and of these, only two - MASON and RePAST - operate with both raster and vector datasets. Two of the DSL don't even allow any sort of direct interaction with GIS data. Spatial result output, particularly, seems to be an area where many spatial simulation tools are yet to reach maturity.

Looking at Program-level tools in general, they can alleviate some of the burdening of directly using a general purpose programming language, but still require good programming skills from the modeller [22]. The full knowledge of one of these code libraries is something achievable only with several months of practice [23]. Today these tools are tendentiously open source, by one way or another operating on several computer platforms and providing good GIS integration. Coupling this characteristic to their wider application scope, Program-level tools usually gather around them large communities of users, that provide informal, but extensive, support.

Model-level support tools tend to be quite specific, and much of the model behaviour and assumptions are hidden in the program and may not be explicit or modified; their use in other application fields is largely impossible. The modeller can in fact dispense programming skills using this kind of tools but gets constrained to a specific field and overall simulation behaviour. They also tend to narrow the interaction with geo-referenced data, by imposing certain formats or in some cases by lacking output functionality. Evolution or generalization of these tools can sometimes become too expensive and fate them to extinction. Traditionally they take advantage of market niches providing for the needs of a specific and restricted group of users, thus the commercial nature of many of them. Community support is usually weak or non-existent; more often, support is a paid service.

The use of DSL facilitates modelling and reduces the build-up time of Spatial Simulations, but existing languages do not avoid the need of programming skills. As with any other programming language, the user has to understand keyword meaning and how to compose a set of instructions into a program. Also, in general, these languages produce final models with lower computational performances than those produced with Program-level support tools. DSL for spatial simulation are found mainly for educational purposes, in some cases more resembling toys than analysis tools. This is also patent in the lack of GIS integration most of them show, some even totally lacking such sort of functionality. Users communities tend to be larger than those of Model-level tools, but on the other hand platform dependency is often an issue.

The survey presented can be used as a guide to choose a spatial simulation tool for GIS applications, but the weight of each comparison vector should always be adapted to each particular case. For applications where GIS integration is a relevant need, with both input and output of geo-referenced data being a requisite then MASON and RePAST are nearly the only options. On the other hand, if ease of use is a more important necessity, then models like LANDIS or SLEUTH can be options if matching the application domain. Somewhere in between can be found SELES, that too imposes a relevant application narrowing, and NetLogo, which essentially trades ease of use for GIS integration and extensibility.

VI. Summary

Techniques for Spatial Simulation have existed in one way or another for many decades, actually preceding the emergence of GIS software. It was only with the maturing of the latter that Simulation was envisioned on large scale spatial datasets. In the wake of the OO maturing process, a host of software tools appeared throughout the 1990s providing support for spatial simulation in most (if not all) GIS fields of application.

The main objective of these techniques is to study Spatial Dynamics, the set of local rules or constructs that when repeatedly applied to the variables and space considered produce unanticipated macroscopic results. Spatial Dynamics analysis is a process composed by two main steps: Modelling and Simulation. The Modelling phase discloses the rules by which the variables in analysis changed the way they did; this is usually a prototyping process against a set of historical data. With the model fully developed, it is then applied to the last known state of the space domain for predictive purposes. The results of this process are the drivers of change (the Dynamics) and future evolution of the spatial domain being studied.

Existing software tools for Spatial Simulation can be classified in three types: Program-level, Model-level and DSL. Program-level tools are code libraries providing specific methods for the rapid coding of models with popular OO languages; usually multi-purpose and cross-platform, they gather large user communities. Model-level tools are parametrizable pre-programmed models aimed at strict fields of application; largely dispensing programming skills, they tend to be used by small groups of users and are usually dependent on commercial software or are commercial themselves. DSL try to bridge between the two other types, providing easier model set-up environments without compromising application scope as much as model-level tools; whilst gathering relevant communities in some cases, DSL tend to be mostly educational tools, with fewer examples of real-life application.

Of a set of twelve different simulation tools surveyed only two showed to be fully matured when it comes to the integration with GIS data, both Program-level libraries: MASON and RePAST. A trend is apparent whereby ease of use implies a loss of functionality regarding geo-referenced data input and output; some Model-level tools show some degree of GIS integration but impose a significant scope limitations. NetLogo is the tool closest to bridge this gap, though impaired by a closed source philosophy and lack of geo-referenced data output.

All the tools considered, with no exception, present important compromises in their choice for spatial simulation in the GIS domain. Space for improvement in the field seems to exist.

References

[1] Kraak, M. and Ormeling, F., "Cartography: Visualization of Spatial Data". Prentice Hall; 3rd edition. 2009.

[2] Batty, M., "Cities and Complexity", MIT Press, 2007.

[3] Wuensche, A. and Lesser, M., "The Global Dynamics of Cellular Automata", Addison-Wesley, 1992.

[4] Ferber, J., "Multi-Agents Systems. In: An Introduction to Distributed Artificial Intelligence", Addison-Wesley, 1999.

[5] Weiss, G., ed., "Multiagents Systems: a Modern Approach to Distributed Artificial Intelligence", MIT Press, 1999.

[6] Messina, J.P. and Walsh, S.J., "The application of a cellular automaton model for predicting deforestation", In: Proceedings of the 4th International Conference on Integrating GIS and Environmental Modelling: Problems, Prospects and Research Needs, Banff, Canada, 2000.

[7] Li, X. and Magill, W., "Modeling fire spread under environmental influence using a cellular automaton approach", Complexity International, 8, 1–14, 2001.

[8] Ermentrout, G.B. and Edelstein-Keshet, L., "Cellular Automata Approaches to Biological Modeling", Journal of Theoretical Biology, 160, 97–133, 1993.

[9] Fall, A. and Fall, J., "A domain-specific language for models of landscape dynamics", Ecological Modelling, 141, 1–18, 2001.

[10] Olde, V. and Wassen, M., "A comparison of six models predicting vegetation response to hydrological habitat change", Ecological Modelling, 101, 347–361, 1997.

[11] Benenson, I. and Torrens, P., "Geosimulation: Automata-based modeling of urban phenomena", London: John Wiley & Sons, 2004.

[12] de Smith, M.J., Goodchild, M.F., and Longley, P.A., "Geospatial Analysis: A Comprehensive Guide to Principles, Techniques and Software Tools", Troubador Publishing Ltd, 2007.

[13] Minar, N., Burkhart, R., Langton, C. and Askenazi, M., "The Swarm simulation system: a toolkit for building multi-agent simulations", 1996.

[14] Collier, N., Howe, T., and North, M., "Onward and upward: the transition to Repast 2.0", In: First Annual North American Association for Computational Social and Organizational Science Conference, Pittsburgh, Pa, 2003.

[15] Luke, S., et al., "MASON: A Multi-Agent Simulation Environment". Simulation: Transactions of the society for Modeling and Simulation International, 82(7), 517–527, 2005.

[16] Merzenich, J. and Frid, L., "Projecting Landscape Conditions in Southern Utah Using VDDT", In: M. Bevers and T.M. Barrett, eds, Systems Analysis in Forest Resources: Proceedings of the 2003 Symposium, Portland, OR, 157–163, 2005.

[17] Mladenoff, D., "LANDIS and forest landscape models". Ecological Modelling, 180 (1), 7 – 19, 2004.

[18] Yi, W. and He, B., "Applying SLEUTH for Simulating Urban Expansion of Beijing", In: 2009 International Joint Conference on Artificial Intelligenc, Vol. 2, 652–656, 2009.

[19] Railsback, S.F., Steven, L.L., and Jackson, J.K., "Agent-based Simulation Platforms: Review and Development Recommendations", Simulation, 82 Issue 9, 2006.

[20] Repenning, A. and Ioannidou, A., "Broadening Participation through Scalable Game Design", In: in Proceedings of the ACM Special Interest Group on Computer Science Education Conference, (SIGCSE 2008), 305–309, 2008.

[21] Ginot, V., Le Page, C., and Souissi, S., "A multi-agents architecture to enhance end-user individual based modelling", Ecological Modelling, 157, 23–41, 2002.

[22] Tobias, R. and Hofmann, C., "Evaluation of free Java-libraries for social-scientific agent based simulation", Journal of Artificial Societies and Social Simulation, 7 (1), 2004.

[23] Samuelson, D. and Macal, C., "Agent-Based Simulation Comes of Age", OR/MS Today, 33 (4), Lionheart Publishing, Marietta, GA, USA, 2006.

TABLE I. COMPARISON OF THE TOOLS SURVEYED REGARDING APPLICATION RANGE.

		Application	**Programming Platform**	**Distribution**	
Program-level tools	**Swarm**	Multi-purpose	Objective-C	Open Source	★★★
	MASON	Multi-purpose	Java	Open Source	★★★
	RePast	Multi-purpose	Java, .NET	Open Source	★★★
Model-level tools	**OBEUS**	Urban Planning	.NET	Shareware	★
	AnyLogic	Several Specific	Java	Commercial	★★
	TELSA	Landscape Management	unknown	Commercial	★
	LANDIS	Forest Succession	.NET	Shareware	★
	SLEUTH	Urban Development	C	Open Source	★
Domain Specific Languages	**NetLogo**	Multi-purpose	Java	Shareware	★★
	AgentSheets	Educational	unknown	Commercial	★
	SELES	General Landscape	unknown	Shareware	★
	MOBIDYC	Population Dynamics	Smallralk	Open Source[a]	★

a. Dependent on commercial software.

TABLE II. COMPARISON OF THE TOOLS SURVEYED REGARDING EASE OF USE.

		Modelling Language	Development GUI	Programming Skills	Community	
Program-level tools	**Swarm**	Objective-C, Java	none	high	Wiki, Mail-list	★
	MASON	Java	none	high	Mail-list	★
	RePAST	Java, C#, others	Eclipse	high	Mail-list	★
Model-level tools	**OBEUS**	C#	yes	low to high	none	★★
	AnyLogic	Diagrammatic, Java	yes	low to high	none	★★
	TELSA	VDDT	yes	none	none	★★★
	LANDIS	Parametric	none	none	Forum	★★★
	SLEUTH	Parametric	none	none	Forum	★★★
Domain Specific Languages	**NetLogo**	Logo specialization	none	low to medium	Mail-list	★★
	AgentSheets	Conversational Prog.	yes	none to high	none	★★
	SELES	Declarative DSL	none	none to medium	Wiki, Forum	★★
	MOBIDYC	Declarative DSL	none	none to medium	none	★★

TABLE III. COMPARISON OF THE TOOLS SURVEYED REGARDING GIS INTEROPERABILITY.

		Input	Output	Vector	Raster	
Program-level tools	**Swarm**	√	×	×	√	★
	MASON	√	√	√	√	★★★
	RePast	√	√	√	√	★★★
Model-level tools	**OBEUS**	√	×	√	×	★
	AnyLogic	√	×	√	×	★
	TELSA	√	×	√	×	★
	LANDIS	√	√	×	√	★★
	SLEUTH	√	√	×	√	★★
Domain Specific Languages	**NetLogo**	√	×	√	√	★★
	AgentSheets	×	×	×	×	-
	SELES	√	√	×	√	★★
	MOBIDYC	×	×	×	×	-