URBAN PLANNING BY SIMULATION OF POPULATION GROWTH

Cirano Iochpe, Flavio Rech Wagner, André Dias Bastos, Guillermo Nudelman Hess, Marcia Aparecida da Silva Almeida
UFRGS – Universidade Federal do Rio Grande do Sul. Instituto de Informática.
{ciochpe, flavio, adbastos, hess, marcia}@inf.ufrgs.br

Abstract: Urban planning is a very important issue to guarantee the sustainable development of modern towns. Many aspects must be considered and one of the most important is the population growth. Having a correct control of this phenomenon it is possible to focus the efforts, resources and investments in the right direction. To anticipate the facts and as a consequence predict the advantages and problems of the growth in a specific region this paper presents a web-based system for the simulation of the populations boom. The system uses a set of rules to predict the development in a given geographic scenario, focused on the needs for public resources such as schools, kinder gardens and public health centers. The outputs are SVG covering maps of the simulated region showing the suitability of the existing resources.

Key words: Geographic Information Systems, urban growth, simulation, geographic scenario

1. INTRODUCTION

Urban systems are becoming ever larger and increasingly complex as urban economies, social and political structures and norms, and transportation and other infrastructure systems and technologies evolve. Scarce resources make efficiency critically important, and in a democratic context that involves many stakeholders with conflicting values and priorities, it is neither feasible nor appropriate to deal with major land use and transportation policies and investments as isolated choices to be decided by planners or bureaucrats within the bounds of a single organization (Waddell and Ulfarsson, 2003)

Urban growth modeling and prediction history essentially started in the 1950's, showed less activity in the 70-80's but has been revived somewhat in the 1990's due to the availability of spatial data and advancements with computer technologies and Geographic Information Systems (GIS) software. Conflicting views of urban system have led to a variety of different growth theories and models (Allen and Lu, 2000).

Simulations based on geographic environments are called Geosimulation (Benenson and Torrens, 2003; Torrens, 2004). A Geosimulation consists on the discrete simulation of dynamic spatial systems, action oriented (Benenson and Torrens, 2003).

Following an ongoing research we intend to concentrate our efforts in simulation applications based on geographic scenarios. This kind of simulations is of interest of a considerable subset of our partners such as the army (strategy war games) and ambient licensing departments (simulation of future environment conditions in case of a polluting enterprise locates in a specific place).

We chose to simulate the urban population growths as our case study. The choice was due the availability of data and the interest of the city planning department, which gave us the data. The main difference between the simulator proposed in this project and the existing ones is the fact that our simulation is influenced not only by external events, such as human intervention, social and economics policies but also by the geographical environment aspects, such as hydrology and relief.

The simulator is intended to be available as web pages on the Internet. It has to allow a user to simulate his own data set and, furthermore, using the growth rules specified by him.

The paper is organized as follows. Section 2 presents some related works. The features and functionalities of the simulator are detailed in section 3. Section 4 explores the system architecture, the simulator kernel and the algorithm for the covering map. Section 5 discusses the output in SVG format. Finally conclusions and future works are presented in section 6.

2. RELATED WORK

A number of simulators have been proposed to address the simulation of the urban growth. Most of them are not available on the internet, that is, they are desktop applications. Furthermore, they are dependent on a proprietary GIS platform and data format.

The UrbanSim (Teerarojanarat, Fairbairn and Chunithipaisn, 2004) was developed to the Oregon Department of Transportation TLUMP. It has been released as an open source software and licensed under the GNU General Public License (GPL). The software is implemented in Java and its user interface is built on top of the Eclipse platform. The key objective of the software is to simulate the development of urban areas, including land-use, transportation, and environmental impacts, over time periods. By determining various comprehensive socio-economic data inputs and user-specified events and actions (e.g. policy planning, environmental constraints), the software can create realistic urban simulation results with different scenarios (Teerarojanarat, Fairbairn and Chunithipaisn, 2004).

The Uplan (Johnston and Shabazian, 2002) simulator is urban growth model that runs in the Windows version of ArcView on a personal computer. The model was designed to rely on a minimum amount of data, but allocates urban growth in several land use types for small (parcel-sized) grid cells. It is a scenario-testing model that can be applied to any county or metropolitan region and that is transparent to the user, making it easy to change the assumptions for land use allocation. The model is rule-based, that is it is not strictly calibrated on historical data and uses no choice or other statistical models. (Johnston and Shabazian, 2002).

2.1 The simulator features and functionalities

The simulator is intended to be available as web pages on the Internet. It has to allow a user to simulate his own data set and, furthermore, using the growth rules specified by him. To use the simulator the user has to be registered in the system. This is required because one user may want to make his scenario and rule set public or private.

Some restrictions are imposed at the moment, in order to build an operational prototype faster. Only schools, kinder gardens and public health centers are human made relevant aspects, as well as only relief and hydrology are the only natural aspects used.

- **Upload new geographic scenario:** In a first implementation, the user can upload his own geographic scenario using shapefile or GML (Open GIS, 2003) files. If he chooses to use shapefiles, he just needs to indicate where they are. If he prefers to upload a GML file, he first receives an ".xsd" file containing the schema he has to follow to instantiate his data. If he uses other schema than the one we provide, the GML file cannot be validated and is considered as invalid. All the information, even in shapefile or GML, may be contained in one or various layers (files). Thus for each user is created a section to enable him to upload all the set of files. Only the user who created a scenario is able to upload or modify the scenario files.

- **Upload a rule set:** The rules must be described in an XML file, based on a schema defined by us. The procedure to upload a rule set is the same used to upload a scenario, that is, he needs to login into the system and the rules file is validated against the schema. The owner of a rule set can update it whenever he wants, adding new rules, modifying or deleting existing ones. The users other than the owner may use and visualize the rule sets, if they are public and use them as a basis to define new ones.
- **Generation of covering maps:** Having chosen a geographic scenario and a rule set, the user can ask for the simulator to generate a covering map. A covering map is an instant static view of the scenario once the rules are applied. Currently, we are generating scenarios to analyze the area covered by a school considering its capacity, the maximum distance ratio between it and the houses and geographic barriers.
- **Perform a simulation:** This is the main functionality of the simulator. Given a geographic scenario, a set of rules, the population growing direction from a center origin, the growing ratio and the number of cycles wanted, the system simulates the population growing and its impacts over the region. In each cycle a number of events are dispatched and the geographic scenario components are updated. At the end of each cycle a new covering map is generated. Optionally, the user can configure the simulator not to refresh the web page in every cycle, but only after a number of cycles.

3. THE SIMULATOR ARCHITECTURE

The global architecture of the simulator is presented in Figure 1.

After comparing some of the open-source GIS libraries such as TerraLib (Câmara et. al, 2002), PostGIS (Ramsey, 2004) and Geotools (Schulz, 2004) we decided for the last one due to its simplicity and because it has lots of examples that can be easily customized to address our needs.

It is true that we first tried to use the PostGIS library, but we could not make work correctly when running together with the GeOS, which is required to perform some of the operations we need. Furthermore, with the PostGIS we were restricted to the PostgreSQL DBMS. Another important point is that it uses the C/C++ programming language and thus it is not totally true to say that it is platform independent. Some customizations have to be done in order to migrate an application from one platform to another.

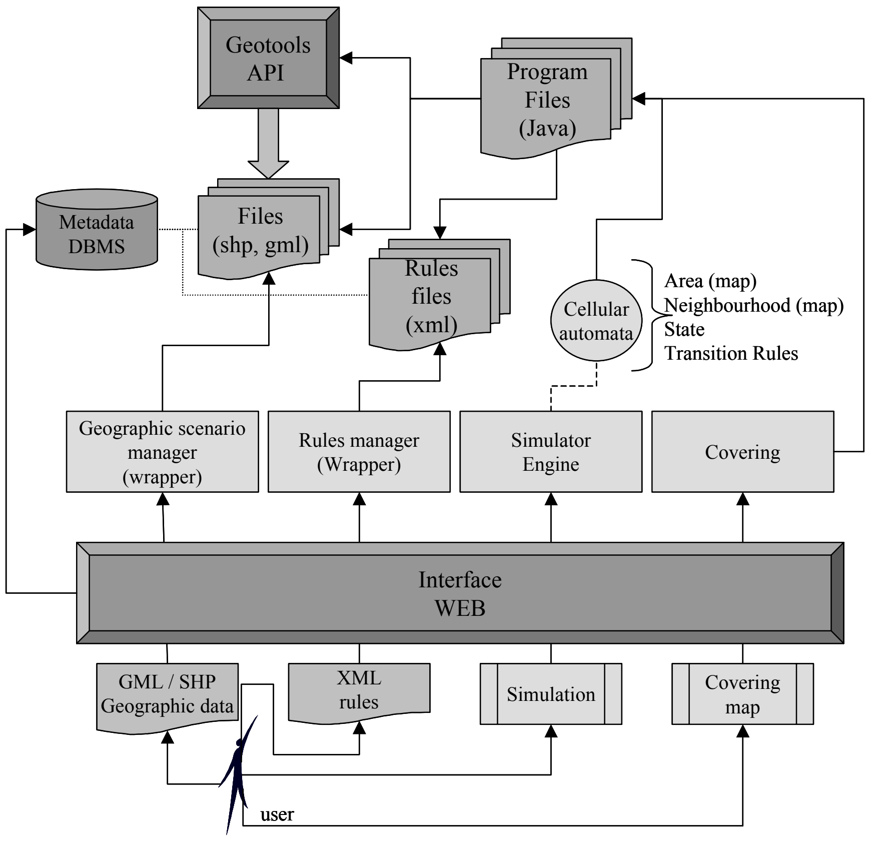

Figure 1. The simulator architecture

On the other hand, the Geotools API is an open source library which has a number of methods to perform spatial operations and manipulation of geographic data. It has a set of classes specifically to manipulate geographic data. Those classes are written in Java and thus it is easy to implement, use and extend them. Furthermore, Java is actually platform independent and with some special classes it is easy to integrate the Geotools in the web environment.

Once having chosen the Geotools API, we decided not to use a DBMS to store and manipulate the geographic data (layers). The reason is because this API does not provide, at the moment, an easy interface to connect and use spatial databases. Thus we decided to keep both the rules and the scenarios in files. These are GML or shapefiles, the ones supported by the Geotools. The database is used only to store the scenarios and rules metadata, such as date of creation, owner, if it is public or private, etc. A simulation scenario, that is, all the configuration of a geographic scenario with a set of rules, the number of cycles and the growth directions is also stored in the database.

3.1 Cellular automata

An alternative that has been widely applied in the urban simulation, specially to model and control the population growing is the use of cellular automata (White and Engelen, 1993). There are many systems based on cellular automata, such as OBEUS (Benenson and Torrens, 2003), jTrend (Huang and Cho, 2002) and URBSIM (Bäck, Dörnemann, Hammel and Frankhauser, 1996). The jTrend is implemented in Java.

Cellular automata are discrete and dynamic systems. Their behavior is totally specified in terms of the location relationships. The space is organized as an uniform grid, and each cell contains a few bits of data. The time advances in discrete steps (non continuous) and the rules that drive the universe are expressed in tables.

The sctructure of each cellular automata can be described as a 4-uple:

$$C = (Sp,\{St\},\{N\},\{TR\})$$

where:

Sp represents the space occupied by the cell.

St represents the set of states possible for a cell.

N represents the number of neighbors of the cell.

TR represents the set of transition rules that drives the changing from one state to another.

The transition rules and, as a consequence, the state of a cell depends on the time instant and on the states of the neighbor cells.

Since each cell of the matrix is totally independent (and the order does not matter), their state update may occur in parallel. There is not a need for a centralized control (Bäck, Dörnemann, Hammel and Frankhauser, 1996).

3.2 The simulator kernel

We decided to implement a new simulator kernel even knowing that a number of simulators already exists. The reason for our decision is because the kernel we need is extremely simple. It only has to count time and call the right methods for each event. Thus, it was easier and faster to develop our own simulator kernel.

The kernel of the simulator is a single Java class called Simulator. As showed in Figure 2, there are some auxiliary classes, separated from the Simulador to keep it independent of the simulations domain. Hence, it is generic and can be easily adapted to any simulation with the same characteristics than ours.

- The Simulator class requires four parameters.
- Events file: An XML file containing the events sequence and when each one is scheduled;
- Number of cycles to consider in the simulation;
- Simulation chronos (cycle time);
- Visualization time (screen refresh interval).

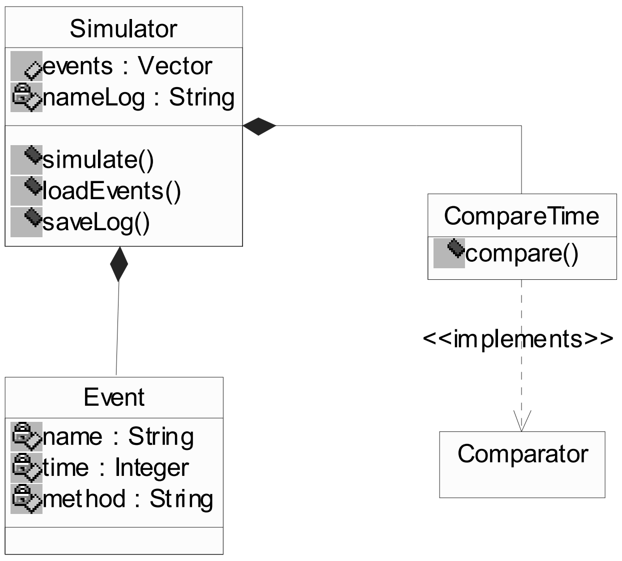

Figure 2. The simulation classes

After reading the events file, the program stores them into an ordered vector of classes called Event. Each Event class has three attributes: the name, time and associated method. The execution of the simulation is a simple loop which stops when reaches the last cycle or the events vector is empty. As illustrated in Figure 3, in each cycle the vector is ordered and the events are tested to find out if they have to be executed or not. When a method is called due to an event, the taken action is added to a log file, which contains the entire simulation summary. The log file's name is "simulacao_AAAAMMDD_HHMM", where AAAAMMDD is given by the simulation's date and HHMM is the simulation's time. In addition, the header of the simulation log contains the input parameters. As a method may add new events, the vector is reordered at each cycle.

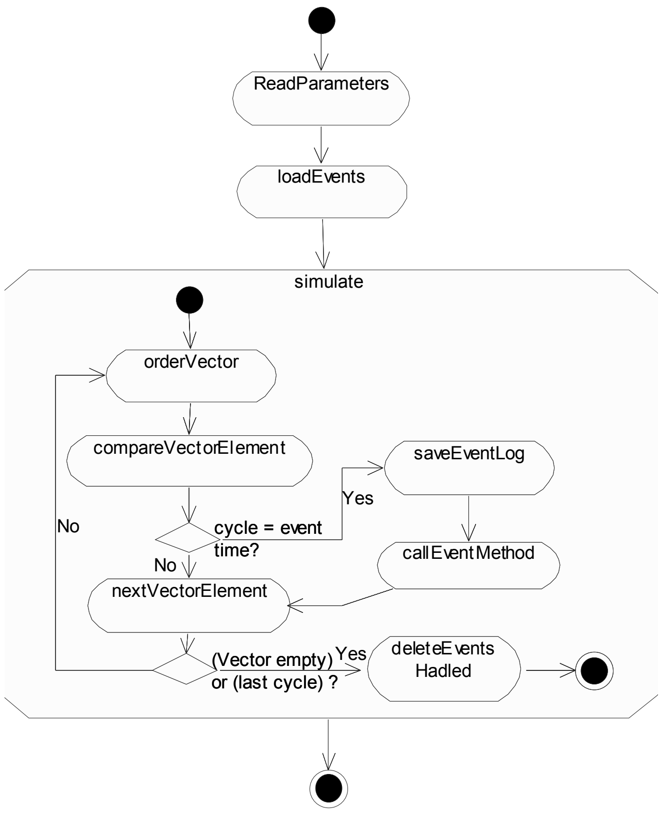

Figure 3. The simulation activity diagram

3.3 The covering map algorithm

The covering map algorithm is the core of the simulator. It presents a view of the environment based on mathematics calculus, which consider the geographic scenario and its components. Both natural aspects such as hydrology and relief and human stuff (equipments) such as schools, kinder gardens and public health center are considered. In this section we detail the algorithm we developed.

Using a web-based interface the user selects the parameters to the generation of the covering map, which are the geographic scenario, the rule set and the equipments. In the first implementation only the schools are being considered. Prior to start the generation, the system verifies if the requisites are satisfied, that is, if in the selected scenario the equipments exist and if there are rules for the equipments in the selected rule set.

At the present moment, the layers are fixed, and the system compels the user to upload a complete scenario. None layers can be missing and additional layers are discharged. The same occurs with the rules set. Actually the rules are fixed. The user can only change the parameters. For the future we intend to make the system more flexible.

The algorithm consists in the creation of a layer. Figure 4 presents an initial activity diagram which contains the main flows and data evolved in the process.

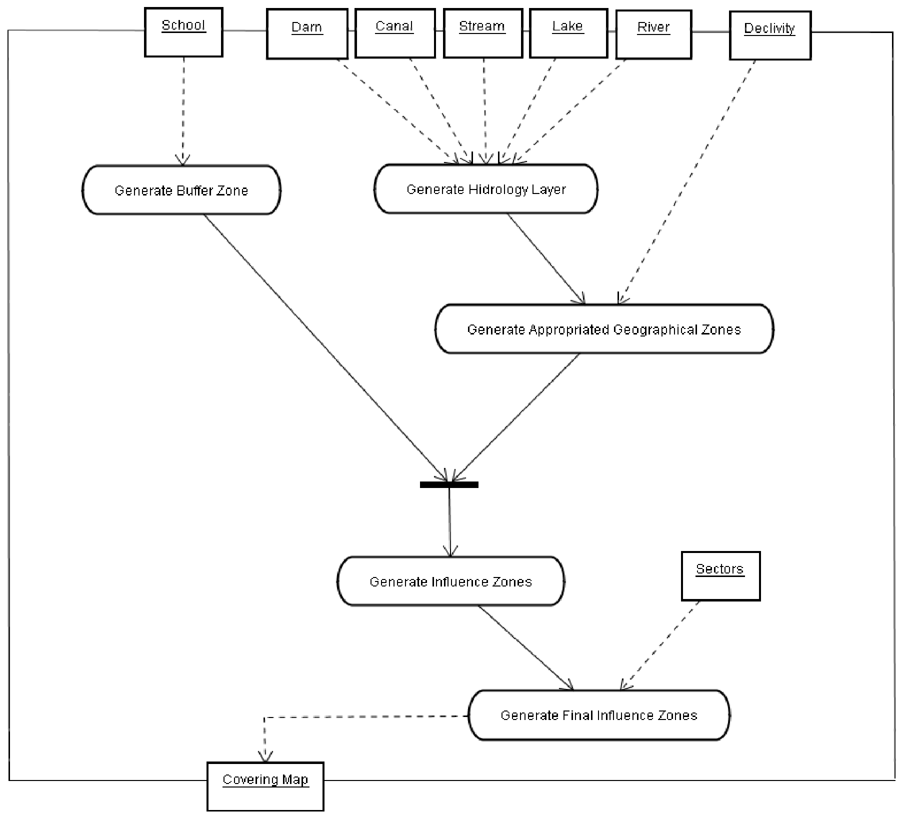

Figure 4. Covering map algorithm's activity diagram.

- **school buffer:** First layer school is generated, creating a buffer zone around each school, given an influence ratio value, , which is one of the rules contained in the set of rules submitted by user .
- **Hydrology layer:** A second is an hydrological layer. A sequence of overlays operations are executed over the different type of hydrological objects (rivers, lakes, streams, etc.).
- **Appropriate locations:** A third layer that is generated consists in perform an overlay with the declivity (relief) layer and the hydrology

layer. A Boolean operation is applied over the resulting layer, marking with value 1 the locations appropriated for a human house and 0 the ones not appropriated. An appropriated location is the one not on an hydrological resource and with the declivity value lower than a certain value (which is also defined by the user in the set of rules).

- **Valid school buffer:** The school buffer and the appropriate locations layers are used to generate the valid school buffers layer, that is, with the buffer modified to cover only the appropriate locations. The operation used is the difference (school buffer – appropriate locations). The non appropriated locations are erased from the school buffer .
- **Real school buffer:** Each school has a maximum capacity of students. In this step the algorithm verifies if the population inside each school's buffer zone is greater than the school's capacity. If the answer is no, the next school is tested. If yes, the buffer zone is decreased until the population inside it is lower than the school's capacity.

The last step is not as simple as it seems to be, because the population is distributed in sectors and each one may have a different amount of people distributed following a growng/distribution model. There is a large number of works whose goal is to propose this kind of modeling (Almeida, 2002). At the present moment, we are focused on studying that related works as well as the interactions between geographical environment aspects and how these factors affect the population distribution in order to obtain a model of urban growth.

4. THE MAP OUTPUT

The map generation of the user input geographic scenario in the GML format can be performed through an XSLT transformation (W3C, 2003) process. Once data are coded in GML it is needed to convert them to a graphic format to enable the visualization as maps and interchange using the web interface. This graphic format has to have a dynamic and interactive support to georeferenced information. The Scalable Vector Graphics (SVG) (W3C, 2004) is suitable for that kind of applications.

The SVG language is supported by most of the browsers simply by the installation of a plug-in and has many advantages comparing to the other image file types used in the internet, such as JPG and GIF. While JPG and GIF stores the image in a raster way, the SVG language represents the images using a vector language. Due to that with SVG is possible to make zoom operations without distortion in the image. Furthermore, SVG is not a proprietary language and the files are smaller than the ones representing the same information with other formats. The SVG format is recommended by the World Wide Web Consortium (W3C) and describes two-dimensional (2D) graphics in XML. A SVG file can be visualized as an image, a text or a set of vectors. (W3C, 2004).

The XSLT stylesheet define how the GML document is transformed in SVG graphic elements. As different stylesheets can be applied over the same document, the GML information coded in it may be visualized differently (Kay, 2001). On the other hand, different XML instances documents can be processed by the same stylesheet if they share a common schema.

To make the geometric representation of a GML file as a map the XSLT model rules were used. Those rules convert each one of the geometric properties to its equivalent in SVG. For instance, the class Point in GML may be represented as a Cicle element in SVG. This transformation is performed by a XSLT processor as illustrated in Figure 5. The XSLT processor's main role is to apply the XSLT stylesheet over the source document and to produce a resulting file. In the case of this paper, the input file is in the GML (Tennakoon, 2001).

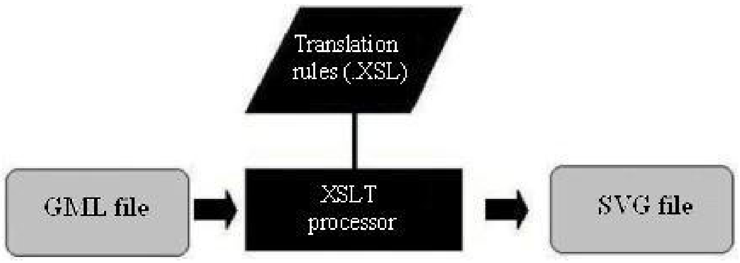

Figure 5. The GML to SVG transformation

A stylesheet is a set of rules (templates) where each rule specifies ho to format certain elements in the file. The templates are apart of the documents, allowing each document to have more than one associated stylesheet. When facing a document against an associated stylesheet each rule matches with an element type of the input document. Xpath expressions are used to refer to specific parts of the document. The original tags are thus substituted by the output tags.

5. CONCLUSIONS

The simulation of the urban growth aids the urban planners to chose the best options for the sustainable development of the city. Our proposal is suitable for that work and furthemore, as it is web based, many people and departments may exchange not only data, but also experiences. Moreover, a planner may search the configuration of a good scenario in order to tune the parameters of his.

At the moment the kernel of the simulator is already in operation and integrated to the web site. The web pages with the parameters to the simulation were implemented using the servlets technology. We already have some test data, and the tools for conversion from shapefiles to GML are implemented, as well as the schema for the data description.

The major next steps are basically in four directions. The first of them is to get a model of distribution of population which allow us to simulate considering geographical aspects. Since we obtain that, it is possible to simulate urban growth satisfactorily. The second is to complete the implementation of the covering map algorithm and integrate it with the simulator kernel. The third remaining work is to convert the results generated by the algorithm into SVG files to be displayed. At least, the rules schema definition as well as the rules XML file validation have to be done.

As future work, we plan to make the simulator more flexible by allowing the user to choose the layers he wants to be considered in the simulation. The is planed to the rules set. We intend to enable the submission of new rules driving the same objects already considered and also new ones.

6. REFERENCES

Allen, J. and Lu, K. S. Modeling and Predicting of Future Urban Growth in the Charleston, South Carolina Area. The South Carolina Sea Grant Consortium. United States. 2000.

Almeida, C; Monteiro, A.M.; Camara, G.; Soares-Filho, B.S.; Cerqueira, G; Pennachin, C; Batty M. Empiricism and Stochastics in Cellular Automaton Modeling of Urban Land Use Dynamics. The Centre for Advanced Spatial Analysis Working Paper Series. 2002b. Available at <http://www.casa.ucl.ac.uk/working_papers/Paper42.pdf>. November 2002.

Bäck, T.; Dörnemann, H.; Hammel, U. and Frankhauser, P. Modeling Urbam Growth by Cellular Automata. In 4th International Conference on Parallel Problem Solving from Nature. Berlin, Germany. 1996

Benenson, I.; Torrens, P. M. Geographic Automata Systems: A New Paradigm for Integrating GIS and Geographic Simulation. In GeoComputation 2003. Southampton, United Kingdom. September, 2003.

Câmara, G. et. al. SPRING and TerraLib: Integrating Spatial Analysis and GIS, Proceedings of the SCISS Specialist Meeting New Tools for Spatial Data Analysis. Santa Barbara, California, USA. May 10-11, 2002.

Geography Markup Language (GML) 3.0. Open GIS Implementation Specification, 2003. Available in http://www.opengis.net. Last access in december 2003.

Hess, G. N, Bastos, A. D., Iochpe, C. A Comparison on open-source GIS Libraries. VI Brazilian Symposium on GeoInformatics (GeoInfo). Campos do Jordão, Brazil. 2004.

Huang, W. and Cho, H. JTrend – a Java-based Cellular Automata Simulation Environment. Techical Report. Iowa State University, United States. October, 2002.

Johnston, R. A. and Shabazian, D. R. Uplan: A Versatile Urban Growth Model for Transportation Planning. TRB Paper, 03-2542. United States. October, 2002.

Kay, M. XSLT 2.0 Programmer's Reference 2nd edition. Wrox, 2001.

Ramsey, P. PostGIS Manual. Available at <http://postgis.refractions.net/docs/>. Last access in July, 2004.

Scalable Vector Graphics (SVG) 1.1. World Wide Web Consortium (W3C) Specification, 2003. Available in http://www.w3.org/TR/SVG/. Last access in february 2004.

Schulz, R. Geotools2 Overview for Users. Available at < http://www.geotools.org>. Last access in July, 2004.

Teerarojanarat, S., Fairbairn, D., and Chunithipaisn, S. Urban Growth Simulation with UrbanSim. Proceedings of the FOSS/GRASS Users Conference. Bangkok, Thailand. September 2004.

Tennakoon, W. T. M. S. B. Visualization of GML data using XSLT. Master dissertation, International Institute for Geo-Information Science and Earth Observation, 2001.

Torrens, P. M. Geosimulation: object-based modeling of urban phenomena. In Editorial of Computers, Environment and Urban Systems. Number 28, 2004.

Waddell, P. and Ulfarsson, G. F. Introduction to Urban Simulation: Design and Development of Operational Models. In Introduction to Urban Simulation. United States, 2003.

XSL Transformations (XSLT) 1.0. World Wide Web Consortium (W3C) Recommendation, 1999. Available in http://www.w3.org/TR/xslt. Last access in december 2003.

White, R. and Engelen, G. Cellular automata and fractal urban form. Environment and Planning A, 25, 1175-1199. 1993.