# TerraML: a Language to Support Spatial Dynamic Modeling

BIANCA PEDROSA [1]
GILBERTO CÂMARA [1]
FREDERICO FONSECA [2]
TIAGO CARNEIRO [1]
RICARDO CARTAXO MODESTO DE SOUZA [1]

[1]INPE—National Institute for Space Research, Caixa Postal 515, 12201 São José dos Campos, SP, Brazil
{bianca,tiago,gilberto,cartaxo}@dpi.inpe.br
[2]School of Information Sciences and Technology, Pennsylvania State University, 1602, State College, PA, USA
ffonseca@ist.psu.edu

**Abstract.** Spatial Dynamic Modeling simulates spatio-temporal processes in which a location on the Earth's surface changes due to some external driving force. This paper introduces TerraML, a dynamic modeling language to be used in environmental applications. TerraML supports both discrete and continuous change processes and generalized neighborhood to accommodate non-local actions.

## 1 Introduction

Cellular models have been used in the last two decades for simulation of urban and environmental models, mostly in connection with cellular automata (CA) (White and Engelen 1997). CA have become popular largely because they are tractable, can replicate traditional processes of change through diffusion, and also contain enough complexity to simulate surprising and novel changes as reflected in emergent phenomena (Couclelis 1997). Early proposals for the use of CA in spatial modeling tended to stress their pedagogic use in demonstrating how global patterns emerge from local actions. In the case of most actual applications to geographic systems, the strict adherence to the basic CA model is inevitably relaxed, and the resulting models are inhomogeneous, where the inhomogeneities may represent such factors as suitability, accessibility, or legal restrictions on land use (White and Engelen 1997). Therefore, in most current applications, the models that have emerged are best called cell-space models rather than CA (Batty 2000).

Currently, most CA-based spatial models are linked to a GIS via loose coupling mechanisms. In this case the GIS is used for data conversion and graphic display and the spatial models are run in an environment external to the GIS. Examples include the models used by Clarke (Clarke and Gaydos 1998) for simulation of US metropolitan growth, the CLUE land-use model (Veldkamp and Fresco 1996) and the DINAMICA landscape model (Soares-Filho, Cerqueira, and Pennachin 2002). This structure allows the use of existing programs but requires substantial work in data conversion and causes problems of redundancy and consistency due to the creation of multiple versions of the same data. Modeling tools also lack sufficiently flexible GIS-like spatial analytical capabilities; as a result, their ability to convey spatial relations is limited. Therefore, the need for a full integration between GIS and dynamic models remains strong. In a tight level of integration, there would be no strict separation between the model and the GIS, and a dynamic model becomes just one of the applications that could be developed using the generic functionality of a GIS toolbox (Wesseling et al. 1996). A strongly-integrated GIS and dynamical model architecture would allow non-specialists, already familiar with GIS interfaces, to experiment with models, reducing the overhead for data conversion and abstracting part of the complexities in model formulation. Furthermore, modeling and GIS could both be made more robust through their connection and co-evolution (Parks 1993).

However, given the limitations of the current generation of commercial GIS systems, substantial investment in the development of tools and functionality is required for full integration of cell-spaces and dynamical modeling into a GIS architecture. This situation is part of a more general problem, in that the GIScience community currently lacks a comprehensive set of open-source tools for development of new ideas and rapid prototyping. To face this challenge, we created an architecture for spatial dynamical modeling using cell-spaces, which has been implemented as software components as part of an open source GIS library.

This paper introduces TerraML, a dynamic modeling language to be used in environmental applications. TerraML supports both discrete and continuous change processes, supports different data formats and is fully integrated with general-use databases. The remainder of this paper is organized as follows. In section 2 the

theoretical foundations of TerraML are explained. In section 3 the TerraML architecture is introduced. In section 4 the main components of TerraML are described. In section 5 an example in TerraML is provided. In section 6 we discuss implementation aspects of TerraML. Section 7 presents conclusions and future work.

## 2 Theoretical Foundations for TerraML

### 2.1 Hybrid Automata

One of the more important challenges in the development of languages to support dynamical spatial modeling in cell-spaces is the need to represent dynamical processes with both discrete and continuous components. For that purpose, the traditional paradigm of discrete cellular automata is no longer sufficient. Therefore, TerraML is based on the theory of hybrid automata (Henzinger 1996). A hybrid automaton is a dynamical system whose state has both a discrete component, which is updated in a sequence of steps, and a continuous component, which evolves over time. Hybrid automata, which combine discrete transition graphs with continuous dynamical systems, can be viewed as infinite-state transition system. A hybrid automaton consists of the following components:

- ***Variables***. A finite set X = { $x_1$,..$x_n$} of real-numbered variables.
- **Control graph**. A finite directed multigraph ***(V,E)***. The vertices in V are called *control modes*. The edges in E are called *control switches*.
- **Initial and flow conditions**. Initial conditions express the starting condition of the automaton. Flow conditions express predicates that are executed in each control mode.
- **Jump conditions**. Jump conditions are used to assign discrete changes between vertices of the control graph. Each jump condition is assigned to a directed edge.

To explain the hybrid automata concept, we can use as an example a water balancing simulation in the hydrology domain. In such a system, rainfall time series are used to fill cells with rain water until their infiltration capabilities be reached, then a runoff flux occur. In Figure 1, a control graph illustrates the mechanism by which the water balance automaton evolves. The nodes of the graph contain flow conditions, which change the variable values.

The conditions labeling the edges are known as jump conditions. Flow conditions are executed until a jump condition is met. The automaton has an initial condition (*soilwater*=0). When a time series is informed, the *soilwater* value is calculated. Then, the value is checked to see if it has reached the cell infiltration capability. If the *soilwater* value is greater than its infiltration capability, the excess is calculated and transported to another cell. This trajectory is processed recursively for all simulation steps. It is important to note that in a hybrid automata, control modes, which are the node names in the graph (dry, wet), are introduced in order to accomplish the cellular automata discrete nature.

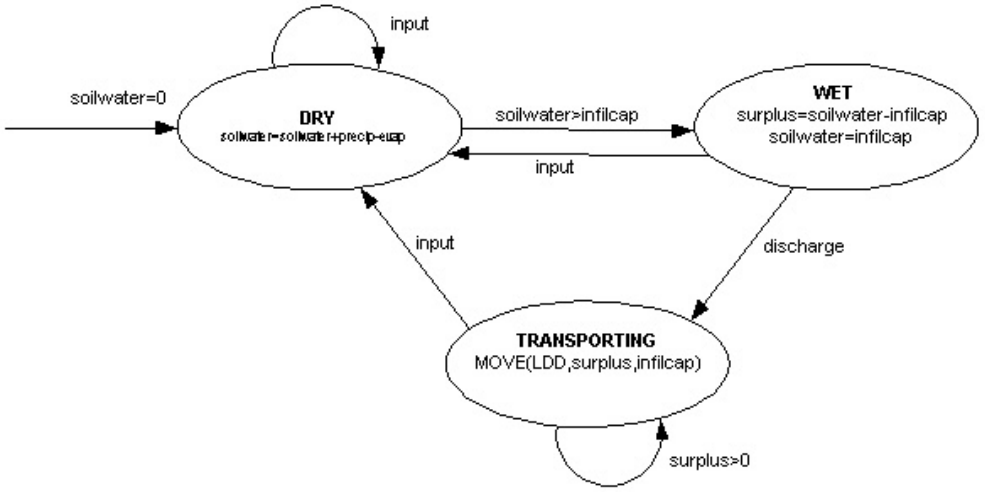


**Figure 1** A water balance hybrid automaton

### 2.2 Non-Proximal Space Definitions

The definition of geographical space in a traditional cellular automaton is based on the assumption that the cellular space is isotropic, and that the neighborhood of interest is entirely local. However, the traditional neighborhoods used in CA such as the Moore 8-neighbor definition have limited usefulness when applied to real-world problems, since the real-world is effectively inhomogeneous. In many situations, action at a distance plays a significant rôle in shaping the processes that define transitions in the cell-space.

In fact, one of the more relevant criticisms to the use of GIS and CA techniques for modeling geographical reality is its over-reliance on proximity conditions. Geographers such as David Harvey (Harvey 1989) consider that the most important impact on human experience is the compression of space and time. Harvey considers that, due to space-time compression, flows of resources, information, organizational interaction and people are essential components of a new definition of space. Other researchers follow the same perspective. Milton Santos (Santos 1996) and Manuel Castells (Castells 2000) talk about "spaces of fixed locations and spaces of fluxes". The concept of "spaces of fixed locations" represents spatial arrangements based on contiguous locations, and the concept of "spaces of fluxes" indicates spatial arrangements based on networks.

To take one example of a inhomogenous space, consider the process of land use change in the Brazilian Amazonia. This process is conditioned by the urban occupation on the region, which has increased significantly in the last two decades. Any model which would aim to project patterns of land use change in Amazonia (as TerraML aims) has to consider that transportation networks (rivers and roads) play a decisive rôle in governing human settlement patterns. As an illustration, Figure 2 shows the urban settlements in Amazonia, shown as white areas, and the road network, as red lines. A realistic model for land use changes in the region has to take into account that the roads establish preferential directions for human occupation and land use changes, which would be impossible to be captured in an isotropic neighborhood definition for a CA-based model. As shown in Figure 2, the neighborhood definitions in any CA that aims at modeling an area such as Amazonia need to be based on a flexible definitions of proximity, that would capture action-at-a-distance. Aiming at supporting action-at-distance in its models, TerraML is based on a flexible neighborhood definition, allowing the user to define her own proximity matrix, according to the needs of the problem at hand.

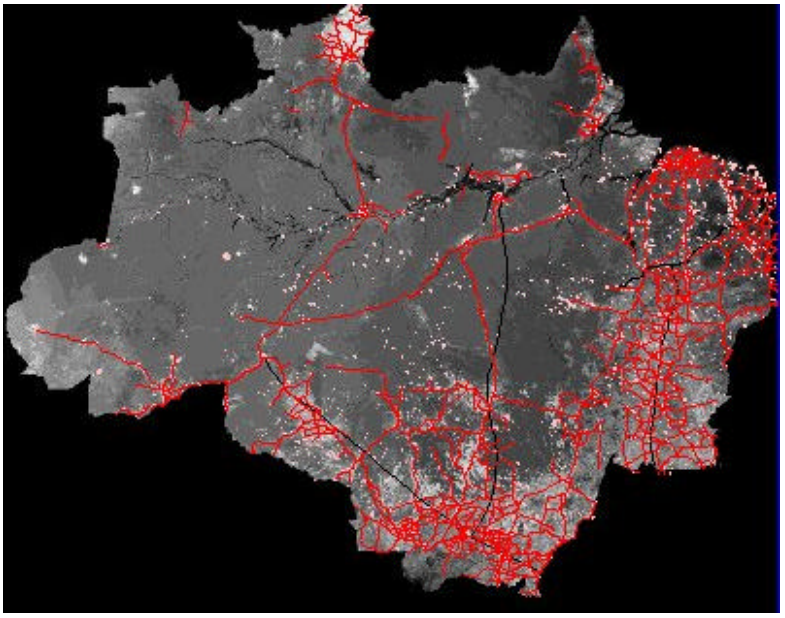

Figure 2 – Spaces of fixed location and spaces of fluxes in Amazonia

### 2.3 Representing Time

Although it has been recognized the importance and need of the temporal aspect in many processes in *GIScience*, the representation of time has not gone beyond a limited prototype stage (Parent, Spaccapietra, and Zimányi 1999; Zipf and Krüger 2001). The reasons for that come (1) from the static cartographic paradigm over which GIS had been constructed, (2) an emphasis on the short-term and implementation-oriented solutions, and (3) the lack of a theory of space-time representation (Peuquet 2001). Most implementations of temporal aspects in GIS have been limited to extending spatial systems to incorporate fragile concepts of time, ignoring (1) the semantic of the space-temporal processes and (2) the underlying aspects of change (Hornsby and Egenhofer 1997).

From the database perspective, there is a broad theory, which has started with the snapshot approach and continued with concepts such as time-stamping, transaction, and valid-time dimensions (Elmasri and Navathe 2000). After that time scales were introduced with the notion of chronons. A chronon is the minimal temporal granularity for a particular application. Most temporal database systems consider only the linear flux of time, although, in theory, there is the notion of cyclic and branching time flows (Worboys 1995).

The issue of representing time in dynamic models goes beyond a matter of extending GIS to incorporate temporal database concepts. At the temporal dimension, as well as in the model and space dimension, the dichotomy between continuous and discrete is a challenging issue. Events such as storms and volcanic eruptions are discrete in both spatial and temporal domains, while temperature and precipitation are spatio-temporal continuous processes (Peuquet 2001). Another strong concept in temporal systems refers to the updating dynamics, which can be synchronous or asynchronous. In a synchronous temporal system all elements is updated simultaneously (Sipper 1999).

Control structures are the most critical support required in a computational environment for dynamic modeling. Iterative control structures work over the entire set of cells in a direct fashion applying a set of operations to one or more attributes of the cell-space. Our purpose is not just to replicate the existing time structures for general-use applications, but to go a step forward in building an integrated spatio-temporal framework, which incorporates processes and focuses on the underlying components of change at the conceptual and implementation levels.

## 3. TerraML Architecture

In the architecture of TerraML, a cell-space is defined as a generalized raster data structure, where each cell holds more than one attribute value. Cell-spaces are a convenient way of managing geographic data in the new generation of spatially-enabled database management systems (DBMS); if required, cells can be handled as individual geographic objects, and operations designed for objects (such as 9-intersection predicates) can be applied to them. The attributes can be presented to the user in the same way as vector geographic objects, and familiar visualization operations can be applied to these data sets.

In terms of implementation, the cell space structure can be divided into two parts called (1) *basic structure* and (2) *extended structure* (Figure 1). The basic structure is static and defined a priori (compile time) representing the set of attributes, which every cell has independently of the model. The extended structure is dynamic, i.e., defined during the simulation process (run time) to accommodate the data provided by a TerraML document.

The basic structure is essentially spatial. Each cell has its spatial reference (cartographic reference), its address in the cellular space (indices), and other attributes such as the cell state and latency.

The extended structure contains attributes which varies from simulation to simulation. For that reason, they are created and attached to the cell structure via a dynamic allocation memory mechanism. These attributes refer to the environmental and socio-economic characteristics of the cell and can be temporal or not. Temporal attributes are the ones that have multiple occurrences in the cell such as the different land uses along the time. They are implemented with a temporal database support for handling their multiple versions.

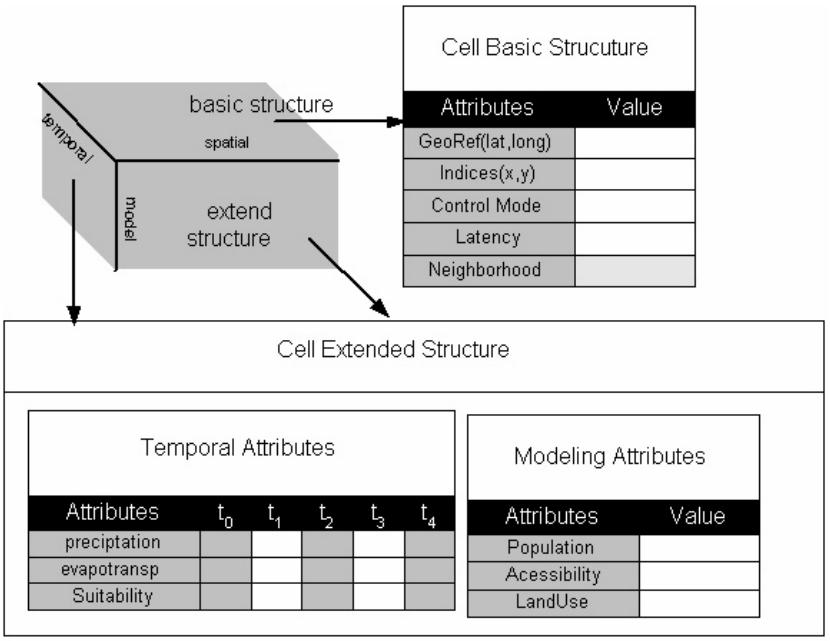


**Figure 3** The Cell-Space Data Structure in TerraML

## 4. The TerraML Structure

A program written in TerraML has a main section called Cellular Processor, which is divided in 5 subsections: *input*, *output*, *transition*, *constraint*, and *simulation*. Figure 4 shows the TerraML Simplified Structure, which describes (1) the elements present in a TerraML document, (2) the order in which they appear, and (3) the content and attributes of each element.

The *input section* is where the data to be retrieved are declared. In TerraML, raster-based maps and images, time series, and non-spatial scalar data at global and cellular scales are suported. It is necessary to inform the values of file names, attributes, and variables in order to make the retrieval and binding processes work.

In the *output section* the data to be saved are declared. These data are generated by the simulation program and added to the cell space as attributes.

*Transition* is the section where the rules upon which the cell states evolve are specifies. In TerraML, discrete and continuous transitions are supported.

In the *constraint section*, restrictions to limit or force a transition are specified.

The *simulation* section is the place to specify the actions to be processed during the execution of the model. First, some cellular space parameters, such as neighborhood, initialization attribute and result name, are configured. After, the actions to be applied over cell attributes, for a determined number of times, are specified. These actions include commands such as updating, calculating, and setting cell attribute values.

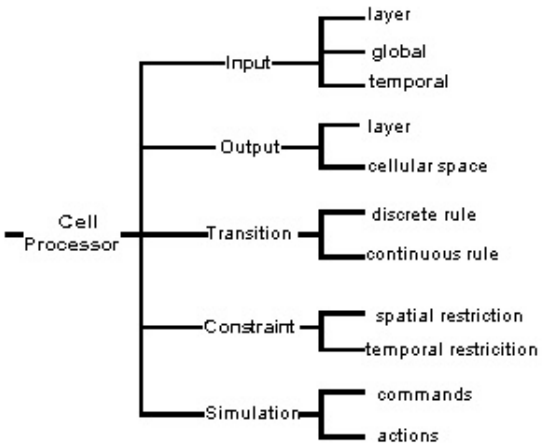


**Figure 4** the TerraML Simplified Structure

**5. A TerraML example**

Now we present an overview of the TerraML syntax by using a simplified example of a deforestation process. In this example, Figure 5, we represent the section names in bold. The **cellprocessor section** is the main section, and it has some attributes for documentation purposes, such as the *author* of the simulation and the name of the *model*.

In the **input section**, two images, *use99* and *road99*, are retrieved and assigned to the *landuse* and *accessibility* variables, respectively. In the **output section**, the variable use is declared as temporal. This variable is directly related to the results to be produced by the simulation section. In the **transition section**, three different transitions are specified: a transition from "forest" to "deforested" state occur if a cell is close to roads (accessibility) or if all its neighbors are in the "deforested" state. A transition from "in regeneration" to "regenerated" happens after 10 years. In the **constraint section**, there are two constraints. The first one is a spatio-temporal constraint restricting the deforestation process to 10% of its current area (spatial) over 20 years (temporal). The second constraint imposes a permanent property to "forest reserve", meaning that a cell in that state cannot be changed to another state in any circumstance. In the **simulation section**, the cellular space is initialized with the *landuse* variable and a "moore' neighborhood (4 neighbors) is specified. The model is processed for 20 time steps that are equivalent to a 20 year period. The results are stored in files called *use2000*, use 2001, and so on, to *use2020* according to the value of the attribute name in the **output section** an the values of the attributes *init* and *end* of the *time* control structure.

```
<cellProcessor author="Bianca" date="3/26/2002" case="Amazon Forest" model="LUCC" >
    <input>
        <layer name="use99"       attribute="class"    >    landuse          />
        <layer name="road99"      attribute="distance">     accessibility    />
    </input>
    <output
        <temporal name="use"   attribute="class"   >
    </output>
    <transition>
        <rule  from="forest"          to="deforested"   >    <event> condition="acessibility=51"        />    />
        <rule  from="forest"          to="deforested"   >    <event> neighbor="all"                    />    />
        <rule  from="regeneration" to="regenerated"  >    <event> time="after 10"                   />    />
    </transition>
   <constraint>
        <restriction state="deforested"    spatial="+10%" temporal="20 years"/>
        <restriction state="forest reserve" type="static"                      />
    </constraint>
   <simulation>
        <cellspace  neighborhood="moore" result="use" init="landuse" />
        <timer init="2000"  end="2020"  timeunit="year" />
            <TRANSIT>
        </timer>
    </simulation>
</cellProcessor>
```

**Figure 5** An example in TerraML showing changes in land use cover

**6. Implementation Aspects**

A TerraML program is mapped to the cellular space architecture presented in section 3. The cellular space architecture is implemented as software components to be provided by a GIS library called TerraLib. TerraLib is an open-source general-purpose GIS library under development at the Brazilian National Institute for Space Research (INPE). TerraLib provides, in its kernel (Figure 6), functionality for handling the different types of geographic data and facilities for data conversion, graphical output, and spatial database management (Câmara et al. 2000). Algorithms that use the kernel structures, including spatial analysis, query and simulation languages, and data conversion procedures are also provided.

TerraLib has been implemented in C++, based on the object-oriented paradigm. This way the cellular space architecture

is implemented in a hierarchy of classes, where each class represents the cellular space main components.

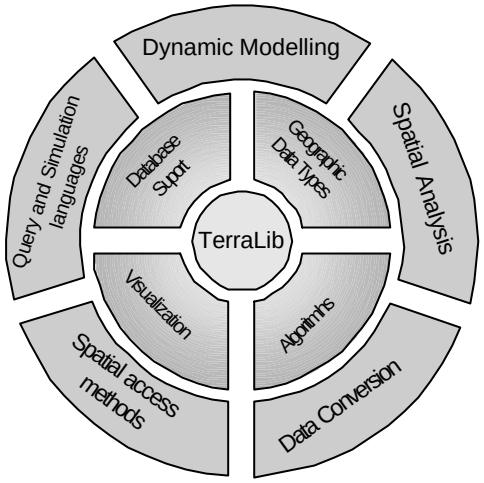


**Figure 6** The TerraLib Structure

In Figure 7, the main class is the cellular space, which is composed by a cellular grid, a set of transitions and constraints. Each cell has a set of attributes and a neighborhood. A cell attribute is an abstract class, which means that it can hold any data type. A cell neighborhood refers to the set of cells which influence the cell state. The set of cells in a neighborhood can has any configuration, be contiguous or not, and has any number of cells.

**7. Conclusions**

In this paper we introduced TerraML, a language to support spatial dynamic modeling in environmental processes. TerraML represents an improvement over other dynamic modeling languages such as PCRaster (Wesseling et al. 1996), MapScript (Pullar 2001), CALANG (Stocks and Wise 2000) and CELLAR (Folino and Spezzano 2000). First, TerraML supports both discrete and continuous change processes. Second, TerraML supports non local actions due to its non-proximal neighborhoods. Third, TerraML supports different data formats and is fully integrated with general-use databases. Fourth,

The development of TerraML and the open source GIS software library is part of an ongoing work. Future efforts will focus on a more complete integration of space and time into the language, and on introducing restrictions to transitions by means of socio-economic variables.

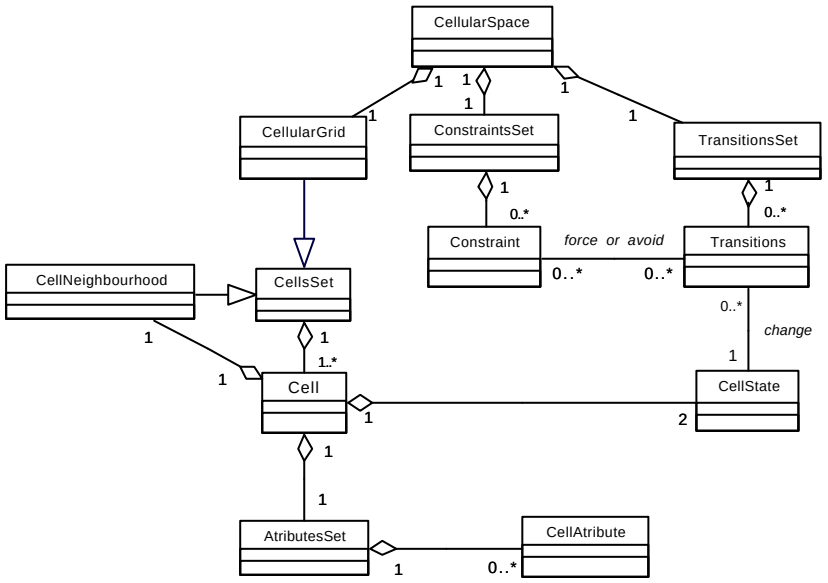


**Figura 7** The cellular space class hierarchy

**References**


Batty, M. 2000. GeoComputation Using Cellular Automata. In *GeoComputation*, edited by S. Openshaw and R. J. Abrahart: Taylor&Francis.

Câmara, G., R.C.M. Souza, B. M. Pedrosa, L. Vinhas, A.M.V. Monteiro, J.A. Paiva, M.T. Carvalho, and M. Gatass. 2000. TerraLib: Technology in Support of GIS Inovation. Paper read at GeoInfo 2000 - II Workshop Brasileiro de Geoinformação, June, 2000, at São Paulo(ed), 2000.

Castells, Manuel. 2000. *The Information Age: Economy, Society, and Culture*. Oxford: Blackwell.

Clarke, K. C. , and L. Gaydos. 1998. Loose-Coupling a Cellular Automaton and GIS: Long Term urban growth prediction for San Francisco and Washington/Baltimore. *International Journal of Geographical Information Science* 12 (7):699-714.

Couclelis, Helen. 1997. From Cellular Automata to Urban Models: New Principles for Model Development and Implementation. *Environment and Planning B: Planning and Design* 24:165-174.

Elmasri, R, and S. B. Navathe. 2000. *Fundamentals of Database Systems*. Edited by A. Wesley. 3rd ed. USA.

Folino, G., and G. Spezzano. 2000. CELLAR: A High Level Cellular Programming Language with Regions. Paper read at Proceedings of 8th Euromicro Workshop on Parallel and Distributed Processing, at Rhodes, Greece.IEEE (ed), 2000.

Harvey, D. 1989. *The Condition of Postmodernity*. London: Basil Blackwell.

Henzinger, T. A. 1996. The Theory of Hybrid Automata. Paper read at Proceedings of the 11th Symposium on Logic in Computer Science (LICS'96).IEEE (ed), 1996.

Hornsby, Kathleen, and Max J. Egenhofer. 1997. Qualitative Representation of Change. Paper read at Spatial Information Theory: A Theoretical Basis for GIS, Proceedings of the International Conference COSIT '97, at Berlim.A. F. S. Hirtle (ed), Lecture Notes in Computer Science (Springer-Verlag), 1997.

Parent, C., S. Spaccapietra, and E. Zimányi. 1999. Spatio-Temporal Conceptual Models: Data Strucutres + Space + Time. Paper read at ACM GIS'99, at Kansas City, MO USA.ACM (ed), 1999.

Parks, B. O. 1993. The Need for Integration. In *Environmental Modelling with GIS*, edited by M. J. Goodchild, B. O. Parks and L. T. Steyaert. Oxford: Oxford University Press.

Peuquet, D. 2001. Making Space for Time: Issues in Space-Time Data Representation. *GeoInformatica* 5 (1):11-32.

Pullar, D. 2001. MapScript: A Map Algebra Programming Language Incorporating Neighborhood Analysis. *GeoInformatica* 5 (2):145-163.

Santos, Milton. 1996. *A Natureza do Espaço (The Nature of Space)*. São Paulo: Hucitec.

Sipper, M. 1999. The emergence of Cellular Computing. *IEEE Computer* 32 (7):18-26.

Soares -Filho, B. S., G. C. Cerqueira, and C. L. Pennachin. 2002. DINAMICA: A New Model to Simulate and Study Landscape Dynamics. *Ecological Modelling (in press)*.

Stocks, C. E., and S. Wise. 2000. The role of GIS in Environmental Modelling. *Geographical and Environmental Modelling* 4:219-235.

Veldkamp, A., and L.O. Fresco. 1996. CLUE-CR: an integrated multi-scale model to simulatie land use change scenarios in Costa Rica. *Ecological Modelling* 91:231-248.

Wesseling, C.G, D. Karssenberg, W.P.A Van Deursen, and P.A Burrough. 1996. Integrating dynamic environmental models in GIS: the development of a Dynamic Modelling language. *Transactions in GIS* 1:40-48.

White, R., and G. Engelen. 1997. Cellular Automata as the Basis of Integrated Dynamic Regional Modelling. *Environment and Planning B: Planning and Design* 24:165-174.

Worboys, Michael F. 1995. *GIS - A Computing Perspective*. Bristol, PA: Taylor & Francis Inc.

Zipf, Alexander, and Sven Krüger. 2001. TGML - Extending GML by Temporal Constructs - A proposal for a SpatioTemporal Framework in XML. Paper read at Procedings of the ACM GIS 2001. The Ninth ACM International Symposium on Advances in Geographic Information Systems, at Atlanta, USA(ed), 2001.