

TerraML – A Cell-Based Modeling Language for an Open-Source GIS Library

Bianca Pedrosa¹, Frederico Fonseca², Gilberto Camara¹, Ricardo Cartaxo¹

{bianca,gilberto,cartaxo}@dpi.inpe.br

¹ Image Processing Division

Brazilian National Institute for Space Research
São José dos Campos, SP, Brazil

ffonseca@ist.psu.edu

² School of Information Sciences and Technology

The Pennsylvania State University
State College, PA, USA

1. Introduction

TerraLib Modeling Language (TerraML) is a spatial dynamic modeling language to simulate dynamic processes in environmental applications. TerraML provides an interface (front-end) from which the end-user can access software components provided by an open-source GIS library called TerraLib (back-end). We decided to implement dynamic models as software components with the objectives of incorporating new functionalities via a flexible and interoperable way, and achieving a tightly coupled integration between the dynamic model and the geographic data. In TerraML we represent space as a cellular model, and represent time as a linear flow divided into discrete intervals. The dynamic model is described by transitions and constraints, where the transition concept corresponds to similar concepts on finite state machines and active databases theories.

2. TerraML Schema

TerraML is based on XML (eXtensible Markup Language), which is a meta-language, i.e., a language to create other markup¹ languages. [1] Figure 1 shows the TerraML Simplified Schema using the XML graphic notation [2]. A program written in TerraML has a main section called Cellular Processor, which is divided in 5 subsections: input,

¹ In a markup language (tag-set) each element is delimited by starting (<>) and ending (</>) tags.

output, transition, constraint, and simulation. The `input` section is where data to be retrieved are declared. In the `output` section, the cellular space is configured. Transition is the section where the user specifies a set of rules upon which the cell states evolve. The `constraint` section contains restrictions that limit, avoid, or force the occurrence of a transition. Finally, the `simulation` section contains actions to be processed during model execution.

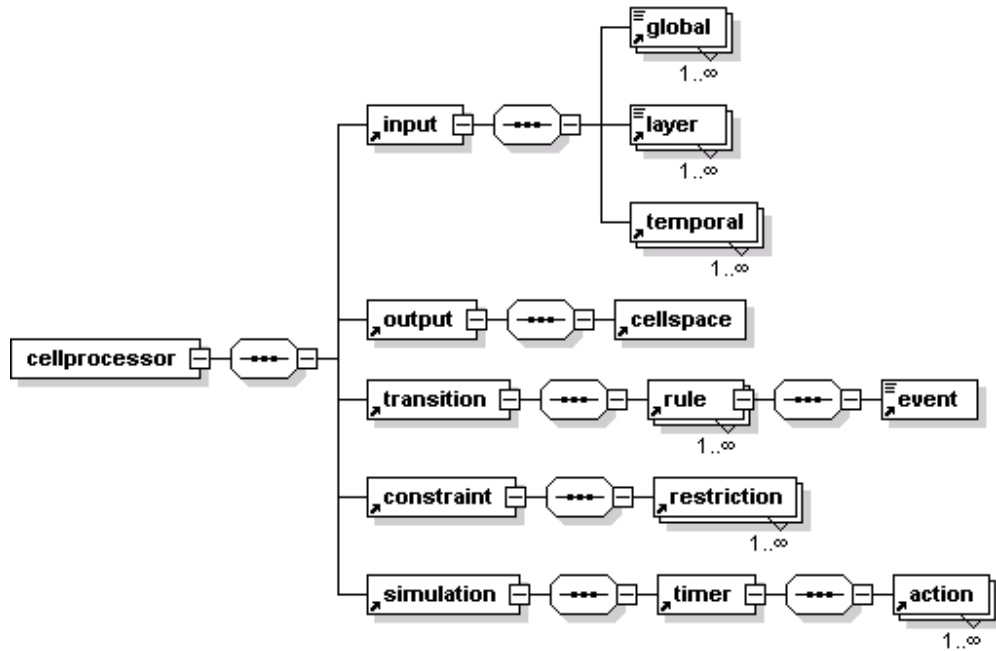


Figure 1 – The TerraML Schema

In Figure 2, we present an overview of the language by using a simplified example of a deforestation process. In this example, two images, `use99` and `road99`, are retrieved and assigned to the `landuse` and `accessibility` variables, respectively. The cellular space is initialized with the `landuse` variable and a von Neumann neighborhood [3] is specified. Three different transitions can happen in this simulation. Transitions 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. Constraints are preconditions to limit, avoid, or force the occurrence of a transition in its spatial, temporal, or both contexts. In this example, constraints restricting the deforestation process to 10% over 20 years and fixing the preserved cells are also specified. The simulation is processed for 20 time-steps that are equivalent to 20 years. The results are stored and displayed on the screen one by one in a sequence giving an animation effect.

```
<cellProcessor author="Bianca" date="3/26/2002" case="Amazon Forest" model="LUCC" >
  <input>
    <layer name="use99" attribute="class">landuse</layer>
    <layer name="road99" attribute="distance">accessibility</layer>
  </input>
  <output>
    <cellspace neighborhood="0,3,6" name="use" init="landuse" />
  </output>
  <transition>
    <rule from="forest" to="deforested">
      <event>condition="accessibility=51"</event>
    </rule>
    <rule from="forest" to="deforested">
      <event>neighbor="all"</event>
    </rule>
    <rule from="in regeneration" to="regenerated">
      <event>time="after 10"</event>
    </rule>
  </transition>
  <constraint>
    <restriction state="deforested" spatial="+10%" temporal="20 years"/>
    <restriction state="forest reserve" type="static"/>
  </constraint>
  <simulation>
    <timer init="2000" end="2020" timeunit="year" />
    <TRANSIT>
    <SAVE>
    <SHOW>
    </timer>
  </simulation>
</cellProcessor>
```

Figure 2 – An example in TerraML showing changes in land use cover

3. TerraML Data Structures

A TerraML program is mapped to a cell space, which is a generalized raster data structure where each cell holds more than one attribute value. Cell-spaces are extensions of cellular automata (CA) [4] to support non local actions [3]. In cell spaces each cell can be handled as an individual geographic object to which traditional visualization operations can then be applied.

In terms of implementation, the cell space structure can be divided in two parts called (1) *basic structure* and (2) *extended structure* (Figure 3). The basic structure is defined *a priori* and contains attributes that are common to all types of simulation models: the georeferenced location for each cell, its location in cellular space, its state and latency. The extended structure is dynamic, i.e., defined during the simulation process (run time) to accommodate the data provided by the user in a TerraML program.

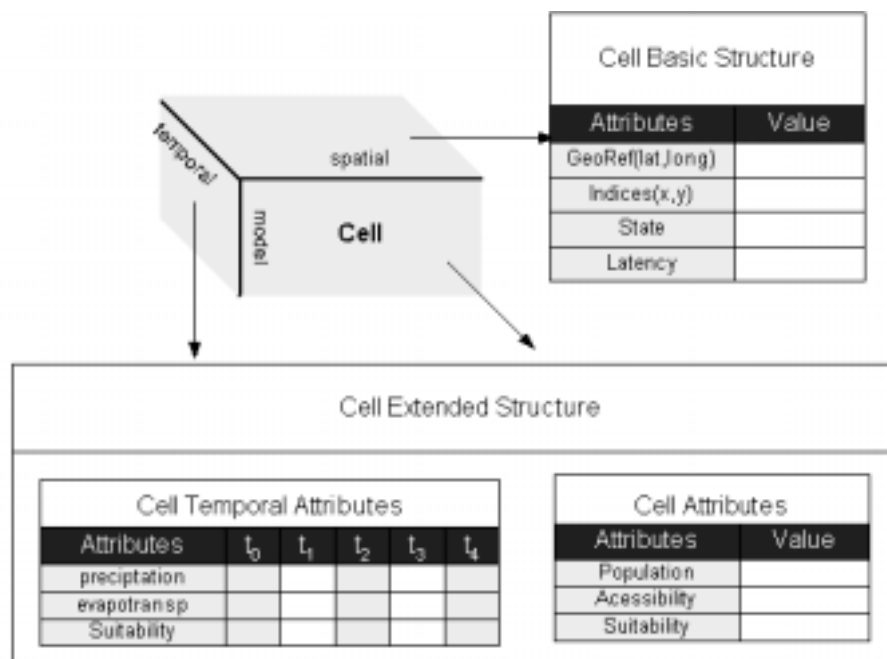


Figure 3 - The Cell Data Structure

The extended structure contains the attributes specified in the input and output sections, which varies from program to program. For that reason, they are created and attached to the cell structure via dynamic memory allocation [5]. 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 in the simulation period. They are implemented with a database temporal support for handling their multiple versions.

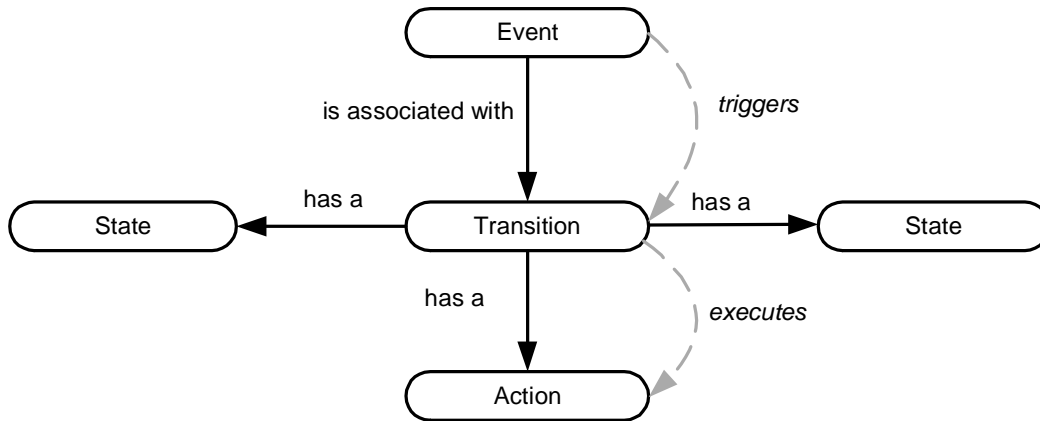


Figure 4 – Structure of a transition rule extended from [6]

Another important data structure present in TerraML is the transition (Figure 4). We decided to model changes using the transition concept found on finite state machines [6] and in active databases [7] theories, which assume that a transition (1) has a source and a target state, (2) is triggered by an event, and (3) can be associated to actions. Events can be expressed by means of relational conditions, neighborhood configurations, or mathematical operations. Actions are performed in order to manipulate cell attributes or to invoke an operation or any other application-dependent action [7]. This structure for transitions can be easily adapted to support continuous behavior, according to the hybrid automata theory. “A hybrid automaton is a formal model for a dynamical system with

discrete and continuous components” [8].

4. TerraLib

TerraLib is an open-source general-purpose GIS application development library under development at the Brazilian National Institute for Space Research (INPE). TerraLib provides, in its kernel (Figure 5), functionality for handling the different types of geographic data and facilities for data conversion, graphical output, and spatial database management [9].

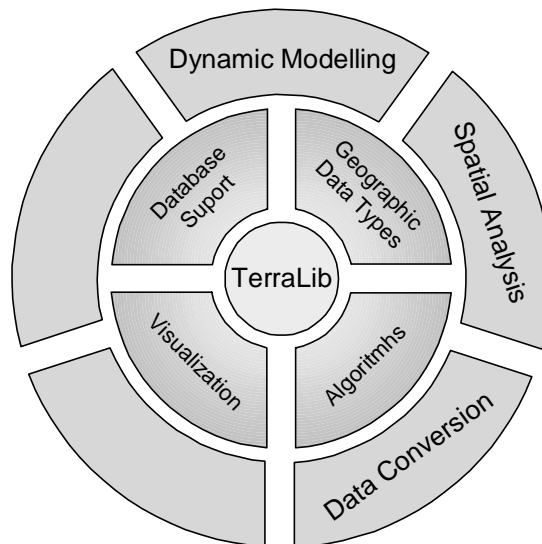


Figure 5 The TerraLib Structure

Algorithms that use the kernel structures, including spatial analysis, query and simulation languages, and data conversion procedures are also provided. In TerraLib, data structures and algorithms are independent following the computational trend of multi-paradigm software development [5, 10, 11].

TerraLib aims to enable the development of a new generation of GIS applications, based on the technological advances on spatial databases [9]. The basic idea behind TerraLib is that the current and expected advances in database technology will enable, in the next

few years, the complete integration of spatial data types in data base management systems (DBMS).

5. Conclusions and Future Work

The advantages of TerraML over other dynamic modeling languages such as PCRaster[12], MapScript[13], CALANG [14] and CELLAR [15] are:

- TerraML supports different data formats and is fully integrated with general-use databases because of its integration with TerraLib.
- TerraML is an XML-based language, which is a global standard capable of generating human-readable and interoperable documents [1].
- TerraML implements an integrated spatio-temporal framework, which incorporates processes and focuses on the underlying components of change at the conceptual and implementation levels.
- TerraML combines emerging and well-consolidate computational technologies, such as finite state machines, hybrid automata, cellular computing [16], components, and event programming in a multi-paradigm software development approach.

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, on modeling continuous behavior, and on proposing mechanisms to manage multiple scales in both spatial and temporal dimensions.

6. References

- [1] S. W. Houlding, "XML - an opportunity for <meaningful> data standards in the geosciences," *Computers and Geosciences*, vol. 27, pp. 839-849, 2001.
- [2] W3C, "World Wide Web Consortium," vol. 2002: www.w3.com, 2002.
- [3] M. Batty, "GeoComputation Using Cellular Automata," in *GeoComputation*, S. Openshaw and R. J. Abrahart, Eds.: Taylor&Francis, 2000, pp. 95-126.
- [4] R. White and G. Engelen, "Cellular Automata as the Basis of Integrated Dynamic Regional Modelling," *Environment and Planning B: Planning and Design*, vol. 24, pp. 165-174, 1997.
- [5] J. Coplien, *Multi-paradigm Design for C++*: Addison-Wesley, 1999.
- [6] J. Van Gorp and J. Bosch, "On the Implementation of Finite State Machines," *Proceedings of 3rd Annual IASTED International Conference Software Engineering and Applications*, pp., Scottsdale, Arizona, USA, 1999.

- [7] S. Cerri and P. Fraternali, *Designing Database Applications with Objects and Rules. The IDEA Methodology*. Harlow, England: Addison Wesley Longman, 1997.
- [8] T. A. Henzinger, "The Theory of Hybrid Automata," Proceedings of Proceedings of the 11th Symposium on Logic in Computer Science (LICS'96), pp. 278-292, 1996.
- [9] G. Câmara, R. C. M. Souza, B. M. Pedrosa, L. Vinhas, A. M. V. Monteiro, J. A. Paiva, M. T. Carvalho, and M. Gatass, "TerraLib: Technology in Support of GIS Inovation," Proceedings of GeoInfo 2000 - II Workshop Brasileiro de Geoinformação, pp. 126-133, São Paulo, 2000.
- [10] M. H. Austern, *Generic Programming and the STL: Using and Extending the C++ Standard Template Library*: Addison-Wesley, 1999.
- [11] Z. Zhang and D. A. Griffith, "Integrating GIS components and spatial statistical analysis in DBMSs," *International Journal of Geographical Information Science*, vol. 14, pp. 543-566, 2002.
- [12] W. P. A. Van Deursen, "Geographical Information Systems and Dynamic Models," in *Faculty of Spatial Sciences*. Rotterdam, The Netherlands: University of Utrecht, 1995, pp. 126.
- [13] D. Pullar, "MapScript: A Map Algebra Programming Language Incorporating Neighborhood Analysis," *GeoInformatica*, vol. 5, pp. 145-163, 2001.
- [14] C. E. Stocks and S. Wise, "The role of GIS in Environmental Modelling," *Geographical and Environmental Modelling*, vol. 4, pp. 219-235, 2000.
- [15] G. Folino and G. Spezzano, "CELLAR: A High Level Cellular Programming Language with Regions," Proceedings of Proceedings of 8th Euromicro Workshop on Parallel and Distributed Processing, pp., Rhodes, Greece, 2000.
- [16] M. Sipper, "The emergence of Cellular Computing," *IEEE Computer*, vol. 32, pp. 18-26, 1999.