TerraLib: An open source GIS library for spatio-temporal databases

GILBERTO CÂMARA[1], RICARDO CARTAXO SOUZA[1], ANTÔNIO MIGUEL MONTEIRO[1],
KARINE REIS FERREIRA[1], LÚBIA VINHAS[1], GILBERTO RIBEIRO DE QUEIROZ[1],
MARCELO TÍLIO DE CARVALHO[2], MARCO ANTONIO CASANOVA[2]
[1]DPI/INPE- Image Processing Division, National Institute for Space Research, Brazil
[2]TECGRAF/PUC-RIO – Computer Graphics Group, Catholic University of Rio de Janeiro, Brazil
{gilberto, cartaxo, miguel, karine, lubia, gribeiro}@dpi.inpe.br
{tilio, casanova}@tecgraf.puc-rio.br

Abstract. The GIScience community currently lacks a comprehensive set of open-source tools for the development of new ideas and rapid prototyping. With this motivation, this work describes **TerraLib**, an open-source GIS software library that extends object-relational DBMS technology to handle spatio-temporal data types. The library includes: (a) support for different DBMS, including Oracle, PostgreSQL and MySQL; (b) an Open GIS compliant spatial feature data model; (c) support for different temporal models (events, moving objects, cell spaces, modifiable objects); (d) facilities for spatial, temporal and attribute queries on the database; (e) dynamic modelling in generalized cell spaces; (e) handling of large image data sets with indexing and efficient visualization; (e) support for generic GIS programming with iterators over spatio-temporal data structures; (f) spatial analysis algorithms, such as space-time clustering tests, regionalization methods, and geographically weighted regression; (i) support for persistent visualization attributes with the concepts of “themes” and “views”. As a research tool, **TerraLib** aims at enabling the collaborative development of GIS prototypes that include recent advances in GIScience. On a practical side, **TerraLib** enables quick development of custom-built applications using spatio-temporal databases.

Keywords. GIS, Small GIS, Software Libraries.

1 Introduction

GIS software development is undergoing substantial change, caused by the availability of database management systems (DBMS) that can handle spatio-temporal data types. This integration enables a transition from the current GIS technology to a new generation of *spatial information appliances*, small systems tailored to specific user needs (Egenhofer 1999). Therefore, an important challenge for the GIS community is finding ways of taking advantage of spatially-enabled DBMS to build innovative applications. A second important challenge is incorporating recent advances from geographical information science into mainstream GIS. A number of important results have been produced in research areas such as spatio-temporal data models (Erwig and Schneider 2002) (Hornsby and Egenhofer 2000), geographical ontologies (Fonseca et al. 2002), spatial statistics and spatial econometrics (Anselin 1999), dynamic modelling and cellular automata (Couclelis 1997), and environmental modelling (Burrough 1998). These results have largely been outside of the reach of the GIS user community, for lack of widely available tools and systems that support them.

One of the possible responses to this challenge would be to establish a co-operative development network based on open source technology. The geographical information science community would have much to benefit from the availability of a general open source GIS library. Similarly to Linux-based solutions, the availability of GIS open source software would allow researchers to share their results. However, such developments do not happen by spontaneous growth, and require a core set of technologies from which further developments would build on. With this motivation, the authors are developing *TerraLib*, an open source GIS software library that extends object-relational DBMS technology to handle spatio-temporal data types. As a research tool, the library aims at enabling the development of GIS prototypes that include recent advances in GIScience. On a practical side, it supports quick development of custom-built applications using spatial databases.

This paper describes the *TerraLib* library, by examining its major components and indicating how the library supports the development of GIS tools that incorporate research results from GIScience. Section 2 presents the requirements and the rationale for the design decisions for TerraLib. Section 3 describes the the spatio-temporal data model and the database interface (archival and retrieval). Section 4 indicates how *TerraLib* handles large image data sets. Section 5 discusses extensibility issues, with an emphasis on the use of generic programming tools for development of spatial analysis

functions. Section 6 presents the dynamical modelling capabilities of TerraLib. Section 7 indicates how the library supports large image data handling. Section 8 describes some geographical applications that have been developed using TerraLib. We conclude the paper by discussing the most important points raised by the research and indicating the future directions of the TerraLib project.

2 Requirements and Design Rationale for TerraLib

We have designed *TerraLib* with a large number of atomic functions and data structures with low granularity. Such an option is more appropriate for open source software products, which aim to cater for a broad range of applications. We have chosen the C++ language because of its flexibility, multi-platform stability, and support for multi-paradigm programming (Coplien 1999). Especially important is the support for *generic programming* with templates available in C++ using the STL library (Austern 1998). Using C++ simplifies linking with object-relational DBMS such as ORACLE, PostgreSQL and mySQL. Since TerraLib aims at helping to bridge the gap between GIScience and GIS applications, a balance between simplicity and expressive power is inevitable. The design of a TerraLib is also an engineering compromise between multiple requirements:

- *Persistence*: The database should be able to store and retrieve not only spatio-temporal data, but also ancillary data such as semantic information, metadata, and visualisation status. The need for *persistence* dictates that the library needs to provide a database schema, with relations that capture the different components of spatio-temporal data.
- *Support for different DBMS*: the library should be able to handle different object-relational databases management systems transparently. There is a basic class for storage and retrieval of spatio-temporal objects in object-relational database management systems, and this class provides a general API which is implemented for each DBMS.
- *Support for temporal applications*: *TerraLib* handles different types of spatio-temporal applications (including events, mobile objects, evolving regions) and supports spatio-temporal queries. Each spatial object has a unique and persistent identifier and a spatial reference system. The descriptive, spatial and temporal components of geographical objects are stored separately on the database and are linked by the object id.

- *Efficiency in large-scale applications:* emerging GIS applications require storage and retrieval of hundreds of thousands of spatial objects, as well as very large satellite imagery. TerraLib supports for spatial indexing of both raster and vector data, including multi-resolution and block segmentation techniques for handling satellite imagery.
- *Extensibility*: The functionality available in a GIS library should be extensible by other programmers, and the introduction of new algorithms and tools should not affect already-existing code. To achieve this goal, *TerraLib* uses iterators over spatial data structures to decouple algorithms from data structures, thus simplifying extensibility.
- *Support for dynamic models*: Models of spatial processes, including patterns of land use and land cover, socioeconomic and demographic characteristics, are becoming increasing important for geographical applications. In *TerraLib*, we have implemented support for dynamic models using cell spaces, which generalize cellular automata by providing multiple attributes and flexible neighborhood definitions.

One of the inevitable problems of having a fixed database schema in TerraLib is a loss of flexibility, since users of the library will have to convert their data to the chosen schema. However, we have kept the database relations as independent as possible, so that application programmer may combine them in multiple ways, allowing the expression of different conceptual models. We consider that the loss of flexibility is balanced by a broad range of functionality.

The vector spatial data structures are designed to be compatible with the Open GIS simple feature model and the dimension-extended 9-intersection operators (Clementini and P. di Felice 1995) are used for spatial queries of vector geometries. In order to better understand the capabilities provided by TerraLib, we have included in Table 1 a comparision with similar products: a commercial library (ArcSDE®) and an open source one (PostGIS). Note that the ArcSDE® is part of a the ArcGIS® family of products, which provides spatial analysis functions by means of separate products.

TABLE 1 – SELECTED PRODUCT COMPARISON

Requirement	ArcSDE	PostGIS	TerraLib
DBMS supported	ORACLE,DB2, SQL Server, Informix	PostgreSQL	ORACLE, PostgreSQL, mySQL
OpenGIS Compatibility	Yes	Yes	Partial
Egenhofer operators	Yes	Yes	Yes
Temporal Support	No	No	Yes
Cell spaces	No	No	Yes
Dynamic modelling	No	No	Yes
Iterators over spatio-temporal data structures	No	No	Yes
Spatial analysis methods	(Yes)	No	Yes
Support for raster data	Yes	No	Yes
Multi-resolution images	Yes	No	Yes

3 The Architecture of TerraLib

A crucial design decision in any open source project is its software architecture. All successful open source products such as Linux, PostgreSQL and Apache have a kernel whose maintenance is the responsibility of a very small team. The kernel has to be stable and to allow easy extension without internal changes. Contributions from the community occur at the external layers of the system. As an example, out of more than 400 developers, the top 15 programmers of the Apache web server contribute 88% of added lines (Mockus, Fielding, and Herbsleb 2002).

However, not all software products have well-defined paradigms as in the case of operating systems and DBMS. The challenge in GI technology is to combine object-relational DBMSes that handle spatial data structures with modules that perform functions such as spatial analysis and dynamic modeling. A typical GI application consists of three steps: (a) querying the spatial database; (b) manipulating the query results to generate new objects; (c) visualizing the resulting objects. These three

capabilities (query, manipulation and visualization) should all be part of a generic GI software library. To respond to these requirements, TerraLib has four main components, as shown in Figure 1:

- **kernel**: the core of TerraLib is composed of: (a) spatio-temporal data structures; (b) support for cartographic projections; (c) topological and directional spatial operators; (d) an API for storage and retrieval of spatio-temporal objects in object-relational DBMS; (e) classes for controlling visualization of spatial data. Kernel maintenance and upgrade is the responsibility of the project core team (INPE and TECGRAF), as typical for other free software projects.
- **drivers**: modules that implement the kernel's generic database API to access DBMS products such as mySQL and PostgreSQL, and external files in both open and proprietary formats. Basic maintenance and upgrade is the responsibility of the project core team, but there is a large scope for external contribution.
- **functions**: algorithms that use the kernel structures. Typical functions include spatial analysis and query and simulation languages. The functions are designed to allow external contribution.
- **interfaces**: consist of different interfaces to the TerraLib components, to allow software development in different environments (C++, Java, COM) and also the support for Open GIS services such as WMS (Web Map Services) protocol.

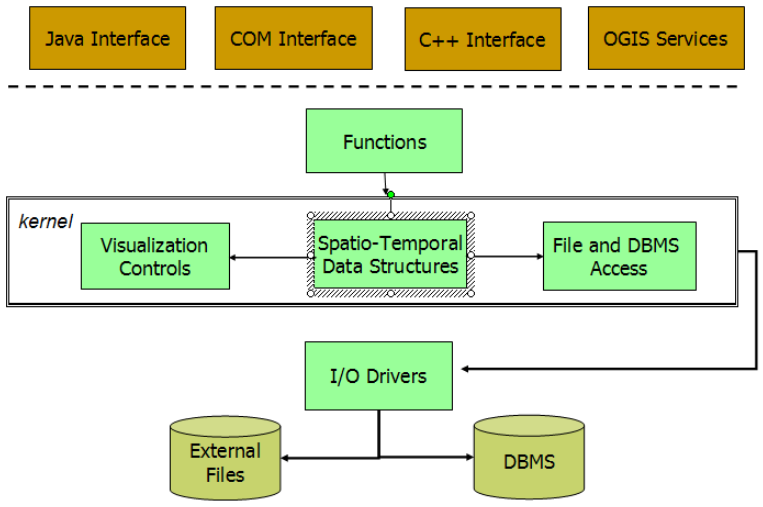


Figure 1 – TerraLib Software Architecture

3.1 The TerraLib Kernel

Some of the key issues involved in the design of the TerraLib *kernel* include:

- *Support for spatio-temporal applications*: In order to handle different types of spatio-temporal applications, TerraLib includes different types of spatio-temporal data types (*events, mobile objects, and evolving regions*) and supports spatio-temporal queries. Each spatial object has a unique and persistent identifier and a spatial location. The descriptive, spatial and temporal components of geographical objects are stored separately on the database and are linked by the object's id.
- *New types of data structures:* Models of spatial processes, including patterns of land use and land cover, socioeconomic and demographic characteristics, are becoming increasing important for geographical applications. To support these models, TerraLib includes *cell spaces* in a GIS database environment. This allows is an important improvement over current GIS technology, where most dynamical models based on cell spaces have a very loose coupling with a GIS. In *TerraLib*, we have implemented support for dynamic models using cell spaces, which generalize cellular automata by providing multiple attributes and flexible neighborhood definitions (Pedrosa et al. 2002)..
- *Efficiency in large-scale applications:* emerging GIS applications require storage and retrieval of hundreds of thousands of spatial objects, as well as very large satellite imagery. TerraLib supports spatial indexing of both raster and vector data, including multi-resolution and block segmentation techniques for handling satellite imagery (Vinhas, Souza, and Câmara 2003).

3.2 The TerraLib Drivers

The interface between the *kernel* and the various DBMS and file formats is done by the modules contained in the *drivers* component of TerraLib. When designing this component, we considered that the library should be able to handle different object-relational databases management systems transparently. This has required careful design of a DBMS interface that handles specific features of each DBMS and allows a single software interface for different systems such as PostgreSQL and mySQL (Ferreira et al. 2002).

3.3 The TerraLib Functions

The interface between the *kernel* and the *function* components has been designed to allow maximum extensibility. TerraLib should be extensible by other programmers, and the introduction of new algorithms and tools should not affect already-existing code. To achieve extensibility, we have adopted the principles of *generic programming*. *TerraLib* uses iterators over spatial data structures to decouple algorithms from data structures, thus simplifying extensibility. For example, for computing a histogram it is not essential to know if the elements are organized as a set of points, a set of polygons, a grid or an image. All that is needed is the ability to look into a list of values, and to obtain, for each element of the list, its values and the indexes of the elements of the list that satisfy a certain property (for example, those that are closer in space than a specified distance). In a similar way, a large number of spatial analysis algorithms can be abstracted away from a particular data structure and described only in terms of their properties. (Vinhas et al. 2002).

3.4 Programming Interfaces

The straightforward way to use TerraLib to develop GIS applications is to embed its C++ classes into a new system. However, the TerraLib designers have to take into account the diverse qualifications and preferences of the GIS programming community. Interfaces for Java and COM programming environments have been designed and are currently being implemented. An interface for the Open GIS web services specification is also under development.

4 Spatio-Temporal Data Structures in TerraLib

4.1 Spatial Objects and Layers

TerraLib supports two basic containers for spatial data: *spatio-temporal objects* and *layers*. A *spatio-temporal object* is an atomic feature whose identity is unique and persists over time. A *layer* is a collection of spatio-temporal objects that share the same geographical projection and the same set of attributes over a temporal period. In an object-relational DBMS, a *layer* is represented by a collection of conventional attribute tables, spatial representation tables and temporal history tables. Retrieval of information about a spatial object requires accessing its container class (a *layer*) and then accessing each of its components. Since a layer contains the temporal evolution of

all its objects, it is an *abstract* concept, and not simply the representational equivalent of a "map" or of a "coverage".

4.2 Spatial Data Types and Relationships

The base class for all spatial data structures in TerraLib is the `TeGeometry`[1] class. Each geometry has a unique identifier. The spatial data structures in TerraLib include: (a) `TePoint2D`: a 2D point. (b) `TeLine`: a vector of 2D coordinates. (c) `TeLinearRing`: a closed line whose last point is the same as its first; (d) `TeLineSet`: a set of lines. (e) `TePolygon`: a polygon is composed of an outer linear ring (its external boundary) and inner rings (its holes); (f) `TePolygonSet`: a set of polygons; (g) `TePointSet`: a set of 2D samples; (h) `TeTIN`: a triangular mesh; (i) `TeArc`: an arc composed by a straight line in the plane; (j) `TeNode`: a 2D point; (k) `TeArcSet`: a set of arcs; (l) `TeNodeSet`: a set of nodes. The raster data structures include: (a) `TeRaster`: a multi-dimensional raster data structure (used for images and grids); (b) `TeCell`: a single cell, used for building cell spaces (discussed in more detail in section 6). The basic geometrical classes are shown in Figure 2.

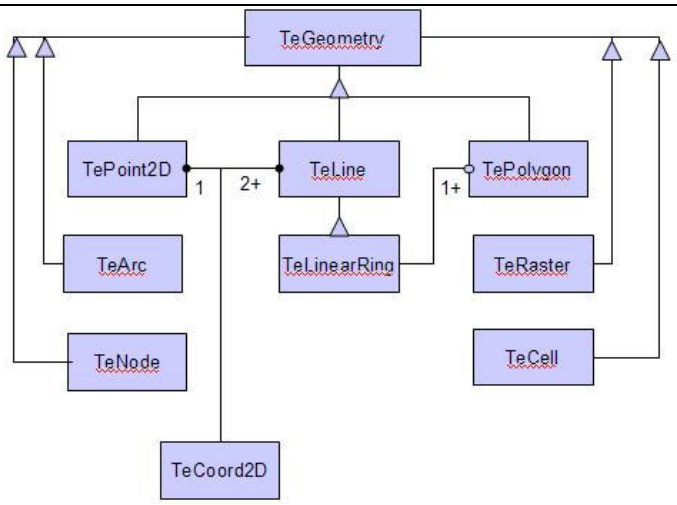


Figure 2– Elementary spatial data structures in TerraLib (partial view)

In order to perform elementary topological operations between spatial data structures, we have implemented the dimension-extended 9-intersection model (Clementini and P. di Felice 1995). These operators have been endorsed by the

[1] Throughout the text, TerraLib classes and functions are indicated with a `Te–` prefix.

OpenGIS consortium (McKee and Buehler 1996), and have been given the mnemonic names: `Equal`, `Disjoint`, `Touch`, `Inside (Within)`, `Overlap`, and `Cross`.

4.3 Spatio-Temporal Model and Queries

There are multiple alternatives for the integration of space and time, which differ on issues such as: (a) the time model considered (point-based or interval-based); (b) whether the database stores event time or transaction time (or both); (c) support for versions. In TerraLib, we have chosen to define temporal queries to operate over time intervals, and to store only event time. The library implements three basic types of spatio-temporal applications:

- *Events*, defined as independent occurrences in space and time, as in the case of crimes. Each spatial object associated with an event will have a unique spatio-temporal identifier.
- *Dynamical objects*, which have a fixed geometry and variable attributes. This is the case in dynamic models based on cell spaces.
- *Moving regions*, whose boundaries and locations can vary in time, as well as their attributes.

We use temporal predicates over time intervals: *before, meets, overlaps, finished, during, starts,* and *equals* (Allen 1983). Recent research indicates that, although it is feasible to define a small number of spatial predicates over objects and temporal predicates over intervals, it is extremely difficult to compute a comparable small size of predicates that can concisely express an elementary set of spatio-temporal relations (Erwig and Schneider 2002). There are too many predicates that can be considered different. Therefore, TerraLib implements a generic type of a spatio-temporal object set, which is used to store the result of a combined spatial, temporal, and attribute query. When using the type `TeSTObjectSet`, the application programmer defines three different restrictions (spatial, temporal, and attribute) and then uses the method `apply` to perform the query. The implementation of the `apply` method takes into account issues such as optimization and facilities available at each specific DBMS supported by TerraLib. The outline of the type definition is shown in Figure 3.

```
type TeSTObjectSet
Functions
setTemporalRel:    layer × interval → void
setSpatialRel:     layer × restriction × spt_rel → void
setAttributeRel:   layer × where_clause → void
apply:              → Boolean
```

Figure 3: Simplified specification of the type "spatio-temporal object set".

5 Spatial Data Storage and Retrieval

In *TerraLib*, there is a basic class (`TeDatabase`) for storage and retrieval of spatio-temporal objects in object-relational database management systems. This class provides a general API that enables the user to: (a) establish the connection with the database server; (b) execute SQL definition and manipulation commands; (c) create the database schema; (d) define indexes; (e) define referential integrity; (f) execute temporal, spatial and attribute queries. For each different DBMS, a sub-class of the `TeDatabase` class encapsulates the internal differences of each system, taking advantage of spatial indexing or in-built optimization, if available. As explained earlier, the *TerraLib* database schema uses the concept of *layers* as containers of spatio-temporal data. Since a layer is an abstract concept, it is represented as a set of conventional attribute relations, spatial representation relations and temporal history relations. To illustrate the database schema, we discuss the relations associated to a set of spatial data structures (points, polygons, cells and rasters), the temporal relations and the overall metadata information.

The *point geometry* relation describes a two dimensional coordinate, or it can be used to represent a three dimensional point, with the object identifier pointing to the record on the attribute table that contains the 3d information. The *polygon geometry* table describes the area representation of all objects associated to a layer. The external and internal parts of the polygons are stored in different records; the internal parts (holes) have information about the external boundary that contains them. The *cell geometry* relation defines a partition of the plane into regular cells, which is very useful for dynamic modelling applications. The *raster geometry* relation stores large images and digital terrain models. We use a combination of *multi-resolution pyramid* and a *tiling* scheme is the most appropriate strategy for handling large image files. The tiling scheme is used as a spatial index, such that when retrieving a section of an image, only

the relevant tiles will be retrieved and decompressed. The multi-resolution pyramid is very useful for visualization of large data sets, to avoid unnecessary data access.

There are four types of temporal predicates supported in TerraLib, each supported by a different relation (or combination of relations):

(a) *Static attributes*, which do not change. In this case, the *static attribute* relation stores the attributes associated to each object.

(b) *Events*, considered as independent occurrences over time. Since events are considered to be independent occurrences, the *event relation* stores each object as a unique entry, together the values of its attributes. When dealing with events, there is a one-to-one relation between an object geometry and its temporal lifetime. This relation is used, for example, in the case of crime and epidemiological studies.

(c) *Evolving objects*, whose geometries are fixed, but whose attributes vary in time. This information is stored on a specific relation, which contains, for a given time interval, the object id and its attribute values valid for the interval. This relation is used, for example, for dynamic models based on cell spaces.

(d) *Moving objects*, whose geometries and attributes evolve over time. The key concept here is the notion of the *state* an object, defined as a unique combination of a time interval, a geometrical representation and a set of descriptive attributes. In this case, we have chosen to use two relations to store the information: the *object state relation* and the *attribute state relation*. At a given time interval, the object's state relation records the object identification (which is unique), its geometry, and an index for its attributes. Considering that the object geometry may vary asynchronously in relation with its attributes, efficiency considerations dictate the need for a second relation (the *attribute state* relation) which stores the attributes of the object at a given time interval.

The overall information for the database schema is stored in four relations: (a) a list of all layers available in the database; (b) a list of all temporal relations associated to each layer; (c) a list of all spatial relations associated to each layer; (d) a list of all spatial reference systems used for the geographical locations. A simplified version of the database schema is shown in Figure 3. The top part shows the overall information relations. The bottom part illustrates some of the spatio-temporal relations. The three linked relations (`te_polygon`, `te_object_state` and `te_attribute state`)

are used in the case of a set of evolving objects. The lowermost three relations cater for the other cases of temporal events.

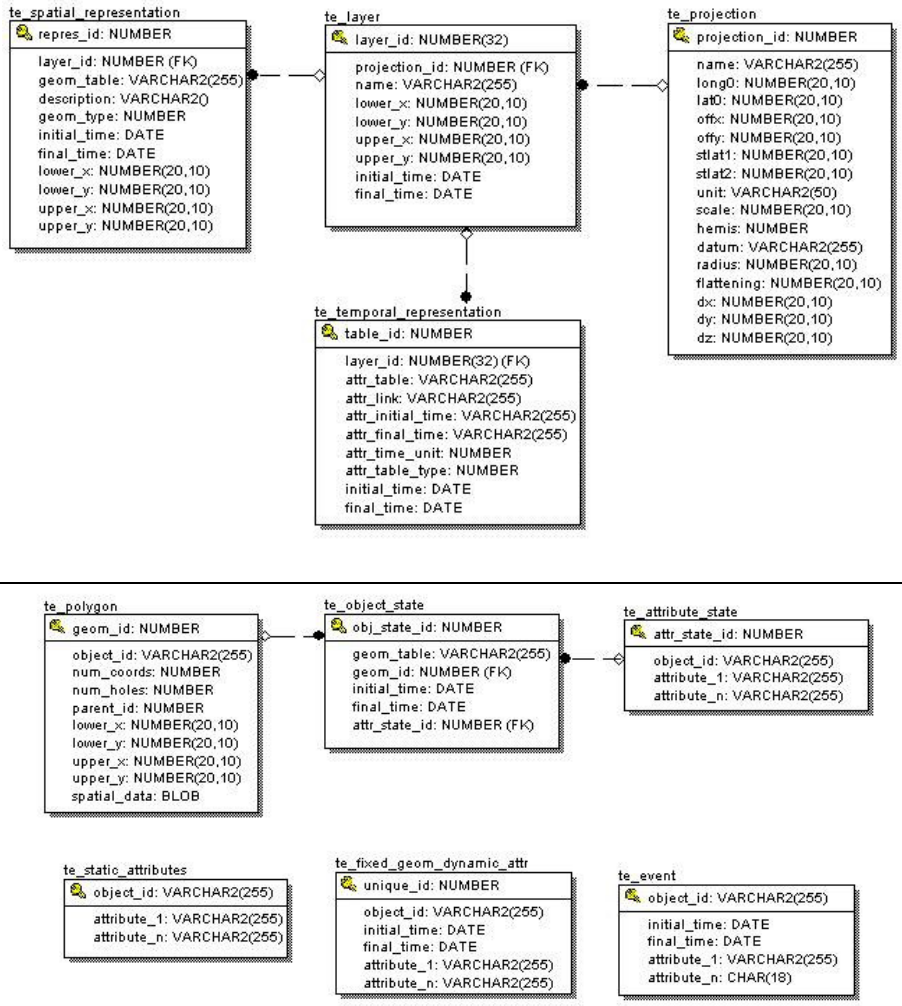


Figure 4 – Simplified ER diagram for the TerraLib database schema

6 Generic Programming and Spatial Analysis in TerraLib

One of the important requirements for an open source product is *extensibility*. The available functionality should be extensible by other programmers, but the introduction of new algorithms should not affect already-existing code. To achieve extensibility, we have adopted the principles of *generic programming*: "*decide which algorithms you want; parametrize them so they work for a variety of suitable types and data structures*" (Stroustrup 1997). Following the example of the STL library, which is part of the C++ standard, we use the concept of *iterators*. Iterators are generalized pointers

that provide glue for connecting algorithms and data structures. For example, for computing spatial autocorrelation indexes it is not essential if the elements are organized as a set of points, a set of polygons, a grid or an image. All that is needed is the ability to look into a list of values, and to obtain, for each element of the list, its values and the indexes of the elements of the list that satisfy a certain property (for example, those that are closer in space than a specified distance). In a similar way, a large number of spatial analysis algorithms can be abstracted away from a particular data structure and described only in terms of their properties.

In *TerraLib*, we have applied the generic programming paradigm to GIS in a four-step process: (a) finding regularities in spatial data handling algorithms; (b) generalising the regularities in this algorithms into requirements for traversal of data structures; (c) providing iterators that support these requirements; (d) designing algorithms that use these iterators. The library provides iterators for all its spatial data structures. For example, the class `TeRaster` provides *iterators* that allow the sequential traversal of an image. The simplest *iterator* traverses the entire image. There are also iterators that traverse the elements of a portion of the image delimited by a polygon.

7 Dynamic Modelling with Cell Spaces in TerraLib

Cell-spaces have been used in the last two decades for simulation of urban and environmental models, mostly in connection with cellular automata (CA). Most cell-space models are linked to a GIS via *loose coupling* mechanisms, where the GIS is used for data conversion and graphic display, and the spatial models are run outside of the GIS database. This structure is best for linking existing programs, but requires substantial work in data conversion and causes problems of redundancy and consistency. Modeling tools also lack sufficiently flexible GIS-like spatial analytical capabilities. As a result, their ability to convey spatial relations is limited. To address these shortcomings, cell-space models need to be strongly linked to the GIS architecture. In a tight level of integration, there would be no strict distinction between model and GIS, and a dynamic model becomes just one of the applications that could be constructed using the generic functionality of a GIS toolbox (Wesseling et al. 1996). Furthermore, modeling and GIS could both be made more robust through their linkage and co-evolution (Parks 1993).

To provide a tight level of integration between models and GIS, we have implemented in *TerraLib* a cell space data structure that, together with support for temporal predicates (as discussed in Section 4) provides the necessary support for the implementation of dynamic models based on cell spaces. Cell spaces are a convenient way of managing geographical data in the new generation of spatially-enabled database management systems (DBMS). Cell spaces can be viewed either as a generalized raster data structure, where each cell holds more than one attribute value, or as a set of non-overlapping polygons. Cell spaces have several advantages over single-attribute raster data structures. Using single-attribute raster structures, describing a complex spatial phenomenon requires information to be stored in different files; this separation results in increased complexity in data management and user interface. In a cell space, such information is kept together in a single structure, with significant benefits in terms of visualization and interface. The attributes can be presented to the user in the same way as vector geographical objects and familiar visualization operations can be applied to these data sets. An example of the use of cell spaces in TerraLib is shown in figue 5.

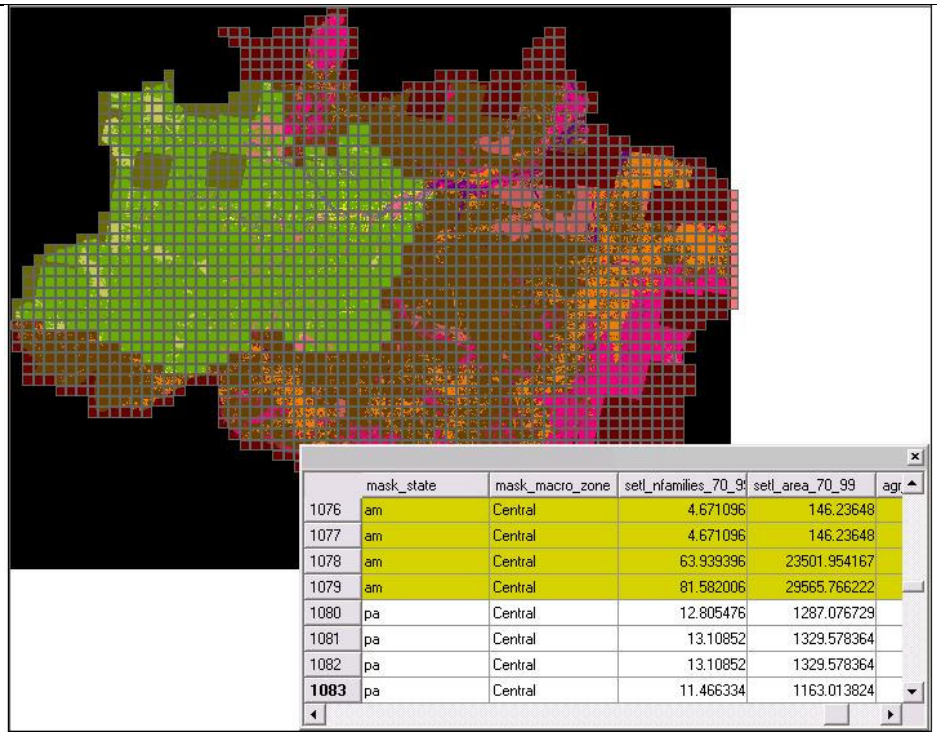
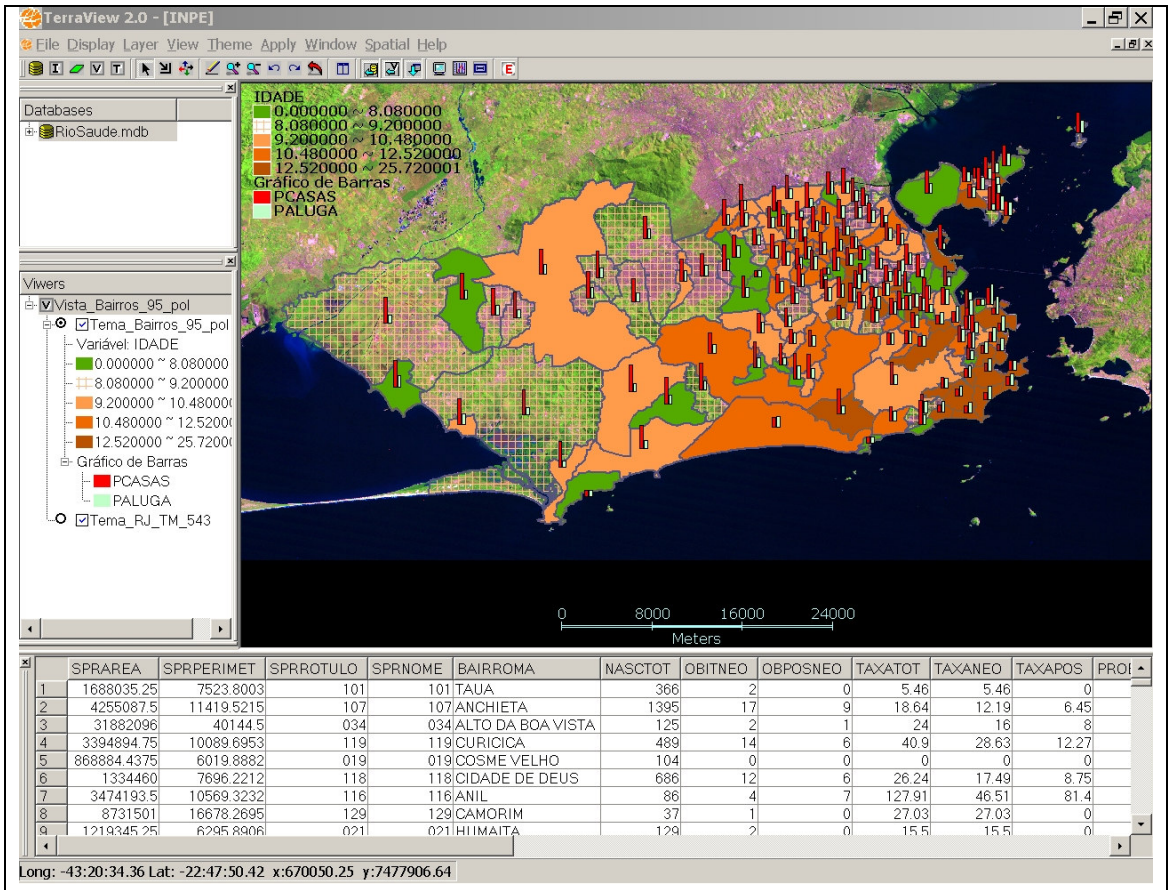


Figure 5 – Use of cell spaces in TerraLib

8 Some Applications of TerraLib

8.1 TerraView

TerraView is a tool for spatial data analysis, which provides the basic functions of data conversion, visualization, exploratory spatial data analysis, spatial statistical modelling and spatial and non-spatial queries. This product is being used as general visualization tool for TerraLib databases, as well as a specialized application for spatial epidemiology and crime analysis by Brazilian public institutions (the Brazilian National Institutes of Health and the National Secretariat for Public Security). The user interface for the TerraView product is shown in Figure 5.



	SPRAREA	SPRPERIMET	SPRROTULO	SPRNOME	BAIRROMA	NASCTOT	OBITNEO	OBPOSNEO	TAXATOT	TAXANEO	TAXAPOS	PROI
1	1688035.25	7523.8003	101	101	TAUA	366	2	0	5.46	5.46	0	
2	4255087.5	11419.5215	107	107	ANCHIETA	1395	17	9	18.64	12.19	6.45	
3	31882096	40144.5	034	034	ALTO DA BOA VISTA	125	2	1	24	16	8	
4	3394894.75	10089.6953	119	119	CURICICA	489	14	6	40.9	28.63	12.27	
5	868884.4375	6019.8882	019	019	COSME VELHO	104	0	0	0	0	0	
6	1334460	7696.2212	118	118	CIDADE DE DEUS	686	12	6	26.24	17.49	8.75	
7	3474193.5	10569.3232	116	116	ANIL	86	4	7	127.91	46.51	81.4	
8	8731501	16678.2695	129	129	CAMORIM	37	1	0	27.03	27.03	0	
9	1219345.25	6295.8906	021	021	HUMAITA	129	2	0	15.5	15.5	0	

Figure 6 – User Interface for the TerraView product

8.2 SIGMUN

SIGMUN is a system for handling cadastral applications in metropolitan areas, used by the cities of Santos, São Sebastião, Caraguatatuba, São José dos Campos, São Bernardo do Campo, Cachoeiro de Itapemirim, Vitória, and in 30 cities of the state of Bahia.

8.3 Forest Fires Database (PROARCO)

PROARCO is a program for Environmental and Forest-fire Risk Monitoring over the Amazon Region, developed by cooperation amongst the Brazilian National Institute for Space Research, INPE, and the Brazilian Enviromental Agency (IBAMA). This is an operational system for the Brazilian Amazon and the Brazilian Cerrado on a daily basis for detection and monitoring of forest fires. The forest fires database has been developed using TerraLib, supports queries over spatial-temporal objects and is available on the Internet (www.dpi.inpe.br/proarco/bdqueimadas). The web interface provides a visualization of forest fires, as well as satellite imagery and land cover and cadastral information.

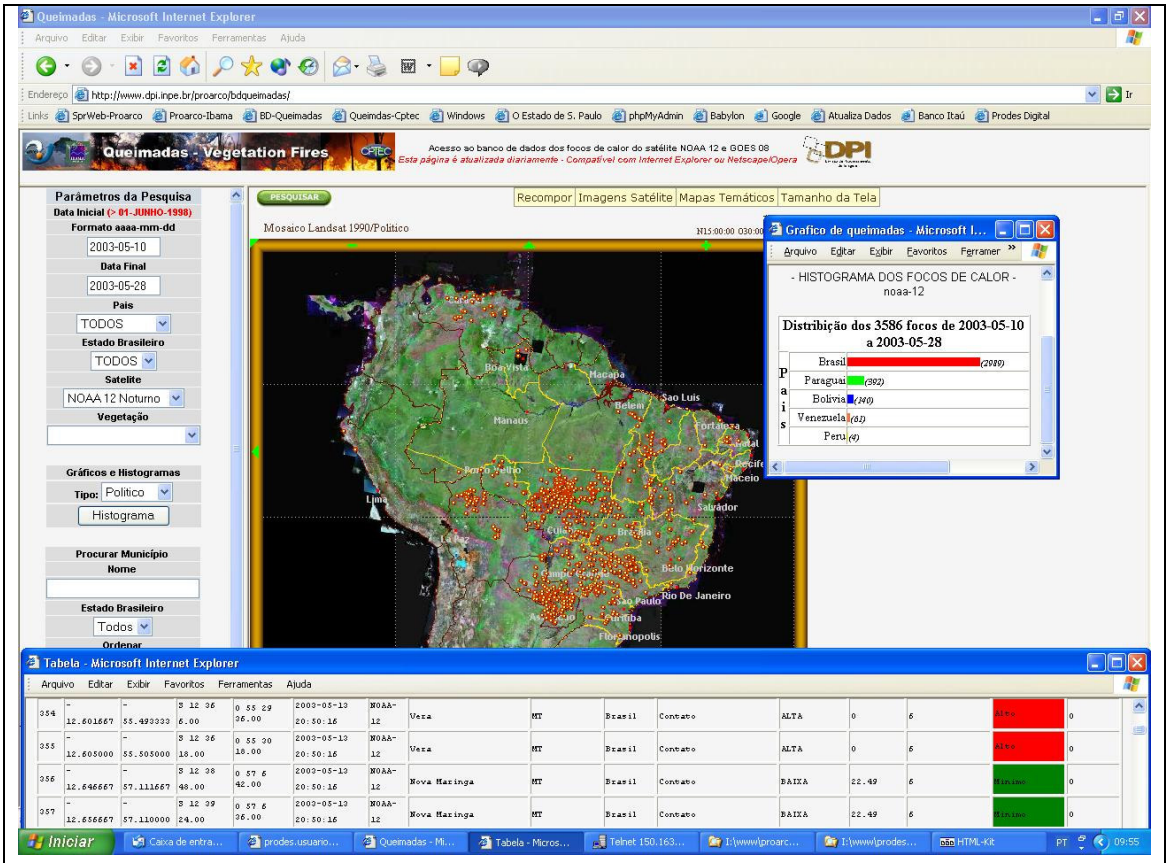


Figure 7 – Forest Fire Database (PROARCO system).

8.4 Emergency Action Plan

InfoPAE is an automated system designed to improve the response to emergency situations for the petroleum industry. The system offers sophisticated action plans, easy access to vital information and tight control over the resources allocated to face an emergency. InfoPAE works with *local emergency action plans* (LEAPs). A LEAP is a structured collection of actions, similar to a workflow, coupled with information stored in geographical as well as conventional databases. During an emergency, the team follows a previously stored LEAP, backed up by its ancillary information. The team registers the actions taken and documents eventual difficulties. Later on, upper level management may use the system to generate reports that are useful to detect eventual problems with the LEAP or to assess the performance of the team. LEAP frameworks are very useful to expedite designing a large number of emergency plans. Current plans include implementing InfoPAE at nearly 80 installations throughout 2003.

9 Conclusions

The design and implementation of TerraLib serve an example of the challenges involved in the bringing together innovative research ideas into a single framework. Many of the design decisions reached a compromise between conflicting requirements of functionality and simplicity, as illustrated by the following issues:

(a) *Database schema*: to be of practical value, a GIS library must have a database schema, even if this results in a lack of flexibility. By keeping all spatio-temporal relations independent, we have tried to allow many alternatives for application implementation, although we recognize that some users might need to adapt their current model to use *TerraLib*.

(b) *Spatio-temporal predicates*: it would be very difficult to design a system that would support all theoretical possibilities of spatio-temporal predicates. As an engineering compromise, we settled for a set of temporal relations that are general enough to support most real-life situations and can be retrieved efficiently.

(c) *Cell spaces*: the inclusion of cell spaces in a GIS database environment is an important improvement on the current situation, where most dynamical models based on cell spaces have a very loose coupling with a GIS.

(d) *Generic programming*: given that a lot of spatial analysis algorithms can be applied to different data structures, the concept of *generic programming* can be of much benefit for GIS application development.

TerraLib is an evolving product. Version 3.0 has been released on early May 2004, with an investment of 40 man-years of work. It comprises 95.000 lines of code in C++, plus 195.000 lines of code of third-party libraries. The software is available at the website http://www.terralib.org.

Acknowledgments

The development of *TerraLib* is a joint effort of INPE (National Institute for Space Research), TeCGraf/PUC-Rio (Computer Graphics Group at the Catholic University in Rio de Janeiro), PRODABEL (Informatics Corporation for the City of Belo Horizonte) and IMPA (Institute for Pure and Applied Mathematics). The library core team, apart from the authors, include Juan Garrido, Lauro Hara at INPE and Paula Frederick, Marcelo Metello and Luiz Gustavo Magalhães at PUC-Rio. The TerraLib project is partially financed by CNPq grant no. 552040/02-9. Gilberto Câmara's research is also financed by a CNPq grant no. 300557/96-5. The project has also received financial support from FAPESP (Fundação de Amparo à Pesquisa no Estado de São Paulo).

support from FAPESP (Fundação de Amparo à Pesquisa no Estado de São Paulo).

References

Allen, J.F. 1983. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM* 26 (11):832-843.

Anselin, Luc. 1999. Interactive techniques and Exploratory Spatial Data Analysis. In *Geographical Information Systems: principles, techniques, management and applications*, edited by P. Longley, M. Goodchild, D. Maguire and D. Rhind. Cambridge: Geoinformation International.

Austern, Matt. 1998. *Generic Programming and the STL : Using and Extending the C++ Standard Template Library*. Reading, MA: Addison-Wesley.

Burrough, P. 1998. Dynamic Modelling and Geocomputation. In *Geocomputation: A Primer*, edited by P. Longley, S. Brooks, R. McDonnell and B. Macmillan. New York: John Wiley.

Clementini, E., and P. di Felice. 1995. A Comparisonn of Methods for Representing Topological Relationships. *Information Sciences* 3:149-178.

Coplien, James. 1999. *Multi-Paradigm Design for C++*. Reading: Addison-Wesley.

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.

Egenhofer, Max. 1999. Spatial Information Appliances: A Next Generation of Geographic Information Systems. Artigo apresentado em First Brazilian Workshop on GeoInformatics, em Campinas, Brazil.

Erwig, Martin, and Markus Schneider. 2002. Spatio-Temporal Predicates. *IEEE Transactions on Knowledge and Data Engineering* 14 (4):881-901.

Ferreira, Karine Reis, Gilberto Queiroz, Joao Argemiro Paiva, Ricardo Cartaxo Souza, and Gilberto Câmara. 2002. Arquitetura de Software para Construção de Bancos de Dados Geográficos com SGBD Objeto-Relacionais. Artigo apresentado em XVII Simpósio Brasileiro de Banco de Dados, em Gramado, RS.

Fonseca, Frederico, Max Egenhofer, Peggy Agouris, and Gilberto Câmara. 2002. Using Ontologies for Integrated Geographic Information Systems. *Transactions in GIS* 6 (3):231-257.

Hornsby, Kathleen, and Max Egenhofer. 2000. Identity-Based Change: A Foundation for Spatio-Temporal Knowledge Representation. *International Journal of Geographical Information Science* 14 (3):207-224.

McKee, L., and K. Buehler, eds. 1996. *The Open GIS Guide*. Wayland, MA: Open GIS Consortium, Inc.

Mockus, Audris, Roy Fielding, and James Herbsleb. 2002. Two case studies of open source software development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology* 11 (3).

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.

Pedrosa, Bianca, Gilberto Câmara, Frederico Fonseca, and Ricardo Cartaxo Modesto de Souza. 2002. TerraML - A Cell-Based Modeling Language for an Open-Source GIS Library. Artigo apresentado em II International Conference on Geographical Information Science (GIScience 2002), em Boulder, CO.

Stroustrup, Bjarne. 1997. *The C++ Programming Language, 3rd ed.*: Addison-Wesley Publishing Company.

Vinhas, Lubia, Gilberto Ribeiro de Queiroz, Karine Ferreira, Gilberto Câmara, and João Argemiro Paiva. 2002. Programação Genérica Aplicada a Algoritmos Geográficos. Artigo apresentado em IV Simpósio Brasileiro de Geoinformática, em Caxambu.

Vinhas, Lubia, Ricardo Cartaxo Modesto de Souza, and Gilberto Câmara. 2003. Image Data Handling in Spatial Databases. Artigo apresentado em V Simpósio Brasileiro de Geoinformática, em Campos do Jordao.

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.