Rule-based evolution of typed spatiotemporal objects

Olga Oliveira Bittencourt[1], Gilberto Câmara[1], Lúbia Vinhas[1], Joice Seleme Mota[1]

[1]Image Processing Division – National Institute for Space Research (INPE)
Avenida dos Astronautas 1758 – 12227-001 – São José dos Campos – SP – Brazil
{olga,gilberto,lubia,joice}@dpi.inpe.br

Abstract. *This paper describes a model for spatiotemporal objects whose location is fixed, but its boundaries and properties change. We refer to these as evolving objects. We consider cases where the evolution of an object is dependent of its type and propose a rule-based approach for description of spatiotemporal object's evolution. By developing semantics of type-based evolution, we can keep a detailed history of change. We present an example where the model is able to represent type conversions and recover the evolution history of a set of objects. The model allows answers to questions about causes of change and thus deals with cases not supported by models based on objects of a unique type.*

1. General Information

A major research topic in GIScience is modelling and representation of geographical objects whose properties change. We distinguish two broad categories of spatiotemporal objects. The first case concerns those objects that change their position and extent continuously. We refer to those as *moving objects*. Moving objects arise, for example, in location-based systems that deal with spatial and temporal position of planes, storms or cars. The second type concerns objects that do not move, but whose geometry, topology and properties change. We refer to those objects as *evolving objects*. Evolving objects arise, for example, in urban cadastre and in land change patterns.

The two categories of spatiotemporal objects need different ways of data modelling, representation and algorithms. Handling moving objects demands notions such as *trajectory* (Güting et al., 2000), plus specialized query methods (Sistla et al., 1997) and indexing techniques (Šaltenis et al., 2000). The widespread availability of location-based systems motivated advances in moving objects databases (Erwig and Schneider, 2002) (Güting and Schneider, 2005). By contrast, handling evolving objects requires tracking the changes that occurred during an object's lifetime, such as creation, splitting and merging (Hornsby and Egenhofer, 2000; Medak, 2001).

One recent technique for handling spatiotemporal objects is event-based calculus (Worboys, 2005; Worboys and Hornsby 2004; Vidal and Rodriguez 2005). Event-based calculus captures the semantics of spatiotemporal objects by specialized events that are external to the objects themselves. Each application is associated to a specialized set of events. For example, the semantics of traffic objects uses events such as *departure*, *arrival*, or *unexpected destination* (Hornsby and Cole, 2007). Event-based techniques have proven useful in applications such as traffic models.

In this paper, we deal with evolving objects. We deal with cases where the simple rules of merging and splitting are not enough to describe their evolution. These

situations arise when objects are defined not only by their shape and properties, but also by their *type*. Consider the case of riverbanks. Definition of what is 'the river' and what is 'the land' changes over the seasons. When a river expands during the wet season, the part of the land that is flooded will be split and merged with the river. The object that matches the flooded area will change its type and properties. In the dry season, this object may become land once again. In this evolution, expansions and contractions produce junctions and splitting which are type-dependent. In this and similar cases, recording the history of changes needs keeping track of type-dependent cases. This requires a higher-level of semantics above that of the basic operations of creation, splitting and merging. We shall refer to those objects as *typed evolving objects*. This raises the question we explore in this paper: "*How can we deal with spatiotemporal objects whose evolution is type-dependent?*"

This paper proposes a rule-based approach for description of object's evolution. The rules arise from knowledge about the application domain. They provide a higher-level semantics layer that uses low-level operations and deals with type dependency. By developing semantics of type-based evolution, we can keep a detailed history of changes. Then, we can recover the evolution history of a set of objects and answer important questions about the causes of changes. Our proposal involves defining a set of object types and a set of functions applicable on those types. This leads to an algebraic formulation, which can be implemented easily in a functional language or translated into an imperative language or a specialized query processor.

The rule-based evolution proposed in this paper has some similarities with the event-based calculus. In both cases, we describe changes in objects by a set of occurrences. The main difference is that rule-based evolution uses functions, which are more general than events. An event can be modelled as a function, but there are some functions that are needed to describe an object's evolution which are not occurrents. For example, the function *history* provides a description of the changes in an object. Therefore, the function-based approach can be seen as a generalization of the event-based calculus to include occurrents (modelled as events) and other types of operations.

This paper is organized as follows. In section 2, we present the idea of rule-based algebras for describing evolution of spatiotemporal objects, and review previous work on the subject. In section 3, we present the operations defined for evolving objects. In section 4, we present a case study on using rule-based evolution. In section 5, we discuss how to implement the proposal. In section 6 we present some conclusions and future work.

2. Rule-based Evolution of Spatiotemporal Objects and its Relation to Previous Work

Pelekis et al. (2004) and Roddick et al. (2004) review the different types of spatiotemporal data models proposed in the literature. They point out the differences between models that describe moving objects and those focused on object lifelines. Moving objects are the ones whose position and extent change continuously. Güting et al. (2003) and Güting and Schneider (2005) propose an algebra for moving objects, composed by a set of spatiotemporal data types, axioms and their operations. Algebraic data types provide a conceptually clean foundation for representation and querying of moving objects (Güting et al., 2003). The specific data types defined to handle moving

objects are *moving points* (objects where only the position in space is relevant) and *moving regions* (objects where the position and the time-dependent extent are relevant). This algebra was implemented using SECONDO (Güting et al., 2004), an extensible and modular DBMS environment created to support development of algebras. The set of data types and operations can answer questions such as: "*Given the trajectories of snowstorms and aeroplane flights, which flights went through a snowstorm?*".

A second area of research concerns object lifelines. These works focus on describing the history of incremental changes in the life of an object. This happens in applications such as urban cadastre, where parcels are created, merged, split and wiped out. To keep track of an object's history, all changes need to be modelled and recorded. Hornsby and Egenhofer (2000) stress the need to preserve an object's identity as it changes its geometry, topology, and attributes in time. This view is also supported by Grenon and Smith (2003). Medak (2001) developed an algebra to model object lifelines. Medak's algebra provides a set of basic operations, which are a foundation for more specific applications. The literature on object lifelines uses three basic ideas: identity, life, and genealogy. Identity is the characteristic that distinguishes each object during all its life. Life is the time period from the object's creation until its elimination. Genealogy implies managing the changes that an object has during its life. Medak (2001) and Hornsby and Egenhofer (2000) propose the use of ideas such as *creation*, *destruction*, *merging*, *splitting*, *death* and *reincarnation* to record the history of the objects, following the changes and keeping its identity.

Our paper focuses on object lifelines. The current work on object lifelines focuses on objects of the same type and the usual examples involve parcels and counties. For more complex cases, we need to consider operations involving objects of different types, which arise in many real-life applications. Consider the question "*when does a tropical storm become a hurricane?"* To answer this question, we need to consider more than the trajectory of the storm. There are many conditions that determine how tropical storm becomes a hurricane. They include the storm's wind-force, the sea surface temperature and the trajectory. These conditions need to be fed into a set of rules that will eventually convert an object of type '*tropical storm'* to an object of type '*hurricane*'.

As a second example, consider the history of Bangladesh during the last 800 years. Islam was introduced to Bengal in the XII century. By the XVI century, the Mughal Empire controlled the area around the Bay of Bengal. The British gained control of Bengal in 1757. When India was partitioned in 1947, Bengal was split along religious lines, with the western part going to India and the eastern part joining Pakistan as a province called '*East Pakistan'*. The people from Bangladesh gained their independence from Pakistan in 1971 (Wikipedia, 2007). Thus, the region changed their status many times during the last 800 years. A history of the region that would consider the Islam culture as its building block would need to account for the various status changes. Consider the Islamic area around the Bay of Bengal as a spatiotemporal object. Its status changed from '*part of an empire'* to '*part of a province of an empire*' and then to ' *province of an independent country*' and finally to '*independent country*'.

These examples lead us to consider how to enrich our models of spatiotemporal objects to capture the complexity of such changes. They lead us to propose the idea of *typed spatiotemporal objects*. Our view of types comes from Computer Science, where

types are tools for expressing abstractions in a computer language (Cardelli and Wegner, 1985). On a theoretical level, a type is a set of elements in a mathematical domain that satisfy certain restrictions. A *typed spatiotemporal object* is an object whose evolution is subject to constraints that are specific to its type. Thus, objects of type '*independent country*' will have different rules for evolution than those of type '*province*'. Using types to model the evolution, we can gather richer models for capturing the semantics of evolving objects.

3. Operations for Evolving Objects

To carry out the evolution in a computer model we defined specific operations to evolving objects. This section informally presents these operations with some definitions and conventions we adopted:

1. Data types, functions and instances use `monospaced` font. Types are in **`boldface,`** their instances and reserved words are shown in `normal` font. For lists associated to types, we use **`list_typename`**.
2. Type definitions follow usual conventions for abstract data types. Types have an externally viewable set of functions and a set of axioms that are applicable to these functions.
3. Each function has a signature, given a
   `function:` **`typeA × typeB → out_type`**.
   `typeA` and **`typeB`** are the types of the input parameters and **`out_type`** is the type of the output.
4. We describe each rule in pseudocode. For attribution, we use ':='. '`As`', '`in`' and '`with`' are reserved words. For sets of objects, we use '[]'. For groups, we use '{}' and the separator is ';'.

We defined the functions *create*, *split*, *merge*, *evolve*, *setType*, *getType*, *getInstance* and *remove* in our experiments. Table 1 presents an informal set of signatures and explanations for these evolving operations.

Table 1. Informal definition of evolving operations

Function	*Signature*
`create`	**`timestamp x type → ST_object`** Given a specific type and its timestamp, create an instance of a spatiotemporal object of the same type.
`getInstance`	**`ST_object x timestamp → S_object`** Recover the static instance of the spatiotemporal object, given a timestamp.

setType	**ST_object x type → ST_object** Set the type of the spatiotemporal object.
getType	**ST_object x timestamp → type** Given a timestamp, retrieve the type of the spatiotemporal object.
merge	**ST_object x ST_object x timestamp → ST_object** Given two evolving objects, merge them produce an object based on the evolution rules.
split	**ST_object x ST_object x timestamp → ST_object x ST_object** Given two evolving objects, split them produce two objects based on the evolution rules.
remove	**ST_object → null** Removes the spatiotemporal object of the model.

We created the *evolve* function to allows grouping evolutions according with similarity ideas or specific actions significant to user. These evolutions can be further recognized during the history recovery. Its signature is:

`evolve` **`identifier timestamp { functions }`**

Further, we defined a parametric function to recover the *history* of objects. This function allows defining richer and relevant forms of recovering information inside the history of objects. It is more than just recovering the static operations. Our distinct signatures give us kinds of looking the history by different points of view, recovering distinct information and combining it with relevant information related to spatiotemporal object's evolution. The basic signature of *history* is:

`history` **`ident_option form_option time_option → [ST_object]`**

Each parameter option and their combinations, presented in Table 2, allow recovering different aspects of evolving object *history*.

Table 2. Parametric function *history*

Parameter	*Signature*
ident_option	**ST_object → [ST_object]** Recover the evolution of a spatiotemporal object using its identifier. This evolution consists in a set of evolving objects with their respective timestamps.

	Alias → [ST_object] Recover the evolution of a spatiotemporal object using the alias defined to the spatiotemporal object.
form_ option	complete **→ [ST_object]** Recover the complete history of the spatiotemporal object, including all evolutions that happened with any split or merge of this object. The idea is recovering the tree with the original object on the root and all subtrees with its evolutions.
	reverse **→ [ST_object]** Recover the reverse history of the object with the focus on the last evolution to its beginning.
time_ option	from **timestamp → [ST_object]** Fix the early timestamp to recover the history. The result will contain information until the last evolution of the spatiotemporal object.
	until **timestamp → [ST_object]** Fix the end timestamp to recover the history. The result will contain information from the spatiotemporal object creation until the evolution before or equal the mentioned timestamp.
	from **timestamp** until **timestamp → [ST_object]** Fix the early and end timestamp to recover the history of the spatiotemporal object.

4. A Simple Example of Rule-based evolution

To evolve our objects, we used a rule-based evolution approach. The idea is to derive a set of rules from domain knowledge. These rules will act as the constraints on the specific types of spatiotemporal objects. In a general case, we propose a series of steps for using the concepts of typed spatiotemporal objects and rule-based evolution for modelling a real-life case, as outlined below:

1. Use a domain expert to elicit the different types of objects needed to model the problem.
2. Use a domain expert to set up the rules that govern the evolution of the different types of objects.
3. Express the object types and the evolution rules in a computer model. Preferably, develop a set of algebraic data types and operations on them.
4. Use the computer model to capture the history of the study area.

In the next subsection, we present a case study where the idea of rule-based evolution was applied to model the evolution of Bangladesh case.

4.1. The Bangladesh case

This section exemplifies the rule-based evolution of spatiotemporal objects. We present a case study focusing on the history of Bangladesh (illustrated in Figure 1) during the last 800 years. The history of the region that would consider the Islam culture as its building block would need to account for the various status changes. It is on that viewpoint that our example focuses on.

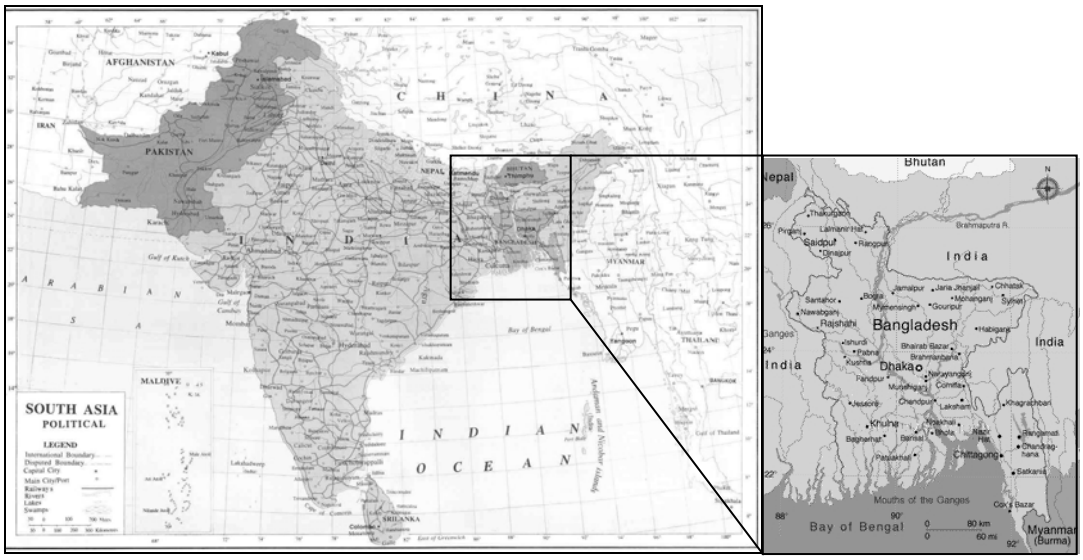

Figure 1. Illustration of Bangladesh in South Asia.

Consider the Islamic area around the Bay of Bengal as a spatiotemporal object. Bangladesh status changed from '*part of an empire'* to '*part of a province of an empire*' and then to '*province of an independent country*' and finally to an '*independent country*'. A model to account for its evolution would have rules, rather defined by a domain expert, such as the following:

R1. *Splitting a province produces two provinces.*

R2. *Splitting a province of one Empire produces one province and one Empire without that province.*

R3. *Splitting a province of one Country produces one province and one Country without that province.*

R4. *Merging two provinces produces one province.*

R5. *Merging a province and a country produces a Country with a new region and one region that is part f the Country.*

R6. *Merging a province and an Empire produces an Empire with a new region and one region that is part f the Empire.*

Table 3 summarizes the main information captured on these rules. It presents rules, functions, input types of evolving objects and output types of resulting evolving objects.

Table 3. Summary of Bangladesh case rules

Rule	Function	Input Types		Output Types	
		Status	Status	Status	Status
Province_ split	split	Province		Province	Province
Province_ independence	split	Province part of an Empire	Empire	Province	Empire
Province_ independence	split	Province part of a country	Country	Province	Country
Province_ conquest	merge	Province	Province	Province	
Country_ conquest	merge	Country	Province	Country	Province part of a Country
Empire_ conquest	merge	Empire	Province	Empire	Province part of an Empire

4.2. Modelling the evolution

Our spatiotemporal objects evolve as *merging* and *splitting* as the result of conquests and independence events. The set of rules defined by the domain expert govern the evolution of these events. The rules are expressed in a computer model through the combination of input types of evolving objects and the possible functions defining the output evolving objects. The evolution of rules is expressed by the operations on them.

To exemplify the evolutions and to be able of querying the history by different viewpoints, we defined some operations that describe the main events occurred. Consider the indications of area_x, for example area_1, the spatial object. The model allows answer questions such as: *"How regions evolve? What happens when two objects merge?"* and understand how changes occurred in Bangladesh.

1. Islam was introduced to Bengal in the XII century.

```
create area_1 as "Ancient Asia" with status := "Empire";
create area_2 as "Bay_of_Bengal";
create area_3 as "Bengal" in "XII century";
setType Bengal with religion := "Islam";
merge "Ancient Asia" "Bengal";
```

The evolution on merge operation will follow the rules for merge and split operations. It will detect that the applied rule is 'Empire_conquest' of 'Ancient Asia' Empire that evolves Bengal to 'Province part of an Empire'.

2. By the XVI century, the Mughal Empire controlled the area around the Bay of Bengal.

```
create area_4 as “Mughal Empire” with status := “Empire”;
evolve Mughal_Empire_control_Bengal in “XVI century”
{
    split “Ancient Asia” Bengal;
    merge “Mughal Empire” Bengal;
};
```

This step contains two different evolutions. The first is a ‘Province_independence’ where Bengal is a Province that is not anymore part of ‘Ancient Asia’ and ‘Ancient Asia’ is an Empire without Bengal Province. The second evolution is an ‘Empire_conquest’ where Bengal is ‘Province part of an Empire’, ‘Mughal Empire’ in this case.

This step also presents the option of grouping evolutions in a meaningful event defined by ‘Mughal_Empire_control_Bengal that favours the understanding of evolutions. Other steps in this example continue representing changes on the area.

3. The British gained control of Bengal in 1757.

```
create area_5 as “United Kingdom” with status := “Empire”;
evolve British_control_of_Bengal in 1757
{
    split “Mughal Empire” Bengal;
    merge “United Kingdom” Bengal;
};
```

4. When India was split in 1947, Bengal was split along religious lines, with the western part going to India and the eastern part joining Pakistan as a province called ‘*East Pakistan*’.

```
create area_6 as “Pakistan” with status := “Country”;
create area_7 as “India” with status := “Country”;
evolve divisao_provincia_Bengala in 1947
{
    split “Britain” Bengala;
    split Bengala area_4 as West_Bengal and East_Pakistan;
    merge East_Pakistan Pakistan;
    merge West_Bengal India;
};
```

5. The people from Bangladesh gained their independence from Pakistan in 1971.

```
evolve Bangladesh_independence in 1971
{
    split Pakistan East_Pakistan;
    create East_Pakistan as Bangladesh with status := "country";
};
```

4.3. Recovering the history

To exemplify some uses of history function, tables 4 and 5 show the result of two history queries. These tables include the most relevant information about the model: (a) the timestamp; (b) operations applied on the spatiotemporal objects; (c) the optional alias on that timestamp; (d) the spatiotemporal object's type; (e) the typed evolution of the spatiotemporal object on that timestamp.

Table 4 shows the answer to the query '`history Bangladesh from 1971`'. This query focuses on changes in the object since its creation in 1971. Then, the only result that this query will recover is the creation of the object in 1971.

Table 4. Result of 'history Bangladesh from 1971'

Timestamp	*Operation*	*Alias*	*Status Type*	*Evolution*
1971	Creation	Bangladesh	Country	Bangladesh_independence

Table 5 shows a summary of the answer to the query '`history Bangladesh reverse`'. This query specifies the interest in all changes that occurred with the object Bangladesh and its predecessors. The query specifies the reverse order, then, the user focuses on looking from the last Bangladesh evolutions until the beginning of the generation.

Table 5. Answers to 'history Bangladesh reverse'

Timestamp	*Operation*	*Alias / Status Type*	*Evolution*
1971	Split Pakistan; creation;	Bangladesh / Country	Bangladesh_independence
1947	Split Britain; merge Pakistan;	East_Pakistan / Part of a Country	divisao_provincia_Bengala
1715	Split Mughal Empire; merge with Britain;	Bengal / Province part of a Province of an Empire	British_control_of_Bengal
XVI century	Split Ancient Asia; merge Mughal Empire;	Bengal / Province part of an Empire	Mughal_Empire_control_Bengal
XII century	Creation; Type religion: = Islam; Merge Ancient Asia;	Bengal / Province part of an Empire	Nondefined

With this approach and the operations we are able of easily answering questions such as *"How do the Bengal region of XVI century evolved?", "What happened when the Bengal region was split?" or "What happened when East_Pakistan merged with Pakistan?"*. These questions show some new possibilities that our approach allows to model and query data.

5. Implementation Road Map

This section presents an implementation road map to the rule based evolution approach. The first step is to create evolution rules composed by: 1) detecting the objects of interest; 2) analysis of evolution cases by the specialist and; 3) development of a set of evolution rules. Through the set of rules it is possible to define the second step: proposing and implementing a group of operations to characterize and evolve the spatiotemporal objects.

The set of evolution rules leads to an algebraic formulation, which can be easily implemented in a functional language or translated into an imperative language or a specialized query processor. From an implementation point of view, the rule-based evolution of typed spatiotemporal objects is possible and simple.

Currently, we are in the second step on the development of a model to land use changes in the Amazon region (Escada et al., 1997, INPE, 2005). This is a complex scenario and future work will present this complete model based on the idea of evolution rules. The experiments are being performed in a prototype developed in the environment TerraHS (Costa et al., 2006). TerraHS integrates the functional programming and the spatial databases for GIS application development. We are using TerraHS because it allows prototyping algebras and we are implementing our evolving objects and functions in an easy and complete manner.

6. Conclusions

In this paper, we deal with *evolving objects*. We are interested in cases where the simple rules of merging and splitting are not enough to describe their evolution and the evolution of objects is dependent of their *types*.

This paper proposes the concept of *typed spatiotemporal objects* and the use of rule-based evolution approach to capture a detailed history of changes of spatiotemporal objects. Rule-based evolution works best when the domain knowledge is well known, and we are able to assign a meaningful type system to the objects. Our proposal involves defining a set of object types and a set of functions applicable on those types. Then, we can recover the evolution history of a set of objects, answer important questions about causes of change and thus deals with cases not supported by models based on objects of a unique type.

Future works will be realized on other areas and scenarios. Currently, we are interested in studying the evolution of land use changes in the Amazon region. For next steps, an algebra of evolving objects will be developed as well as new operations to advance our model and to characterize other problems.

Acknowlegment

Gilberto Camara's work is partly funded by CNPq (grants PQ – 300557/19996-5 and 550250/2005-0) and FAPESP (grant 04/11012-0). Olga Oliveira Bittencourt's work is funded by CAPES.

References

CARDELLI, L. and WEGNER, P., 1985, On Understanding Type, Data Abstraction, and Polymorphism. *ACM Computing Surveys,* **17,** 471-552.

COSTA, S. S., CÂMARA, G., and PALOMO, D., 2006, TerraHS: Integration of Functional Programming and Spatial Databases for GIS Application Development. In: *Advances in Geoinformatics, VII Brazilian Symposium in Geoinformatics, GeoInfo2006*, Davis, C. A. Jr. and Monteiro, A. M., eds. (Campos do Jordão, Brazil: Springer), **24**, 127-148.

ERWIG, M. and SCHNEIDER, M., 2002, Spatio-Temporal Predicates. *IEEE Transactions on Knowledge and Data Engineering,* **14,** 881-901.

ESCADA, M. I. S., MONTEIRO, A. M., AGUIAR, A. P., CARNEIRO, T. and CAMARA, G., 2005, Análise de padrões e processos de ocupação para a construção de modelos na Amazônia (Analysis of land use patterns and processes for the construction of models in Amazonia: Experiments in Rondonia). In *XII Brazilian Symposium on Remote Sensing*, (Goiania, Brazil: SELPER), 2973-2983.

GRENON, P. and SMITH, B., 2003, SNAP and SPAN: Towards Dynamic Spatial Ontology. *Spatial Cognition & Computation,* **4,** 69-104.

GÜTING, R. H., BEHR, T. and ALMEIDA, V., 2004, SECONDO: An Extensible DBMS Architecture and Prototype. Report (Hagen: Fernuniversität Hagen).

GÜTING, R. H., BOHLEN, M. H., ERWIG, M., JENSEN, C. S., LORENTZOS, N., NARDELLI, E., SCHNEIDER, M. and VIQUEIRA, J. R. R., 2003, Spatio-temporal Models and Languages: An Approach Based on Data Types. In *Spatio-Temporal Databases*, Koubarakis, M., ed., (Berlin: Springer).

GÜTING, R. H., BÖHLEN, M. H., ERWIG, M., JENSEN, C. S., LORENTZOS, N. A., SCHNEIDER, M. and VAZIRGIANNIS, M., 2000, A Foundation for Representing and Querying Moving Objects. *ACM Transactions of Database Systems,* **25**.

GÜTING, R. H. and SCHNEIDER, M., 2005, *Moving Objects Databases.* (New York: Morgan Kaufmann).

HORNSBY, K. S. and EGENHOFER, M., 2000, Identity-Based Change: A Foundation for Spatio-Temporal Knowledge Representation. *International Journal of Geographical Information Science,* **14,** 207-224.

HORNSBY, K. S. and COLE, S., 2007, Modeling Moving Geospatial Objects from an Event-based Perspective. *Transactions in GIS*, **11(4)**, 555-573.

INPE, 2005, Monitoramento da Floresta Amazônica Brasileira por Satélite (Monitoring the Brazilian Amazon Forest by Satellite). Report (São José dos Campos: INPE).

MEDAK, D., 2001, Lifestyles. In *Life and Motion of Socio-Economic Units. ESF Series*, A. U. Frank, Raper, J., & Cheylan, J.-P., ed., (London: Taylor & Francis).

PELEKIS, N., THEODOULIDIS, B., KOPANAKIS, I. and THEODORIDIS, Y., 2004, Literature review of spatio-temporal database models. *The Knowledge Engineering Review*, **19,** 235-274.

RODDICK, J., EGENHOFER, M. and HOEL, E., 2004, Spatial, Temporal and Spatiotemporal Databases Hot Issues and Directions for PhD Research. *SIGMOD '93, SIGMOD Record,* **33,** 126-131.

ŠALTENIS, S., JENSEN, C. S., LEUTENEGGER, S. T. and LOPEZ, M. A., 2000, Indexing the positions of continuously moving objects. *ACM SIGMOD Record*, **29,** 331-342.

SISTLA, A. P., WOLFSON, O., CHAMBERLAIN, S. and DAO, S., 1997, Modeling and Querying Moving Objects. *Proceedings of the Thirteenth International Conference on Data Engineering,* 422-432.

VIDAL, C. and RODRIGUEZ, A., 2005, A Logical Approach for Modeling Spatio-temporal Objects. *2nd International Workshop on Conceptual Modeling for Geographic Information Systems, CoMoGIS 2005*, 218-227.

WIKIPEDIA, 2007. Wikipedia, The Free Encyclopedia. Retrieved in October, from: http://en.wikipedia.org/wiki/Bangladesh.

WORBOYS, M. F., 2005, Event-oriented approaches to geographic phenomena. *International Journal of Geographical Information Science,* **19(1)**, 1-28.

WORBOYS, M. F. and HORNSBY, K. S., 2004, From Objects to Events: GEM, the Geospatial Event Model. In: *Third International Conference, GIScience 2004*, Egenhofer, M. J., Freksa, C. and Miller, H. J. eds., (Adelphi, USA: Springer), 327-343.