

# FORMAL MODELLING OF TEMPORAL-SPACE SYSTEM IN OBJECT-Z

Cristian Vidal

Business Informatics Administration School  
Universidad de Talca  
Talca, Chile  
cvidal@utalca.cl

Philip Sallis

Geoinformatics Research Centre  
Auckland University of Technology  
Auckland, New Zealand  
psallis@aut.ac.nz

**Abstract**—In general the formal modelling of software systems is both non-traditional and a seldom undertaken task. Partly this is due to the increased development cost it incurs. It is also due somewhat to a lack of appreciation of the value formal methods can bring to the software development process in terms of producing timely and error-free programming. Additionally, the deficit of competent professionals to undertake the work militates against the adoption of these methods and yet there is general acceptance of the proposition that greater robustness can be brought to software products using formal modeling thus, reducing post-implementation costs. Of concern is that little attention has in recent years been paid to using formal methods in the modelling of high cost (often critical) systems. In this paper, we will demonstrate for Spatio-Temporal information systems, (an example critical system class), how by using a traditional modelling approach and a current object oriented formalism, (Object-Z), we can produce a robust and error-free data design.

**Keywords**- *Modelling, Spatio-Temporal, States, Events, Object-Z.*

## I. INTRODUCTION

The formal modelling and formal requirements specification of a temporal-space system (*spatio-temporal system*), sometimes referred to as *temporal geospatial system*, is a non-traditional task in software engineering, computer science and geoinformatics. In [1] a formal model of a spatio-temporal system is presented, in a deductive logic database system context, using the *Object Event Calculus* (OEC) [2] formalism.

In the modelling of a temporal geospatial system, the following modelling approaches are used: the Snapshot-based approach, the State-based approach and the Event-based approach.

The goal of this paper is to present a formal specification model for spatio-temporal system requirements. A formal object oriented requirement specification language (Object-Z) is used [3]. Being an extension of the formal language Z [4], Object-Z allows the inclusion of object oriented and temporal modelling elements.

In *Section 2* of this paper the different conceptual modelling approaches of a spatio-temporal system are presented. In *Section 3* the Object-Z formalism used here is

described. In *Section 4* the application of Object-Z for the formal specification of spatio-temporal system is described. In *Section 5*, the main conclusions from this paper are given and some future work is presented.

## II. SPATIO-TEMPORAL MODELLING

In this paper, it's presented a formal modelling of spatio-temporal system using the approach that combines the State-based Model and Event-base Model described in A and B sections, respectively.

### A. State-based Model

The state-based model introduces the idea of using a timestamp to the set of objects being considered. In a model based on states of the objects, this concept records all the temporal states of an object. The state of an object is completely defined according to the value of its attributes. So, when an object changes the value of any of its attributes that object experiences a change of state.

The traditional state-based model associates a timestamp with each change of state for any objects present in the model of the system. This means that changes of state for each object are reflected in the model. The initial value of each attribute is registered as a change of state of the object. For instance, if we have a *Person Class* with an identity attribute *Person* and attributes *Attribute* and *State* then a change of state associated with the object of this class (eg. Person: Cristian), this can be seen as represented by the following table:

**Table 1.** Example of the classic state-based model relation Change\_State.

*Class Person - Object Cristian*

Person	Attribute	State	Vs
Cristian	Job	Programmer	01/01/2000
Cristian	Job		01/03/2000
Cristian	Job	Programmer	01/01/2001
Cristian	Job		07/03/2001
Cristian	Job	Engineer	01/08/2005
Cristian	Location	Serviu	01/01/2000
Cristian	Location		01/03/2000
Cristian	Location	Serviu	01/01/2001
Cristian	Location		07/03/2001
Cristian	Location	CBB	01/08/2005

In practical implementations of temporal databases a timestamp based in intervals of time is used to represent transaction time (Ts-Te and Vs-Ve, respectively) [9]. These timestamps can be associated with any dynamic relation in which the objects participate. Using a dynamic relation attribute (say, Works\_At) we can see how in Table 1 an associated timestamp could be applied to each tuple. It should be noted that the temporal granularity for all the tuples in the relation is daily.

**Table 2.** Example of the relation Works\_At in a state-based model [8].

Persona	Empleo	Lugar	V <sub>s</sub> - V <sub>e</sub>
Cristian	Programador	Serviu	01 de Enero 2000 - 01 de Marzo de 2000
Cristian	Programador	Serviu	01 de Enero 2001 - 07 de Marzo 2001
Cristian	Ingeniero	Cementos Bio Bio.	01 de Julio 2005 - NOW

In the previous example the utilization of the value **NOW** for one side of the time interval is shown, which allows it to represent a valid interval open on its right side (in other words of current validity) [10]. When responding to a query that includes attributes with a **NOW** value, this value should be replaced by the current time value.

In general, the state-based model answers a wide range of spatio-temporal queries. Nonetheless, in its conception, the model leaves out an important set of information regarding the phenomena that cause the change of states of an object. This model only focuses in the states of the objects. It doesn't say anything about the events that affect the states of the objects being modeled.

### B. Event-based Model

A model based only on events recognizes only spatio-temporal event occurrences. From these occurrences the temporal state of an object is obtained. At a conceptual level

an event-based model enables the modelling of *Participation* and *Involvement* relations between objects and events [11, 12]. At a logical and physical database level an event-based model includes the state of the objects, in the style of the Object-based Event Calculus [1, 2, 7] (OEC).

A model based on events is a complete model because it reflects all occurrences of spatio-temporal events. This model type usually restricts the set of events to information explicit in the problem domain. This way the loading of unnecessary and irrelevant information is avoided, which in theory reduces the otherwise high computational cost of data transfer and object association.

In fact, although this model makes possible the completeness of spatio-temporal queries, it has been empirically confirmed that it does have an operationally high computational cost. This is due mainly to the non existence of a state base for each object in the model. This in turn requires further processing to retrieve the number of tuples required to answer what are often basic spatio-temporal queries [8].

The literature emphasizes the following types of events associated to objects with spatial variability [8]:

- **Create:** An event that creates an object.
- **Change:** An event that changes a property of an object without changing the objects identity. It is distinguished between attribute changes (internal change) and membership change (external change).
- **Destroy:** An event that destroys an object.
- **Split:** An event that creates one, two or more new objects from an existing object, which suffers a change on his spatial limits (and possibly in some non spatial attributes) caused by the occurrence of this event.
- **Merge:** An event that can result in the creation of a new object, due to a fusion of two or more objects which disappear after , or due to a change in the limits of an existing object (and possibly to some non spatial attributes), caused by the fusion of this existing object with one, two or more objects which disappear.

### III. OBJECT-Z

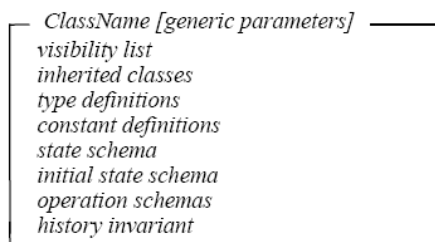
Object-Z is a formal requirements specification language of object oriented software, being an extension of the language Z [3, 13, 14, 15] with the incorporation of object oriented concepts [16]. Z is a requirements specification model-oriented language with features for modelling complex data structures and their operations. Object-Z improves the clarity and power of modelling Z specifications, due to its object oriented features (encapsulation, inheritance and polymorphism) and temporal modelling features.

Object-Z, just like the formal language Z, is a model-oriented language and its main root is in set theory; its most important feature is the class diagram. A class diagram has

the shape of a box with a Z style name, and optional generic parameters; including an optional section for temporal invariants of the class. *Figure 1*, illustrates the shape of an Object Z diagram class box.

The box components are:

- A visibility list that restricts access to attributes and operations;
- A class list from which its attributes and methods are inherited (superclasses);
- A definition list of the classes types and constants;
- A state diagram where the class state attributes and invariants are defined.
- An initial state diagram which specifies the initial state of the class object.
- A set of operation diagrams; where in each one of them the pre-conditions and post-conditions of each operation of the class are specified.
- A temporal invariant or history invariant that restricts the operations order by using temporal logic.



**Fig. 1.** Object-Z Scheme Class [3, 13, 14].

The class diagram extends the definitions of *types* described in the language Z, allowing the classes to act like types.

As mentioned before, the temporal or history invariant of an Object-Z class diagram makes possible to define a temporal order on the possible occurrence of the class methods. The operators  $\square P Q$  and  $\Delta P Q$  are used, where **P** is a method of the class; indicating that the method **P** is always in condition **Q**, or eventually in condition **Q** respectively, where the condition **Q** could be **Enabled** or **Occurs** [3].

The use of a formal requirements specification is necessary for the implementation of complex software applications [17], like a spatio-temporal information software application. In this work, the modelling of a spatio-temporal information system is presented, using the modelling approaches described in the *section 2*.

#### IV. OBJECT-Z STATE + EVENTS SPATIO-TEMPORAL MODELLING

An approach to modelling spatio-temporal data system with Object-Z, using the states of an object, together with the record of the event that originates the change of state, is presented in this work. The spatial information, attributes and events, are presented in a simple form, similar to non-spatial information, but with some identifying information.

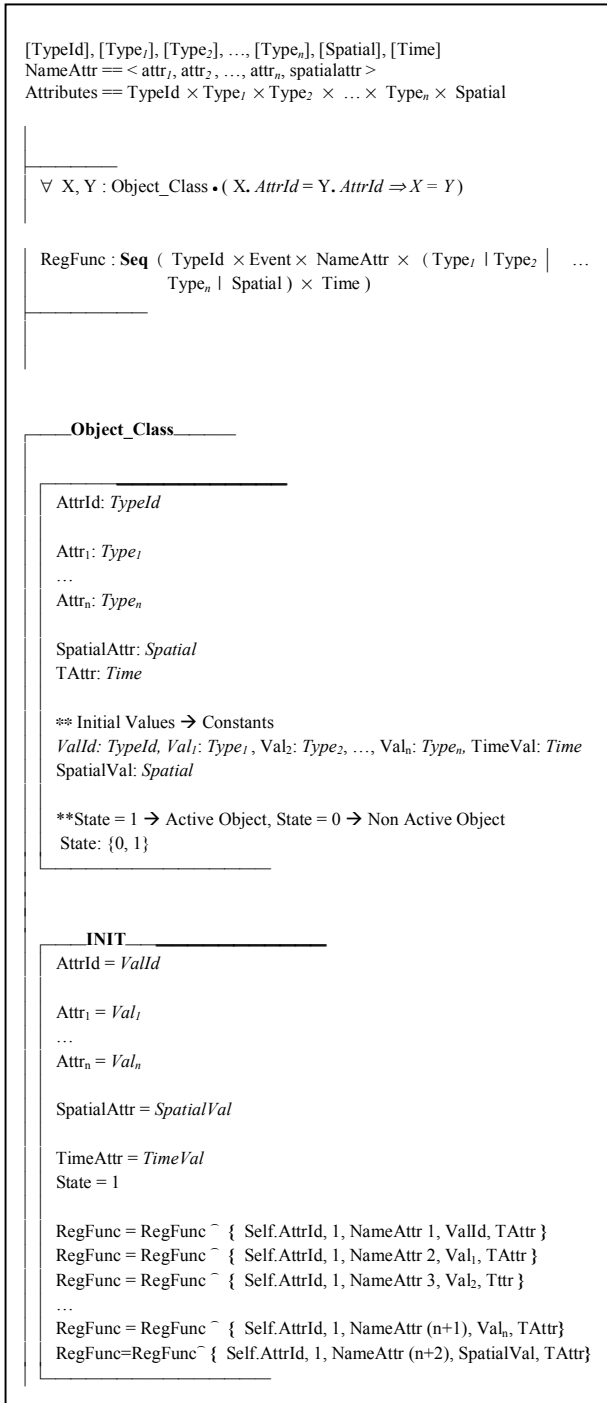
In this modelling approach, the class *Object\_Class* is presented with the necessary attribute types definitions and some global rules. For example, the identity attribute must not have the same value for two different objects of this class. This rule is indicated immediately before the *Object\_Class* modelling, in the formal specification. As will be mentioned later, there is a *Succession* of five important attributes for recording some of the objects' information, including spatio-temporal information (time, event, attribute, value), for answering some spatio-temporal queries (state, changes of state).

The approach based on an object's state is often used in the implementation of real systems reflecting all the possible state changes of an object. The extension presented in this section is in the registration of a change of state. In such cases the event that causes the change of state is also registered. In this modelling, the attributes affected by the change of state are registered when an event causes changes in the object, where a global set of tuples with five attributes is maintained. These are: the identity attribute of an object, the event that causes the change of state (value change of one of the objects attributes), the attribute affected by the change of state, the new value of that attribute and the point in time of the event.

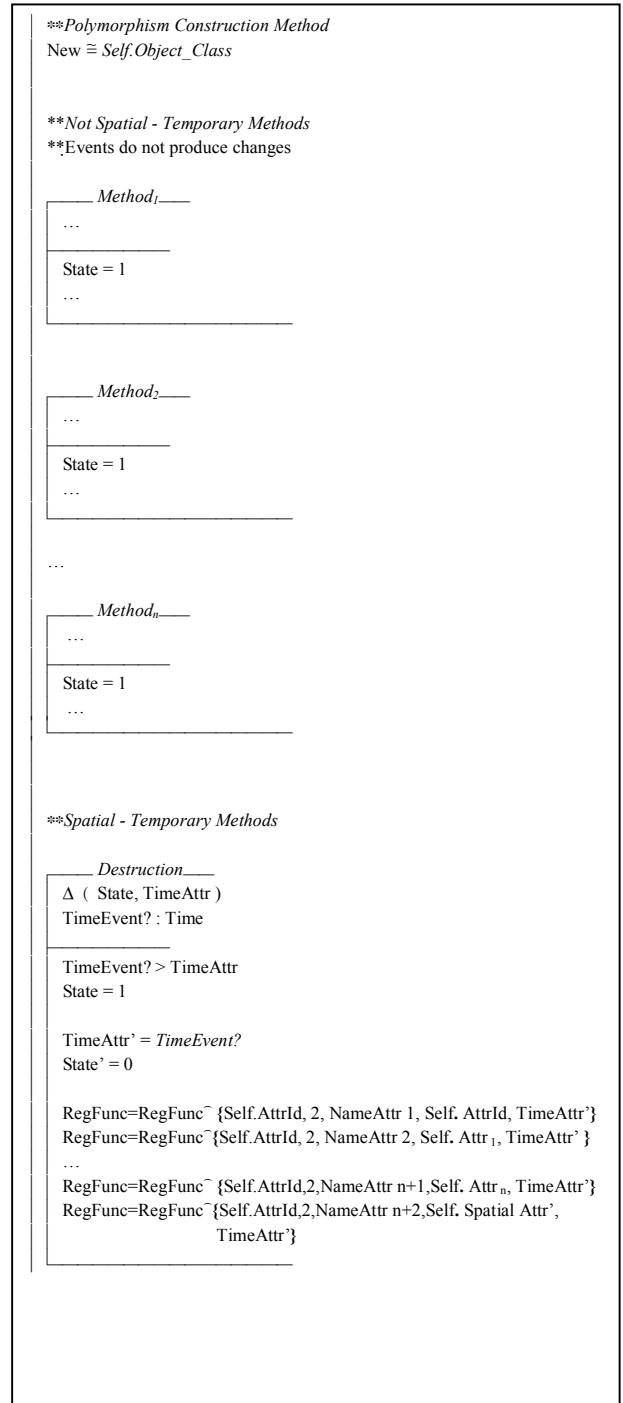
An object class *Object\_Class* exists, with "normal attributes" (non-spatial attributes), *AttrId* (identity attribute) *Attr<sub>1</sub>*, *Attr<sub>2</sub>*,..., *Attr<sub>n</sub>*, *State* (1→Active and 0→ Non Active) and some non-spatial methods *Method<sub>1</sub>*, *Method<sub>2</sub>*,..., *Method<sub>n</sub>*. The spatial attribute *SpatialAttr* of type *Spatial* and the Spatio-Temporal methods (events) presented in *Section 2.B* are considered as part of the *Object\_Class* class, too. The *TimeAttr* attribute is used as a temporal attribute and It is associated with the current time of the object. The *New*, *Delete* and *Change* methods have a direct performance only on one object (the current and/or affected object), while the *Split* and *Merge* methods work with two affected objects (one of them is the current object). The *Delete* event gives the *State* attribute the value Non Active.

The *Split* event uses an input parameter which belongs to the newly created object from the event. The *Merge* event uses an input parameter which belongs to the object that is spatially merged with the affected object. All the methods have the input variable *TimeEvent* (*TimeEvent?*) and as a precondition *TimeEvent? > TimeAttr*. Initially in the *Object\_Class* class, *TimeAttr* is equal to an initial value *TimeVal*.

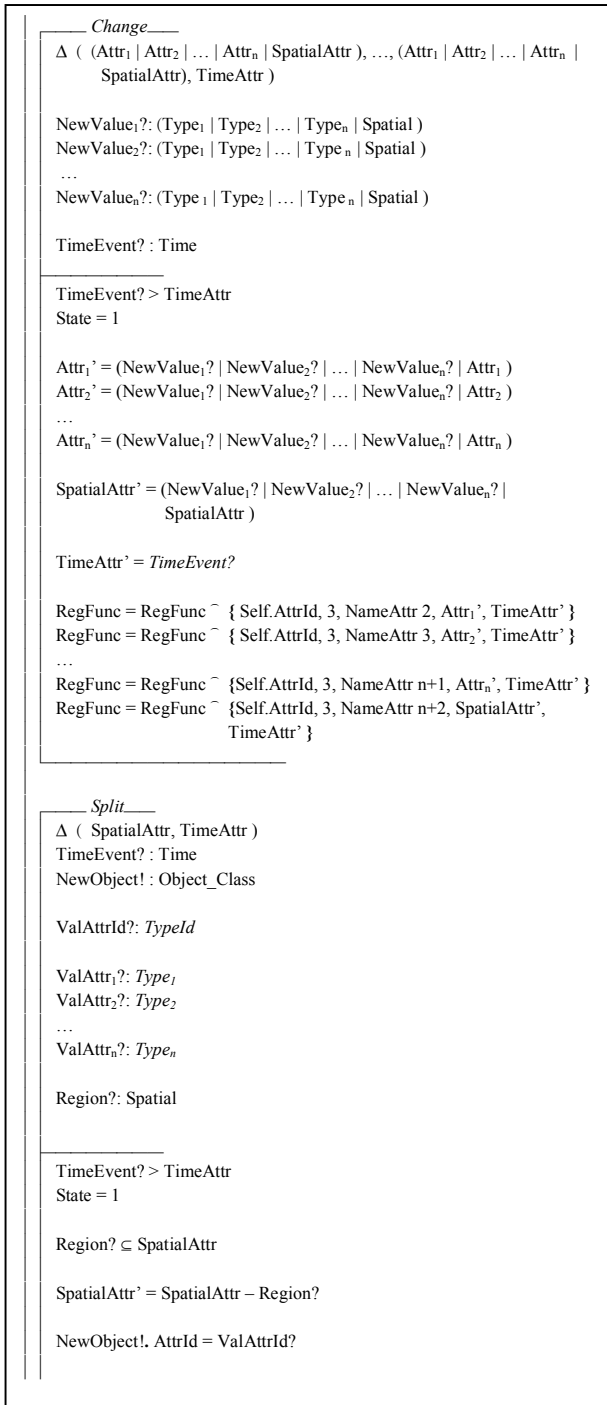
Figures 2, 3, 4, and 5 show a formal Object-Z spatio-temporal modelling using the (state+event) approach for the *Object\_Class* class. In this modelling, the five spatio-temporal events analyzed in *Section 2* are presented.



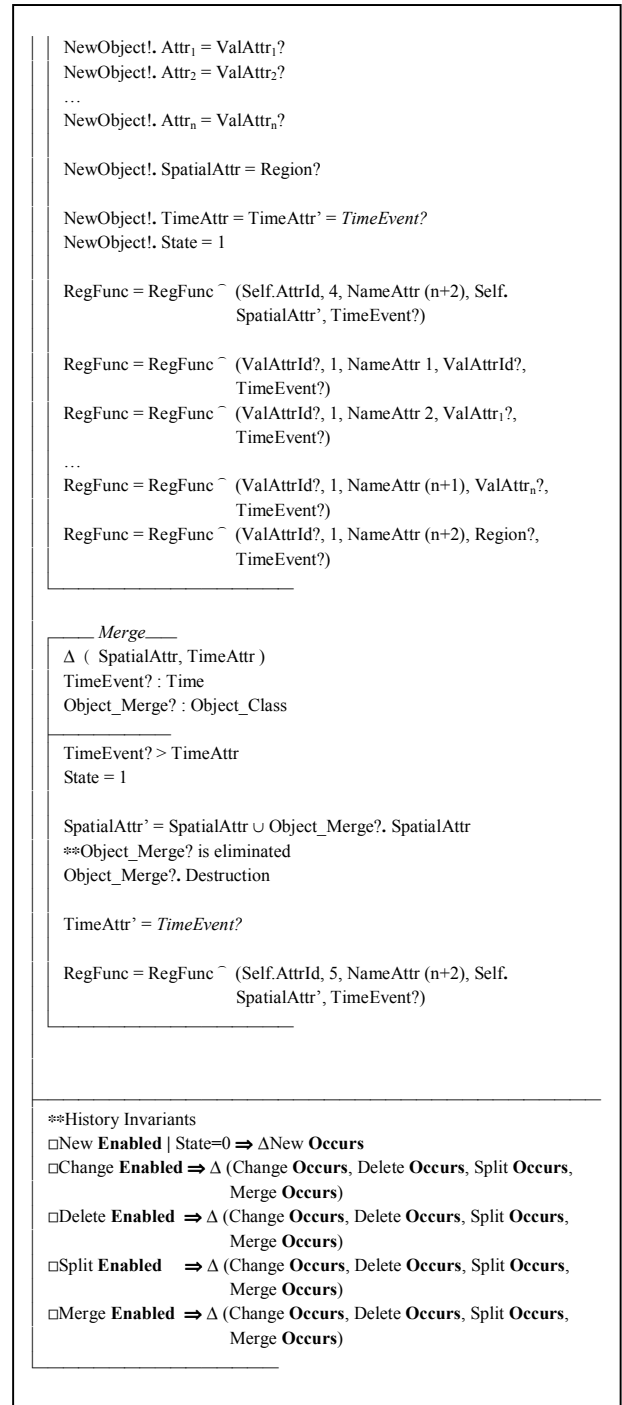
**Fig. 2.** Object-Z Object\_Class Modelling (States + Events) Approach



**Fig. 3.** Object-Z Object\_Class Modelling (States + Events) Approach (Cont...)



**Fig. 4.** Object-Z Object\_Class Modelling (States + Events) Approach



**Fig. 5.** Object-Z Object\_Class Modelling (States + Events) Approach

This modelling presents Object\_Class with its attributes, general methods (Method<sub>1</sub>, Method<sub>2</sub>, ..., and Method<sub>n</sub>) and others methods for the spatio-temporal events described in the *Section 2.B* (Destruction, Change, Merge and Split). History invariants are indicated for these spatio-temporal methods/events, with some rules for them. For example, if an object has a Non Active State value then a *New* event can occur, but not in any other case. On the other hand, if other spatio-temporal events are enabled, then any one of them could occur without restriction. With this formal modelling the semantics are detailed for each one of these events so as not to generate mistakes and not to produce false interpretations of the data.

## V. CONCLUSIONS AND FUTURE WORK

The formal modelling of information systems is most appropriate in the development of critical systems, where a high cost is involved in their development and subsequent operation, where points of failure need early recognition and rectification. Spatio-temporal information systems are an example of these high cost systems because they usually reflect massive resource bases such as municipal infrastructure information. Much of this is geographic information with associated transportation, public utilities, revenue and other human resource data. According to studies performed in software engineering, the utilization of formal techniques in software development is necessary to reduce post-implementation maintenance costs, particularly high cost or critical systems. In this, the most important and influential factor is the requirement specifications phase of a software application. In response to this reality, the paper presented here proposes a formal method for modelling spatio-temporal information systems. Object-Z was used as a formal specification language, taking advantage of object orientation for a future implementation of the specified systems. The most complex aspect of formal modelling for software systems is the mathematical notation used. However this notation is not complex in general terms, since a basic knowledge of logic and set theory, is all that is needed in a formal Object-Z specification. If greater emphasis is placed on this topic in Computer Science curriculum, more graduates would be prepared to use formal methods in the design and development of information systems, which in turn would improve the robustness of these systems and reduce the failure rate and post-implementation maintenance costs.

The specification presented here demonstrates formal modelling by using directly and indirectly the modelling approaches described in *Section 2*. The spatio-temporal snapshot-based modelling approach is the simplest of these approaches, while the state-based and (state+event)-based approaches are more complex, due to the methods that must be specified as a part of the modelled classes. As future

work, extensions to the models presented is expected in order to determine the period of existence of an object (its temporal value) and valid period of the state of the object. In future it is expected to incorporate other modelling frameworks and include some features described in other modelling approaches such as Initial-Final time marks associated to the states of the object and other spatio-temporal elements. The complete implementation of a spatio-temporal system has been considered, using the formal modelling presented in *Section 4*, and with this the ability to improve and validate the formal specification proposed in this work.

## ACKNOWLEDGMENT

To Dr. Claudio Rojas Miño, Mr. Andrés Ruiz-Tagle and Ms. Brianna Hill from FACE (Universidad de Talca) and to the people from Geoinformatics Research Centre (Auckland University of Technology), especially to Mr. Ghobakhlou, Mrs. Zandi, Mrs. Ko and Mrs. Shanmuganathan. All of them helped with this work and current ideas for future research.

## REFERENCES

- [1]. C. Vidal and A. Rodriguez. A Logical Approach for Modeling Spatio-Temporal Objects and Events. Perspectives in Conceptual Modeling, ComoGIS'05, pages 218–227. Springer-Verlag, LNCS 2005.
- [2]. N. Kesim and Marek Sergot. A logic programming framework for modeling temporal objects. IEEE Transactions on Knowledge and Data Engineering, 8(5):724–741, 1996.
- [3]. G. P. Smith. An Object-Oriented Approach to Formal Specification. PhD thesis, Dept. of Computer Science, University of Queensland, October 1992.
- [4]. Davies and J. Woodcock. Using Z. Prentice-Hall, 1999.
- [5]. P. Armstrong. Temporality in spatial databases. Proceedings: GIS/LIS'88, 2:880-889, 1988.
- [6]. M. Worboys. A unified model of spatial and temporal information. Computer Journal, 37(1):26-34, 1994.
- [7]. F. E. Petry, T. M. Schmidh and R. Ladner. The object event calculus and temporal geographic information systems. Developments in Applied Artificial Intelligence, 799-813, IEA/AIE 2003, 2003.
- [8]. C.Vidal and A. Rodriguez. Un Marco Formal para Base de Datos Espacio-Temporales Basado en Objetos y Eventos. Magíster en Ciencias de la Computación, Departamento de Informática, Universidad de Concepción, September 2007.

- [9]. Weidong Chen and David Warren. C-logic of complex objects. ACM SIGACT-SIGMOD-SIGART Symp. Principles of Database Systems, pages 369-378, 1989.
- [10]. H. Gregersen and C. S. Jensen. Temporal Entity-Relationship Models—A Survey. IEEE Transactions on Knowledge and Data Engineering, 11(3):464–497, 1999.
- [11]. Grenin and B. Smith. Snap and span: Towards dynamic spatial ontology. Journal of Spatial Cognition and Computation, 4(1):69-103, 2004.
- [12]. P. Grenin. The formal ontology of spatio-temporal reality and its formalization. AAAI, 2002.
- [13]. Hayes. Specification Case Studies. International Series in Computer Science, Prentice-Hall, 1987.
- [14]. J.M. Spivey. The Z Notation: A Reference Manual. Prentice-Hall, 1992.
- [15]. R. Duke, G. Rose, and G. Smith. Object-Z: a specification language advocated for the description of standards. *Computer Standards and Interfaces*, 17:511–533, 1995.
- [16]. B. Meyer. Object-Oriented Software Construction. Prentice-Hall, 1998.
- [17]. M. Holloway. Why Engineers Should Consider Formal Methods. NASA Langley Research Center, Octobre 1997.