

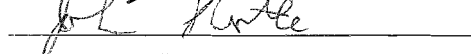
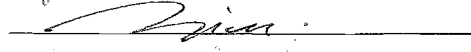
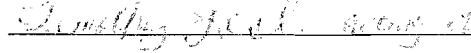

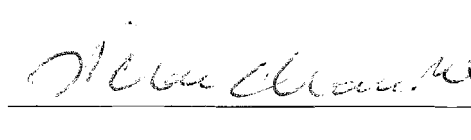


A CONSTRAINT-BASED MODEL FOR 3D SPATIAL-TEMPORAL DATA
MANAGEMENT

by

Jing Li
A Dissertation
Submitted to the
Graduate Faculty
of
George Mason University
in Partial Fulfillment of
The Requirements for the Degree
Of
Doctor of Philosophy
Earth Systems and Geoinformation Sciences

Committee:

	Dr. David Wong, Dissertation Director
	Dr. Chaowei Yang, Committee Member
	Dr. John Shortle, Committee Member
	Dr. Jyh-Ming Lien, Committee Member
	Dr. Peggy Agouris, Department Chair
	Dr. Timothy L. Born, Associate Dean for Academic and Student Affairs, College of Science
	Dr. Vikas Chandhoke, Dean, College of Science

Date: 7/13/2012 Summer Semester 2012
George Mason University
Fairfax, VA

A Constraint-Based Model for 3D Spatial-Temporal Data Management

A thesis submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University

by

Jing Li
Master of Science
George Mason University, 2009

Director: David Wong, Professor
Department of Geography and Geoinformation Science

Summer Semester 2012
George Mason University
Fairfax, VA

Copyright 2012 Jing Li
All Rights Reserved

DEDICATION

This is dedicated to my parents.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. David Wong for his mentorship during my graduate studies at George Mason University.

I would like to thank all committee members for their guidance in the dissertation.

I would like to thank Dr. Lance Sherry and his student, Akshay Belle, for providing me the data to conduct the case study in the dissertation.

I would like to thank all members of the Center for Intelligent Spatial Computing for Water/Energy Science (CISC) at George Mason University.

Finally, and most importantly, I would like to thank my parents. They have been always supporting me in my life.

TABLE OF CONTENTS

	Page
List of Tables	vii
List of Figures.....	viii
List of Abbreviations	ix
Abstract	x
Chapter One Introduction	1
Chapter Two Literature Review	9
2.1 Spatial-Temporal Data Modeling and Management	9
2.1.1 Conceptual Models.....	10
2.1.2 Logical Implementations	18
2.2 The Constraint-Based Approach for Data Modeling and Management	24
2.2.1 Background.....	24
2.2.2 Constraints in spatial sciences	26
Chapter Three A Conceptual Model for Managing 3D Spatial-Temporal Data	31
3.1 An Extended Object-Oriented (OO) Model for Spatial-Temporal Objects	31
3.2 The Constraint-Based Approach	38
3.2.1 Types of Constraints	40
3.2.2 Formalization of Constraints	47
3.2.3 Enforcement of Constraints	49
3.2.4 Apply constraints in spatial-temporal queries	51
Chapter Four A Prototype System to Handle 3D Spatial-Temporal Objects	56
4.1 Conceptualization of a system for managing 3D spatial-temporal objects	56
4.1.1 General Requirements.....	56
4.1.2 Constraints for managing 3D Data	57
4.1.3 Constraints for Spatial-Temporal Querying	58
4.1.4 Formulation of 3D spatial-temporal queries.....	60

4.1.5 Enforcing constraints with spatial-temporal geometric computational algorithms.....	64
4.2 Design and Implement a 3D Spatial-Temporal Prototype System	75
4.2.1 Data Format	76
4.2.2 Core Modules	77
4.2.3 Development Platform and Related Issues	84
Chapter Five A Customized System Supporting DAC	85
5.1 Background.....	85
5.2 Contextualize the Conceptual Model for DAC	89
5.2.1 Modeling the 3D Spatial-temporal objects	89
5.2.2 Describing the Constraints.....	91
5.3 Customizing the Prototype System to support DAC	94
5.3.1 Data	95
5.3.2 Customizing Geometric Algorithms	96
5.3.3 Calculating Workloads.....	101
5.3.4 Reconfiguration of Airspace Sectors	102
5.3.5 System Demonstration	104
Chapter Six Conclusion and Discussion	109
6.1 Summary.....	109
6.2 Contributions	112
6.3 Future Work.....	113
References.....	116

LIST OF TABLES

Table	Page
Table 1 Comparisons of conceptual spatial-temporal models: Part I (“Partial” indicates partial support)	16
Table 2 Comparisons of conceptual spatial-temporal models: Part II (“Partial” indicates partial support)	17
Table 3 Comparisons of GIS systems with spatial-temporal or 3D capabilities.....	22
Table 4 Classification of spatial-temporal objects	32
Table 5 Spatial-temporal test cases and corresponding constraint formalization	55
Table 6 Types of Spatial-temporal queries supported by the system (*: Valid after specifying time period)	61
Table 7 Selected 3D topological relations (*: Planar polygon).....	65
Table 8 Selected spatial-temporal topological relations	66
Table 9 Selected spatial-temporal objects and possible swept objects (*: Assuming linear movement, no rotations)	68

LIST OF FIGURES

Figure	Page
Figure 1 Object hierarchy of the extended 3D spatial-temporal OO model.....	38
Figure 2 A systematic view of constraints.....	50
Figure 3 Relations between constraints with objects and constraints specified by a query (O: constraints associated with objects; Q: constraints specified by a query)	54
Figure 4 The process of formulating spatial-temporal queries (a) and the graphic user interfaces supporting the formulation process (b).....	63
Figure 5 A 3-step identification process of spatial-temporal relations between two moving points	69
Figure 6 Identifying the spatial-temporal relations between moving objects and objects changing spatial properties at discrete time points	74
Figure 7 Identifying the spatial-temporal relations between two objects changing at discrete time points	75
Figure 8 An overview of the system architecture	78
Figure 9 Space-time trajectories generated from different spatial-temporal objects.....	79
Figure 10 Constraint Management Interface a) Constraint Editor; b) Constraint Graph ..	83
Figure 11 Three constraints enforced when changing the boundaries of sectors.....	93
Figure 12 A simplified UML diagram of objects in DAC ("*" denotes the 1-N condition)	94
Figure 13 Snapshot of airspace sectors and flights in the 3D default view (a) view from above; b) oblique view from the upper southwest corner; c) side view from the north) ...	96
Figure 14 Calculating the levels of workloads	98
Figure 15 Enforce the non self-intersection constraint	99
Figure 16 Change the shared boundary with polygon clipping.....	100
Figure 17 Enforce no intersection constraint between sectors	101
Figure 18 Workload profiles for selected sectors generated from the system	102
Figure 19 A simulated DAC process in the system (WL=workload)	104
Figure 20 A report of enforcing workload constraints before reconfiguration	105
Figure 21 Snapshots of sectors (a) before and (b) after reconfiguration	106
Figure 22 A report of verifying workload constraint after reconfiguration	108
Figure 23 Workload profile for "Sector5" after reconfiguration.....	108

LIST OF ABBREVIATIONS

Air Traffic Control	ATC
Air Traffic Management	ATM
Axis-Aligned Bounding Boxes	AABB
Computer Aided Design	CAD
Constraint Query Language	CQL
Constraint Satisfaction Process	CSP
Department of Transportation	DOT
Dynamic Airspace Configuration	DAC
Entity-Relationship	ER
Extensible Markup Language	XML
Federal Aviation Administration	FAA
Flight Level	FL
Future ATM Concepts Evaluation Tool	FACET
Geographic Information System	GIS
High Occupancy Vehicle	HOV
Instrument Flight Rules	IFR
Minimum Distance Separation	MDS
National Airspace System	NAS
Nautical Mile	NM
Next Generation Air Transportation System	NextGen
Object Constraint Language	OCL
Object-Oriented	OO
Object-Relationship	OR
Region Connection Calculus	RCC
Spatial-Temporal	ST
Standard Query Language	SQL
Unified Modeling Language	UML
Virtual Geographic Environment	VGE
Virtual Reality	VR

ABSTRACT

A CONSTRAINT-BASED MODEL FOR 3D SPATIAL-TEMPORAL DATA MANAGEMENT

Jing Li, Ph.D.

George Mason University, 2012

Dissertation Director: Dr. David Wong

Describing structures of geospatial objects as models is essential for understanding geographic processes. Efforts to develop such models started from decades ago but a model for 3D spatial-temporal (3D space plus 1D time) objects has not been well formulated. This dissertation describes the formalization of a spatial-temporal data model for managing 3D spatial-temporal objects. This model extends the spatial-temporal Object-Oriented model by incorporating a behavioral description to the definition of the spatial-temporal objects to better characterize the dynamics of these objects. Besides, spatial-temporal rules and conditions, which are expressed as constraints, are integrated as important components of the model. These rules and conditions serve as the foundations of maintaining data integrity and enabling complex spatial-temporal queries. A set of constraints related to the spatial-temporal characteristics of 3D spatial-temporal objects were identified and defined.

The conceptual model served as the theoretical basis towards the building of a Geographic Information System (GIS) for managing 3D spatial-temporal objects. Based on the conceptual model, a prototype system was developed that provides interactive data management and query functions for 3D spatial-temporal objects. A subset of spatial-temporal constraints identified in the conceptual model were captured and formalized through extended 3D computational geometry algorithms to ensure data integrity and facilitate spatial-temporal queries. Using the dynamic repartitioning of airspace sectors as a case study, this research shows that the proposed framework is effective to solve problems involving 3D spatial-temporal objects.

CHAPTER ONE INTRODUCTION

Describing the spatial structure, arrangement and relationships of geospatial objects as models is essential to understand the driving factors of various geospatial processes, to simulate the interactions among objects over time and space, and to discover the patterns of movements in human and environmental systems (Yuan and Hornsby, 2008). The earliest effort in research field starts with modeling 2D geospatial objects, assuming a static nature for them, such as representing county boundaries by polygons. Since late 1980s, spatial-temporal models that incorporate temporal information have been proposed to support the exploration of the geographic processes. After several decades of developments, scientists have proposed a variety of spatial-temporal models, most of which have been implemented for managing 2D spatial-temporal (2D space plus 1D time) objects (Pelekis et al., 2004; Abraham and Roddick 1999).

However, many objects and phenomena have explicit height dimension, such as a flying aircraft or a dust storm. These 3D objects and phenomena may be represented adequately in 2D for a temporal snap-shot when the elevation of the objects or phenomena is fixed. When the temporal dimension is taken into account, elevations of many 3D objects and phenomena do change over time. For example, chemical dispersion or atmospheric transport models produce 3D spatial-temporal (3D space plus 1D time)

data with concentration levels vary by elevation and time, but often only the footprints of plumes were captured and used in GIS for impact assessments (e.g., Chakraborty and Armstrong, 1996). When depicting the phenomena, many conceptual modeling approaches tend to simplify the modeling process by eliminating the elevation dimension (Yuan et al., 2003). This simplification reduces the complexity of the phenomena but discards the valuable or even critical information about the elevation dimensions. As a result, the accuracy of analysis in later time is not guaranteed. In managing 3D spatial-temporal objects, ignoring the third dimension or compressing the height dimension to 2D may lead to erroneous conclusions.

On the other hand, as 3D spatial-temporal datasets become more accessible, geospatial tools for 3D spatial-temporal datasets are needed to support the management and analysis of these datasets. For example, in estimating the damages caused by a hurricane, scientists analyze massive amount of data from multiple sources, some of which are 3D spatial-temporal data such as wind speeds at different elevation levels. Although a few packages and software have been developed to support the analysis of spatial-temporal problems (e.g. Shaw and Yu, 2009), most of these tools only provide interfaces to handle 2D spatial-temporal objects mainly because they are built upon 2D spatial-temporal data models. While existing 3D geovisualization platforms such as virtual globes offer reasonably strong rendering capability for 3D data, including volumetric data, they are short of providing efficient data management and query-analytical functions. Existing spatial-temporal data management systems were not designed to accommodate 3D spatial-temporal objects. Scientists lack effective tools to

facilitate the analysis on these 3D spatial-temporal objects. The gap between the requirements for 3D spatial-temporal analysis and the availability of effective models and corresponding tools for 3D spatial-temporal objects has to be addressed appropriately.

Therefore, the primary objective of this research is to propose a data model to facilitate the management of 3D spatial-temporal objects superior to the existing modeling approaches in describing such objects. The proposed model is designed to provide reasonable and formal descriptions of 3D spatial-temporal objects to capture their spatial characteristics, temporal properties, dynamic behaviors and attributes. Besides providing object descriptions, this model integrates spatial-temporal rules and constraints, which facilitate data management and support preliminary analyses on 3D spatial-temporal data.

Geospatial data for spatial-temporal objects or phenomena should meet relevant constraints. In the processing of digitizing county boundaries with polygons, the underlying topological assumption in using polygons to depict county boundaries is that polygons cannot overlap because county boundaries cannot cross each other in real world. If non-overlapping condition is not satisfied, a topological error will occur during later operations and analysis. Such non-overlapping condition may be verified as a part of the data integrity in a spatial database where the data are stored (e.g. ArcInfo arc-node data structure).

When extending 2D modeling to 3D domain such as in the development of a Virtual Geographic Environment (VGE, Lin et al., 2009), where 3D objects are involved, certain topological integrity should also be considered. In a VGE, users should generate,

modify and manipulate 2D and 3D geospatial objects such as buildings, roads, trees and facilities according to certain rules. For example, a bridge cannot be placed through a building under a normal condition. Trees can be planted along the roads rather than on the road. These rules are examples of constraints pertaining to the spatial properties, in this case, the positions and relations to objects. If these rules are not followed in the environment, these objects may not be constructed properly and the accuracy of the system is not guaranteed. The enforcement of these constraints can detect objects violating integrity constraints within the system to ensure the accuracy and correctness of data.

The requirements for 2D and 3D spaces can also be extended to the temporal dimension. When both time and space are considered, spatial objects undergo some spatial-temporal processes. Such dynamics are essentially movements or evolutions of objects. Due to the additional temporal dimension, the conditions should be enforced for both the spatial and temporal dimensions. The constraint “Minimum Separation Distance” (MSD) in air traffic control requires all aircrafts to maintain a safety distance among each other to avoid potential crashes. In order to enforce such MSD constraint, the distance between any pair of aircrafts in 3D space is frequently monitored and compared with the MSD to ensure that such safety control is achieved. These examples, from the 2D static to 4D dynamic situations, all illustrate the importance of constraints, both spatial and temporal, in a modeling process but their importance has not been fully investigated in previous modeling efforts.

This research exploits two usages of constraints in enhancing the capabilities of the spatial-temporal model for 3D spatial-temporal objects. On the one hand, constraints can be used to describe rules imposed on 3D spatial-temporal data to maintain data integrity. By enforcing constraints, 3D spatial-temporal objects violating data integrity can be identified. On the other hand, considering a query as a process of finding objects meeting a set of conditions, constraints support complex spatial-temporal queries. Query criteria can be expressed as constraints and processed using constraint solving techniques (Kanellakis et al., 1992).

Several concepts formulated in the conceptual model provide foundations of describing 3D spatial-temporal objects and managing data representing these objects. These concepts serve as the theoretical basis towards the building of a GIS for 3D spatial-temporal data. Based on the conceptual model, I developed a prototype system that provides interactive data management and query functions for 3D spatial-temporal objects. This system supports essential data management and query-analytical tasks on 3D spatial-temporal data. Basic functions include 3D rendering and animation, manipulating 3D objects interactively and issuing spatial-temporal queries through a visual-graphics interface. A set of spatial-temporal rules and conditions as constraints were implemented to maintain data integrity and support spatial-temporal queries. If constraints related to data integrity are imposed during an interactive editing session, modifications of data representing 3D spatial-temporal objects cannot violate any rules pertaining to data integrity. Conditions specified by queries can be recorded as constraints. Solving a query then becomes a constraint satisfaction process (CSP) that

identifies objects or conditions meeting the constraint specifications. To implement these constraints, this research has extended existing 3D computational geometry algorithms to deal with both geometric and temporal relationships between 3D spatial-temporal objects.

To demonstrate the usages of the proposed concepts and the prototype system, the interactive data management prototype system was customized to handle the Dynamic Airspace Configuration (DAC). DAC is a process of repartitioning airspace adaptable to changes in traffic demands, weather and other factors to achieve a demand-capacity balance (Kopardekar et al., 2007). The key strategy is to accommodate new constraints dynamically using a constraint-based approach with spatial-temporal components to represent an airspace system (Lee et al., 2008). A typical DAC task is to eliminate the excessive controller workload of an airspace or combine two airspaces with low workload when traffic demand levels change (i.e., the number of flights as 3D objects inside a sector as 3D prism). In this process, controllers should be able to identify the workload of each sector and make adjustments of the sector boundaries accordingly. This process involves enforcing different categories of constraints on demand. Various approaches have been proposed to divide airspace into sectors given the status of workloads (e.g. Yousefi and Donohue 2004). While these approaches can produce optimal configurations of sectors automatically, a few reconfigured sectors may be significantly different from the initial spatial settings of sectors. Such differences increase the difficulties of airspace controlling after reconfiguration. Manual adjustments are necessary in this case. Besides, most approaches were implemented in a 2D spatial framework without considering the vertical hierarchy of sectors in a 3D setting. Since the

repartitioning of airspace sectors involves manipulations on 3D spatial-temporal objects under various constraints, a 3D spatial-temporal system that can handle multiple constraints is necessary. Through providing visualization, manipulation, querying and constraint solving functions, the interactive data management prototype system developed in this research can be used to identify the overloaded sectors, changing the boundaries of sectors interactively and verify the accuracy of reconfiguration instantly.

This research contributes to the spatial-temporal modeling in the following aspects. First, a conceptual model is formalized for 3D spatial-temporal objects to enhance the capacities of existing spatial-temporal models in depicting 3D spatial-temporal objects. Second, a constraint-based approach is provided to facilitate spatial-temporal data integrity checking and querying of 3D spatial-temporal objects. Third, the prototype system fills the gap between conceptual models and logical implementations of 3D spatial-temporal data management systems by offering several fundamental modules essential to the management of 3D spatial-temporal data.

The rest of the dissertation proceeds as follows. In Chapter 2, I review the current spatial-temporal models and GIS, and discuss some major weaknesses in existing models and systems. An overview of concept and usages of constraints in data modeling is given in this section. Chapter 3, I present the formulation of a conceptual data model using an object-oriented approach and the integration of spatial-temporal constraints with the data model. In Chapter 4, the developments of the prototype system are described in details, particularly in terms of the use of constraints in handling 3D spatial-temporal data and how 3D computational geometry algorithms can be extended to support spatial-temporal

queries. In Chapter 5, I demonstrate how the conceptual model and the prototype system can be customized to facilitate the repartitioning of airspace sectors. Conclusions and ideas for future work are discussed in Chapter 6.

CHAPTER TWO LITERATURE REVIEW

2.1 Spatial-Temporal Data Modeling and Management

The development of effective and expressive models for spatial-temporal data is an important topic in spatial-temporal research (Pelekis et al., 2004). Spatial-temporal modeling is the process of creating a data model by applying formal descriptions to represent spatial-temporal data (Parent et al., 1999, Longley et al., 2010). Major components of such data models are data structures, rules and relations between spatial-temporal objects (Steiner, 1998). After several decades of development effort, a variety of spatial-temporal data models have been proposed by scientists from various fields. Most of these models were at the conceptual level to provide formal descriptions of spatial properties, temporal attributes and relations of these objects. In addition, some of these conceptual models have been implemented in GIS or packages supporting geospatial data management, analysis and explorations. These spatial-temporal GIS usually offer the capabilities to view, represent, manage and analyze spatial-temporal objects. By reviewing these conceptual models and their logical implementations, I can identify the current status and major deficiencies in theories and techniques in handling spatial-temporal data. In this section, I first introduce popular conceptual spatial-temporal models and then discuss the developments of existing GIS tools related to spatial-temporal data.

2.1.1 Conceptual Models

Conceptual data modeling is a process of abstracting objects from reality through defining the representations and important properties of objects and capturing possible relationships between objects (Batra and Maraks, 1995; Tryfona and Jensen, 1999). Since conceptual models provide a high level of abstracted information of objects and phenomena, these models should have a certain level of reusability for different application scenarios and should be independent of implementation platforms.

Conceptual modeling is considered as the first step towards constructing a data model.

Efforts in developing conceptual models of spatial-temporal data can be dated back to late 1980s when spatial data models were first extended from describing 2D static objects to depicting 2D spatial-temporal objects. Some earlier approaches focused on capturing the spatial and temporal information of the spatial-temporal objects. As one of the earliest spatial-temporal models, the snap-shot model deals with how spatial-temporal objects change their properties at different time points (Langran and Chrisman, 1988).

Spatial-temporal objects are grouped by different time-stamped layers. Elements on the same layer represent states of objects within the same temporal frame. A spatial-temporal object may be stored in different temporal layers representing its entire lifespan.

However, the connections between these objects in different temporal layers are not included in the model. As a result, such model fails to support queries for spatial relationships between objects over time. In addition, a change in any object in a layer will create a new time-stamped layer, but data of unchanged objects are duplicated in the new layer. When the model is used to describe a mixed set of spatial-temporal objects, consisting of some objects with changing properties and others remain unchanged, the

new layer will include redundant data, another major drawback of the model (Yuan, 1996).

A few models were developed to capture changes of spatial-temporal objects. One such model is the event-based model (e.g., Peuquet and Duan, 1995; Worboys, 2005; Hornsby and Cole, 2007). Instead of only recording the positions of spatial-temporal objects at different time points, such model records events associated with spatial-temporal objects to facilitate the analysis of changing patterns of objects overtime. How an object changes is a function of time. A time-stamped layer is used to store changes of objects or events happened within a specific time frame, and multiple time-stamped layers are used to record events or objects changes over multiple periods. An event is described by two components: a component descriptor storing the new value of a property after a change and the spatial location, which is a group of spatial coordinates depicting the boundaries of the geographic region within which the event occurred. A new event appears when a significant change of an object is detected. As time progresses, new events can be added to the layer one by one to show the entire process of a spatial-temporal object. This model supports performing temporal oriented analysis such as handling temporal based queries, retrieving historical spatial and attribute information and estimating the changes over time, but is inefficient in terms of evaluating topological relations within a temporal cross-section.

Similar to the event-based model, the moving object model also captures changes of objects, though, in an abstract manner (Erwig et al., 1999). The positions of objects are recorded in the moving object model whereas other properties such as shapes are

abandoned. Besides recording the positions of objects over time, such model describes the behaviors of a spatial-temporal object through tracking its trajectory. As a result, this model may offer a solution to spatial-temporal behavioral queries to examine the status of a spatial-temporal object during its lifespan (Hornsby and King, 2008). In a moving object model, an abstract data type is defined to represent objects (e.g., moving points for flight trajectories) and only two geometric primitives, moving points and moving regions, are modeled (Güting et al., 2000). The moving volumetric objects are not included in the model, so it is not sufficient to describe general 3D spatial-temporal objects (Pelekis et al., 2004). Another major deficiency is that the moving object model does not distinguish two objects of identical trajectories whereas these objects have to be differentiated by their properties in certain situations. For example, the driver and truck are recognized as the same object in a moving object model as their movements are identical. Such modeling approach is not applicable in the cases that require identifying the shapes of objects.

Focusing on the interactions and relations between objects, scientists extend the Entity-Relationship (ER) and Object-Relationship (OR) models by including spatial and temporal entities (Tryfona and Jensen, 1999; Parent et al., 1999). The relations, to some extent, reflect the processes and changes associated with objects or caused by interactions between objects. For example, a spatial-temporal object A split into two objects B and C at time t_l . The change “split” is the relationship between original object A and the two new objects. Given different scenarios, the relationships may include spatial relations specifying the spatial settings, changes between objects of different statuses, and

temporal relations within a specific time frame. Although these models provide descriptions of processes involving the interactions between multiple objects, none of these models define data types and operations required by logical implementations.

Based upon concepts of Object-Oriented (OO) modeling in software engineering, a spatial-temporal OO model utilizes the OO concepts of classes, attributes, properties and methods to build OO framework for spatial-temporal objects. The spatial-temporal properties are the attributes of objects and the changes are considered as methods. With such a model, properties and relations of a spatial-temporal object can be described regardless of the complexity of the spatial or temporal properties. The major weakness of the OO approach is that the description of object behaviors, which refer to the transformations, movements or changes in attributes of objects over time, is not clear and precise (Twumasi 2002; Pelekis et al., 2004). In most OO modeling practices, versioned objects are created to describe spatial-temporal objects. Defined by the triplet of (time, shape, location), each versioned object is a temporal object with spatial attributes defined by the triplet of (time, shape, location) depicting the status of a spatial-temporal object at a time point or time period. A spatial-temporal object usually is made up a series of objects sorted by the temporal dimension.

As the behavioral description of a spatial-temporal object is not included in the triplet, changes in behaviors between two versioned objects are unknown. Knowing the changes is critical in supporting behavioral queries and analyses. For example, to identify the motion path between two versioned objects, scientists need to examine movement behaviors such as motion status and directions of these two versioned objects. However,

such information is not captured by the versioned object. As a result, using versioned objects to represent spatial-temporal objects fails to support any queries of spatial-temporal behaviors. While a few OO modeling approaches are able to provide explicit descriptions of the states of objects, only a few categories of behaviors and events associated with objects can be captured (e.g. exist or not exist, Hornsby and Egenhofer, 2000). A comprehensive description of object behaviors should be integrated into the OO model.

A hybrid model that combines the objects with events, called the GEM model, has been proposed to describe the processes and events associated with spatial-temporal objects (Worboys and Hornsby 2004). Such model defines “object” and “event” as two data structures and links these two structures through the concept of “object participation”. When an event occurs, objects associated with the event will be recorded as two fields of the event structure. The changes of an object over time can be obtained through examining the events that the object has participated. However, since the model has not defined a concrete standard for “new event”, critical processes may not be captured.

Despite various modeling approaches to describe and represent spatial-temporal objects, Goodchild et al. (2007) proposed a more general representation of spatial-temporal objects with four concepts: “geo-atoms”, “geo-objects”, “geo-fields” and “geo-dipoles”. A geo-atom is expressed as a tuple of (point in space-time, a property, value of the property). A geo-field summarizes the changes of a property over a set of geo-atoms. If the geo-atoms meet certain criteria, for example, the values of temperature of geo-

atoms from stratosphere falling into a certain range, they may be aggregated to form a geo-object. Upon the formation of a geo-object, geo-dipoles are created to record the relations between the values of the property of any pair of geo-atoms. Through capturing the spatial, temporal, relations and changes from an atomic level, the four data structures serve as a general form of depicting geographical dynamics. While Goodchild et al. (2007) has demonstrated the applicability of using the four conceptual abstractions to represent spatial-temporal point sets, the practical usages for more complex spatial-temporal objects especially for 3D moving volumetric objects have not been evaluated systematically. The “point in space-time” in the tuple of a geo-atom is not sufficient to represent the geometric shape of complex 3D spatial-temporal objects.

As the conceptual models were defined and formulated through different modeling approaches, their capabilities of depicting objects for advanced analysis vary significantly. A relatively comprehensive evaluation on spatial-temporal models has been conducted by Pelekis et al. (2004). I further extended the evaluation by adding two more models described the literatures published after 2004 (Table 1; Table 1Table 2). Still, none of the models discussed above is comprehensive enough to manage 3D spatial-temporal objects. The major deficiencies of these models in respect to 3D spatial-temporal objects can be summarized into three categories. First, the properties, behaviors of and relations between spatial-temporal objects cannot be captured comprehensively in any of these models. Models using time-stamped layers (e.g. the snapshot model) cannot handle continuous temporal data. Second, some models are not capable of handling 3D spatial-temporal objects. As these models treat time rather than the height as the third

dimension, they are not able to describe the third spatial dimension of 3D spatial-temporal objects (e.g. Tryfona and Jensen, 1999). Third, querying capabilities of most models are limited to a few basic types of queries. For example, OO models originally designed for 2D cases may be applied to model 3D objects, but they are not comprehensive enough to support queries of object behaviors because these models do not provide clear and precise behavioral descriptions such as transformations of objects over time. Additional efforts should be put into extend these models to support the analysis of 3D spatial-temporal data.

Table 1 Comparisons of conceptual spatial-temporal models: Part I (“Partial” indicates partial support)

Model	Spatial			Temporal	
	Spatial dimensions	Property	Relation	Discrete	Continuous
Snapshot model	3D	Yes	No	Yes	No
Event-based Object	3D	Partial	No	Yes	No
Moving-object model	2D	Partial	Partial	Yes	Yes
Entity-Relation	2D	Yes	Yes	Yes	No
Object-Relation	3D	Yes	Yes	Yes	Yes
Object-Oriented	3D	Yes	Yes	Yes	Yes
Object-Event	3D	Yes	Yes	Yes	Yes
Geo-Objects	3D	Yes	Yes	Yes	Yes

Table 2 Comparisons of conceptual spatial-temporal models: Part II (“Partial” indicates partial support)

Model	Spatial-temporal properties			Data Structures	Support Data Integrity
	Thematic attributes	Spatial Relations	Processes		
Snapshot model	No	No	No	Time stamped layer	No
Event-based Object	Yes	No	No	Event; Raster	No
Moving-object model	No	Partial	Yes	Moving points/regions	No
Entity-Relation	Yes	No	Partial	Entity constructs	Yes
Object-Relation	Yes	No	Partial	Object	Yes
Object-Oriented	Yes	Yes	Partial	Object	Yes
Object-Event	Yes	Yes	Partial	Object/Event	No
Geo-Objects	Yes	Yes	Yes	Geo-objects	No

Therefore, existing spatial-temporal models have to be extended in the following aspects to support the management of 3D spatial-temporal data. First, data structures should include different types of 3D spatial-temporal objects ranging from moving points to moving volumetric objects. Most models cannot efficiently handle complex objects that can be described accurately only using multiple geometric data types (Lohfink et al., 2010). Second, the model should provide complete and accurate descriptions of the

processes and changes associated with objects. These changes and processes are also considered as important spatial-temporal properties of objects (Pelekis et al., 2004). Third, such model should define a mechanism to maintain data integrity when the spatial properties, temporal information or attributes of objects are manipulated and modified frequently. Without imposed any data integrity constraints, these operations may introduce errors so that data quality is compromised. Fourth, an efficient 3D spatial-temporal data model should provide interfaces to facilitate the explanation, synthesis, presentation and analysis of processes (Ellul and Haklay, 2006). Designing a model to meet all three requirements is extremely challenging when various 3D spatial-temporal objects have to be considered.

One of the primary objectives of this research is to design a conceptual data model to improve the description and management of 3D spatial-temporal objects. The proposed data model should be able to capture spatial-temporal properties, relations and attributes of 3D spatial-temporal objects. It should define a mechanism to maintain data integrity at the modeling stage, ensuring the consistency, completeness and accuracy. Meanwhile, it should offer reasonable support for different types of spatial-temporal queries, serving as the basis of advanced analyses.

2.1.2 Logical Implementations

Compared to the relative abundant conceptual work on model formalization, only a few logical implementations as GIS systems have been developed. While most spatial-temporal models reviewed in the previous section are conceptual, they do serve as the foundations, for example, object representations and data organizations, in subsequent

logical implementations in 2D cases (e.g. Goodchild et al., 2007; Pultar, 2010; Shaw and Yu, 2009). Some of the earlier research on spatial-temporal GIS started in the late of 1980s, including how the objects and their variations are described at different time points (e.g. Peuquet 1994; Yuan 1996). How an object changes over space and time, and the spatial-temporal relations among objects are on-going research topics. However, in most of these discussions, spatial objects were limited to 2D, and thus time was treated as the third dimension. In other words, results are applicable only to 2D, but not to 3D spatial-temporal objects.

Recently, analytical functions were included in some spatial-temporal systems. They show the moving trajectories of spatial-temporal objects based on the moving object models. One of the successful implementations is the Space-Time extension for ArcGIS developed by Shaw and Yu (2009). By connecting sequential locations over time with line segments, the system creates a trajectory of a moving object. Time is specified as the third dimension of an object, and the trajectory of an object is converted to a series of 3D line segments within the space-time prism - an implementation of the Time Geography concept (Hägerstrand 1970). A horizontal slice may be inserted cutting across object trajectories to identify the locations of objects at that time. By comparing the trajectories of multiple objects, hypotheses about the relations between these moving objects may be formulated. Moving trajectory is a straightforward way to show the changes of locations overtime, especially when elevation is not important, for instance, in describing certain types of human activities.

When approaches for handling 2D spatial-temporal objects are used to explore 3D dynamics, they simplify the modeling process by eliminating the vertical dimension in the analysis (Abdul-Rahman and Pilouk 2008). These 2D data handling methods may fail to provide accurate analytical results when the elevation information is crucial in the analysis. For example, vehicles inside a garage are represented by points in a 2D system environment. If two points are too close such that one is "on the top" of the other, the situation cannot be resolved easily in the 2D setting. One may interpret that the two vehicles collided, or one sits on the other, but in fact, the two vehicles are on different floors. More confusing situations will arise if data include a temporal dimension tracking the movement of these vehicles inside the garage. Therefore, these inherent 2D systems are not sufficient to support the analysis of 3D spatial-temporal objects, and developing 3D temporal GIS technology is necessary.

Several 3D GIS have been developed for specific applications. For example, Kwan and Lee (2005) developed a network-based 3D GIS to facilitate emergency responses. In this system, 3D networks were constructed to model the path structure of multi-level buildings. Using the building internal network structure, shortest paths for evacuations within the building can be computed based on the static path information. Temporal variations, for example, the accessibility of stairs can vary over time, are not taken into account. Besides the specialized ones, several generic 3D GIS have been developed to manage the spatial properties of 3D static objects such as shape, dimensions, and positions in 3D space (e.g. Lee and Zlatanova 2009; Ravada et al., 2009). These 3D GIS with graphic interfaces can often display the geometric properties

of 3D objects in a virtual environment (Verbree et al., 2000; Shi et al., 2003; Lin, 2009). In addition, these systems also provide multiple visual interfaces and widgets to facilitate interactive manipulations of the 3D objects (Schmidt et al., 2008). When these systems are integrated with spatial databases, they support basic 3D spatial queries such as determining the intersections between two 3D objects (Gröger et al., 2004; Chen et al., 2007). Unlike the 2D approaches that squash the third dimension of 3D objects, 3D GIS can accommodate information of three spatial dimensions and provide specific functions to support the 3D framework but the temporal dimension of spatial-temporal objects are not considered in typical 3D GIS.

While a limited number of systems have the capabilities to manipulate spatial-temporal objects and to handle spatial-temporal data, these systems have some deficiencies. Properties of 3D spatial-temporal features or objects are not well captured because the height dimension is usually eliminated in some of these systems (e.g. Shaw and Yu, 2009). As a result, errors may be introduced in data manipulation and analyses. Besides failing to describe the geometric characteristics of objects comprehensively, these systems provide very limited capabilities to handle spatial-temporal relationships among 3D spatial-temporal objects, particularly on evaluating spatial and temporal topologies to support 3D geometric queries. Identifying these relationships through queries is considered to be the early stage of analysis toward more advanced spatial-topological analyses, as queries are foundations of all spatial and aspatial analytical functions (Breunig and Zlatanova, 2011; Shekhar and Chawla, 2003; Ellul and Haklay, 2006; Nocera et al., 2009). Most existing 3D GIS also do not have a mechanism to verify

the integrity of spatial-temporal data, which refers to the consistency, validity, completeness and accuracy of data. These criteria for data integrity can often be expressed as a set of rules. Without such a safeguard mechanism, data quality may be undermined or in jeopardy when data undergo editing or manipulations. Spatial and/or temporal conflicts may emerge during the data manipulation processes, and they have to be reconciled before data are committed to permanent changes. These are some of the challenges in handling 3D spatial-temporal data in GIS to be addressed.

Table 3 Comparisons of GIS systems with spatial-temporal or 3D capabilities

Systems	2D Spatial-Temporal Systems	3D Database	3D Spatial-temporal systems	My system prototype
Examples	ArcGIS Space-Time Extension	Oracle Spatial	GeoDec	
Object types	2D moving points	3D Geometries	Simple 3D spatial-temporal objects	3D Moving Objects
Space-Time Representation	Time as third dimension	No temporal	Time as an additional attribute	Time as an 4 th dimension
Data Management	Edit georational tables	Edit georational tables	Interactive editing	Interactive editing
Data Integrity	Yes; ArcGIS only	Yes	No	Yes
Visualization	2D/3D view	No	3D view	3D view
Analytical and querying functions	Simple query on tables; other functions under developments	3D Query	Visibility; queries on 3D buildings, moving points	Spatial-temporal queries

Given the current developments of GIS tools (Table 3), logical implementation of a 3D spatial-temporal model is not straightforward, and requires the developments in several aspects (Pelagatti et al., 2009). First, data structure should be adaptable to multiple types of 3D geospatial objects. So far, only the OO approach is believed to be able to fully describe 3D spatial-temporal objects at the conceptual level but its capacities of supporting support spatial-temporal behavioral queries are far from sufficient as discussed in previous section. Second, algorithms need to be developed to support operations such as spatial and/or temporal queries. These algorithms, which are based upon geometric computations of 3D objects, have not been implemented in spatial databases or existing 3D GIS prototypes regardless of their computing efficiency. Third, data integrity and consistency have to be maintained when these objects undergo manipulation (Cockcroft, 1997).

In addition to proposing a conceptual model for 3D spatial-temporal objects, another objective of this study is to adopt a data management perspective to address some of the limitations of existing systems in handling 3D spatial-temporal data as discussed above. Particularly, the focus is to develop a framework to ensure the integrity of spatial-temporal data. Additionally, the framework should be able to facilitate spatial-temporal queries of 3D spatial-temporal objects.

2.2 The Constraint-Based Approach for Data Modeling and Management

2.2.1 Background

A constraint-based approach may offer some solutions to address the open issues discussed above. Constraints, also known as filters or principles, are conditions that need to be satisfied during a modeling process (LaCharité and Paradis, 1993). The importance of constraints has been widely recognized in database management and applications.

Constraints can be used to express data from an infinite set of which elements are uncountable (Kuper et al., 2000). For example, in the context of sending a package, given various combinations of package weights and destinations, the postage fee charged for a package to all destinations within the world is an infinite set of values (Revesz, 2009).

Since these fees are computed by multiplying the unit rate with the weight of the package, instead of listing all possible combinations and fee values, the post office usually determines the fee using an equation. In this example, all fees should satisfy a mathematical relation between unit rates and weights, which is an equality constraint in essence. In addition to equality constraints, a more frequently used constraint type is inequality constraint where a loose condition is specified (Rigaux et al., 2002). For example, the spatial coordinates of all points falling inside the unit sphere with a hole inside are summarized with two inequality constraints, $x^2+y^2+z^2 < 1$ and $x^2+y^2+z^2 > 0.5$. In this case, not all possible solutions need to be identified and recorded, but the set of solutions is defined by mathematical conditions. Such unique feature, expressing infinite data with conditions, saves the storage of databases and improves the efficiency of data retrieval when applying constraints in database management as not all values need to be stored explicitly.

Constraints facilitate the implementation of queries. Query conditions can be formulated as constraints. Extended from standard relational languages, constraints are inherently a type of query language (Grumbach et al., 1995). The Constraint Query Language (CQL), a formal language that represents query conditions as constraints in databases and information systems, was first proposed in 1990. It has been implemented in popular database software (e.g. Oracle 10g). Solving queries expressed as constraints is a process of finding objects meeting the requirements specified by the constraints. This is also known as CSP. In this process, instead of executing query criteria in a brute force manner, a set of constraints can be simplified into simpler equivalent forms (Robin et al., 2007). For example, any real number r from the set R should meet two relational constraints: $a < r < b$ and $c < r < d$. Given two additional constraints $a < c$ and $b < d$, the original constraints can be simplified into $c < r < b$. Executing the simplified constraint may improve the efficiency of evaluating constraints, which further improve the overall efficiency of querying (Caballero et al., 2010; Buscemi et al., 2008).

Constraints can assist the maintenance of data integrity. Integrity constraints have been adopted in relational databases and can be specified by database administrators before the creation of a database (Camossi et al., 2006). Given the presence of integrity constraints, the database will verify if rules pertaining to data integrity are satisfied when any changes are made to the data. For example, a temporal constraint that the check-in action always happens before the check-out action is created in a hotel management system. When inserting a new lodging record into the system, the system compares the date of check-in and check-out to verify if the record meets the temporal constraint. If the

new record fails to meet the temporal constraints, a warning message can be issued by the system to notify users about the violation condition. Upon receiving the error message, users then can delete or modify the record. Enforcing data integrity ensures the quality of data stored in a database.

2.2.2 Constraints in spatial sciences

The concept of constraint attracts the attention of spatial scientists. Incorporating constraints for modeling 3D static objects is not new. Practices from Computer Aided Design (CAD) have demonstrated that constraints are essential in the creation of 3D objects. Ma et al. (2003) proposed a constraint-based modeling approach employing geometric and topological requirements to help manufacturing process such as placing a hole inside a wheel precisely. In the geospatial domain, Louwsma et al. (2005) developed a Virtual Reality (VR) system where multiple constraints were created and enforced to assist users place new objects in the landscape precisely.

One example in spatial-temporal scenarios is to describe the trajectory of objects moving in 2D space (Grumbach 1999; Grumbach 2001). A linear equality of variables x , y and t describes the trajectory of moving objects represented by a collection of points. The position of such an object at a given time can be retrieved from the formula. When an object moves within a rectangular region, inequality constraints can be used to describe its trajectory. The inequalities can be expressed as $a < x < c$ and $b < y < d$ where a , b , c and d are the points defining the bounding rectangle. Both inequality and equality constraints offer a relative precise description of movement. This example provides sound evidence that constraints can be used to characterize spatial-temporal dynamics.

In reviewing literature on constraints in spatial domain, three major roles of constraints can be identified. Constraints can capture information of objects regardless of their dimensionality (Kuper et al., 2000). Spatial or temporal information, of objects if not explicitly provided but in the form of constraints, can be inferred from interpreting the constraints. This nature of constraints can be reflected by the capabilities of describing spatial positions of moving objects with linear equality discussed earlier (Grumbach 1999; Grumbach 2001). Constraints may be imposed upon different aspects of an object. The complexity and semantic of these constraints may vary significantly, ranging from the simple constraints of restricting the numerical values of an attribute (e.g. the maximum load of a bridge) to the more complicated ones conditions that control occurrences of a process (e.g. a High Occupancy Vehicle, HOV lane) (Brodsky and Kornatzky, 1995).

Constraints define the rules of constructing objects, interactions among objects and operations that can be performed on objects (Ma et al., 2004). In the process of creating or editing an object, any changes applied to the object will invoke the process of enforcing constraint to determine if the object violates system integrity (Tarquini and Clementini, 2008). If the non-overlapping constraint is in effect when digitizing the county boundaries, a newly generated polygon feature will be subjected to a process of verifying the topological relationships between a polygon and existing polygons in the system. Similar to the data integrity constraints in database management, these constraints help ensure the accuracy, completeness and validity of geospatial data within a system (Camossi et al., 2006).

Constraints facilitate queries on spatial objects using computing techniques for multiple constraints. Querying criteria can be expressed as constraints and constraints are preprocessed by assessing whether the specified conditions are consistent. If these conditions create conflicts, the querying criteria are inconsistent. For example, an inconsistent spatial query could be “finding any pair of objects that are disjoint and intersection at the same time”. The preprocessing step terminates invalid queries to avoid unnecessary geometric computations (Egenhofer, 1994). In solving spatial queries, indexing techniques are often used to speed up spatial searches. Given the trajectory of a 2D moving object described by a linear or polynomial expression with variables x , y and t , which is an example of arithmetic equality constraints, its minimum bounding rectangle (MBR) can be derived from the extreme values of x , y and t conveniently. Identify the MBR of moving objects is an early step in generating an R-tree, which is a spatial index facilitating searches of a large number of spatial objects within a specific geographic region (Revesz, 2009). Without the constraint based representation, it takes considerable amount of time to sort all possible values of x , y and t in order to determine the MBR. Applying constraints in this case improves the efficiency of determining the MBR and indexing.

While the examples discussed above have demonstrated the unique roles of constraints in spatial domain, they only show how constraints have been used in the management of 3D static or 2D spatial-temporal data. Employing constraints in 3D spatial-temporal data management may encounter some difficulties: a) determine a reasonable complete set of constraints designed for 3D spatial-temporal objects (Salehi et

al., 2011). While scientists have explored the types of constraints that can be imposed on spatial objects in general, existing literature has not provide a comprehensive set of constraints for 3D spatial-temporal objects. Due to the complexity and variety of properties of spatial temporal objects, constraints appropriate for different properties also vary significantly; b) define a logical sequence of spatial-temporal constraints, especially those involving multiple filters, accurately and precisely so that they can be interpreted by the constraint computing tools (Jaffar and Lassez, 1987; Parent et al., 1999).

Constraint statements expressed in the form of natural languages are imprecise and may introduce ambiguities. How to eliminate the ambiguities while express the constraint conditions effectively remains a problem; c) implement these constraints in a systematic manner. This is the process starting from issuing constraints, enforcing constraints with appropriate algorithms and tools, to notifying the users that the objects violate some constraints (Louwsma et al., 2005).

This research aims to address several challenges of applying constraints to maintain 3D spatial-temporal data integrity and support querying through a) defining and formalizing appropriate constraints for 3D spatial-temporal objects; b) developing efficient algorithms to enforce these constraints imposed on 3D spatial-temporal objects; and c) providing interactive constraint management tools through which users can interact with constraints and receive instant results after enforcing constraints. With the formal definitions of various constraints pertaining to 3D spatial-temporal objects, both integrity constraints and query conditions can be expressed in a logical manner. The corresponding constraint solving techniques, involving multiple spatial-temporal

algorithms, assist the maintenance of data integrity and evaluation of queries. Finally, the interactive interface allows users with minimum knowledge of the constraint concept to perform data management functions and analyses.

CHAPTER THREE A CONCEPTUAL MODEL FOR MANAGING 3D SPATIAL-TEMPORAL DATA

3.1 An Extended Object-Oriented (OO) Model for Spatial-Temporal Objects

According to Pelekis et al. (2004), the OO data model is probably the most comprehensive type of data model for describing 3D spatial-temporal objects in terms of its capabilities in depicting spatial features, temporal properties and attributes. The major deficiency of existing OO modeling approaches is the lack of sufficient and accurate descriptions of object behaviors, which leads to inadequate support on the analysis of behaviors as discussed in Chapter 2. Therefore, the formulation of the proposed conceptual model was based on an OO modeling approach, but it improved upon the OO approach by including the descriptions of object behaviors which will be discussed later. An important feature of the OO model is to group objects into different classes based on their attributes and to differentiate one class from the other based on their important characteristics (Banerjee et al., 1987; Meyer 1988; Rumbaugh et al., 1991). Objects from the same class usually share the same attributes and behaviors. Being able to classify spatial-temporal objects is the entry point of designing such data model. Classifications of spatial-temporal objects vary by the ways that these objects are perceived. Spatial-temporal objects are considered as spatial objects changing over time in general. According to Güting and Schneider (2005, p23), a spatial-temporal object is “being embedded in a space that is the cross-product of the original spatial domain and of time”.

Based on this definition, they divide spatial-temporal objects into ten categories and objects in each category consist of shape element and temporal element (Güting and Schneider, 2005). Besides such shape-based classification, a few other classifications have been discussed in the literature. For example, according to the variation-based classification, objects are classified into deformable or not. Deformable objects such as water flows are called soft objects. Non-deformable objects are rigid objects of which the spatial shape remains constant under external forces, such as buildings (Jaillet et al., 1998). A widely recognized classification of geospatial objects in GIS is the representation-based one. Gradient vector flow in physics and fluid vector flow in brain tumor segmentation are all examples of vector representation. Raster representation employs a scalar field to capture the objects, which is made up of a 4D (3D space plus 1D time) matrix. All three classifications are applicable to both 2D spatial-temporal and 3D spatial-temporal objects. Table 4 shows the three classifications and gives examples for each classification.

Table 4 Classification of spatial-temporal objects

Criteria	Types	Examples	References
Shape	Shapes: Points, regions, entities Time: instant/period	events in space and time, locations for certain period, set of location events,	Güting and Schneider 2005.
Variation	Rigid/Soft	Building/Droplets	Shen et al, 2005
Representation	Raster/Vector	Fluid flows described by vector field/Dust density records with scalar fields	Helman and Hesselink, 1989

Among these classifications, the shape-based classification is preferred for building an Object-Oriented (OO) data model. As discussed in Chapter 2, a versioned object, expressed as a tuple of (shape, location, time), is usually defined in an OO model to represent the status of a spatial-temporal object within a time frame. A spatial-temporal object consists of a series of its versioned objects sorted by temporal information. With the shape-based classification, the spatial and temporal information of a spatial-temporal object can be easily identified to generate versioned objects since this classification clearly indicates the shape element and time element of spatial-temporal objects (Wachowicz and Wachowicz 1999). In the context of 3D spatial-temporal objects, the shape of 3D spatial-temporal objects is characterized by the basic geometric primitives that make up the objects. They are 3D points, 3D lines, 3D planes and volumetric shape bounded by 3D planes. By recording the types and spatial coordinates of the geometric primitives of a spatial-temporal object, the model depicts spatial information of that spatial-temporal object. The shape and the location of a 3D spatial-temporal object are considered as the spatial information. On the other hand, the temporal properties of a spatial-temporal object are characterized by its occurrence or creation, lifespan, transactions (from one status to another status) and termination. These temporal properties can be recorded as time points, time periods and frequencies. A time point refers to a moment of an object instance or an event while a period refers to duration of time that makes up of a pair of time points. A frequency indicates the number of occurrences of a repeating phenomenon or event in every unit of time. A frequency that

specifies the repeating rate can be attached to a time point or a time period. For example, an Orange Line train leaves the first station every three minutes (frequency) from 8:00 AM to 10:00 AM (time period). The frequency is attached to the Orange Line trains. Then a spatial-temporal object is a spatial object that its shape, position, and/or attributes change over at discrete time points or time intervals. Below, I will give the formal definitions of spatial information and temporal information of a versioned object which represents a 3D spatial-temporal object.

Definition 1 Basic geometric types: Given a 3D space R^3 :

A point p is defined as

$$p = (x, y, z) \quad x \in R, y \in R, z \in R$$

A line l is defined as and a point p and a vector \vec{d} which start from the point

$$l = (p, \vec{d}) \quad p \in R^3, \vec{d} \in R^3$$

A plane s is defined as a flat surface with a point p on that surface and a normal vector \vec{d} of the surface such that

$$s = (p, \vec{d}) \quad p \in R^3, \vec{d} \in R^3$$

Given a plane s , a region r is defined as a closed chain of n line segments (p_i, p_{i+1}) for $0 \leq i \leq n-1$ and (p_{n-1}, p_0) on that same plane.

$$r = (p_1, \dots, p_n, p_1) \quad n > 2, \forall p_i \in s$$

A polyhedron v is a set of enclosed planes

$$v = (s_1, \dots, s_n) \quad n \geq 3$$

The spatial information of a versioned object at time α is characterized by its geometric primitives SP_α which is defined as

$$SP_{\alpha} = \langle p_{\alpha}, l_{\alpha}, s_{\alpha}, r_{\alpha}, v_{\alpha} \rangle$$

Definition 2 Temporal data types are time-point, time-interval and time-frequency.

A time point is defined as a real number

$$T^p = t \quad t \in R$$

A time interval is made up of two real numbers

$$T^i = \{T^p | a \leq T^p \leq b; \ a, b \in R, a < b\}$$

A frequency is defined as a positive value indicating rate of repetitive patterns

$$f(t) = k \quad k > 0$$

Temporal information TP of a versioned object α is a tuple of two sets: the set of time points and the set of time intervals. Frequencies can be attached to each set.

$$TP_{\alpha} = \{(t, f)\} \quad t \in \{T^p\} \cup \{T^i\}$$

As discussed in Chapter 2, to overcome the deficiency of existing OO models in supporting behavioral analysis on spatial-temporal objects, the proposed model also depicts the dynamics of an object as the behavioral description of 3D spatial-temporal objects. As an extended feature to existing OO models, such description records the evolution, movement and changes between the two versions of an object during its lifespan; i.e. from its creation to its destruction (Lohfink et al., 2010). With such behavioral description, researchers can estimate or predict the changes between the observed sequence of a spatial-temporal object (e.g. Worboys and Hornsby 2004; Vidal and Rodriguez 2005). For example, a car started slowing down at Location A at time t_1 and reached a complete stop at Intersection B at time t_2 . Assuming a constant reduction of

speed v_x between t_1 and t_2 , I can derive a polynomial equation establishing the relationship between distances, speed and time. The location of the car at any time can be derived using this equation.

The behavioral description is added to the triplet expression of the versioned object. The behavior of a versioned object is defined as “changes” when one version of the object is compared to the previous version. Typical changes are geometric transformation (morphology), topological changes, and changes in attributes. The geometric transformation refers to the deformation process such as rotation, proportional and differential scaling, skewness and translations, or a combination of the four processes. A topological change refers to a change in the topological structure of the object, such as “self merge”, “self split”, and formation of holes. Change in an attribute refers to a change in a property of the object, such as an increase or decrease in velocity of a moving object. These changes are not mutually exclusive of each other as all three types of changes can happen to the same object, such as iceberg, all at once. By adding behavioral descriptions to versioned objects, the model establishes the relations between a sequential pair of versioned objects, and eventually, the relations between any two versioned objects of the spatial-temporal object.

Definition 3 A behavioral description of an object of version i is the joint of three types of changes

$$B_i = \langle CT_i, CG_i, CA_i \rangle$$

Where CT_i is the topological change of the object between version i and version $i-1$; CG_i records the geometric changes same as above and CA_i is the description of changes in attributes;

Figure 1 illustrates the structure of a versioned object which is a combination of geometric properties, temporal properties, attributes, and behavior. A new versioned object is created when any of its geometric properties, behavior characteristics, or attribute values have changed. For example, each flight track is a versioned object because the location of each track varies. A spatial-temporal object can be defined with multiple versioned objects sorted by temporal information.

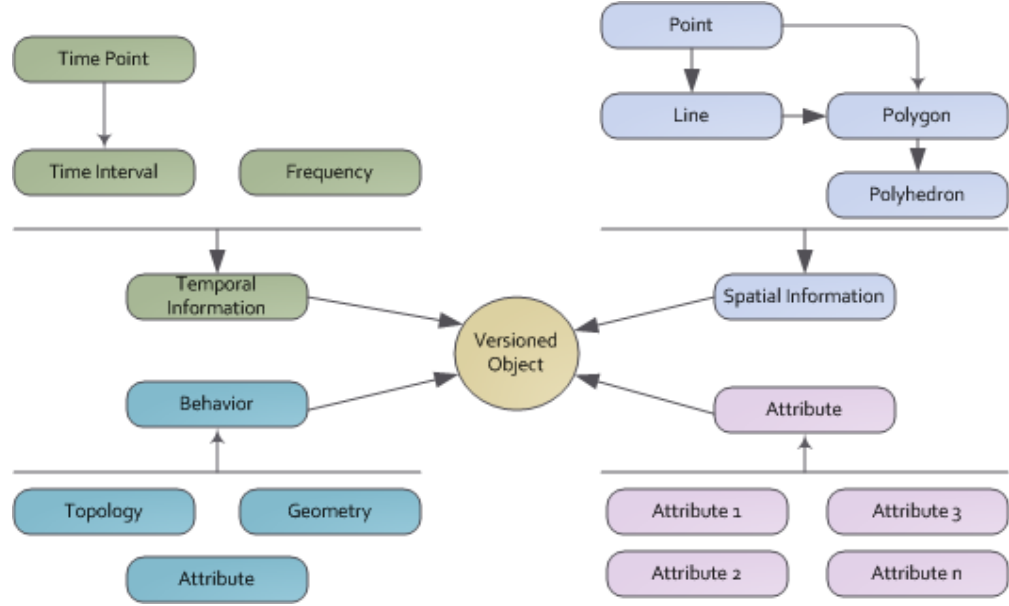


Figure 1 Object hierarchy of the extended 3D spatial-temporal OO model

Definition 4 A spatial-temporal object is made up of different versions of the same object sorted by their temporal descriptions.

$$O = \{VO_i | 1 \leq i \leq n\}$$

Definition 5 The i_{th} versioned object VO is defined as a tuple of spatial information SP_i , temporal information TP_i , behavioral description B_i and attributes A_i

$$VO_i = \langle SP_i, TP_i, B_i, A_i \rangle$$

3.2 The Constraint-Based Approach

While the OO model described in 3.1 depicts the spatial, temporal and behavioral information of an individual 3D spatial-temporal object, it does not capture permissible relations between objects nor provide functions to support analysis such as querying. A

constraint-based approach is proposed and integrated with the OO model in improving the capabilities of the OO model. As discussed in Chapter 2, a constraint-based approach facilitates the maintenance of data integrity. Most database applications have provided functions that allow database administrators to create rules that data entered in to the database should follow. Even for 3D databases (e.g. Oracle spatial), they do not provide a mechanism to ensure data integrity of 3D spatial-temporal objects. However, such critical mechanism is missing in existing spatial-temporal data models and the management systems built upon the models. This is partly due to the fact that integrity constraints for spatial-temporal data that define permissible relationships between 3D spatial-temporal objects are not well formulated (Salehi et al., 2011). A precise classification of constraints related to data integrity is necessary to provide guidance to specify these constraints in a spatial-temporal database. Due to the presence of the third spatial dimension – height or elevation, the spatial-temporal semantics typically applied to 2D spatial-temporal modeling have to be expanded such that the scope of the resultant constraints can accommodate the third spatial dimension. With different categories of constraints defined in the model, integrity constraints on spatial-temporal properties of spatial-temporal objects can be easily declared during the modeling stage. Besides declaring these rules in a reasonable manner, maintaining data integrity heavily relies on the design of the enforcement of these integrity constraints. In a spatial-temporal model, certain integrity constraints are enforced all the time within the system whereas other integrity constraints are imposed on the objects only during their life span. Therefore, the process of evaluating constraints should be carefully designed.

Spatial-temporal constraints are employed to implement the complex queries. Constraint-based solutions for queries outperform traditional ways in processing queries mainly because multiple constraint process techniques are often employed to optimize the evaluations of queries. An important constraint computing technique related to the conceptual model could be integrating constraint-based descriptions for objects with queries expressed as constraints. If constraints pertained to spatial-temporal properties of spatial-temporal objects exists, these constraints serve as additional criteria when processing queries. As a result, the constraint-based approach can better characterize the querying conditions by adding relevant constraints. How such an approach is realized at the conceptual level will be further illustrated in Section 3.2.4.

This section proceeds as follows. Section 3.2.1 discusses the definitions and content of different types of constraints. A formal description of different types of constraints will be provided in Section 3.2.2, followed by the discussion on constraint enforcements. Finally, basic types of spatial-temporal queries with constraints are formalized.

3.2.1 Types of Constraints

In the past decade or so, scientists have developed different taxonomies of constraints in geospatial applications (e.g. Belussi et al., 1997; Currim and Ram 2010; Mas and Reinhardt 2009; Revesz 2009; Salehi et al., 2011; Weder 2009). In conceptual data modeling, objects are described with data structures defined by the model and constraints pertained to objects are in fact imposed on the data structures (Currim and Ram 2010). Therefore, the components of a data structure determine the classifications of

constraints. In this model, a spatial-temporal object is described with a series of versioned objects and each version object consists of four components. Constraints can be divided into four general categories corresponding to the spatial, temporal, behavioral and attribute components of a versioned object. The similar classification is found in Mas and Reinhardt (2009) and Salehi et al (2011) where spatial, temporal, thematic and spatial-temporal/ change constraints are discussed. While complex constraints have been considered as a new category according to their classifications, such constraints, which are created by combining more than two constraints from the four categories, are not treated as an individual category in my classifications. Besides, I also extend the content of spatial-temporal constraints by adding rules related to spatial-temporal topology, which has not been systematically defined. Below I will introduce representation, spatial, temporal, attribute and spatial-temporal constraints.

Representation constraints: One of the earlier steps in building a data model for geographic features or phenomena is to represent these features or phenomena in an abstract manner. Different approaches can be used to represent objects. For example, in a moving object model, an airplane is described as moving points in which the geometric information such as shape and volume is not recorded. But the geometric information of an airplane is included in an OO model because the geometric properties are the most important features to distinguish one object from another in the OO model. Therefore, depending on the approach adopted, how a specific type of objects should be represented can be regarded as a representation constraint. Representing an airplane as moving points or moving volumetric objects is determined by the modeling approaches. This is an

example of representation constraints. In a modeling process, representation constraints refer to the rules guiding the selection of appropriate abstract spatial and temporal data types (e.g. geometric primitives) to represent an object (Werder 2009). Representation constraints are usually enforced when researchers describe an object with models.

Temporal constraints: Temporal constraints can be absolute or relative. An absolute temporal constraint is a direct measure of time or a temporal attribute of a spatial-temporal object (e.g., transaction time, valid time). By contrast, temporal relations are derived from the comparison of the temporal information of multiple objects. For example, the relation “A is before B” implies the occurrence of A is prior to that of B. Allen (1983) identified 13 basic temporal relations some of which are before, equal, meets, overlaps, during, starts, finishes and the reverse cases. As the representations of temporal relationships are typically relative (not absolute dates), they contain a certain degree of uncertainty during the derivation. The uncertainty issue is beyond the scope of this study. Considering the importance of the two types of temporal constraints, the proposed model supports both absolute as well as relative temporal constraints.

Temporal constraints can also be expressed as temporal logic too. Unlike the date or before, a direct way to depict temporal conditions, temporal logic hides the references to time. With temporal logic, the temporal semantics is inferred by interpreting temporal expressions. These expressions of such logic are any rules for representing and reasoning about the properties of time (Venema, 2001). Operators such as “*until*”, “*always*”, “*sometimes*” and “*next*” are usually found in the expressions and are considered as symbols of temporal logic. For example, *car A* is not allowed to enter *Zone X* until *car B*

leaves, implying a temporal condition involving two cars. Temporal logic can be integrated with absolute or relative temporal conditions. An example could be “Car A is always in front of Car B from 8:00 AM to 12:00AM”.

A temporal constraint can be imposed onto any spatial temporal objects regardless of their spatial dimensionality. For example, the lifespan can be used to describe the melting of iceberg or the duration of a flood event. From the perspective of space, iceberg is a 3D volumetric object while the flooded area may be captured as polygons with boundary changing over time in a 2D context. Temporal constraints can be applied to all 2D and 3D spatial temporal objects.

Spatial constraints: Parallel to the temporal constraints, spatial constraints regulate the spatial properties of objects. To differentiate them from representation constraints which primary regulate the abstracted representation of objects, spatial constraints are defined as geometric properties of objects and spatial relations meeting certain conditions. Geometric properties include shape, size, extent, geometrical structures of objects. Spatial relationship of objects or features can be described by distance, either absolute or relative. Certain types of relationship can also be described as topological, if the actual distance is not important. Similarly, the directional relations (east or north of) specify the order in space. These types of spatial relationships can be regarded as spatial constraints. In Grumbach et al (1999), linear equations over speed, time and location belong to a type of metric relations and are used to represent standard geometry.

Among different types of spatial relations, the spatial topology between objects is probably the most complex, especially when 3D objects are involved. Two common models for defining spatial topology are the 9-intersection model proposed by Egenhofer et al. (1994) and the Region Connection Calculus (RCC) model proposed by Cui et al. (1993). Further exploration of 3D topology can be found in Zlatanova (2000). Zlatanova examined 3D topological relations of simple objects. Simple objects are usually generalization of triangles or tetrahedra of arbitrary dimension (Buekenhout and Parker, 1998). By eliminating 26 unrealistic relationships, Zlatanova concluded 69 possible topological relationships between simple objects in 3D space. The spatial relationships found in complex objects are more complicated (Schneider and Behr, 2006). A complex object is built from simple objects. For example, the simplified complex, a type of complex object, can be created by joining 1-simplex (points), 2-simplex (triangles) or 3-simplex (tetrahedrons) spatially. If complex objects are present, such as volumetric objects with holes, 82 different topological relations can be defined (Losa and Cervelle, 1999). This dissertation is not intended to identify or formalize new spatial relations for 3D objects, but utilize the spatial topological relations defined in the classic 9-intersection model described in Egenhofer et al. (1994). According to the 9-intersection model, the fundamental spatial topologies are containment, intersection and disjunction. Others are extended from the three basic relations and many more can be found in Li (2006). In this research, three basic spatial topological constraints, which are containment, intersection and disjunction, will be implemented in the prototype system discussed in Chapter 4.

Thematic attribute constraints: In addition to the constraints imposing on the spatial and/or temporal properties, constraints on attributes are more frequently employed to condition the scale, ranges and variations of attribute values over time (e.g., Speed limits) Compared to the type of spatial or temporal constraints, attribute constraints may exist in many ways (forms) due to the various nature of attributes. This research does not intend to enumerate all of them but illustrate just a few typical ways to specify conditions imposed on attributes of objects.

- *Equal:* Specifies the exact value for an attribute. E.g.: The train should departure every 3 minutes.
- *Enumeration:* Specifies a list of values for an attribute. E.g.: Special use airspace can be prohibited areas, restricted areas, warning areas, military operations areas, alert areas, and controlled firing areas.
- *Exclusion:* provide a list of values that are not valid for an attribute. E.g.: The ID of a flight cannot include symbols such as “.”,”\$”.
- *Range:* specify extreme value(s) for an attribute. E.g.: The ranges of altitude levels for a specific class of airspace.
- *Comparison:* relate the attributes with another reference attribute with comparison operators such as “less than” and “larger than”. E.g.: Car A should travel faster than car B.

Spatial-temporal constraints: Constraints in this category are usually the conditions imposed on dynamic processes captured by the behavioral descriptions of the versioned objects. For example, to ensure that pilots can react to a stall in fixed-wing

flight (a sudden reduction in lift), the changes of speed should be less than a value or an alarm will be triggered. Spatial-temporal constraints can be imposed on spatial-temporal relations such as spatial-temporal topological relations, which are the combination of spatial topological relations and temporal relationships (Pelekis et al., 2004).

Constraints can be applied to 2D and 3D objects because constraints capture information of objects regardless of their spatial dimensionality. While the same constraint can be imposed on different objects, enforcement of constraints may vary with different objects. Here I use a simple example to illustrate the use of constraints in regulating 2D and 3D spatial-temporal objects respectively.

Example 1: Treating a flying car as an object, it has the following states:

State1: stop (S), driving (D), flying (F)

State2: non-operational (NO), operational (O).

Several constraints can be imposed on the object: the range that the car can travel in D, in F, and in the combined state of D and F. In certain conditions, some constraints on attributes, behavior or states are translated into spatial and temporal constraints. The constraint on the range in D can be translated into a distance/metric and time, how long and how far the object car can travel. Road network and airspace boundaries serve as constraints on the object in State1 = D and State1 = F respectively.

Driving (assuming no skyways) and flying, which are also states of the car, correspond to 2D and 3D behaviors respectively. In the 2D case, a crash indicates a constraint violation when the car travelled outside of the network, causing changes to the geometric representation of the object. The crash occurred when State1 = D and State2 =

O. After the violation occurred, State2 = NO. The flying behavior can be used to illustrate the 3D case. Flying violates the network constraint in the 2D case because the flying car does not follow the road network. When flying is allowed, the airspace constraint becomes effective where the movement of car is no longer constrained by a predefined 2D road network. Unlike the “crash”, “flying” violates the constraint enforced to 2D objects, but not for 3D objects. This implies that if 3D objects are treated as 2D objects as in the literature, properties for 3D objects are likely misinterpreted. The proposed model designed for managing 3D spatial-temporal objects depicts the properties of these objects more accurately.

3.2.2 Formalization of Constraints

Constraints expressed in natural languages in the examples above are understandable to humans, but cannot be used in database implementations. Implementations require formal and standard modeling languages. Using formal languages (e.g. Object Constraint Language, OCL) to record constraints is not just logical but also helps standardize various constraints (Werder 2009). According to Davis et al. (1999), “a constraint is a sentence consisting of a predicate applied to constant symbols”. Within the proposed OO model, a predicate describes conditions for spatial-temporal properties, features and relations such as spatial relations. Constant symbols stand for the entities described by the predicate. If the predicate of a constraint includes a single filter, such constraint is called a “*simple constraint*”. Sometimes, multiple constraint types are included in the predicate and these types are connected with each other in a logical manner to form a “*complex constraint*”. Operators connecting these constraints are

constraint connectors. Previous studies have identified several typical connectors in constraint based models such as “*union*” and “*and*” (Mandel and Cengarle, 1999). Current available connectors do not explicitly take into account the situations for spatial-temporal modeling. To accommodate spatial-temporal objects, two additional sets of connectors are needed: spatial connectors and temporal connectors. The first set deals with spatial constraints implicating the spatial relations between the outcomes of two constraints. The second set deals with temporal constraints establishing the temporal logic or priority among constraints such as “*before*”, “*after*” and “*when*”(Ziemann and Gololla, 2003). For example, the connector “*when*” means that one constraint should be enforced when another constraint is satisfied.

Definition 6 A simple constraint is defined as

$$C: < O, E >$$

where O is the entity to which a predicate is applied and E is a simple logical predicate, including a filter.

Definition 7 A complex constraint is formed by a logical conjunction of multiple filters and can be written as

$$MC: \{ < c_i, cn_k > | c_i, c_j \in C, cn_k \in CN \}$$

where C is a set of constraints; c_i is a constraint from C ;

CN is the set of constraint connectors and can be written as:

$$CN: \{ \text{logical connectors}, \text{spatial connectors}, \text{temporal connectors} \}$$

Example 2: A constraint that all cars are not allowed to enter a restricted area A during t_1 and t_2 can be expressed as

O: { All cars, A }

E: NotInside During (t1, t2)

3.2.3 Enforcement of Constraints

Enforcements of different types of constraints are triggered by different mechanisms, controlled by the constraint hierarchy, depending upon the priority of enforcements and level of object aggregations. I investigate the hierarchy of constraints in controlling the enforcement process in two ways: the priority of enforcements and level of object aggregations (Demuth and Hussmann, 1999). Figure 2 shows different types of constraints (right) and the hierarchy of constraints (left). Any constraint, either simple or complex, can be mapped to the components in object hierarchy and/or elements in constraints hierarchy (Cockcroft 1997; Pinet et al., 2007).

The constraint hierarchy arranges constraints from low priority to high priority adopted from classifications of integrity constraints designed for database managements (Elmasri and Navathe 2000). The static constraints should be satisfied at any “snapshot” of the system which is labeled as “*Anytime*”. With a looser condition, the dynamic ones are triggered when state of objects changes which are usually associated with the movements or transformations of objects. The last category, functional constraints are triggered when certain operations are applied to the objects such as the rate of reducing speed during a landing process of flights. Enforcing different levels of constraints within a system sustains the robustness and correctness of that system as violations of business rules are filtered out accordingly.

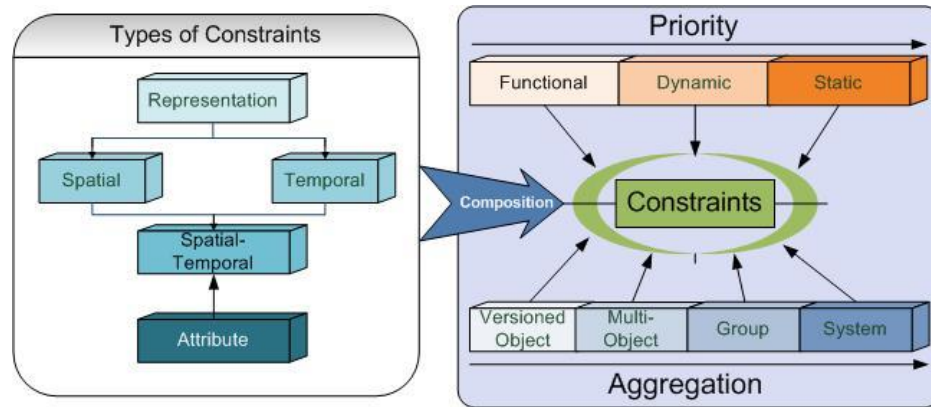


Figure 2 A systematic view of constraints

On the other hand, the object hierarchy arranges objects based on level of aggregations from individual level to system level. Constraints can be associated with a versioned object, multiple versioned objects from a spatial-temporal object, a group of spatial-temporal objects and/or all objects within a data management system. The number and types of objects involved in the corresponding constraint enforcement may vary. If a constraint is imposed on a versioned object of a spatial-temporal object, that constraint is effective to that versioned object only. If the same constraint is imposed on the entire spatial-temporal object, then the constraint is effective for all versioned objects of that temporal object and these versioned objects are subjected to the enforcements.

When a constraint is imposed on an object, such constraint indirectly reflects a property or behavior of the object and offers additional information about the object. Compared to the traditional OO approaches that record object properties explicitly, the constraint-based approach provides information about the properties in the form of rules,

conditions or restrictions to be satisfied under certain situations. Such types of constraints usually are group constraints involving relational declarations between two spatial-temporal objects or two versions of a spatial-temporal object. For example, two moving vehicles *Car A* and *Car B*, the position of *Car A* is known and the position of *Car B* is unknown. A spatial-temporal constraint that *Car A* is always 200 meters north of *Car B* is specified in the system. I can derive the position of *Car B* which is 200 meters south of *Car A* at each individual time point. This spatial-temporal constraint is a “group” constraint imposed on two moving objects. By interpreting the conditions described by constraints, I obtain additional spatial information of one of the moving objects.

While describing objects is one of essential usages of constraints in data modeling as discussed in Chapter 2, this research does not intend to explore how constraints can be implemented to describe 3D spatial-temporal objects comprehensively. To do so, I have to deal with several challenges. While linear equality constraints have been used to describe trajectories of moving objects in a few studies, movements of these spatial-temporal objects are relatively simple as compared to those geophysical phenomena such as fluid flows. Formulating an accurate and precise mathematical equality constraint to characterize such complex physical processes is difficult due to the deformable nature of these objects in 3D space (Montagnat and Delingette, 2005). Therefore, this topic will be addressed in future work.

3.2.4 Apply constraints in spatial-temporal queries

A spatial-temporal query extends a spatial query with temporal variants called “temporal lifting” (Güting et al., 2000). For example, I can add a temporal condition,

from “10:00AM to 12:00AM”, to a spatial query “finds objects in front of object B in a 3D space” to form a spatial-temporal query. Erwig and Schneider (2002) identified four fundamental types of queries: projection to space and time, spatial-temporal state, spatial-temporal aggregation and relation between multiple objects. Being able to support these four types of queries is a key indicator of a viable spatial-temporal data model (Pelekis et al., 2003).

Similar to general queries implemented in databases, executing a spatial-temporal querying is a two-step process involving the formulation and evaluation of query criteria (Shekhar and Sanjay, 2003). As discussed in Chapter 2, the constraint-based query approach may assist in both steps. On the one hand, query conditions can be expressed as a combination of multiple spatial-temporal constraints. On the other hand, solving queries is a process of applying constraint enforcements to identify objects meeting the constraints. In this section, I mainly illustrate how constraints can facilitate the formulation of four basic types of spatial-temporal queries in general. Since evaluating spatial-temporal queries expressed as constraints depends on the implementation of computational geometry algorithms, which belongs to the implementation of the proposed model. Chapter 4 will explain the evaluation process in detail.

Table 5 shows four types of spatial-temporal queries expressed in natural language and constraints. In these examples, query criteria as filters (denoted in italic text) are recorded as complex constraints. These complex constraints consist of at least two constraints from the five basic categories of constraints. When a complex constraint includes the logical constraint connector “and” in connecting several simple constraints,

it means these conditions have to be satisfied at the same time (e.g. Query 1 in Table 5). According to these examples, all query criteria have been converted into constraints successfully.

Besides describing the filters of query statements, constraint-based queries sometimes include the additional restrictions associated with objects to better characterize the query conditions. In Table 5, Query 4 is converted into two constraints. The first constraint is a temporal constraint explicitly described by the original query statement. The second constraint is a spatial constraint. If faculty are only allowed to park their cars on level 1 and level 3, I use a spatial constraint “on level 1 and level 3” to substitute the condition “cars belong to faculty in Deck A”. In this way, only the cars on level 1 and level 3 are treated as potential solutions to the query whereas cars on the other levels are not considered since they do not belong to faculty.

Adding a constraint may not always help eliminate unqualified candidate objects during in a query given the relations between constraints associated with objects (denoted as O) and the constraints specified by the query (denoted as Q) (see Figure 3). Four types of relationship are identified, including covered by, cover, overlap and disjoint (Figure 3). In Query 4, the O, faculty are only allowed to park their cars in level 1 and level 3, is considered as “overlap” with Q as cars satisfying O may not meet the temporal constraint of Q, which is from 8:00AM to 8:00 PM. If the constraints specified by Q describe a more restricted condition as compared to O, the relation should be considered as Q is covered by O. No additional constraints will be attached to Q. If O is inconsistent with

Q, for example, faculty members are not allowed to park their cars in Deck A, no results will be returned to the query. As a result, the query process is terminated.

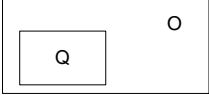
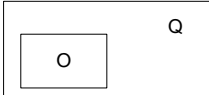
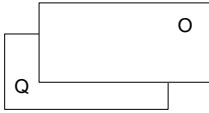

	Covered by	O: Faculty are allowed to park cars in all levels in all decks except Deck D
	Cover	O: Faculty are allowed to park cars in level 1 and level 3 in Deck A from 9:00AM to 5:00PM
	Overlap	O: Faculty are only allowed to park their cars in level 1 and level 3
	Disjoint (Inconsistent)	O: Faculty are allowed to park cars in Deck A

Figure 3 Relations between constraints with objects and constraints specified by a query (O: constraints associated with objects; Q: constraints specified by a query)

If Query 4 is solved using the traditional Structured Query Language (SQL) approach, a query processor selects cars in the deck during that time period as one set and selects cars belong to faculty from all cars as another set. The cars found in both sets will be returned as the query results. On the other hand, to enforce the two constraints of Query 4, the process may depend on the available information about the objects. When the level information associated with cars is available, a constraint solver first selects the cars on level 1 and level 3, and then verifies if the temporal information of these cars falls within the range of 8:00 AM to 8:00 PM .If the temporal selection is performed on a

smaller set of candidates, cars on level 1 and level 3, the constraint-based approach may outperform the SQL approach. If car level information is not provided, the constraint solver has to implement a spatial selection to identify the cars are on level 1 and level 3. As a spatial selection is usually more time consuming than an attribute selection, the evaluation in this case is more time consuming. While the constraint-based approach can better characterize the conditions specified by a query, the enforcement of constraints may not be always more efficient.

Table 5 Spatial-temporal test cases and corresponding constraint formalization

Query	Type	Examples	Constraint Formalization
1	Projection to space and time	<i>Trajectory</i> of a flight during a <i>time frame</i>	MC:<Flight A; Temporal: During (t1,t2) > and <Flight A; Shape: line>
2	Spatial-temporal state	When and where the <i>enrouted</i> flight reaches <i>speed S</i>	MC: <Flight A; Attribute: speed= <i>S</i> > and <Flight A; SpatialTemporal: movement equations derived from enrouted flight plan>
3	Spatial-temporal aggregation	When the snow reaches <i>largest spatial coverage</i>	MC: <Snow; Attribute: extent(t(x))> extent(t(i)) >
4	Relation between multiple objects (moving)	Number of cars belong to faculty in Deck A from 8:00AM to 8:00PM	MC:<Cars; Temporal: During(8:00AM, 8:00PM)> and <Cars; Spatial: on level 1 and level 3 of Deck A>

CHAPTER FOUR A PROTOTYPE SYSTEM TO HANDLE 3D SPATIAL-TEMPORAL OBJECTS

This chapter presents the conceptualization and development of a prototype system. Such system demonstrates the concepts of the extended OO conceptual model described in Chapter 3 and spatial-temporal constraints in supporting the management of 3D spatial-temporal objects. I will first review the general requirements of a spatial information system followed by the discussion on usages of constraints in developing the prototype system. Examining the general requirements and applicability of constraints can help identify the core functionalities of the prototype system. Based on the requirements, the detailed design of the system is introduced at a modular level.

4.1 Conceptualization of a system for managing 3D spatial-temporal objects

4.1.1 General Requirements

According to Thomas and Cook (2006), geovisual analytics serve four purposes: data representation and transformation, visual representation and interactive techniques, visual-driven reasoning and mining, and production, presentation and dissemination of analytical results. Although these four purposes pertain to geovisual analytics, they, to a large degree, reflect some fundamental requirements of a spatial information system that supports geospatial analysis. Data representation and transformation resolve conflicts and ensure the compatibility of data from different sources in order to support visualization

and analysis. Visual representation utilizes interactive techniques to render the concerned phenomenon and allows users to manipulate visual products generated from the data. Visual-driven reasoning and mining help discover relations, patterns and trends hidden in visual products using analytical tools. Finally, analytical results are presented in visual displays.

To fulfill some of these requirements, the prototype system should be able to perform basic functions such as viewing 3D objects described by multiple data types, manipulating these objects interactively and performing spatial-temporal queries. As discussed in Chapter 2, existing 3D GIS provide reasonably good 3D rendering capabilities and interactive manipulations techniques. The challenges of implementing such prototype system are to develop tools to ensure effective manipulations of geospatial 3D objects and to support spatial-temporal queries, especially for complex ones. The constraint-based approach has the potential to address these two critical challenges.

4.1.2 Constraints for managing 3D Data

To implement of interactive capabilities to manipulate 3D objects, intuitive visual interfaces and a data management module are needed. Visual interfaces are usually considered as a part of the front-end functions through which users can alter objects graphically (Schmidt et al., 2008). Upon the modifications of objects, data management module updates the modified data accordingly. However, the accuracy and correctness of such modifications are not guaranteed if rules pertaining to the acceptable operations of objects are not imposed on, say “free style” modifications. Although some data management tools may take into consideration the positional information of the objects

on the fly during the data manipulation process such that the accuracy of the modified spatial features may be safeguarded, the underlying topological relations among features are not always considered and maintained when users alter the geometric properties of objects (Grumbach 2001).

A constraint-based approach for 3D geospatial data management can help ensure of data integrity. Previous studies have demonstrated that incorporating constraints in the visual manipulations of geospatial objects can help maintain data integrity in 3D modeling in VGE (Tarquini and Clementini, 2008, Lin et al., 2010). Although this prototype deals with 3D spatial-temporal objects, the constraint-based approach can be extended to handle the temporal dimension as constraints can describe objects regardless of their dimensionality. In developing visual interfaces, users should enjoy a higher level of accuracy in manipulating data if spatial-temporal constraints are incorporated into a verification module, which will be triggered through the interactive visual interface for data manipulation while not violating the system's data integrity.

4.1.3 Constraints for Spatial-Temporal Querying

Four types of constraints, spatial, temporal, spatial-temporal and attribute constraints, discussed in the Chapter 3 encompass all types of conditions applicable to each 3D spatial-temporal object and the relationships between objects over space and time. Therefore, these constraints can also be used to formulate queries for 3D spatial-temporal objects, as all possible conditions for all objects and their relationships are bounded by these constraints. Ideally, when a query is formulated, it can be recorded and then executed repeatedly. Thus using a query language is a common practice. Standard

query languages have been extended to handle 3D spatial-temporal objects and have been implemented in database platforms (e.g., Mirbel et al., 2003; Sourina 2006). However, these systems can only support simple queries, which are insufficient to express the semantics of complex spatial-temporal relationships where multiple filters are involved (Li and Chang 2004). In order to formulate complex queries, some of these systems require users to have a good understanding of the technical details of the query language structure and semantics, including the knowledge of selecting the appropriate logical connectors to connect different filters. In addition, most systems can only handle a few simple geometric data types such as boxes and cylinders, partly due to the limited capabilities of the underlying databases in processing topological queries for 3D objects. Query functions for other basic geometric data types such as 3D points, 3D polylines, 3D polygons, and polyhedron were not supported in these systems. When a query involves spatial topology over time, the algorithms become more complex, involving geometric computations on various spatial-temporal objects. Previous systems have not yet advanced to the stage of handling this type of complicated 3D spatial-temporal queries and efficient 3D spatial-temporal algorithms have yet to be developed.

The approach adopted to manage queries is more user-friendly and interactive than previous approaches so that users can handle the complex 3D spatial-temporal queries more efficiently. A three-step process to implement spatial-temporal queries for 3D spatial-temporal objects is proposed here. These steps include formalizing queries by means of graphic interfaces, recording queries with a formal query language, and implementing query processing algorithms. Instead of requiring users to use or learn a

query language, I proposed developing a graphic interface for users to formulate queries interactively. Users can first generate a simple spatial query and then specify additional spatial and/or temporal constraints to complete the formulation of a complex query. Specifics of the interface and processes will be discussed in the implementation section.

Queries can be expressed with constraint query languages such as OCL which is a formal language that describes rules applied to objects (Kanellakis et al., 1990). But to handle 3D spatial-temporal data, I developed an enhanced version of the OCL to record constraints (Warmer and Kleppe 1999). Any query can be translated into constraints stored in OCL, and constraints can be validated and modified. Processing queries that are expressed as constraints is essentially a constraint satisfaction process. Objects meeting the conditions described by constraints are returned as the results to a query after evaluation. Algorithms designed for evaluating certain constraints can be reused to solve similar queries. Constraint programming techniques, such as identifying a reasonable sequence of enforcing constraints rather than using a brute force approach can improve the efficiency of processing constraints (Caballero et al., 2010).

4.1.4 Formulation of 3D spatial-temporal queries

Table 6 lists the types of spatial-temporal topological queries that have been formulated in the current work and implemented in a prototype system. Query of attributes, which has been discussed thoroughly in the literature, is excluded here. Three components are needed to support the formulation of a spatial-temporal query: a window interface for query input, spatial topological constraints and temporal constraints. The query window is used to define the 3D space inside which the spatial topological

constraints will be operated. Spatial topological constraints include those fundamental ones for 2D as discussed in Egenhofer and Herring (1994), but I have also included 3D topologies discussed in Ellul and Haklay (2006). However, I have to extend the topology categories to handle spatial-temporal objects (Table 6). Temporal constraints include time period or time point and temporal topologies. The three components together support the formulation of spatial, temporal and spatial-temporal queries.

Table 6 Types of Spatial-temporal queries supported by the system (*: Valid after specifying time period)

Mode	Query Window	Spatial Topology	Temporal Constraint
With a new geometry	3D Moving Point	Intersect(trajjectory); Inside;	Time Period
			Temporal Topology
	Cube	Inside; Intersect;	Time Period
			Temporal Topology*
	3D Rectangle	Intersect; On surface; Inside	Time Period
			Temporal Topology*
	3D Planar Polygon	Intersection; On surface; Inside;	Time Period
			Temporal Topology*
With existing objects	3D Spatial-temporal object: 3D point, line, 3D planar polygon, regular volumetric objects, bounding volume;	Given the object types, multiple topological relations supported	Time Period
			Temporal Topology
			Combined
		Not specified	Time Period
			Temporal Topology

Spatial or temporal queries can be formulated in the following manners. To formulate a spatial query, the temporal constraint should be set to “N/A” (i.e., not

applicable), meaning that the spatial constraint is true at all time. Then the temporal information of the objects will not be taken into account when solving the spatial queries. Such spatial queries can be used, for instance, to find the flights with the same route as flight 355. In this case, as long as a flight taking the same route as flight 355 at any time will be identified. On the other hand, temporal queries are formulated to identify objects satisfying only the temporal conditions specified in the queries, disregarding the spatial properties of the objects. Even if the object is a spatial-temporal object, only the temporal information of the object will be evaluated. An example of such a query is to find all the elevators that left the ground floor 5 seconds before a given elevator, regardless of where the elevators are located within in the building.

Figure 4 describes the process of formulating a spatial-temporal query, and the associated user-friendly interfaces. To formulate a spatial-temporal query, the user can either select an existing spatial-temporal object or create a new 3D object, such as a cube using the graphic interfaces (Figure 4 (b)). If a new 3D object is created, its temporal attribute (time period or time point) has to be defined first (Figure 4 (a)). Treating the selected or the newly created 3D temporal object as the reference object, temporal topology can be specified for the query, identifying objects that meet the query conditions before, during or after the temporal attribute value of the reference object. Then the spatial topology in respect to the reference object is needed to determine the spatial relationship to be used in implementing the spatial-temporal query.

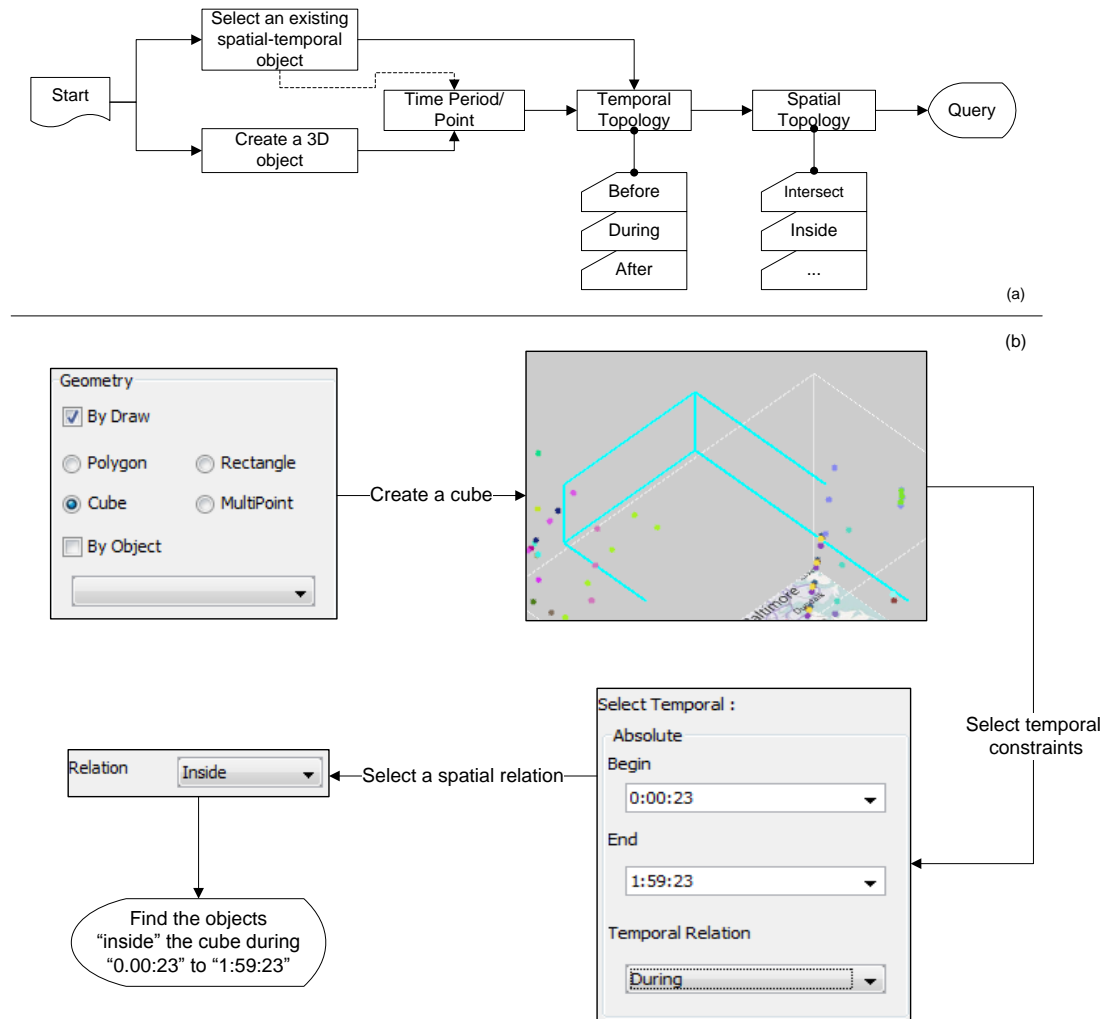


Figure 4 The process of formulating spatial-temporal queries (a) and the graphic user interfaces supporting the formulation process (b).

Queries formulated using the graphics interfaces in Figure 4 are translated into constraints, and these constraints are recorded and stored in OCL. Below is an example of the extended version of OCL that describes a spatial-temporal constraint for the object Flight.KHA382 that it should always be outside of Sector.Sector3.

Context: Flight.KHA382

inv: self.**Spatial_Outside**.Sector.Sector3

and **Temporal_AllTime**

4.1.5 Enforcing constraints with spatial-temporal geometric computational algorithms

While newly formulated queries are translated to constraints to be executed, constraints pertaining to different objects have to be enforced. Enforcing spatial-temporal constraints is a two-step process: determining the current statuses of relevant spatial-temporal objects (e.g., spatial relations between objects) and verifying whether their statuses meet the conditions described by the applicable constraints. Among the four types of constraints discussed above, the spatial-temporal constraints are probably the most difficult to evaluate because algorithms for spatial-temporal topology have not been well developed. Identifying topological relationships between 3D objects over time is a challenging task and this research extends existing 3D computational geometry algorithms to tackle this challenge.

Literature in computational geometry has documented concepts and algorithms in handling 3D relationships. Existing 3D spatial algorithms can be used to identify fundamental relationships of intersections, containments and disjunctions among different geometric primitives commonly found in 3D GIS (Egenhofer and Herring, 1994¹). Being able to identify intersections between 3D objects is critical in determining

¹ Information of the 3D geometric algorithms can be found at: <http://paulbourke.net/geometry/>; <http://www.geometritools.com/Documentation/Documentation.html>; <http://java3d.java.net/>; <http://www.softsurfer.com/Archive/>

more complicated topological conditions, such as “on the edge” and those listed in Table 7.

Table 7 Selected 3D topological relations (*: Planar polygon)

	Point	Line	Polygon*	Cube	Polyhedron
Point	Equal	On line	On surface	Inside	Inside
	Disjoint	Disjoint	Disjoint	Outside	Outside
Line		Intersect	On surface	Inside	Inside
		Disjoint	Intersect	Intersect	Intersect
			Disjoint	Disjoint	Disjoint
Polygon*			Intersect	Intersect	Intersect
			Disjoint	Inside	Inside
				On Facet	On Facet
Cube				Intersect	Intersect
				Inside	Inside
				Disjoint	Disjoint
Polyhedron					Intersect
					Inside
					Disjoint

These extended 3D geometric algorithms are designed not only to evaluate spatial topological relationships, but also are used detect spatial-temporal topological relationships (Table 8, Erwig and Schneider 1999). Multiple spatial-temporal relations can be found between two moving objects. One of the most difficult one to detect is spatial-temporal intersection when an exact “collision” occurs. Collision detection usually involves starts with a refinement process that detects possible collisions followed by “exact collision detection” that finds the spatial-temporal intersections (Cohen et al., 1995; Hubbard 1996). Spatial-temporal disjoint can be determined at the refinement stage

(e.g.: the trajectories of two moving objects do not intersect with each other) or during the “exact collision detection”. Similar to collision detections, spatial-temporal containment that one object completely falls into another object at a time point or time interval can be implemented with a two-step approach: refinement and exact containment. Instead of identifying intersections, the spatial-temporal containment examines possible containments and verifies the containment corresponds to a temporal overlap. Therefore, implementing collision detections can facilitate the identification of all other spatial-temporal relations. According to Jiménez et al. (2001), four general approaches have been implemented for collision detections, including “spatio-temporal intersection”, “swept volume interference”, “multiple interference detection” and “trajectory parameterization”. In all four approaches, 3D geometric algorithms are used to identify the spatial relationships between spatial-temporal objects (consider a trajectory is special type of swept objects).

Table 8 Selected spatial-temporal topological relations

Temporal relations when the spatial relation detected	Spatial relations of spatial-temporal objects		
	Containment	Disjoint	Intersection
Before/After/Meet	Spatial-Temporal Disjoint	Spatial-Temporal Disjoint	Spatial-Temporal Disjoint
Equal; Overlap; Start; Finishes; During	Spatial-Temporal Containment	Spatial-Temporal Disjoint	Spatial-Temporal Intersection

Given the various geometric primitives of 3D spatial-temporal objects, this research designs the process of detecting spatial-temporal relations based on “swept

volume interference”. The sweeping process creates a swept object, which is defined as the spatial coverage of a spatial-temporal object when the object moves through a 3D space (Parida and Mudur 1994). Relations identified between swept objects are spatial relations without imposing temporal constraints. For example, a spatial intersection found between two flight paths, which are the swept objects of two flight objects over time, does not imply that the two flights travelled at the same time, but they did share part of the flight path at different time. The geometric characteristics of a swept object are determined only by the behaviors and spatial dimensionality of the moving object (Abdel-Malek et al., 2001).

Multiple approaches can be used to create swept objects, such as deriving a mathematical equation for contour-spine volume (Parida and Mudur 1994). I assume a linear movement of spatial-temporal objects in between time points when I developed a function to create a swept object. Rotations of objects are not considered. Due to the limitations of the linear movement assumption, more sophisticated approaches in developing the swept objects may be more accurate and realistic. However, as the objective is to demonstrate the utilities of using the swept object concept to evaluate topological relationships between spatial-temporal objects, the simplistic linear assumption is sufficient without losing the generality of the concept. Table 9 shows four basic spatial-temporal objects and their corresponding swept objects that the system can create and handle. Given the different geometric characteristics of the swept objects, the system will perform appropriate 3D geometric algorithms to detect the spatial relationships between these swept objects accordingly.

Table 9 Selected spatial-temporal objects and possible swept objects (*: Assuming linear movement, no rotations)

Spatial-temporal object	Swept Object	Creation*
3D moving point	line segments	Connect individual points in a temporal order
3D moving line segment	surface	Connect end points in a temporal order
3D moving polygon	Prism	Connect the vertices in a temporal order and build facets to close the prism
3D moving polyhedron	3D polyhedron	Connect the vertices in a temporal order, create bounding volume and merge inner polyhedrons when necessary

While each of these 3D geometric algorithms performed on swept objects is quite efficient, putting them to work together to detect 3D spatial-temporal relationships accurately, is not a trivial matter. The process varies with geometric primitives and types of changes of spatial-temporal objects. In this research, I mainly focus on detecting spatial-temporal relations in three scenarios: a) two moving points; b) one moving object and one object changing spatial properties at discrete time points; and c) two objects changing spatial properties at discrete time points. In all three scenarios, the behaviors of moving objects are linear movements and maintaining constant speed between any two consecutive versioned objects. The spatial information (position) SP_x at any time point T_x^p between i_{th} and $(i+1)_{th}$ of a moving point is derived with the following expression:

$$SP_x = \frac{SP_{i+1} - SP_i}{T_{i+1}^p - T_i^p} \times (T_x^p - T_i^p) + SP_i;$$

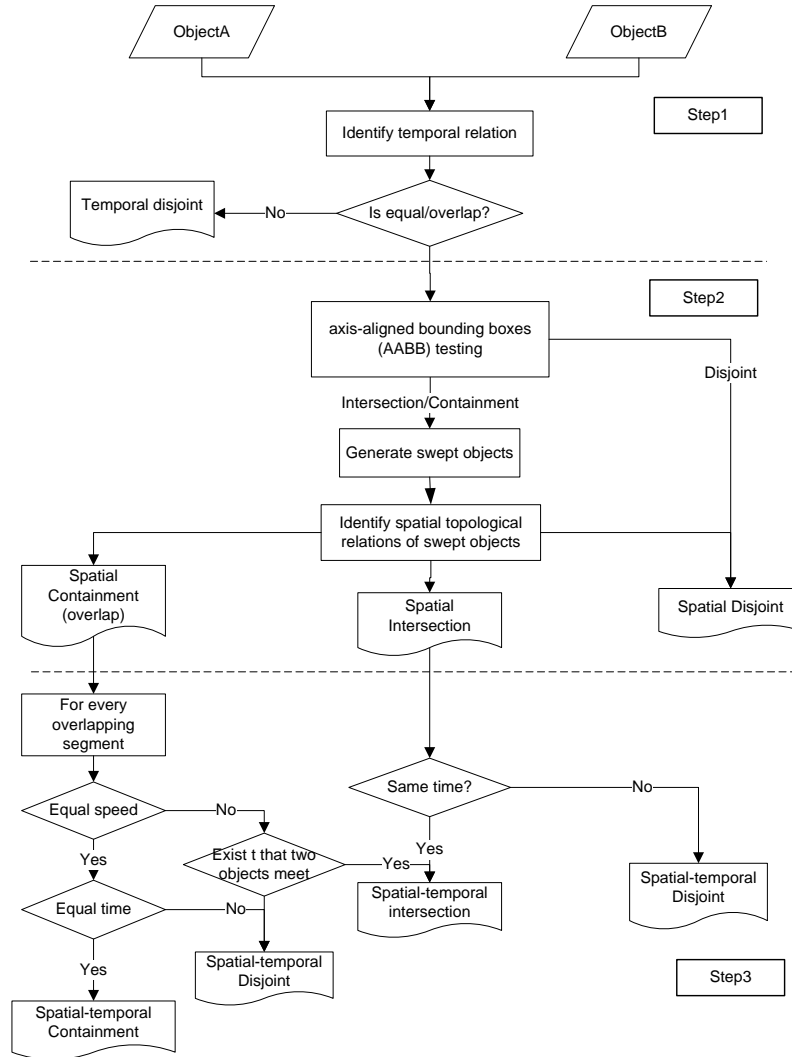


Figure 5 A 3-step identification process of spatial-temporal relations between two moving points

I propose three major steps in executing the 3D geometric and related algorithms to ensure the correct identification of spatial-temporal relations between two moving points (Figure 5). The first step is to evaluate the spatial-temporal objects with the

temporal constraints. Assuming that the objective of the query is to identify features “overlap” in space and a time period, say between t_a and t_b , a temporal constraint will be issued first to identify objects having temporal overlaps during that time period (note that relations other than “overlap” follow the same logic). If a temporal topology, such as “after” instead of a time period is used in the temporal constraint, a temporal range needs to be interpreted from the temporal topology. The temporal range will be used to determine if a temporal overlap between two spatial-temporal objects exists. If a temporal overlap is not identified, I may conclude that no object meets the temporal constraints, and the query will return no results.

When a temporal overlap is identified, the second step is to determine the topological relations between the swept objects derived from the two spatial-temporal objects. To further improve the efficiency, an Axis-Aligned Bounding Boxes (AABBs) testing is performed to identify a spatial-temporal disjoint (van den Bergen 1998). Recall that a “swept object” is the 3D geometric shape depicting the spatial extent of that object in 3D over time. In other words, the 3D extents of the object over time are all projected to one 3D space.

If one or more intersections are found between the swept objects, the two 3D objects may intersect in space and time. For example, an intersection between the swept objects of two flight trajectories implies that the two flights crossed the same location within the specified time period, but may be at different time. In other words, the results from this step may be used to answer a query such as “identifying flights with intersecting trajectories between 1:00AM to 2:00AM within a specific airspace sector”. If

no intersection is found, then further evaluations are needed to determine whether containment or disjunction relation exists between the two generic objects.

After one or more intersections are found between swept objects, the third step is to implement “exact collision detection”. The intersection between swept objects does not imply a collision between these two objects (Jiménez, 2001). This final step maps the intersections of the two swept objects to the temporal dimension for further evaluations. Time periods/points covering the swept object intersections can be extracted from the two spatial-temporal objects to examine when they crossed the intersecting point or collocated within the intersecting region. For instance, the two one dimension swept objects of two 3D spatial-temporal objects intersect at the 3D location (x_0, y_0, z_0) . If the two objects arrived that location at time t_a and t_b , respectively, these two spatial-temporal objects missed each other if $t_a \neq t_b$. If the difference between t_a and t_b is less than a threshold value, then the two objects might have established a “visual contact”. If t_a and t_b are very similar or identical, then they ran into each other. If the two objects were aircrafts, then a collision would have occurred.

If overlap segments are identified from two trajectories, the two moving points may overlap in space and time. Further evaluations are performed to verify spatial-temporal overlapping. If the overlapping segments consist of at least two continuous line segments from more than two versions, the evaluations are performed on each segment of overlapping segments. If two objects maintain the same speed at any time when moving along the segment and start and finish the movement along that segment at the same time,

a spatial-temporal overlapping is identified. Otherwise, the spatial-temporal intersection testing is performed to find a time point t that:

$$t = \frac{v(A) \times t_s(A) - v(B) \times t_s(B)}{v(A) - v(B)} ;$$

$$t \in [\min(t_e(A), t_e(B)), \max(t_s(A), t_s(B))]$$

where v is the speed of an object;

t_s is the start time point when the overlapping condition starts;

t_e is the end time point when the overlapping condition ends;

If such time point is found, two objects meet in space and time. Following these three major steps in analyzing the geometric relationships between the swept objects, the spatial-temporal relationships of two points moving in 3D space can be determined.

Besides identifying the spatial-temporal relations between moving points, the 3-step algorithm can be modified to identify such relations between a moving object and an object changing spatial properties at discrete time points (Figure 6). Since *Object B* changes at discrete time points, every versioned object can be treated as a static object and a conditional sweeping is applied to *Object A* only. For every versioned object of *Object B*, B_i , its temporal information is a time period T_i^p . Based on T_i^p , the conditional sweeping connects the versioned objects of *Object A* within that time period to generate a conditional swept object rather than all versioned objects. Then the algorithm identifies the spatial relations between that swept object of *Object A* and B_i . Due to the static nature of B_i , the spatial containment and spatial disjoint conditions identified at this stage are also considered as spatial-temporal containment and spatial-temporal disjoint. However,

potential spatial-temporal disjoints and containments may not be fully detected after spatial intersections are found in step 2, which requires further refinements.

The geometric algorithms discussed in Table 7 can be modified to identify spatial-temporal relations between two objects changing spatial properties at discrete time points (Figure 7). For every versioned object of *Object A*, A_i , its temporal information is a time period T_i^p . Based on T_i^p , a set of versioned objects of *Object B* within that time period are selected as a new set, B' . For each B_k from B' , the algorithm perform geometric computations to identify the relations between A_i and B_k . Spatial relations identified in this process reflect the spatial-temporal relations.

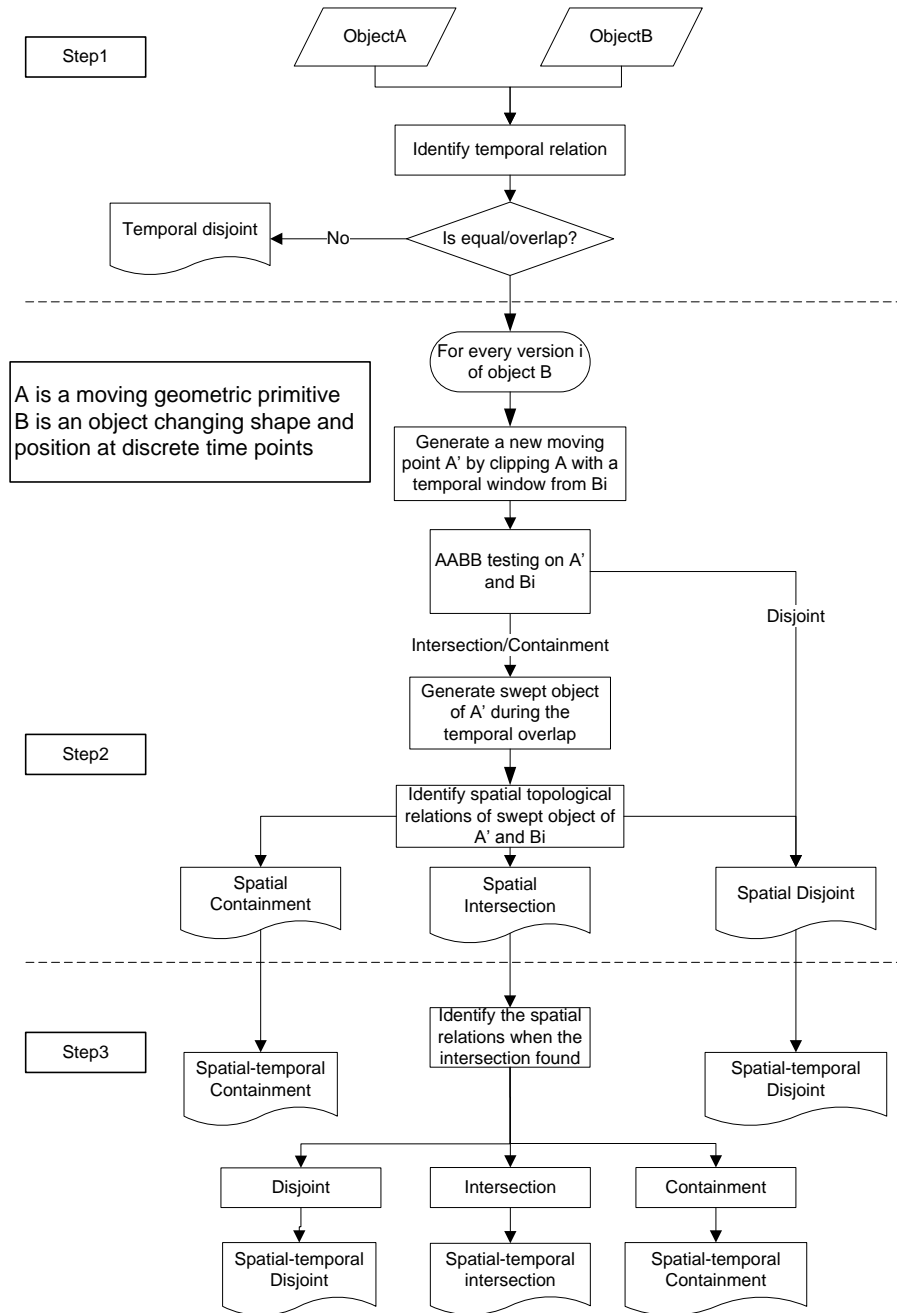


Figure 6 Identifying the spatial-temporal relations between moving objects and objects changing spatial properties at discrete time points

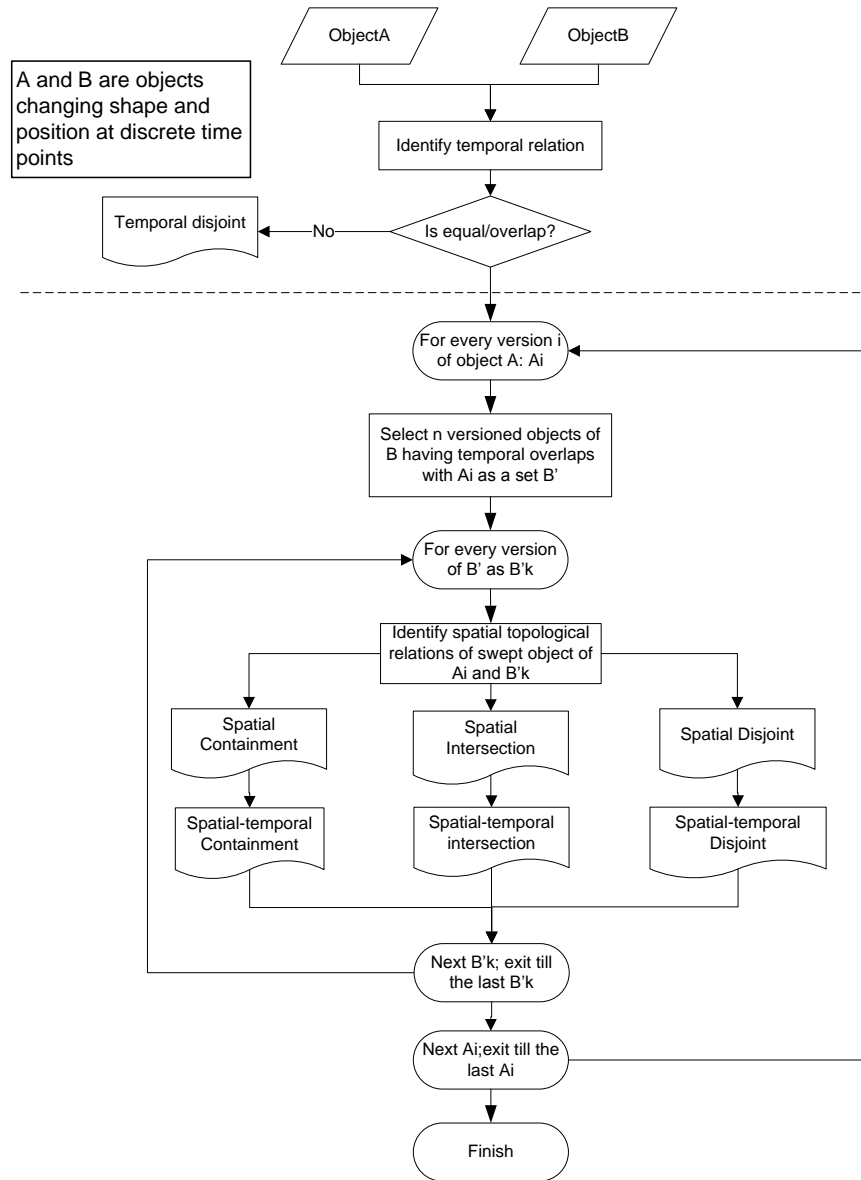


Figure 7 Identifying the spatial-temporal relations between two objects changing at discrete time points

4.2 Design and Implement a 3D Spatial-Temporal Prototype System

As a proof of concept, a prototype system was designed and developed to implement concepts and ideas discussed earlier to manage 3D spatial-temporal data using a constraint-based approach. Rules and restrictions on feature objects are described as constraints to ensure to data integrity. In addition, spatial-temporal queries can be expressed as constraints and processed by constraint computing techniques. The system is able to handle 3D spatial-temporal geometric feature types of points, lines, polygons and certain types of polyhedrons. The temporal data types are time points and time periods. The system includes four main modules, supporting three major types of functions: 3D rendering, data management and interactive querying. Many operations in the data management and interactive querying modules need to take constraints into consideration. Therefore, a constraint management module is also developed to maintain the constraint subsystem.

4.2.1 Data Format

In developing the system, I evaluated viable data formats, which define how data are organized and stored physically. The two popular formats are raster and vector. An important criterion to select a data structure is whether the chosen structure can support typical operations to be performed by the system. As spatial-temporal queries and analyses will be the major types of operations, the effectiveness of data structure in handling spatial topological relationships is critical. Although raster data structure can handle certain types of topological relations in 2D space, in general, it is not very effective to deal with topological analysis. When raster structure is extended to a 3D space, several options such as octree structure are available. However, their effectiveness

in handling topological relations is limited. On the other hand, several topological models have been identified, capturing and representing topological relations of data in vector formats in both 2D and 3D (e.g., Zlatanova et al., 2004). Therefore, in this study, the vector data format is chosen as the data format and functions are implemented based on such data format.

4.2.2 Core Modules

The four core modules are described in Figure 8. The 3D rendering module supports fundamental capabilities of rendering objects in 3D space found in most 3D visualization platforms. It provides the environment to simulate movements of objects based on the spatial-temporal data stored in the database. The graphical display environment in the module also provides an interface through which users can interact, manage and modify 3D features, such as dragging objects to new locations during the editing mode. The default mode of display, which provides a static 3D view of objects, serves as the graphic interface through which the 3D data can be managed interactively. Animation, space-time trajectory and space-time matrix are additional display modes supported by the module to visualize and compare the dynamics of objects over time. These three display modes are very popular techniques for rendering 3D spatial-temporal data in existing 3D systems.

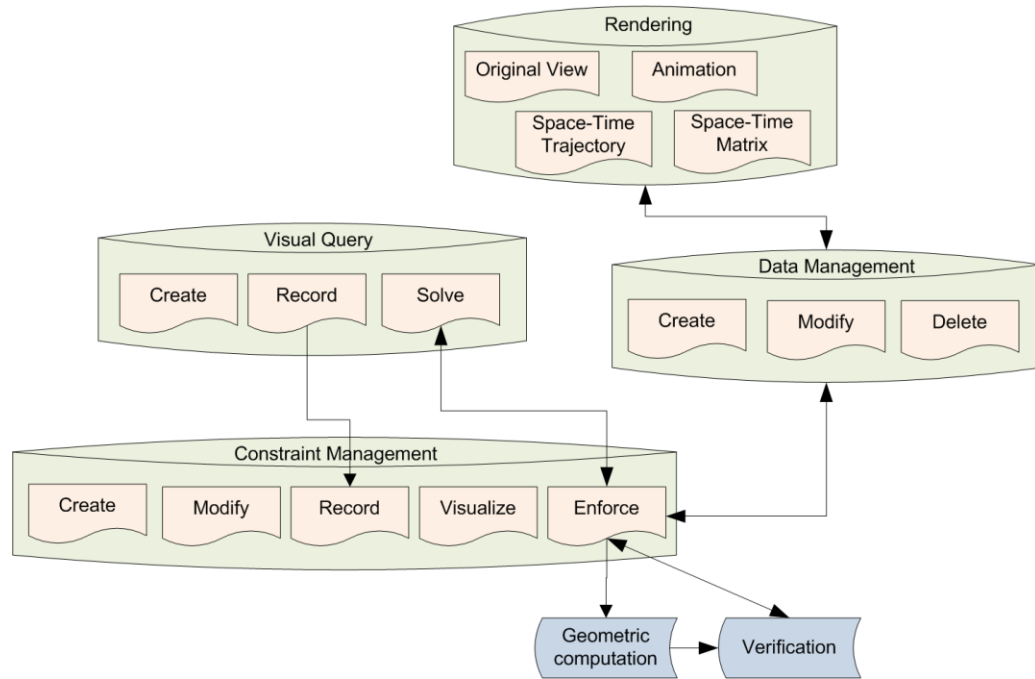


Figure 8 An overview of the system architecture

3D Rendering: Frequently used for displaying 4D data (3D plus time), the animation mode renders data at each time point or period in a temporal sequence but displays the visual products continuously as a movie. The space-time trajectory mode shows the trajectories of moving objects. Unlike the animation that simulates the movements between time steps, the space-time trajectory mode establishes the linkages between any two versions of the objects at consecutive time points by connecting their positions in these two versions (Figure 9). In other words, the display in the space-time trajectory mode is a static graphic showing the tracks of objects, while the animation shows the object positions in each frame of display. To a large extent, the space-time

trajectory displays are the swept objects of 3D objects. The space-time matrix mode uses a panel of four sub-windows to display the 3D temporal objects in specific time frames. In the sub-windows, user specifies time points or periods such that objects that are present during the specified periods are rendered in corresponding windows. This mode is primary used to investigate the positions of objects in specific time frames or to compare the changes of objects at different time frames.

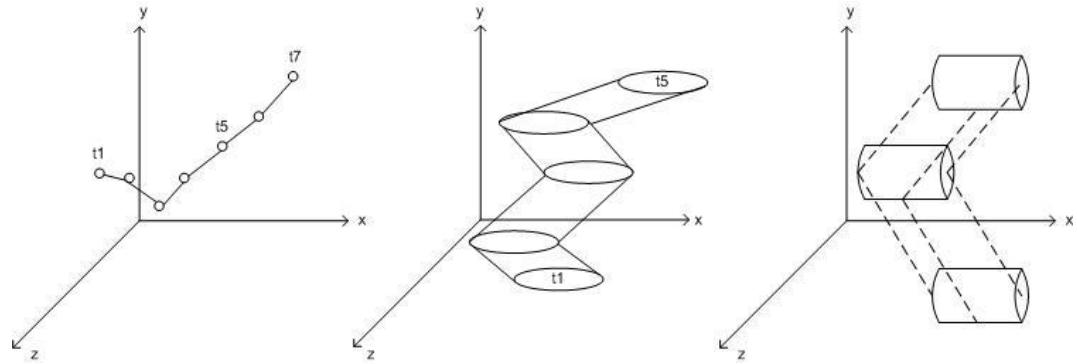


Figure 9 Space-time trajectories generated from different spatial-temporal objects

The applicability of different visualization and rendering methods depends on the nature and characteristics of the spatial-temporal objects. Animation is a universal approach to represent dynamics this approach provides limited information about objects. Space-time trajectory can show the movement of an object during its lifespan. When an object does not move but change its shape over time, the space-time matrix can be used to show the differences of the geometric characteristics of two versioned objects. Complementary to the default view, the three modes offer continuous and dynamic

(animation), continuous-static (space-time trajectory) and discrete (space-time matrix) views of the spatial-temporal objects.

Data management: The data management module is considered to be the core module of any GIS. This module provides functions to allow users to create, edit and delete objects at the front-end for the module. Modifications of objects also invoke a data updating process in the data management module at the back-end. To ensure that data describing objects are correct, following the relevant operating rules depicted by the constraints, the constraint management module interacts with the data management module to maintain data integrity. The functions to manage attribute values are not developed because this research attempts to demonstrate how topological integrity, the most difficult one among all constraints discussed in this dissertation, is maintained during the interactive manipulation.

The creation process of 3D spatial-temporal objects is similar to digitizing features in cartography but the difference is that constraints are enforced during the creation process. The system supports the creation of geometric shapes such as 3D point, 3D polyline, 3D polygon, cube, sphere and polyhedron closed by several planes. Both time point and time interval are supported as temporal properties of the objects. When a user finishes creating an object, the new object is a candidate object in the system and subjected to integrity checking. Spatial-temporal algorithms are applied to that object and existing objects in the system to derive relations, both spatial and temporal. These relations are then compared with the constraints to make sure that the candidate object

will not negatively affect the system integrity. If not, the candidate turns into an object within the system.

As for editing, the process starts when a user selects an object from a management tree. The status of the selected object is changed in to the editable status, which is ready for interactive editing. User can move points of the geometry to change the shape and location of the object. The edited parts of the feature will be updated in the system simultaneously. Upon finishing editing, the object is changed to a candidate object and a constraint verification process is performed as in the object creation process.

Different from creating and editing an object, deleting one or more objects will not invoke geometric computations but constraints are updated within the system. Upon removing objects from the data system, any constraint specific to that object is deleted from the constraint system. If a constraint is shared between the deleted object and any other within the system, such constraint will not be removed because other objects should still meet the conditions specified by the constraint. The non-overlapping constraint is one example of such constraint. In this way, system integrity is maintained when users create, delete and edit objects/constraints.

Constraint management: Although previous constraint-based systems provide interfaces for users to specify constraints, these interfaces are far from adequate. Interactive creating, editing and deleting constraints in the system are necessary to assist inexperienced users to manipulate these constraints. Users should have the options to trigger constraint enforcement. Once the evaluation process is finished, the system should report the results of enforcing constraints in an intuitive manner. (Louwsma et al 2005).

The constraint management module maintains a subsystem recording all constraints imposed onto the objects. To facilitate interactions between users and the system in respect to constraint management, two graphic user interfaces (GUIs) have been developed: constraint editor and constraint graph (Figure 10). Using the constraint editor, users can first select an object, and then specify any of the four types of constraints to be imposed. Once the constraints are defined or modified, the constraint management system will update the constraint records accordingly. The constraint graph interface displays constraints between objects via a linked graph. By default, all objects in the system and constraints can be displayed on the graph through which users can view all constraints and linkages among objects. By selecting objects and specifying a time window, a filtered linked graph shows only constraints that are valid to the selected object within the specified time frame.

Besides, the system also offers an option for users to trigger the evaluation of constraints on demand. Once the constraints are evaluated, the system generates a report showing objects violating the constraints. In this module, users interact with constraints through the GUIs. Therefore, this module can accommodate general users with little knowledge of constraint programming.

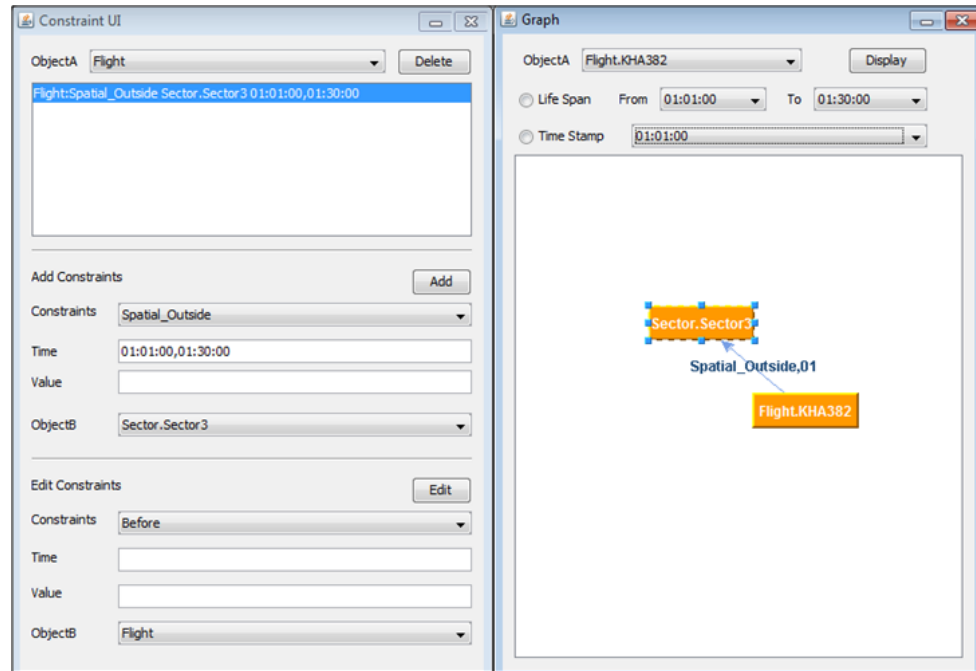


Figure 10 Constraint Management Interface a) Constraint Editor; b) Constraint Graph

Spatial-temporal querying: The query module allows users to create complex spatial-temporal queries through the visual and graphical interfaces by operating on the 3D static and spatial-temporal objects (Figure 4). Upon formalizing a query, the visual query module translates the query criteria into constraints, and constraints are submitted to the constraint enforcement sub-system at the back-end to execute the constraints. Using various spatial-temporal algorithms, including those extended spatial algorithms I have implemented to analyze spatial-temporal relationships, the constraint management module determines which objects meet the requirements described by the query as part of the constraint enforcement process. If the query involves identifying objects, these

objects will be highlighted in the default view once the enforcement process is completed.

4.2.3 Development Platform and Related Issues

The prototype system was developed using Java Development Toolkit (JDK) and multiple Java-based open source libraries. Thus, this system can operate under different operation systems. Two types of files are created: the files in the Extensible Markup Language (XML) format store data and the files in the OCL format store constraints. Different from other geospatial data formats, which records only one type of geometry per file, the XML format is able to accommodate all types of geometry within a file. In this prototype, a simplified version of geospatial XML is developed to record the spatial and temporal properties of an object. On the other hand, OCL describes the rules applied to objects and records the query criteria.

CHAPTER FIVE A CUSTOMIZED SYSTEM SUPPORTING DAC

This chapter illustrates how the proposed framework and prototype system can be customized to facilitate the Dynamic Airspace Configuration (DAC) process, an emerging problem in air traffic management systems involving 3D spatial-temporal objects. I will first briefly explain the problem of DAC and review existing approaches to solve such problem in Section 5.1. To facilitate the DAC problem, the first step is to contextualize the OO model to describe the major spatial-temporal objects relevant to the DAC process and a few critical constraints pertained to these objects. Such conceptualization process will be discussed in section 5.2. Based upon the conceptualization, the generic prototype system has been customized in several aspects, which will be discussed in Section 5.3.

5.1 Background

The DAC has attracted substantial attention recently in the air traffic management with an objective to explore and identify solutions to configure flexible, dynamic and adaptable airspace based on traffic demands, weather and other factors (Kopardekar et al., 2007). The three major tasks of DAC are a) organizing/initiating the configuration of airspace for existing situation. By making use of techniques such as self-separation, airspace sectors can be restructured and categorized by the specific allowable operations. For example, the sector “corridors-in-the-sky” could be a type of high altitude sector in

which the aircrafts are responsible to conduct self-separation. This reduces the workload within the sector. b) Modifying the current configuration to accommodate new demands or new situational factors. When flights are rerouted due to changing weather conditions or other factors, some sectors may experience an overloaded condition which could cause safety issues. To equalize the controller workloads among sectors, boundaries of sectors are subjected to split (and merge) based on their capacities. c) Defining generic airspace by removing site-specific configurations. A generic airspace sector refers to the sector that can be used and managed by any controller of any facilities. The presence of such sector will improve the efficiency of monitoring because such generic characterization can minimize operational differences.

Among these three tasks, changing the boundaries of airspace dynamically is the most challenging due to various technological and human factor issues. First, the coordination of workload is difficult, requiring an accurate monitoring and calculation of workload. The workload of an airspace is directly associated with the flights within that airspace in a defined period. Overload happens when the number of flights exceeds the capacity, usually characterized by controlling workload of that airspace. Algorithms supporting the workload calculation, likely a spatial temporal one should be developed. Second, efficient management of multiple constraints enforced during the DAC process, including specifying, modifying and updating constraints, is challenging. Along with the changes in boundary, the underlying constraints may also change. The original constraints that maintain the data integrity are no longer useful when boundaries change and a new round of conflict detection for all objects in the system is necessary. Third, 4D

methods are needed to adjust the airspace when 3D trajectories are present. Fourth, scalability can be an issue especially when the number of airspace increases with increasing flights (Klein et al., 2008).

Although the DAC problem was proposed in 2007, researchers in the past decade have explored several methods that can be used to facilitate the DAC process mainly through the optimization of sectorization of airspaces. These methods include linear and mixed integer programming, computational geometry, genetic algorithms, clustering methods and heuristic algorithms (Yousefi and Donohue, 2004; Martinez et al, 2007; Delahaye et al., 2006; Klein et al., 2008). Yousefi and Donohue (2004) developed a linear and mixed programming algorithm that assigns the workload to sectors to equalize the distribution of workload among sectors. Assigning the workload to sectors can be done using weighted computation and flow graph partition (Martinez et al, 2007). The weight is generated by calculating (or evaluating) the maximum number of aircrafts within in a unit space, the node in the flow graph, and also sectors in a certain period. The weight of a graph is the total weight of each cell. If the weight exceeds the maximum capacity of a sector, the flow graph is decomposed into sub-graphs. Along this direction, the heuristic algorithm selects appropriate seed locations (such as major airports) to start the assigning process .By taking precedent constraints into account, an evolutionary algorithm was discussed in Delahaye et al (2006). The algorithm selects a series of constraints and establishes a fitness model based on the constraints to identify the sectors having the best fitness in a stochastic process. Although these methods address some aspects of the DAC problems, a systematic solution has not been developed.

Despite abundant theoretical work, a limited number of tools are available for researchers to implement the DAC process. Generic systems such as NASA's Future ATM Concepts Evaluation Tool (FACET) were developed for air traffic management (ATM). These tools usually provide functions such as data processing, visualization and animation. However, they are short of providing interactive manipulation interfaces through which users can dynamically modifying the boundaries of sectors in a 3D environment. Also the tools do not have an underlying mechanism to check the validity of the modifications. For example, the workload during an interactive modification is not available. This is partly because the corresponding 3D data management systems were not appropriately designed.

The extended OO model and the data management prototype proposed in previous chapters may provide a systematic solution to the DAC problem. First, the proposed model can describe the objects within an aviation system quite accurately. The airspace and flights are inherently 3D spatial-temporal objects. Second, the algorithms developed in the prototype system facilitate the identification of overload sectors. A critical DAC task is to eliminate excessive controller workload of an overloaded airspace sector or to combine two airspace sectors with low workloads. The maximum capacity may be regarded as a constraint of a sector, and identifying an overloaded sector is a constraint solving problem. The identification involves two steps, first calculate the workload of each sector and then compare the workload with the capacity of that sector. Third, the constraint-based modeling approach helps maintain the data integrity of an aviation system. With the verification of constraints in place, changing boundaries of

airspace during the reconfiguration will not introduce inconsistency to the system.

Therefore, this section will discuss how the proposed model can be customized to address the DAC problem.

5.2 Contextualize the Conceptual Model for DAC

5.2.1 Modeling the 3D Spatial-temporal objects

In the past several decades, agencies such as Federal Aviation Administration (FAA) and Department of Transportation (DOT) have developed various air transportation systems. The most notable system is probably the National Airspace System (NAS), which is likely evolved into the Next Generation Air Transportation System (NextGEN) in the future. The NAS is the largest national aviation system consisting of facilities, controllers, equipments and procedures used by over 5000 flights on a daily basis. In such systems, a variety of data are collected and maintained to support aviation practices, traffic controlling and decision making. In general, two types of data are provided: static and dynamic. Static data include air traffic control (ATC) sector boundaries, facilities, and air routes. Spatial-temporal data represent the availability of sectors and air routes in real time (or of specific flights). In a DAC problem, spatial-temporal data are more critical as the reconfiguration is the process to accommodate the changing traffic demands among different sectors. These demands are reflected by spatial-temporal data such as newly generated flight routes within the airspace. Therefore, airspace sectors and flight trajectories are the most important objects to be modeled in the proposed model.

With the OO modeling approach, moving points are chosen to represent the geometric properties of flight trajectories whereas the exact shape of aircrafts is not captured in the DAC process. Ignoring the shape of airplanes may introduce some minor errors when the ATM system identifies the time that a flight is entering/leaving a sector. If the shape of an airplane is explicitly recorded in the system with a 3D volumetric object (e.g polyhedron), the “entering” is a process starting from the time point that 3D volumetric object first touches the boundaries of the sector to the time the object is completely inside the sector². With a point-based representation, the “entering” happens when the moving point touches the boundaries of sector, which can be recorded as a time point. Although the point-based representation has limitations, such representation has been used in existing DAC research. As the size of aircrafts is much smaller than the volume of airspace sectors, aircrafts can be treated as moving points (e.g. Delahaye et al., 2006).

In the ATM system, each flight trajectory consists of multiple flight tracks sorted by their timestamps. Each track is a versioned object of that flight represented as a spatial-temporal point p with a time stamp t . Assuming that the geometric structure of airplanes does not change, the behavior of flights is set to “translation”. For any flight A with n tracks, the i_{th} track at time t can be described as

$$VO_i = \langle SP_i, T_i, B_i, A_i \rangle \text{ where}$$

$$SP_i: \langle x, y, z \rangle; T_i: t; B_i: \text{Translation, linear movement}; A_i: \text{attributes}$$

Then the trajectory of flight A is:

² Entering time ~0.2 seconds based on the aircraft data for Airbus A330-200: length:~59m; maximum cruise speed: 880km/h

$$O = \{VO_i \mid 0 < i < n\}$$

On the other hand, solid shapes (e.g. polyhedron) are used to represent airspace sectors. The reconfiguration process requires geometric changes at an individual time point, involving an increase or a decrease in the size of the sector and changes of the geometric shape and topological connectivity of sectors. Therefore, time period is used to record the temporal information of a versioned sector. Different sectors have different geometric properties. For middle-high altitude sectors, where the DAC occur, the geometric properties can be described by a 3D right prism. A 3D right prism is a special geometric shape of which the footprint and the top are polygons and all facets are perpendicular to the top and base polygons.

For any airspace sector S has n versioned objects. Each versioned object starting from t_1 to t_2 of sector S is recorded as:

$$VO_i = \langle SP_i, T_i, B_i, A_i \rangle \text{ where}$$

$$SP_i: \langle \text{polygon}_i, h_i \rangle, h_i \text{ is the height of the right prism;}$$

$$\text{polygon}_i: \langle p_1, \dots, p_k \rangle, p_k \text{ is the } k_{th} \text{ 3D point of the base polygon;}$$

$$T_i: \langle t_1, t_2 \rangle ;$$

$$B_i: \text{Geometric and topological changes;}$$

$$A_i: \text{A list of attributes;}$$

5.2.2 Describing the Constraints

Within the National Airspace System (NAS), operational rules or constraints are attached to different flights and sectors. For example, the minimum distance between two flights should be larger than a threshold value. Flights in Class A sector, which a layer

from Flight Level³ (FL) 180 to FL 600, should file an Instrument Flight Rules (IFR) flight plan. No flights are allowed to enter the Special Use Airspace (SUA) in a certain time period. Related to the DAC process, I summarize the following constraints.

The most important constraint is the maximum workload among sectors which can be treated as an attribute constraint for sectors in operation. This constraint ensures that the maximum workloads across all sectors should be smaller than a threshold workload level. This threshold value can be adjusted by users via the constraint editor. When modifying the boundaries of sectors, three topological constraints are needed (Figure 11): 1) Subdivision of space: no two sectors are allowed to overlap; 2) Boundary of a polygon (airspace sector) cannot intersect itself (i.e., no self-intersection); 3) Boundaries of sectors can be changed only horizontally such that right prisms can always be maintained. In other words, heights of sectors (right prisms) cannot be changed. Although the third constraint is not required in the prototype system, it is preferred in a DAC process to avoid the difficulties in dealing with varying elevation restrictions within a sector. While these three constraints pertaining to airspace sectors could be enforced for selective periods in the system, they were enforced in all time to maintain the geometrical accuracy of the data. Therefore, the three topological constraints are in fact spatial-temporal constraints.

³ A Flight Level (FL) is a standard nominal altitude of an aircraft, in hundreds of feet.

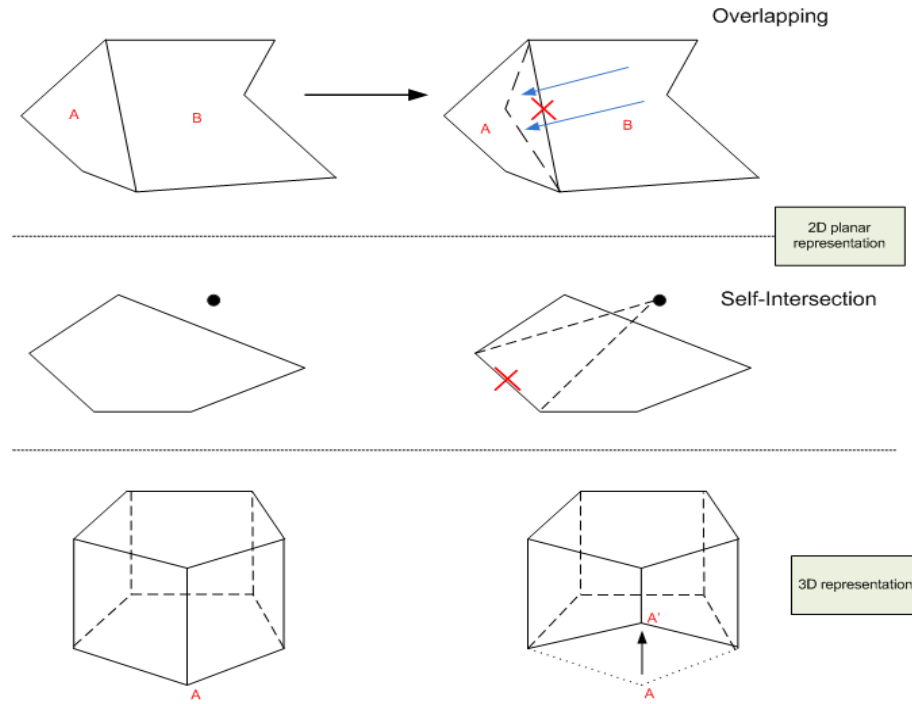


Figure 11 Three constraints enforced when changing the boundaries of sectors

Using the definitions in Section 3.3, constraints are recorded as:

Constraint 1: Any sector: Attribute: Self. Workload \leq Twl

Constraint 2: Any sector: Spatial: Self.Geometry. Not_Intersect self.geometry

Constraint 3: Any sector: Spatial: Self.Geometry.Right Prism

Constraint 4: All sectors: Spatial: No intersection

A real time system should monitor the workload of each sector constantly. The reconfiguration is triggered when the workload within a sector exceeds the maximum value of workload. Other integrity constraints are enforced when a reconfiguration process starts. Figure 12 shows a diagram using Unified Model Language (UML) to describe the airspace sectors, flights and critical constraints involved in the

reconfiguration process. This figure also summarizes the descriptions of objects and constraints involved in the DAC process.

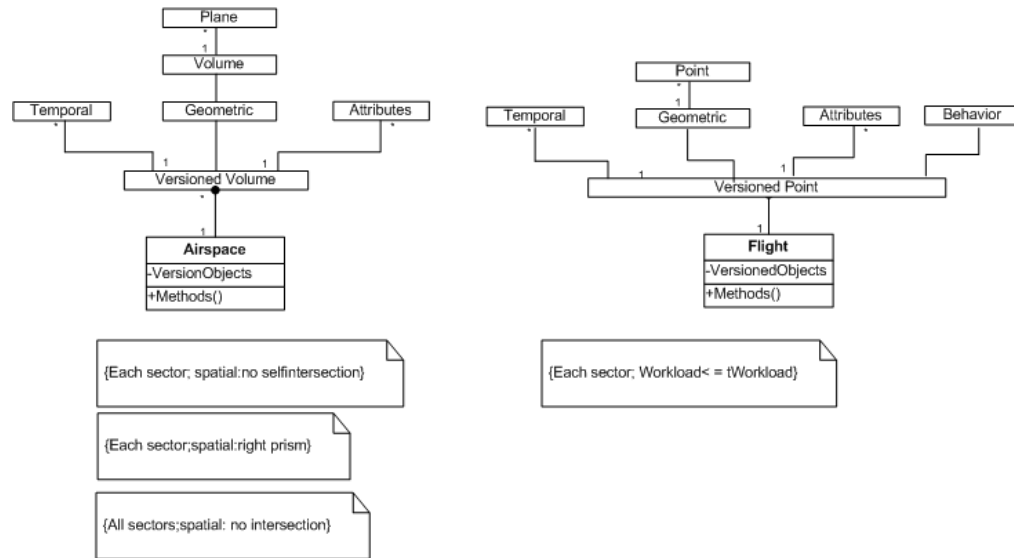


Figure 12 A simplified UML diagram of objects in DAC ("*" denotes the 1-N condition)

5.3 Customizing the Prototype System to support DAC

The prototype system described in Chapter 4 was customized to facilitate the DAC process. The customized visual analytical system is not intended to replace the existing operational traffic control systems, but to show how the implemented spatial-temporal framework may facilitate the DAC process and thus the implemented methods may contribute to the development of the NexGen. This prototype system also serves as an exploratory system that system developers and planners can implement and evaluate the modifications within the simulated DAC process before these modifications are

adopted for and deployed to daily operations. Lessons learned during the process will also contribute to the development of generic 3D spatial-temporal GIS.

A typical DAC scenario is set up as follows using data of airspace sectors in the Washington, DC metropolitan area and simulated flight tracks. I first set the maximum workload level for each sector. Then, the system calculated the workload for each sector. By enforcing the workload constraint, sectors with workloads exceeding the maximum levels were identified and highlighted in the display to be reconfigured. These overloaded sectors were reconfigured by the user via the data management interface. Meanwhile, geometric computation algorithms were executed repeatedly to update the workload status of each sector. The process of reconfiguring the airspace sectors that violated the constraints initially continued until workloads were below the maximum levels in all concerned sectors.

5.3.1 Data

In this demonstration, two major datasets are used: flight tracks data and boundary data of airspace sectors. Due to the sensitivity of real-time flight data, I used flight track data simulated from the FACET (Bilimoria et al., 2001). For each flight track, simulated data items include flight ID, latitude, longitude, elevation and time. Flight tracks sharing the same flight ID formed the trajectory of that flight, and moving points were used to represent flight tracks and trajectories. For airspace sectors, I compiled data of the ATC boundaries and extracted six high altitude sectors where the DAC process takes place. These six sectors cover the three major airports in the Washington, DC metropolitan area. Each sector was represented by the geometric primitive of a right

prism, a convex or non-convex polygon extruded to a certain elevation (Figure 13). The horizontal coordinates of the polygon boundary, the minimum elevation, and maximum elevation were used to create the right prism.

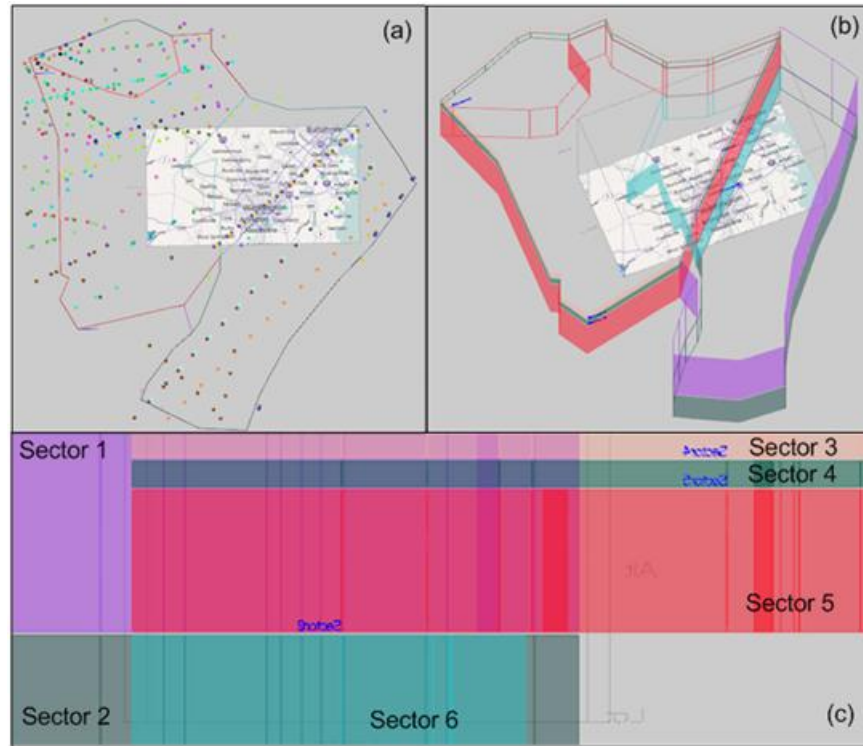


Figure 13 Snapshot of airspace sectors and flights in the 3D default view (a) view from above; b) oblique view from the upper southwest corner; c) side view from the north)

5.3.2 Customizing Geometric Algorithms

The algorithms discussed in Chapter 4 served as the basis of developing domain specific spatial-temporal algorithms. Given the geometric characteristics of flight trajectories and airspace sectors, to enforce the workload constraints, the first step is to identify the levels of workloads of each sector. Several definitions of workload can be

found in Yousefi and Donohue (2004). In the prototype system, workload level of a sector is defined as the number of flights within a sector at a given time point or period. Calculating the workload of a sector becomes a process of determining the relationship between spatial-temporal points, which represent the 3D locations of flights at a given time (the positions of flights are reported every minutes), and right prisms, which represent airspace sectors. The algorithm detecting the relationship between a 3D point and a 3D planar polygon was adopted here. The process of calculation is described in Figure 14. If the spatial relation of “inside” is true between a point and a right prism at the specified time point or period, the flight is counted toward the workload level of that sector. After calculating the workload levels of all sectors, the system enforces the workload constraints by comparing the actual workload levels with the maximum and minimum levels of each sector. After evaluating all relevant constraints, the system generates a report showing which sectors are overloaded at specific time points.

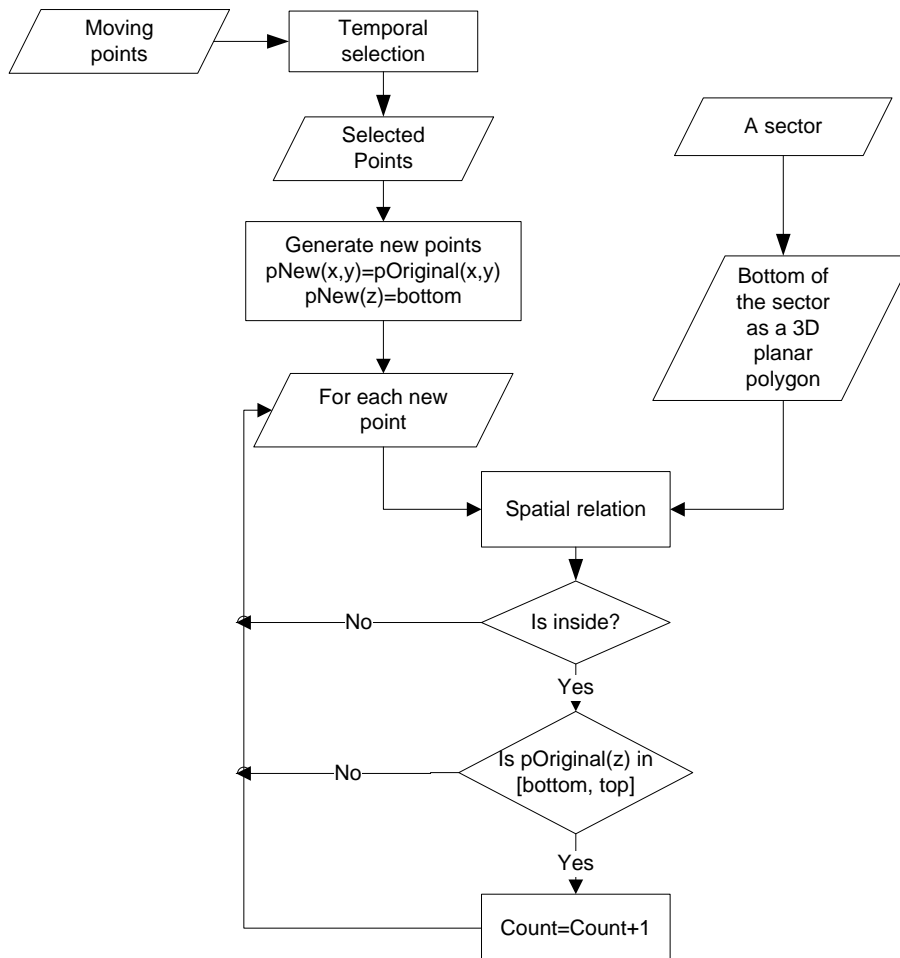


Figure 14 Calculating the levels of workloads

To ensure that airspace sectors satisfied the topological integrity constraints, I modified existing algorithms for detection of self-intersections (Nievergelt and Preparata 1982) and intersections between polygons described in Chapter 4 (Figure 15). When the boundary of a sector is changed or modified such as adding a new vertex to the base polygon of a sector, the self-intersection algorithm is triggered to examine if any segment

of the sector boundary intersects with another segment of the same boundary. If one or more intersections are found, the modification is considered as invalid.

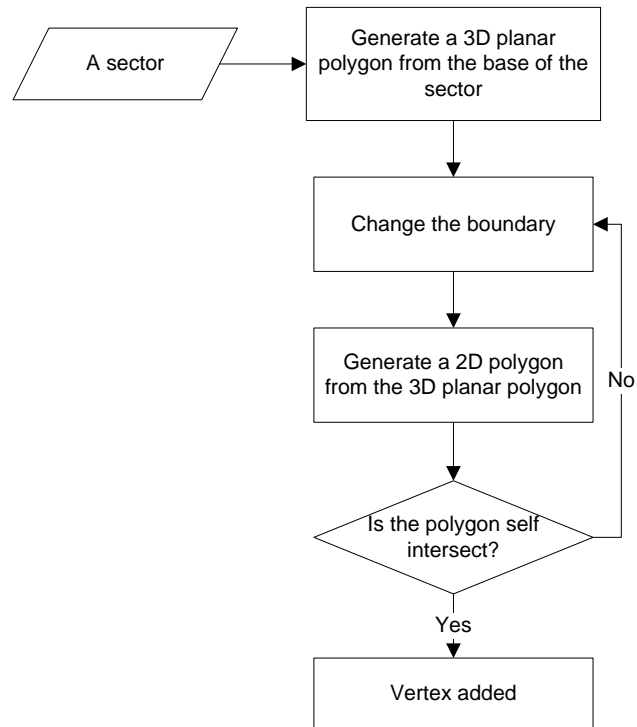


Figure 15 Enforce the non self-intersection constraint

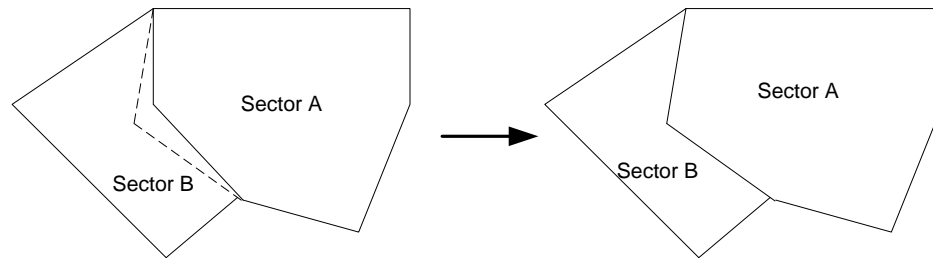


Figure 16 Change the shared boundary with polygon clipping

Meanwhile, another process is performed to detect the intersections between two sectors. Containment is not considered due to the nature of reconfiguration, which is a process changing the boundaries of sectors gradually. How to determine if the modification is a significant change or not and condition the modification is beyond the scope of this research. As sectors are subdivisions of airspace, an exception with the intersection detection is that expanding the boundary of a sector always leads to an intersection with its adjacent sectors. When users change the boundary of a sector, its shared boundaries of other sectors are updated (Figure 16). Therefore, a polygon clipping algorithm was applied to base polygons of two adjacent sectors before implementing the intersection detection (Sutherland and Hodgman 1974). The workflow of enforcing no intersection constraint between sectors is described in Figure 17.

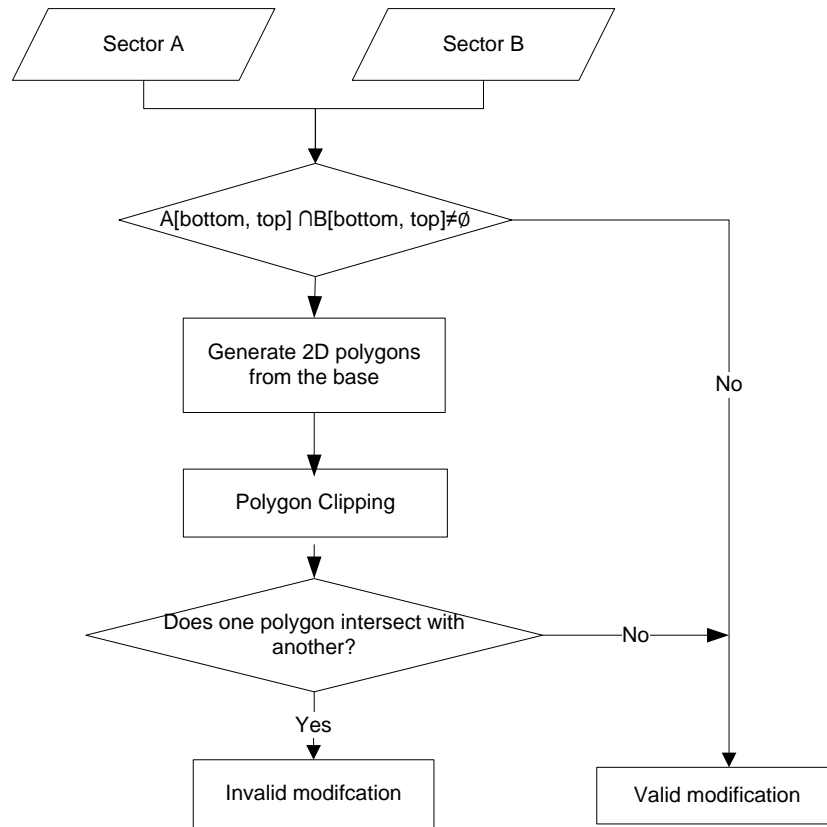


Figure 17 Enforce no intersection constraint between sectors

The last integrity constraint, maintaining the right prism during modifications, is enforced by a coordinate transformation. When a boundary is modified, the system keeps the elevation unchanged but allows the vertical boundaries to shift horizontally.

5.3.3 Calculating Workloads

Reconfiguration of an airspace sector is triggered when the sector is overloaded. Thus, monitoring the workload level of each sector constantly is necessary to determine the need and timing of reconfiguration. To identify these time points, the system may

monitor the situation by calculating the workload level in each sector every minute. This is a spatial-temporal query in determining the number of flights inside a sector. I extracted the first 3-hour data from a 24-hour simulation to demonstrate how the reconfiguration of a sector may take place. Figure 18 shows the workload levels of three selected sectors as depicted by the simulated data without imposing any workload constraint. The x-axis shows the time when the workload was calculated and the y-axis shows the levels of workload at different times. According to Figure 5, workload levels for each sector range from 0 to 6, increasing gradually after 01:00 AM. Sector “Sector5” has significantly higher workload levels as compared to other sectors at various time points.

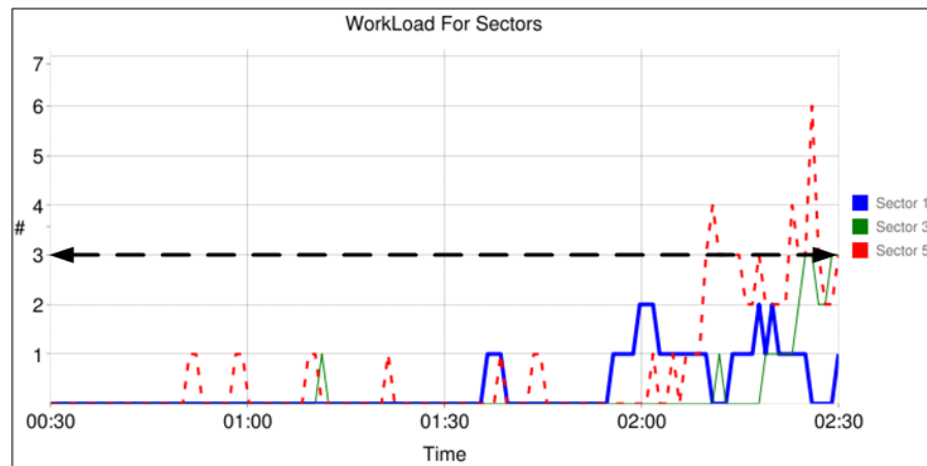


Figure 18 Workload profiles for selected sectors generated from the system

5.3.4 Reconfiguration of Airspace Sectors

The DAC process is set up as Figure 19 shows. The animation mode is used to simulate the movements of flights within sectors. Before the animation starts, the user

can create a workload constraint by setting the maximum number of workload for each sector (WL_{max}). Once the animation starts, the system will constantly calculate the workload of each sector every minute and compare the workload with WL_{max} . If a violation is detected, namely, the workload of a sector exceeds WL_{max} , the system stops animation and promotes a dialog to ask whether the reconfiguration will be implemented. Users can choose to continue the animation, ignoring the overloaded sectors.

If the user agrees to proceed to the reconfiguration process, the system will display objects for the next time step and change the objects into an editing mode temporarily. In this mode, the user is allowed to change the boundaries of any sector by adding, moving and deleting vertices of the sector. In addition, the user can discard the modifications made to sectors at any time during the editing session. During the modification, the user can choose whether the three integrity constraints should be enforced and evaluate these constraints on demand. If the user is committed to change the boundary of a sector, a new versioned object of the sector will be created of which the start and end time will be assigned by the user. The user can continue the modifications until a balanced workload among sectors is achieved. When the editing session is finished, the system will continue animation starting from the next time step.

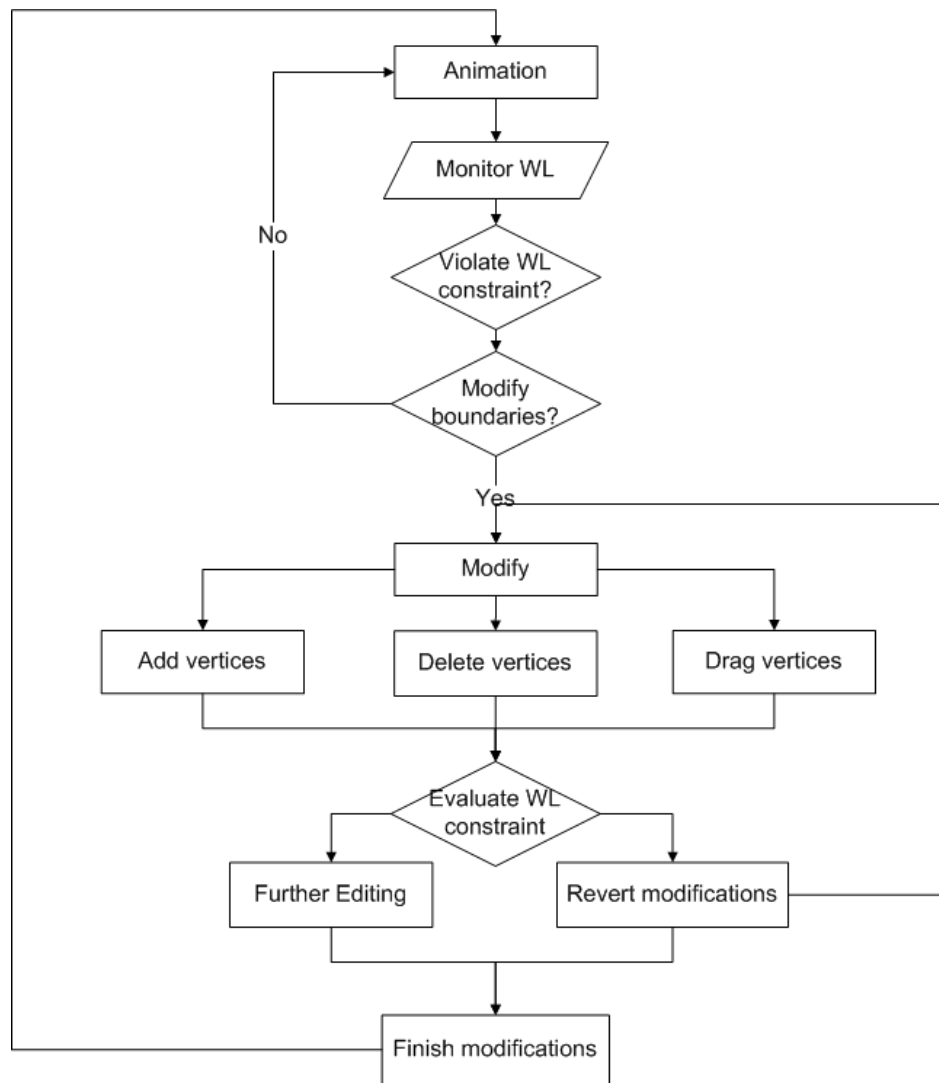


Figure 19 A simulated DAC process in the system (WL=workload)

5.3.5 System Demonstration

In this demonstration, I assume that the maximum workload of all sectors was 3 (the dotted horizontal line in Figure 18). When the system ingested the historical data as if they were real-time data, I also activated the real-time monitoring function such that

the system evaluated the workload constraint every minute. This frequency of monitoring can be adjusted. After the constraint was evaluated and a violation was found, the system created a report showing the latest result of evaluation. The report indicated that the workload level of sector “Sector5” at 2:10:23 was at level 4, violating the workload constraint and requiring a reconfiguration (Figure 20). Then the user could start the editing mode. The editing mode allows the user to add, delete and modify vertices of the footprint of the sector boundaries. Also the user can specify if constraints should be enforced during an editing session. Once finished editing, the user have the option to cancel changes made during the editing session. By default, the three integrity constraints described under section 5.3 were enforced in real time.

Sector	Current WL	Time	Max WL Allowed	Violate
Sector1	0	2:10:23	3	<input type="checkbox"/>
Sector2	2	2:10:23	3	<input type="checkbox"/>
Sector3	0	2:10:23	3	<input type="checkbox"/>
Sector4	1	2:10:23	3	<input type="checkbox"/>
Sector5	4	2:10:23	3	<input checked="" type="checkbox"/>
Sector6	0	2:10:23	3	<input type="checkbox"/>

Figure 20 A report of enforcing workload constraints before reconfiguration

In Figure 21(a), the view from above shows more than four flights in Sector 5 before the reconfiguration, but in fact, one flight was in the lower sector and another one was in the upper sector (see Figure 21(c) for a side view of the sector structure). During this period, most flights were at the western corner of the sector. When the overload was

detected, modifying the boundaries of Sector 5 can reduce the workload level of that sector at that time point.

Although the boundaries can be changed in several ways, a simple solution is to delete two consecutive vertices at that northwestern corner and to close the polygon of that sector by adding an edge. Then one flight was reassigned to its adjacent sector. During the process, the “right prism” constraint was enforced so that only the footprint of a sector was modified. Also the deletion of vertices apparently did not cause any self-intersection, nor did the new boundary of the modified sector overlap with other sectors. Figure 21(b) shows the modified boundary corresponding to the northwest corner of "Sector5".

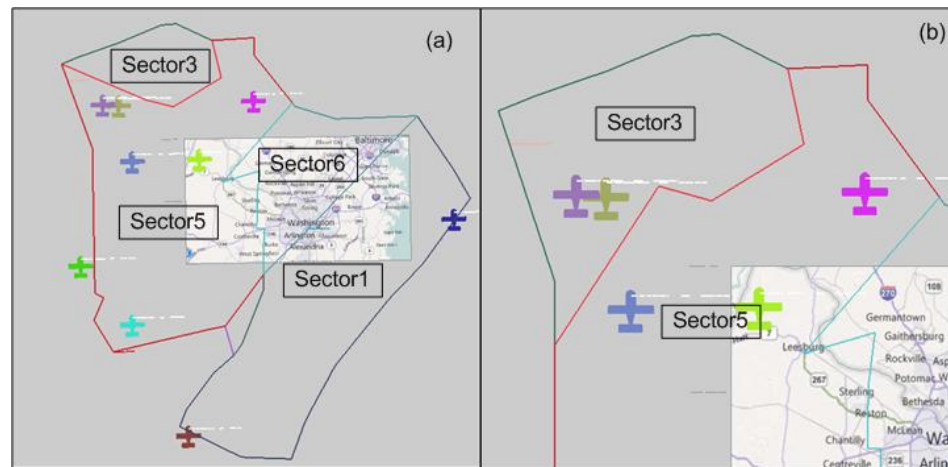


Figure 21 Snapshots of sectors (a) before and (b) after reconfiguration

Constraints were evaluated again to verify if the boundary modification resolved the workload constraint violation. Figure 22 shows the new report from constraint

verification, indicating that no sector has a workload exceeding the limit of 3. After the reconfiguration, workload of "Sector5" was reduced to 3, and all other sectors met the workload constraint. A new profile of sector "Sector5" was generated to verify that after the reconfiguration, workloads of Sector 5 were limited to 3 or below and the profile for Sector 5 with the reconfiguration is shown in Figure 23. Note that after the reconfiguration took place at around 2:10, the workload never exceeded the limit of 3(Figure 23). If the workload constraint was not imposed and sectors were not reconfigured, the workloads of relevant sectors would follow the profiles as depicted in Figure 18.

Sector	Current WL	Time	Max WL Allowed	Violate
Sector1	0	2:11:23	3	<input type="checkbox"/>
Sector2	1	2:11:23	3	<input type="checkbox"/>
Sector3	1	2:11:23	3	<input type="checkbox"/>
Sector4	1	2:11:23	3	<input type="checkbox"/>
Sector5	3	2:11:23	3	<input type="checkbox"/>
Sector6	0	2:11:23	3	<input type="checkbox"/>

Figure 22 A report of verifying workload constraint after reconfiguration



Figure 23 Workload profile for "Sector5" after reconfiguration

CHAPTER SIX CONCLUSION AND DISCUSSION

This chapter summarizes my dissertation and highlights my major contributions to spatial-temporal data modeling and management. As this research is designed to demonstrate the preliminary usages of constraints in handling 3D spatial-temporal data, a few open issues to be addressed in the future will also be discussed in this chapter.

6.1 Summary

Built upon a constraint-based approach, this research formalizes a conceptual data model for 3D spatial-temporal objects. In developing the model, the existing OO model is extended by introducing a new behavioral description, which was a unique feature of moving object models, to capture the transformation, evolution and movement of spatial-temporal objects. In this model, the versioned object is redefined to include the behavioral description, depicting the status of a spatial-temporal object at a time point/period. Then the spatial-temporal object is composed of multiple versioned objects sorted by their temporal information.

Besides object descriptions, spatial-temporal rules and conditions are integrated into the model. Five categories of spatial-temporal constraints are defined and formulated in a logical manner. They are representation, spatial, temporal, spatial-temporal and attributes constraints. These types of constraints are appended to the objects to condition,

regulate and restrict the spatial-temporal properties of these objects. The enforcements of different types of constraints are discussed as well.

The extended OO model has the following characteristics. First, the model provides a reasonably comprehensive description for 3D spatial-temporal objects, especially the description of object behaviors. Second, interpreting constraints appended to the spatial-temporal properties of a spatial-temporal object can obtain additional descriptions of the objects. Third, as some of these constraints are used to describe data integrity, the constraint-based approach is used to maintain the 3D spatial-temporal data integrity at any stage of modeling. Fourth, the constraint-based approach allows expressing queries as a set of constraints to support spatial-temporal querying, including complex queries and improves the efficiency of evaluating these queries with constraint programming techniques.

Based on the conceptual model, a prototype system was developed to support the visualization, manipulation, and querying of 3D spatial-temporal objects. Four categories of spatial-temporal constraints, defined in the conceptual model, are implemented except the representation constraints because such constraints are associated with how objects are represented at the conceptual modeling stage. Through these constraints, users can define various spatial-temporal conditions for objects to maintain data integrity.

Constraints are enforced with the spatial-temporal algorithms extended from existing 3D geometric algorithms. To support the capability of querying 3D spatial-temporal objects, the system allows users to formulate complex spatial-temporal queries through a user-friendly 3D graphical query interface, while 3D spatial constraints can

also be managed through the same query window. Queries are expressed as constraints and algorithms enforcing constraints are also re-used to solve queries.

To demonstrate the utilities of my modeling framework, I explore using such model to facilitate the DAC process. In this example, the typical objects, airspaces and flights, are represented in the proposed model. The associated operational rules are described as constraints appended to the objects. Conceptually, air traffic controllers could modify the spatial and temporal attributes of airspaces to balance the workload among different airspaces. When the boundaries of airspaces are modified, the controllers can evaluate if the workload constraint is satisfied by comparing the workload of each sector with the maximum level specified by the workload constraint.

Using the DAC as a case study, this research shows that the prototype system has the potential to be customized to deal with real world problems involving 3D spatial-temporal objects. With the customized prototype system, traffic controllers could visualize the airspace sectors and flights in a 3D space, query the workload level of each sector and modify the boundaries of airspace sectors interactively. During the sector reconfiguration process, I demonstrated how two attribute constraints and three spatial-temporal constraints worked together while these five constraints are only a subset of a larger array of constraints that are supported by the system. The major intent of this research is not to evaluate the potential that this prototype system can be used in operation, but to demonstrate the potential utilities of using a constraint-based approach together with constraint-oriented languages in handling spatial-temporal data in 3D spaces.

6.2 Contributions

This research contributes to the spatial-temporal modeling in several aspects.

First, the extended OO model enhances the capacities of existing spatial-temporal models in depicting 3D spatial-temporal objects. Besides describing the spatial and temporal properties of objects using traditional OO concepts, the extended model includes another property, the “behavior” of spatial-temporal objects, to the versioned objects. To some extent, the proposed model is a combination of the moving object models and OO models because the proposed model possesses the key features of both types of models. Therefore, the extended OO model provides more accurate object descriptions and capabilities to support advanced analyses.

Second, this study demonstrates the usages of constraints in 3D spatial-temporal modeling. While constraints have the potentials to facilitate spatial modeling, such use of constraints in 3D spatial-temporal modeling has not been fully explored yet. A systematic classification of spatial-temporal constraints is proposed to help scientists identify the constraints in specific problems. This research also summarizes these constraints using formal modeling language (e.g.: OCL). The proposed constraint concepts and the associated 3D computational geometry algorithms were implemented through the prototype system. The system clearly has the potential to deal with various real world problems at various scales involving 3D spatial-temporal objects, from tracking moving objects within a building at a local scale, to the monitoring of atmospheric transport phenomena across cities and regions. These constraint concepts and algorithms restrict the spatial-temporal behaviors of objects, and ensure data integrity by controlling how the data can be modified. They also provide the foundation to

formulate and implement efficient 3D spatial-temporal topological queries, which can support more advanced 3D spatial-temporal analyses. The graphical interface concepts to formulate queries and the process of translating the graphical queries into constraints and recorded by OCL facilitate the management and querying of 3D spatial-temporal data tremendously.

Third, the prototype system fills the gap between conceptual models and logical implementations of 3D spatial-temporal data management systems by offering several fundamental modules essential to the management of 3D spatial-temporal data. Existing GIS are generally weak in handling 3D geospatial features. Adding the temporal dimension to the 3D features will make the data managing tasks incomprehensible. Several critical functions such as 3D spatial-temporal queries were neither provided by existing 3D GIS nor supported by existing spatial-temporal analysis packages. Built upon the extended OO model, the prototype is able to manage various 3D spatial-temporal objects consisting of the basic geometric primitives and temporal data types discussed in Chapter 3. The four modules of the prototype can be inserted into other packages as well. Such prototype system can be customized to help solve real world problems (e.g. DAC). Compared to the systems discussed in Table 3, the proposed system is quite competent in handling and querying 3D spatial-temporal objects.

6.3 Future Work

A few problems need to be addressed in the future. The current model focuses on basic geometric types. It has to be modified to accommodate complex objects and objects with fuzzy boundaries. In the DAC example, moving points are used to represent flights

to simplify the geometric representations and computations of topological relations. If the flights were defined as 3D moving volumetric objects, closer to the real world objects, higher levels topological relations have to be employed. To handle such spatial-temporal objects and related constraints will require more detailed investigations in modeling spatial-temporal objects.

The current model only includes a subset of behaviors whereas all possible changes are not enumerated. Similarly, types of changes on the spatial-temporal objects in the prototype system are limited to positional changes for flights and horizontal movements for sector footprints. Extensions to behavior descriptions are necessary to accommodate more types of changes. Relaxing the types of geometric change could create challenging topological relations to evaluate. Other types of changes for the spatial-temporal objects may involve alternations of behavioral characteristics, which in turn, may affect various constraints applicable to the objects. For example, when a flying car changes from the driving mode to the flying mode, not just the behavior of the object has changed, but the associated spatial-temporal constraints are also different.

The prototype system extends a few 3D geometric computation algorithms to facilitate the enforcement of major spatial-temporal topological constraints. When constraints imposed on more complex spatial-temporal objects are involved, the set of algorithms to evaluate constraints has to be enriched. For example, currently only non-concave polyhedron can be handled and algorithms to identify the spatial relations between concave polyhedron have to be considered. In addition, this research has not explored how constraint computing techniques can improve the efficiency of enforcing

spatial-temporal constraints. When a large number of objects are involved, the computing intensity of evaluating these constraints imposed on these spatial-temporal objects can be a potential bottleneck in this prototype system. Nevertheless, future investigation can build upon the foundation provided by the proposed framework work, implementation approach, and the prototype system.

REFERENCES

REFERENCES

- Abdel-Malek, K., J. Yang, and D. Blackmore. "On Swept Volume Formulations: Implicit Surfaces." *Computer-Aided Design* 33, no. 1 (January 2001): 113–121.
- Abdul-Rahman, Alias, and Morakot Pilouk. *Spatial Data Modelling for 3D GIS*. Berlin: Springer, 2008.
- Abraham, Tamas, and John F. Roddick. "Survey of spatio-temporal databases." *GeoInformatica* 3, no.1 (1999): 61–99.
- Allen, James F. "Maintaining Knowledge about Temporal Intervals." *Communications of the ACM* 26, no.11 (1983): 832-843.
- Banerjee, Jay, Hong-Tai Chou, Jorge F. Garza, Won Kim, Darrell Woelk, Nat Ballou, and Hyoung-Joo Kim. "Data Model Issues for Object-oriented Applications." *ACM Transactions on Information System* 5, no. 1 (January 1987): 3–26.
- Batra, Dinesh, and George M. Marakas. "Conceptual Data Modelling in Theory and Practice". *European Journal of Information Systems* 4, (1995): 185–193.
- Belussi, Alberto, Elisa Bertino, and Barbara Catania. "Manipulating spatial data in constraint databases". *Lecture Notes in Computer Sciences* 1262 (1997): 113-141.
- Bilimoria, Karl. D, Sridhar Banavar, Gano B Chatterji, Kapil S Sheth, and Grabbe Shon. "FACET: Future ATM Concepts Evaluation Tool." *Air Traffic Control Quarterly* 9, no.1 (2001):1–20.
- Breunig, Martin, and Sisi Zlatanova. "3D Geo-database Research: Retrospective and Future Directions." *Computers &Geosciences* 37, no.7 (2011): 791–803.
- Brodsky, Alexander, and Yoram Kornatzky. "The LyriC Language: Querying Constraint Objects." *SIGMOD Rec.* 24, no. 2 (May 1995): 35–46.
- Buscemi, Maria Grazia, and Ugo Montanari. "A Survey of Constraint-based Programming Paradigms." *Computer Science Review* 2, no.3 (2008): 137-141.
- Buekenhout, F., and M. Parker. "The Number of Nets of the Regular Convex Polytopes in Dimension ≤ 4 ." *Discrete Mathematics* 186, no. 1–3 (May 15, 1998): 69–94.

- Caballero, Rafael, Yolanda García-Ruiz, and Fernando Sáenz-Pérez. "Applying Constraint Logic Programming to SQL Test Case Generation." In *Functional and Logic Programming*, edited by Matthias Blume, Naoki Kobayashi, and Germán Vidal, 6009:191–206. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.
- Camossi, Elena, Michela Bertolotto, and Elisa Bertino. "A multigranular object-oriented framework supporting spatio-temporal granularity conversions." *International Journal of Geographical Information Science* 20, no.5 (2006): 511-534.
- Chakraborty, Jayajit, and Marc P. Armstrong. "Using Geographic Plume Analysis to Assess Community Vulnerability to Hazardous Accidents." *Computers, Environment and Urban Systems* 19, no. 5–6 (November 1995): 341–356.
- Cockcroft, Sophie. "A taxonomy of spatial data integrity constraints." *Geoinformatica* 1, no.4 (1997):327-343.
- Cohen, Jonathan D., Ming C. Lin, Dinesh Manocha, and Madhav Ponamgi. "I-COLLIDE: An Interactive and Exact Collision Detection System for Large-scale Environments." In *Proceedings of the 1995 Symposium on Interactive 3D Graphics*, 189–ff. I3D '95. New York, NY, USA: ACM, 1995. <http://doi.acm.org/10.1145/199404.199437>.
- Cui, Z., A. Cohn, and D. Randell. "Qualitative and Topological Relationships in Spatial Databases." In *Advances in Spatial Databases*, edited by David Abel and Beng Chin Ooi, 692:296–315. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 1993. <http://www.springerlink.com/content/f210355632661ku7/abstract/>.
- Currim, Faiz, and Sudha Ram. "Modeling Spatial and Temporal Set-Based Constraints During Conceptual Database Design." *Information Systems Research* (November 18, 2010). <http://isr.journal.informs.org/content/early/2010/11/18/isre.1100.0306>.
- Davis, Ernest, Nicholas Mark Gotts., and Anthony G. Cohn. "Constraint networks of topological relations and convexity." *Constraints* 4, no.3 (1999):241–280.
- Demuth, Birgit., and Heinrich Hussmann. "Using UML/OCL Constraints for Relational Database Design." In *Proceedings of UML 99 - The Unified Modelling Language: Beyond the Standard*. Berlin, Heidelberg: Springer-Verlag, 1999.
- Delahaye, Daniel, and Stéphane Puechmorel. "3D Airspace Sectoring by Evolutionary Computation: Real-world Applications." In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, 1637–1644. GECCO '06. New York, NY, USA: ACM, 2006. <http://doi.acm.org/10.1145/1143997.1144267>.
- Egenhofer, Max J. "Pre-Processing Queries with Spatial Constraints." *Photogrammetric Engineering & Remote Sensing* 60, no.6 (1994): 783-790.

- Egenhofer, Max J, and John Herring. "Categorizing binary topological relations between regions lines and points in geographic databases." In *The 9-intersection: Formalism and its use for natural-language spatial predicates*, edited by Max. Egenhofer, David.M. Mark, and John. R. Herring. Santa Barbara, CA: National Center for Geographic Information and Analysis, 1991.
- Ellul, Claire, and Muki Haklay. "Requirements for Topology in 3D GIS". *Transactions in GIS* 10, no.2 (2006):157-175.
- Elmasri, Ramez, and Sham Navathe. 2000. *Fundamentals of Database Systems*. New York: Addison-Wesley.
- Erwig Martin, Ralf Hartmut Güting, Markus Schneider, and Michalis Vazirgiannis. "Spatio-Temporal Data Types: An Approach to Modeling and Querying Moving Objects in Databases." *GeoInformatica* 3, no.3 (1999): 265-291.
- Erwig, Martin, and Markus Schneider. "Spatio-temporal Predicates." *Knowledge and Data Engineering, IEEE Transactions On* 14, no. 4 (August 2002): 881 –901.
- Erwig, Martin, and Markus Schneider. "A Visual Language for the Evolution of Spatial Relationships and Its Translation into a Spatio-temporal Calculus." *Journal of Visual Languages & Computing* 14, no. 2 (April 2003): 181–211.
- Goodchild, Michael F., May Yuan, and Thomas J.Cova. "Towards a general theory of geographic representation in GIS." *International Journal of Geographical Information Science* 21, no.3 (2007): 239-260.
- Gröger, Gerhard, Reuter M., and Lutz Plümer.,2004. "Representation of a 3-D city model in spatial object-relational databases." In *Proceedings of the 20th ISPRS Congress*, Istanbul, Turkey, 12-23 July 2004.
- Grumbach, Stéphane, Jianwen Su, and Christophe Tollu. "Linear Constraint Query Languages Expressive Power and Complexity." In *Logic and Computational Complexity*, edited by Daniel Leivant, 960:426–446. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 1995.
<http://www.springerlink.com/content/72x54x5881471345/abstract/>.
- Grumbach, Stéphane, Philippe Rigaux, and Luc Segoufin. "On the Orthographic Dimension of Constraint Databases." In *Database Theory — ICDT'99*, edited by Catriel Beeri and Peter Buneman, 1540:199–216. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 1999.
<http://www.springerlink.com/content/qkqy9vhnx97we2dg/abstract/>.

- Grumbach, St éphane, Philippe Rigaux , Le Chesnay , and Luc Segoufin. "Spatio-Temporal Data Handling with Constraints." *Geoinformatica* 5, no.1 (2001):95-115.
- Gütting, Ralf Hartmut, and Markus Schneider. *Moving objects databases*. San Francisco: Morgan Kaufmann, 2005, 389.
- Gütting, Ralf Hartmut, Michael H. Böhlen, Martin Erwig, Christian S. Jensen, Nikos A. Lorentzos, Markus Schneider, and Michalis Vazirgiannis. "A Foundation for Representing and Querying Moving Objects." *ACM Trans. Database Syst.* 25, no. 1 (March 2000): 1–42.
- Hägerstrand, Torsten. "What About People in Regional Science?" *Papers of the Regional Science Association* 14, no. 1(1970): 7–21.
- Helman, James, and Lambertus Hesselink. "Representation and Display of Vector Field Topology in Fluid Flow Data Sets." *Computer* 22, no. 8(1989): 27–36.
- Hornsby, Kathleen, and Max.J Egenhofer. "Identity-based Change: a Foundation for Spatio-temporal Knowledge Representation." *International Journal of Geographical Information Science* 14, no.3 (2000): 207–204.
- Hornsby, Kathleen Stewart, and Stephen Cole. "Modeling Moving Geospatial Objects from an Event - based Perspective." *Transactions in GIS* 11, no. 4 (July 27, 2007): 555–573.
- Hubbard, Philip M. "Approximating Polyhedra with Spheres for Time-critical Collision Detection." *ACM Trans. Graph.* 15, no. 3 (July 1996): 179–210.
- Jaffar Joxan., and Jean Louis Lassez.1987. "Constraint Logic Programming." In *Proceedings of the 14th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, Munich, Germany, 21-23 January 1987. New York, NY, USA: ACM.
- Jaillet F., Shariat B., Vandorpe D., 1998. "Deformable Object Reconstruction with Particle Systems." *Computers & Graphics* 22, no. 2–3 (1998):189-194.
- Jiménez, P., F. Thomas, and C. Torras. "3D Collision Detection: a Survey." *Computers & Graphics* 25, no. 2 (April 2001): 269–285.
- Kanellakis, Paris C., Gabriel M. Kuper, and Peter Z. Revesz. "Constraint Query Languages (preliminary Report)." In *Proceedings of the Ninth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, Nashville, TN, 2-4 April, 1990. New York: ACM.

- Klein, Alexander, Mark D. Rodgers, and Hong Kaing. 2008. "Dynamic FPAs: A New Method for Dynamic Airspace Configuration." In *Integrated Communications, Navigation and Surveillance Conference, 2008. ICNS 2008*, Bethesda, MD, 5-7 May, 2008.
- Kopardekar, Parimal, Karl Bilimoria, and Banavar Sridhar. "Initial Concepts for Dynamic Airspace Configuration" In *7th Aviation Technology, Integration and Operations (ATIO) Seminar. AIAA*. Belfast, Northern Ireland, 18-20 September 2007.
- Kuper Gabriel, Leonid Libkin, and Jan Paredaens. *Constraint databases*. (Berlin : Springer-Verlag, 2000), 428.
- Kwan, Mei-Po, Jiyeong Lee. "Emergency Response After 9/11: The Potential of Real-time 3D GIS for Quick Emergency Response in Micro-spatial Environments." *Computers, Environment and Urban Systems* 29, no. 2 (2005): 93-113.
- LaCharit é Darlene, and Carole Paradis. "The Emergence of Constraints in Generative Phonology and a Comparison of Three Current Constraint-based Models." *The Canadian Journal of Linguistics* 38, no. 2 (1993): 127-150.
- Langran, Gail, and Nicholas R Chrisman. "A Framework For Temporal Geographic Information." *Cartographica The International Journal for Geographic Information and Geovisualization* 25, no. 3 (1988): 1-14.
- Lee, Jiyeong, and Sisi Zlatanova, eds. *3D Geo-Information Sciences*. Lecture Notes in Geoinformation and Cartography. Berlin : Springer-Verlag, 2009), 446.
- Li, Sanjiang. "A Complete Classification of Topological Relations Using the 9-intersection Method." *International Journal of Geographical Information Science* 20, no.6 (2006):589-610.
- Lin, Hui, Fengru Huang, and Guonian Lu. "Development of virtual geographic environments and the new initiative in experimental geography." *Acta Geographica Sinica* 64, no.1(2009): 7-20.
- Lohfink, Alex, Duncan McPhee, and Mark Ware. 2010. "A UML-based Representation of Spatio-Temporal Evolution in Road Network Data." *Transactions in GIS* 14, no.6 (2010): 853-872.
- Longley, Paul A., Mike Goodchild, David J. Maguire, and David W. Rhind. *Geographic Information Systems and Science*. John Wiley & Sons, 2010.

- Losa, Arnaud, and Bernard Cervelle. "3D Topological Modeling and Visualisation for 3D GIS." *Computers & Graphics* 23, no. 4 (August 1999): 469–478.
- Louwsma, Jildou, Sisi Zlatanova, Ron Lammeren, and Peter Oosterom. "Specifying and Implementing Constraints in GIS—with Examples from a Geo-Virtual Reality System." *GeoInformatica* 10, no. 4(2005): 531-550.
- Ma, Weiyin, Yongmin Zhong, Shiu-Kit Tso, and Tianxiang Zhou. "A Hierarchically Structured and Constraint-based Data Model for Intuitive and Precise Solid Modeling in a Virtual Reality Environment." *Computer-Aided Design* 36, no. 1(2004): 903-928.
- Mandel, Luis, and Maria Victoria Cengarle. "On the Expressive Power of OCL." In *FM'99 — Formal Methods*, edited by Jeannette M. Wing, Jim Woodcock, and Jim Davies, 1708:854–874. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999.
- Martinez , Stephane, Gano Chatterji, Dengfeng Sun and Alexandre Bayen, "A Weighted-Graph Approach for Airspace Dynamic Configuration, " In *AIAA Paper 2007-6448, AIAA Guidance, Navigation and Control Conference and Exhibit*, Hilton Head, SC, August. 20-23, 2007
- Mas, S., and W. Reinhardt. "Categories of Geospatial and Temporal Integrity Constraints." In *Advanced Geographic Information Systems Web Services, 2009. GEOWS '09. International Conference On*, 146 –151, 2009.
- Meyer, Bertrand. *Object-Oriented Software Construction*. 1st ed. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1988.
- Mirbel, Isabelle, Barbara Pernici, Babis Theodoulidis, Alex Vakaloudis, and Michalis Vazirgiannis, "Advanced Uses: Composing Interactive Spatio-temporal Documents." In *Spatio-Temporal Databases: The CHOROCHRONOS Approach*, edited by Koubarakis Manolis, Timos K. Sellis, Andrew U. Frank, Stéphane Grumbach, Ralf Hartmut Güting, Christian S. Jensen, Nikos A. Lorentzos, Yannis Manolopoulos, Enrico Nardelli, Barbara Pernici, Hans-Jörg Schek, Michel Scholl, Babis Theodoulidis, and Nectaria Tryfona. Berlin, Heidelberg: Springer-Verlag, 2003.
- Montagnat, Johan, and Hervé Delingette. "4D Deformable Models with Temporal Constraints: Application to 4D Cardiac Image Segmentation." *Medical Image Analysis* 9, no. 1 (February 2005): 87–100.
- Nievergelt, Jürg, and Franco P. Preparata. "Plane-Sweep Algorithms for Intersecting Geometric Figures". *Communications of the ACM* 25(1982): 739-747.

- Nocera, Luciano, Arjun Rihan, Songhua Xing, Ali Khodaei, Ali Khoshgozaran, Farnoush Banaei-Kashani, and Cyrus Shahabi. "GeoDec: A Multi-Layered Query Processing Framework for Spatio-Temporal Data." In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 546–547, Seattle, WA, 4-6 November, 2009. New York: ACM, 2009.
- Parent, Christine, Stefano Spaccapietra, and Esteban Zimanyi. "Spatio-temporal Conceptual Models: Data Structures + Space + Time." In *Proceedings of the 7th ACM International Symposium on Advances in Geographic Information Systems*, Kansas City, MO 2-6 November 1999.
- Parida, Laxmi, and S.P. Mudur. "Computational Methods for Evaluating Swept Object Boundaries." *The Visual Computer* 10, no. 5 (1994): 266-276.
- Pelagatti, Giuseppe, Mauro Negri, Alberto Belussi, and Sara Migliorini. "From the Conceptual Design of Spatial Constraints to Their Implementation in Real Systems." In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 448–451. GIS '09. New York, NY, USA: ACM, 2009.
- Pelekis, Nikos, Babis Theodoulidis, Ioannis Kopanakis, and Yannis Theodoridis. "Literature Review of Spatio-temporal Database Models." *The Knowledge Engineering Review*, 19(2004): 235-274.
- Peuquet, Donna. J. "It's About Time: A Conceptual Framework for the Representation of Temporal Dynamics in Geographic Information Systems." *Annals of the Association of American Geographers* 84, no.3 (1994): 441-462.
- Peuquet, Donna J., and Niu Duan. "An Event-based Spatiotemporal Data Model (ESTDM) for Temporal Analysis of Geographical Data." *International Journal of Geographical Information Systems* 9, no. 1 (1995): 7–24.
- Pinet Francois, Magali Duboisset, and Vincent Soullignac. "Using UML and OCL to Maintain the Consistency of Spatial Data in Environmental Information Systems." *Environmental Modelling & Software* 22, no. 8 (2007):1217-1220.
- Pultar, Edward, Thomas J. Cova, May Yuan, and Michael F. Goodchild. "EDGIS: a Dynamic GIS Based on Space Time Points." *International Journal of Geographical Information Science* 24, no. 3 (2010): 329–346.
- Ravada, Siva, Baris Kazar, and Ravi Kothuri. "Query Processing in 3D Spatial Databases: Experiences with Oracle Spatial 11g." In *3D Geo-Information Sciences*, edited by Jiyeong Lee and Sisi Zlatanova, 153–173. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.

- Revesz, Peter. *Introduction to Databases: From Biological to Spatio-Temporal*. Berlin: Springer, 2009, 754.
- Rigaux, Philippe, Michel Scholl, and Agnes Voisard. *Introduction to spatial databases: with application to GIS*. San Francisco: Morgan Kaufmann, 2002, 411.
- Robin, Jacques, Jairson Vitorino, and Armin Wolf. "Constraint programming Architectures: review and a new proposal." *Journal of Universal Computer Science* 13, no.6 (2007):701-720.
- Rumbaugh, James, Michael Blaha, William Premerlani, Frederick Eddy, and William Lorensen. *Object-oriented Modeling and Design*. Englewood Cliffs, NJ: Prentice-Hall, 1991. <http://cdsweb.cern.ch/record/237250>.
- Salehi, Mehrdad, YvanBedard, MirAbolfazlMostafavi, and JeanBrodeur. "Formal classification of integrity constraints in spatiotemporal database applications". *Journal of Visual Languages and Computing* 22, no.5 (2011):323-339.
- Schmidt, Ryan, Karan Sign, and Ravin Balakrishnan. "Sketching and Composing Widgets for 3D Manipulation. " *Computer Graphics Forum* 27, no.2 (2008): 301-310.
- Shekhar, Shashi, and Sanjay Chawla. *Spatial Databases: A Tour*. Upper Saddle River, NJ: Prentice Hall, 2003. Print.
- Shaw, Shih-Lung, and HongboYu. "A GIS-based Time-geographic Approach of Studying Individual Activities and Interactions in a Hybrid Physical-virtual Space." *Journal of Transport Geography* 17, no. 2 (2009): 141-149.
- Shen, Dayong, Kaoru Takara, Yuji Tachikawa, and Yueh-Lin Liu. "3D Simulation of Soft Geo-objects." *International Journal of Geographical Information Science* 20, no.3 (2006):261-271.
- Shi, Wenzhong, Bisheng Yang, and Qingquan Li. "An Object-oriented Data Model for Complex Objects in Three-dimensional Geographical Information Systems." *International Journal of Geographical Information Science* 17, no. 5 (2003):410-430.
- Schneider, Markus, and Thomas Behr. "Topological Relationships Between Complex Spatial Objects." *ACM Trans. Database Syst.* 31, no. 1 (March 2006): 39-81.

- Sourina, Olga. 2006. "Visual 3D Querying of Spatio-Temporal Data." In *International Conference on Cyberworlds, 2006. CW '06*, 147–156, Lausanne, 28-29 November 2006.
- Steiner, Andreas. "A Generalisation Approach to Temporal Data Models and Their Implementations". Ph.D thesis, Swiss Federal Institute of Technology, Zurich, 1998. <http://e-collection.library.ethz.ch/eserv/eth:22598/eth-22598-01.pdf>.
- Stewart Hornsby, Kathleen, and Kraig King. "Modeling Motion Relations for Moving Objects on Road Networks." *Geoinformatica* 12, no. 4 (December 2008): 477–495.
- Sutherland, Ivan, and Gary W.Hodgman. "Reentrant Polygon Clipping". *Communications of the ACM* 17, no.1 (1974):32-42.
- Tarquini, Francesco, and Eliseo Clementini. "Spatial Relations between Classes as Integrity Constraints." *Transactions in GIS* 12, no.s1 (2008):45-57.
- Thomas, James, and Cook, Kristin. "A visual analytics agenda". *IEEE Computer Graphics and Applications* 26, no.1 (2006):10-13.
- Tryfona, Nectaria, and Christian, Jensen. "Conceptual Data Modeling for Spatiotemporal Applications", *GeoInformatica* 3, no. 3(1999): 245-268.
- Twumasi, Bright Osei. "Modelling Spatial Object Behaviours in Object-relational Geodatabase", Master thesis, ITC, Netherland, 2002.
www.itc.eu/library/Papers/msc_2002/gfm/osei_twumasi.pdf.
- van den Bergen, Gino. "Efficient Collision Detection of Complex Deformable Models Using AABB Trees." *J. Graph. Tools* 2, no. 4 (January 1998): 1–13.
- Venema, Yde. 2001. "Temporal Logic". In *The Blackwell Guide to Philosophical Logic*, edited by Goble Lou. Wiley-Blackwell, 2001.
- Verbree, Edward, Gert Van Maren, Rick Germs, Frederik Jansen, and Menno-Jan Kraak. "Interaction in Virtual World Views-linking 3D GIS with VR." *International Journal of Geographical Information Science* 13, no.4 (1999): 385-396.
- Vidal, Cristian, and Andrea Rodriguez. . "A Logical Approach for Modeling Spatio-temporal Objects and Events." In *Perspectives in Conceptual Modeling*, edited by Jacky Akoka, Stephen W. Liddle, Il-Yeol Song, Michela Bertolotto, Isabelle Comyn-Wattiau, Willem-Jan Heuvel, Manuel Kolp, Juan Trujillo, Christian Kop, and Heinrich C. Mayr, 3770:2 18–227. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005.

- Wachowicz, John, and Monica Wachowicz. *Object-oriented Design for Temporal GIS*. Bristol, PA: Taylor & Francis, 1999.
- Warmer, Jos B., and Anneke G. Kleppe. *The Object Constraint Language: Precise Modeling with UML*. Boston MA Addison Wesley, 1999.
- Werder, Stefan. "Formalization of Spatial Constraints." In *12th AGILE International Conference on Geographic Information Science 2009* Hannover, Germany, 2-5 June, 2009.
- Worboys, Michael, and Kathleen Hornsby. "From Objects to Events: GEM, the Geospatial Event Model." In *Geographic Information Science*, edited by Max J. Egenhofer, Christian Freksa, and Harvey J. Miller, 3234:327–343. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004.
- Worboys, Michael. "Event-oriented Approaches to Geographic Phenomena." *International Journal of Geographical Information Science* 19, no.1 (2005):1-28.
- Yousefi, Arash, and George L Donohue. "Temporal and Spatial Distribution of Airspace Complexity for Air Traffic Controller Workload-based Sectorization". In *AIAA 4th Aviation Technology, Integration and Operations (ATIO) Forum*, Chicago, Illinois, 20-22 September 2004.
- Yuan, May, 1996. "Temporal GIS and Spatio-temporal Modeling". in *International Conference/Workshop Integrating GIS and Environmental Modeling*, Sante Fe, 21-25 January 1996.
- Yuan, May, Melany Dickens-Micozzi, and Michael A. Magsig. "Analysis of Tornado Damage Tracks from the 3 May Tornado Outbreak Using Multispectral Satellite Imagery". *Weather and Forecasting* 17, no.3(2003):382–398.
- Yuan, May, and Kathleen Hornsby. *Computation and Visualization for Understanding Dynamics in Geographic Domains: a Research Agenda*. Boca Raton, USA: CRC/Taylor and Francis, 2008, 120.
- Ziemann, Paul, and Martin Gogolla. "OCL Extended with Temporal Logic." In *Perspectives of System Informatics*, edited by Manfred Broy and Alexandre Zamulin, 2890:617–633. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2003.
- Zlatanova, Sisi. "On 3D Topological Relationships." In *Database and Expert Systems Applications, 2000. Proceedings. 11th International Workshop On*, 913 –919, 2000.

Zlatanova, Sisi, Alias Abdul Rahman, and Wenzhong Shi. "Topological Models and Frameworks for 3D Spatial Objects." *Computer & Geosciences* 30, no.4 (2004): 419-428.

CURRICULUM VITAE

Jing Li received her Bachelor of Science in Geographic Information Science from Wuhan University in 2007.