INFORMATION VISUALIZATION METHODS FOR GIS, CONSTRAINT AND SPATIOTEMPORAL DATABASES

by

Shasha Wu

A DISSERTATION

Presented to the Faculty of
The Graduate College at the University of Nebraska
In Partial Fulfillment of Requirements
For the Degree of Doctor of Philosophy

Major: Computer Science

Under the Supervision of Professor Peter Revesz

Lincoln, Nebraska

June, 2005

INFORMATION VISUALIZATION METHODS FOR GIS, CONSTRAINT AND SPATIOTEMPORAL DATABASES

Shasha Wu, Ph.D.

University of Nebraska, 2005

Advisor: Peter Revesz

Information visualization is as natural as querying for many database and geographic information system (GIS) applications. However, recursively defined concepts such as drought areas based on the standardized precipitation index (SPI) and long-term air pollution areas based on safe and critical level standards are hard to visualize using traditional database and GIS systems. This dissertation contributes to this topic in the following ways.

First, we develop novel mathematical methods to represent and visualize recursively defined concepts. Our mathematical development takes advantage of constraint databases which can conveniently represent Spatiotemporal data.

Second, we apply and test the mathematical method in two different recursive information visualization systems (RecIVs). These two systems are the Drought Online Analysis System (DOAS), which can compute and analyze drought conditions in any region based on SPI values, and the West Nile Virus Information System (WeNiVIS), which can visualize and reason about epidemiology information and was already applied to real West Nile virus data from the state of Pennsylvania.

Third, we analyze some general software design methods and apply the best method to the redesign of the MLPQ constraint database system. For the problem of data entry for spatiotemporal databases via distributed sensor networks, we propose an efficient routing algorithm that reduces the total energy cost for routing the information from the sensors to the central spatiotemporal database store.

ACKNOWLEDGEMENTS

Contents

1 Introduction **1**

1.1 Overview of Contributions 3

2 Related Work **6**

2.1 Relational Databases 6

2.2 Spatiotemporal Databases 7

2.3 Constraint Databases 8

2.4 Geographic Information Systems 8

2.5 Interpolation Methods in Databases 9

2.6 Visualization and Querying 11

3 Information Visualization for Recursively Defined Concepts **13**

3.1 Introduction 13

3.2 The Outline of the RecIV System 15

3.3 Implementation of the RecIV system 17

3.3.1 Software Architecture of the RecIV System 17

3.3.2 Naive Algorithm and Optimization 19

3.3.3 Extension for Linear Combination and Time Delay 23

4 Drought Online Analysis Application **27**

4.1 The DOAS System Architecture 28

4.2 Implementation and Analysis 30
4.3 Comparison 36
4.4 Other Similar Applications 38
4.4.1 Air Pollution Visualization Problem 38
4.4.2 Performance Analysis 40

5 West Nile Virus Application 46
5.1 The West Nile Virus Data 46
5.1.1 Data Source 46
5.1.2 Epidemiological Data Stored in Constraint Databases 50
5.2 Methodology 52
5.2.1 Recursive Definitions and Implementation 52
5.2.2 The WeNiVIS System 53
5.3 Results 55
5.3.1 Flexible User Interfaces with a High-Level Language 55
5.3.2 Enhancement of Tracking and Reasoning about Epidemics . . 57
5.4 Discussion 59

6 Design Methods for Spatiotemporal Databases 62
6.1 Software Modeling 63
6.1.1 The Study 64
6.1.2 The Stability Approach 69
6.2 Design Patterns 72
6.2.1 Context 72
6.2.2 Problems 72
6.2.3 Solution 74
6.2.4 Case Study 76
6.3 Improvement and Maintenance of the MLPQ System 77

6.4 Wireless Sensor Networks 80
6.4.1 The EDIVER Algorithm 81
6.4.2 Simulations 83
6.4.3 Main Results 85

7 Conclusions and Future Work **95**

Bibliography **99**

List of Figures

2.1 Triangulated map based on the 48 weather stations in Nebraska. . . . 10

3.1 The functionalities of the RecIV. 15
3.2 The homepage of the RecIV system. 16
3.3 The software architecture of the RecIV system. 18

4.1 The three-tier software architecture based on MLPQ Web Accessible System 29
4.2 Homepage of the DOAS: Drought Online Analysis System 41
4.3 Color bands for SPI value 42
4.4 Drought region in Nebraska at week 10515 (the 27th week of 2001) . 42
4.5 Process of finding the drought places at time 10515 (the 27th week of year 2001). 43
44
4.7 Visualization of polluted regions. 44
4.8 Comparison of the naive and optimized methods on drought data. . . 45

5.1 The comparison of time shifts between the infections on human and various types of animal hosts. 49
5.2 The triangulated network of sample points. 51
5.3 The constraint-based interpolated weekly WNV infected birds dataset. 52
5.4 The WeNiVIS system in analyzing WNV infections. 54

5.5 The distribution of West Nile Virus infections in week 139. 58
5.6 Left side: The predicted high-risk areas (dark color areas) based on the infected wild bird data. Right side: The actual distribution of human infections (darker color areas mean more infections) three weeks after the wild bird data to its left. 59

6.1 Typical class model for open-pit mining. 65
6.2 Class diagram of conventional model for conveyor belt transportation. 67
6.3 Class diagram of conventional model for pipeline transportation. . . . 68
6.4 Class diagram of stability model for transport system. 70
6.5 Current approach of implementing design patterns. 73
6.6 Stability Model Structure. 75
6.7 Design Patterns for Transaction. 77
6.8 Instance of transaction patterns on orders). 78
6.9 Stability model for Transaction Design Patterns on order instance. . . 87
6.10 A Stability Model for Constraint Database Management Systems. . . 88
6.11 The sequence of events when a node receives a Request packet. 89
6.12 The sequence of events when a node detects an event. 90
6.13 The sequence of events when a node receives an Event Agent packet. 91
6.14 Normalized routing load for AODV and EDIVER. 92
6.15 Average End-to-End Delay for different number of quires. 92
6.16 Packet delivery fraction for AODV and EDIVER. 93
6.17 Avg. Energy Level (a) 13 Requests (b) 20 Requests and (c) 50 Requests. 94

List of Tables

2.1 The point-based spatiotemporal data that records weekly SPI values of 48 major weather stations in Nebraska 7

2.2 The Constraint-based spatiotemporal data that records weekly SPI values of Nebraska. 11

Chapter 1

Introduction

Geographic Information System (GIS) (Demers 2000, Longley, Goodchild, Maguire & Rhind 2001, Worboys 1995) applications increasingly require the use of spatiotemporal data, that is, data that combine both space and time (Langran 1992). For example, the spread of virus through time is a typical spatiotemporal data. Future GIS systems need to efficiently manage spatiotemporal databases (STDBs) containing such data. The study of the representation and the visualization methods for spatiotemporal data is still a growing research area.

Infectious disease outbreaks are critical threats to public health and national security (Damianos, Ponte, Wohlever, Reeder, Day, Wilson & Hirschman 2002). With greatly expanded travel and trade, infectious diseases can quickly spread across large areas causing major epidemics.

Efficient computerized reasoning about epidemics is essential to detect their outbreak and nature, to provide fast medical aid to affected people and animals, to prevent their further spread, and to manage them in other ways.

Several characteristics of epidemics make them special in terms of computer reasoning needs. First, epidemiological data is usually some kind of spatiotemporal data, that is, it has a spatial distribution that changes over time. Second, epidemiological

data is recursive in nature. This means that the best predictions of the spread of infections are based on the current as well as some earlier situations. Third, we need a fast response from any knowledge-base that contains epidemiological data.

The above three characteristics in combination are fairly rare. Geographic information systems generally can represent only static objects that do not change over time, or if they change, then they change only slowly, for example, the population density of counties. Such a slow change may be represented in a geographic information system by a limited number of separate maps. However, continuous change over time is not easy to represent and is hard to reason about in geographic information systems.

We propose new methods to visualize and reason about epidemiological data. The major contributions and novel features of our article are the following:

- **Epidemiological data stored in constraint databases:**

  Relational databases and geographic information systems can not easily manage epidemiological data because of its inherently spatiotemporal nature. Constraint databases (Kanellakis, Kuper & Revesz 1995, Kuper, Libkin & Paredaens 2000, Revesz 2002), which are very suitable for spatiotemporal data, were proposed as extensions of both relational databases and geographic information systems. There are software tools that can export any relational database or geographic information system data into a constraint database (Chomicki, Haesevoets, Kuijpers & Revesz 2003, Chomicki & Revesz 1999).

- **Recursive epidemiological definitions:**

  We propose a new general method to express recursive epidemiological definitions and predictions about the spread of infectious diseases.

- **Implementation using recursive SQL:**

  The Prolog language is the choice for recursive definitions in many knowledge-base systems. However, Prolog is not good for querying spatiotemporal data. It is also less well-known than the widely-used SQL language, which is the standard query language for both relational and constraint databases. The latest SQL standard added to the SQL language a form of recursion, enabling the expression of the needed recursive definitions. It is expected that the latest SQL standard will be implemented in all major relational database products. As part of our contributions, we also implemented for the first time in the MLPQ (Revesz, Chen, Kanjamala, Li, Liu & Wang 2000) constraint database system, which is one of the most sophisticated constraint database prototype systems, the SQL recursive queries.

- **WeNiVIS: The West Nile Virus Information System:**

  We developed an example epidemic information system for reasoning about West Nile Virus infections. This system can show visually the spread of the epidemic and any other spatiotemporal data that may be generated by the system. We chose this example, because it has a typical infection pattern, it is currently still spreading through the United States, and data for it was readily available from Pennsylvania's West Nile Virus Control Program (Pennsylvania's West Nile Virus Control Program 2004).

1.1 Overview of Contributions

The main contributions of this dissertation are the following.

- We give a general framework to mathematically represent and efficiently visualize recursively defined spatiotemporal concepts. This is a theoretical contribution that has potentially a wide applicability.

- We design two different applications, one in epidemiology and another in drought monitoring, that use visualization of recursively defined data. This contribution show the applicability of our theoretical contribution.

- We extend software and wireless sensor network design methods and apply them to the MLPQ constraint database system. This is the basis of many other software systems that may use either spatiotemporal database querying or visualization.

The detailed contributions of this dissertation can be described chapter-by-chapter as follows.

Chapter 2 describes some basic concepts and previous work related to constraint databases, geographic information systems, and spatiotemporal database systems.

Chapter 3, which is based on (Revesz & Wu 2004), proposes a framework to mathematically define recursive spatiotemporal concepts. This chapter also shows how the naive mathematical description can be transformed into an efficiently evaluable logical description. The evaluation also leads to a fast computer visualization of the recursively defined spatiotemporal data.

Chapter 4, which is based on (Wu & Revesz 2004), describes the design and a web-based implementation of a drought online analysis system called DOAS. It also describes the use of the DOAS system on an set of real precipitation data from Nebraska. The DOAS system can be used to analyze and visualize the drought conditions in Nebraska.

Chapter 5, which is based on (Revesz & Wu 2005), describes the design and web-based implementation of the a system called WeNiViS that can analyze epidemiology data. In particular the WeNiViS system is applied to the analysis of West Nile virus (WNV) data from the state of Pennsylvania.

Chapter 6, which is based on (Fayad & Wu 2002, Hamza & Wu 2004, Hamza, Wu & Deogun 2005, Revesz & Wu 2003, Wu, Fayad & Nabavi 2002, Wu, Hamza & Fayad 2003, Wu, Mahdy & Fayad 2002), describes various software design issues and methods that are related to and applicable to spatiotemporal database systems, such as the MLPQ system (Revesz et al. 2000, Wu 2003). It also considers distributed sensor networks that provide the input data for spatiotemporal database systems.

Chapter 7 gives some conclusions and suggestions for future work.

Finally, an extended set of bibliographic references is given in the bibliography section of the dissertation.

Chapter 2

Related Work

This chapter reviews some related work and basic concepts in the area of database management. Relational databases are reviewed in Section 2.1, spatiotemporal databases in Section 2.2, constraint databases in Section 2.3, and geographic information systems in Section 2.4. These four types of databases are closely related. For example, interpolation methods described in Section 2.5 provide an interesting relationship between relational and constriant databases. Finally, visualization and querying of these four types of databases is reviewed in Section 2.6.

2.1 Relational Databases

Relational databases, which were introduced by E. F. Codd (Codd 1970) at IBM's San Jose Research Center, are today the most commonly used type of databases. We give only a brief introduction to relational databases (Elmasri & Navathe 2003, Ramakrishnan 1998, Silberschatz, Korth & Sudarshan 2005).

Consider Table 2.1, which describes a point-based spatiotemporal data set that consists of *Standardized Precipitation Index* (SPI) records collected at 48 main weather stations spread out all over the state of Nebraska. (Chapter 4 explores the use of the

SPI values in a drought online analysis system.)

Table 2.1: The point-based spatiotemporal data that records weekly SPI values of 48 major weather stations in Nebraska

station	x (east)	y (north)	year	week	SPI
BUTTE	-231.4	2214.9	2001	1	-0.62
BLOOMFIELD	-134.6	2179.1	2001	1	-0.83
ONEILL	-214.9	2164.1	2001	1	-0.83
⋮	⋮	⋮	⋮	⋮	⋮
BUTTE	-231.4	2214.9	2001	12	-0.14
BLOOMFIELD	-134.6	2179.1	2001	12	0.07
ONEILL	-214.9	2164.1	2001	12	-0.15
⋮	⋮	⋮	⋮	⋮	⋮
BUTTE	-231.4	2214.9	2002	1	-0.38
⋮	⋮	⋮	⋮	⋮	⋮

In relational databases the terms *table* and *relationa* are used interchangeably. In each table the top row contains the names of the *attributes* that the table uses. Below each named attribute are the values of the entiries corresponding to that attribute within each row which is called a *tuple*. Each *tuple* describes an some entity by its associated set of attribute values. For example, the second row or tuple describes a spatiotemporal entity, namely the fact that at the station called *Butte* which is located at -231.4 easting and 2214.9 northing coordinates, during the first week of 2001 the SPI value was -0.62.

2.2 Spatiotemporal Databases

Relational databases are limited because they cannot express infinite spatiotemporal data. For example, note that Table 2.1 expresses only a finite number of spatiotemporal entities. That is not enough to represent what is the situation over time at some other locations than those that are recorded. Usually the other spatiotemporal entities are also recordable by remote sensing satellites or estimated using interpolation

methods.

Spatiotemporal databases are a class of data models that extend relational databases to be able to record combinations of both space and time (Langran 1992). One class of spatiotemporal databases is *constraint databases*, which are discussed in Section 2.3. Another class of spatiotemporal databases are Geographic Inforamtion Systems, which are discussed in Section 2.4.

2.3 Constraint Databases

A *constraint database* is a finite set of constraint relations. A constraint relation is a finite set of *constraint tuples*, where each constraint tuple is a conjunction of *atomic constraints* using the same set of attribute variables (Revesz 2002). Hence, constraints are hidden inside the constraint tables, and the users only need to understand the logical meaning of the constraint tables as an infinite set of constant tuples represented by the finite set of constraint tuples. Typical atomic constraints include linear or polynomial arithmetic constraints.

The MLPQ system is a constraint database system that implements rational linear constraint databases and queries. Among other functionalities, It supports both SQL and Datalog queries, and minimum/maximum aggregation operators over linear objective functions (Revesz et al. 2000). Other constraint database systems include DEDALE (Grumbach, Rigaux & Segoufin 1998), CCUBE (Brodsky, Segal, Chen & Exarkhopoulo 1999) and CQA/CDB (Goldin, Kutlu, Song & Yang 2003). .

2.4 Geographic Information Systems

Geographic Information Systems (GISs) (Demers 2000, Longley et al. 2001, Worboys 1995) are computer systems that deal with spatially (sometimes spatiotemporally)

related data. The GIS systems are used to combine multiple layers of spatial information in order to aid the user in performing complex analysis. The layers may correspond to different time instances to give the temporal dimensionof the GIS data. GISs are designed for static data and are limited in expressing continuously moving objects, which is a more complex spatiotemporal information than for example annual changes in population of counties (Langran 1992, Worboys 1995). The relationship between GIS and constraint-based spatiotemporal databases is further described in (Li & Revesz 2003).

A method to enhance GIS with spatiotemporal data is given by Theophilides et al. (Theophilides, Ahearn, Grady & Merlino 2003), who developed DYCAST, which is an epidemic spread prediction system based on spatiotemporal interpolation. The DYCAST system was used to predict human West Nile Virus infections based on dead bird surveillance data. However, the DYCAST system does not provide a flexible reasoning method.

Another GIS enhancement is given by Raffaetà et al. (Raffaetà, Renso & Turini 2002), who use MuTACLP, which is a temporal annotated constraint logic programming language.

2.5 Interpolation Methods in Databases

In a 2-D spatial problem, a point-based spatiotemporal relation has the schema of (x, y, t, w_1, w_2, ..., w_m), where the attributes (x, y) specify point locations, t specifies a time instance, and w_i $(1 \leq i \leq m)$ records the features of each location.

A point-based spatiotemporal data set only stores information of some sample points. To represent the features beyond those finite sample points, it is necessary to do spatiotemporal interpolation on them. A shape function based spatiotemporal method (Li, Li & Piltner 2004, Li & Revesz 2004) was used to interpolates and trans-

lates the original point-based spatiotemporal information into a *constraint relation*.

The translation has two steps. The first step is the triangulation of the sample points. Several efficient algorithms have been developed to generate triangular meshes. A popular method is the "Delaunay Triangulation" (Goodman & O'Rourke 1997, Shewchuk 1996). Figure 2.1 shows the triangulation result of the 48 weather stations in Nebraska. Each of these stations is an extreme point of at least one triangle in the map.

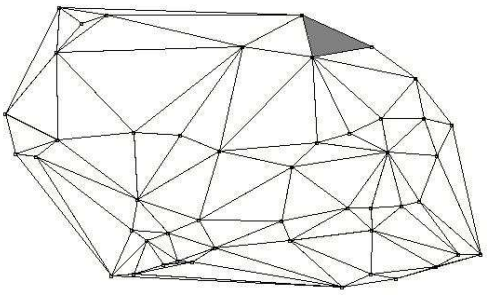

Figure 2.1: Triangulated map based on the 48 weather stations in Nebraska.

The second step defines for each triangle a linear interpolation function that represents the amount of indicator for any point within it (Li & Revesz 2004).

Table 2.2 shows the constraint data set that represents the relational data set shown in Table 2.1. It is a constraint relation with many constraint tuples. Each constraint tuple contains four linear constraints. The first three inequality constraints over x and y represent the area of a triangle. The last linear equation over x, y, and w represents the SPI value of the point at location (x, y) within this area. The field *week* represents the weekly time starting from January 1st, 1800.

For example, the first tuple with $id = 1$ in Table 2.2 is interpolated from the first three weather stations (BUTTE, BLOOMFIELD and ONEILL) in Table 2.1 and represented as the red region in Figure 2.1. The same triangular region in Figure 2.1 has different tuples at different time in Table 2.2. We showed two tuples for the region of $id = 1$ in Table 2.2. One is for time $week = 10489$, the other is for time

$week = 10500$. The time unit $week = 10489$ in Table 2.2 corresponds to the first week of 2001 in Table 2.1, $week = 10500$ represents the 12th week of 2001, and so on. The only difference between these tuples is the last linear equation over x, y, and s.

Table 2.2: The Constraint-based spatiotemporal data that records weekly SPI values of Nebraska.

id	east	north	week	SPI	
1	x	y	10489	w	$3.0788x + y \geq 1502.47$, $-0.1868x + y \geq 2204.24$, $0.3698x + y \leq 2129.32$, $0.7280x - 3.8974y + 1000w = -9420.80$.
2	x	y	10489	w	$-5.6429x + y \geq 5432.03$, $-1.4370x + y \leq 3040.58$, $-0.9106x + y \geq 2694.89$, $12.6524x + 0.8319y + 1000w = -6314.32$.
⋮	⋮	⋮	⋮	⋮	⋮
1	x	y	10500	w	$3.0788x + y \geq 1502.47$, $-0.1868x + y \geq 2204.24$, $0.3698x + y \leq 2129.32$, $-2.5483x - 1.0246y + 1000w = -1819.61$.
⋮	⋮	⋮	⋮	⋮	⋮

Li and Revesz (Li & Revesz 2002, Li & Revesz 2004) did an extensive comparison and proved shape functions to be the best in a test example concerning house price estimation.

2.6 Visualization and Querying

The MLPQ system is able to visualize spatiotemporal data in 2-dimensional space and one-dimenstional time using an *animation* which shows on the computer screen how the objects move in the plane over time by using a series of snapshots at regular time intervals.

As described in (Revesz 2002), Structured Query Language (SQL) is a popular query language for relational database systems. It is a complex language with many

functions and options. A basic SQL query is consisted by three sentences as follows:

$$\begin{aligned} SELECT &\quad a_i, \ldots, a_{i_l} \\ FROM &\quad R_1, R_2, \ldots, R_m \\ WHERE &\quad C_{on_1}, \ldots, C_{on_k} \end{aligned}$$

where R_i for $1 \leq i \leq m$ are relation names (table names), a_{i_k} for $1 \leq k \leq l$ are attribute names, and C_{on_j}s for $1 \leq j \leq k$ are atomic constraints.

Following is a sample SQL query on the SPI database represented by Table 2.1

Query 2.6.1 Find the *east* and *north* coordinations of weather station $BUTTE$:

$$\begin{aligned} SELECT &\quad x,\ y \\ FROM &\quad weather \\ WHERE &\quad station = \text{“}BUTTE\text{”} \end{aligned}$$

As described in (Revesz 2002), Datalog is a rule-based language that integrates the Prolog language, which is a popular language for implementing expert systems, with database technology. A Datalog rule is a Horn clause satisfying a number of properties, among which is the absence of function symbols. Each rule in Datalog saying that that if some points belong to some relations, then other points must also belong to a defined relation (Revesz 2002).

Each Datalog query consists of a finite set of rules in a form as follows:

$$R_0(x_1, \ldots, x_k) \ :- \ R_1(x_{1,1}, \ldots, x_{1,k_1}, \ldots, R_n(x_{n,1}, \ldots, x_{n,k_n}).$$

where each R_i is either an input relation name or a defined relation name, and the xs are either variables or constraints.

For example, the Query 2.6.1 can be represented in Datalog as follows:

$$\begin{aligned} LocationOfButte(x, y) \ :- \ & weather(station,\ x,\ y,\ year,\ week,\ SPI), \\ & station = \text{“}BUTTE\text{”}. \end{aligned}$$

Chapter 3

Information Visualization for Recursively Defined Concepts

3.1 Introduction

In Geographic Information Systems (Longley et al. 2001) we frequently need to visualize on a map the area where a given property P holds, where the area is defined using the following *non-recursive* form.

Definition 3.1.1 An area has property P during time unit T, if during T we measure an amount k or more of an indicator of property P.

However, many properties cannot be defined in this simple way. Usually, these complex properties are defined based on a series of observations in time. Their definitions have the following general *recursive* form.

Definition 3.1.2 An area has property P during time unit T if during T we measure either an amount

(i) k or more of an indicator of property P or

(ii) between k_1 and k of the same indicator
and the area has property P during time unit $T-1$.

where k_1 is less than k. The first part of the recursive definition is like Definition 3.1.1. The second part adds more areas. Hence while an area with only a measurement value between k_1 and k during time T does not have the property P according to the non-recursive definition, it *may* have the property according to the recursive definition. Definition 3.1.2 is appropriate when the indicators measured at time unit $T-1$ do not disappear completely by time unit T.

Example 3.1.1 Suppose we would like to find the counties on a map that have a *disease out-of-control* at time T, and suppose the only indicator that we have available is the number of new infections. Answering this query is not a simple matter of finding the counties that have more than some k (e.g., $k = 10$) new infected persons because some people infected with the disease at time $T-1$ will continue to be infected at time T. Suppose we expect about half of the infected persons to continue to be infected after a time unit. If at time $T-1$ a county had a disease out-of-control, then it is reasonable to assume that the disease is still out-of-control at time T if during T we only have five to nine new infected persons. Hence the recursive definition, written in the form of Definition 3.1.2, is as follows:

A county has *a disease out-of-control* during week T if during week T it reports either

(i) 10 or more new infected persons or

(ii) between 5 and 9 new infected persons
and it is highly-infected during week $T-1$.

In this paper we describe an information visualization framework for recursively defined spatiotemporal concepts and implement the framework in the *RecIV* sys-

tem. It is designed to visualize all recursively defined concepts expressible by Definition 3.1.2. Revesz and Li provided constraint-based visualization for spatio-temporal data in (Revesz & Li 2002) but did not consider recursively defined concepts.

This section is organized as follows:

Section 3.2 describes the main functions of the visualization system from the user's perspective. Section 3.3 describes the implementation of the system.

3.2 The Outline of the RecIV System

The RecIV system is a web-based general solution for problems definable in the form of Definition 3.1.2. The abstraction of the RecIV system is shown in Figure 3.1. There are five inputs for the system. One is the relation $M(x, y, t, w)$, where the attributes (x, y) specify 2-D locations, t specifies a time instance, and the last attributes w records the measurement of the indicator of property P of each location. The next three inputs are T, k, and k_1, which are defined in Definition 3.1.2. The last input is an optional overlay object. The output of the system is the visualized image(s) of a spatio-temporal relation $P(x, y, t)$ that represents the area with property P at time t. This can be overlaid by the overlay objects.

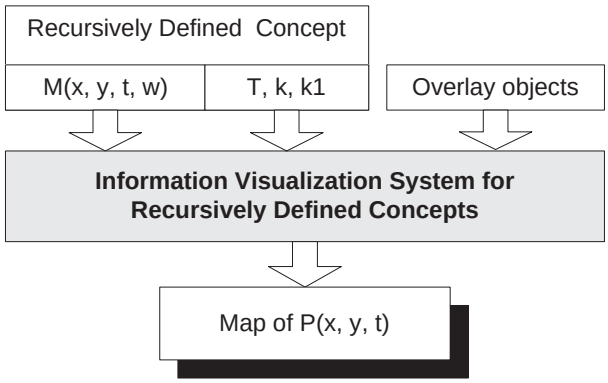

Figure 3.1: The functionalities of the RecIV.

The web interface of the system is shown in Figure 3.2. It provides an English

description of the general problem that this system can solve and several input areas for the users to specify arguments related to their own applications.

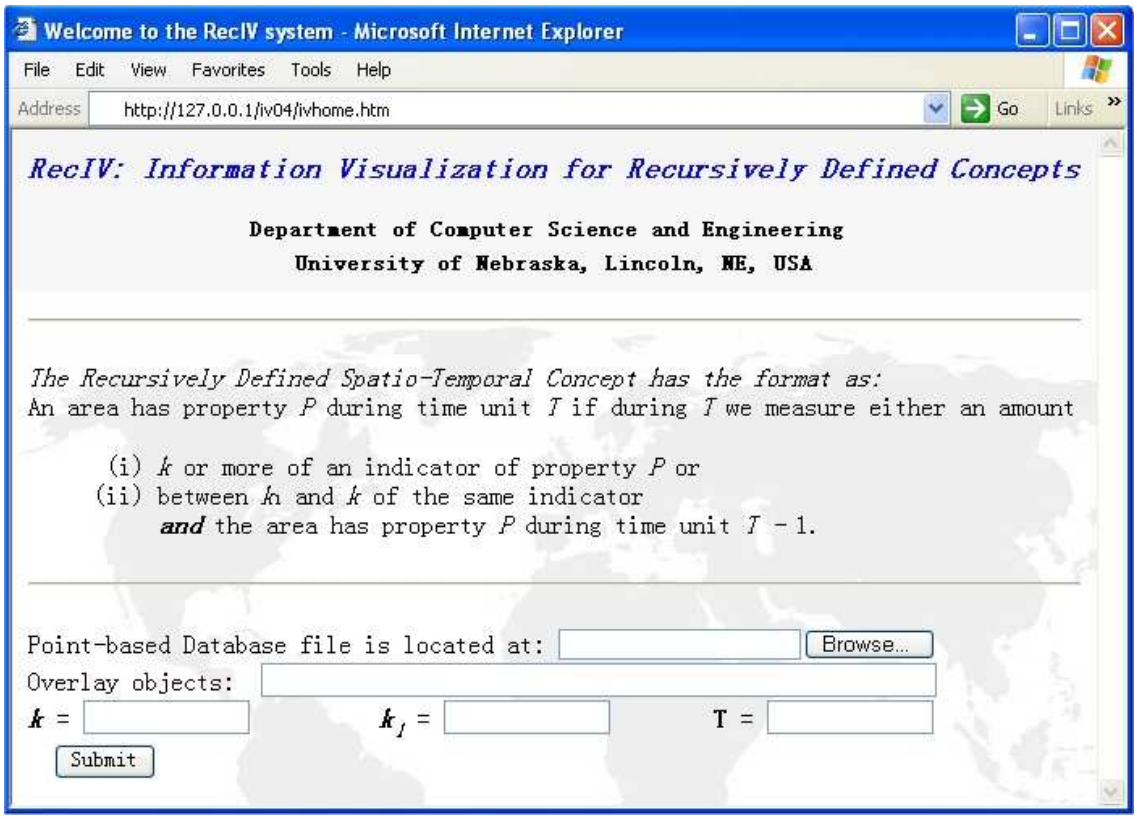

Figure 3.2: The homepage of the RecIV system.

The general solution for the problem defined in Definition 3.1.2 can be formally expressed as follows:

Given $M(x, y, t, w)$, we have:

$$A = \{(x, y, t) \mid M(x, y, t, w) \wedge \ w \geq k \ \wedge \ t \leq T\}$$
$$B = \{(x, y, t) \mid M(x, y, t, w) \wedge \ k_1 < w < k \ \wedge \ t \leq T\}$$

that is, A is the part of M that is greater or equal to k, and B is the part that is between k and k_1. We define the areas having property P at time t using Definition 3.1.2 as follows:

Definition 3.2.1

$P = \{(x, y, t) \mid A(x, y, t) \ \vee (B(x, y, t) \wedge P(x, y, t-1))\}$

Applying a general solution based on the above definition enables the system solve similar problems without modifying the program. The user does not need to search for different solutions for each specific application. Our experience shows that this implementation largely increases the usability and maintainability of this system in practice. As we explain in Section 4.4, the applications of drought visualization and air-pollution visualization have different contexts. However, both of them can be described in the form of Definition 3.1.2. That enables us to solve them with the same system without changing the program.

3.3 Implementation of the RecIV system

In Section 3.3.1, we describe the software architecture of the system and the data translation process. In Section 3.3.2, we illustrate the naive implementation and the optimized algorithm.

3.3.1 Software Architecture of the RecIV System

Recursively defined spatial-temporal information is difficult to visualize in most systems. However, constraint databases (Revesz 2002) provide an efficient way to store the spatio-temporal data, and the Datalog query language supports recursion. We combine 2-D interpolation functions and recursive Datalog with MLPQ constraint database system. Furthermore, we realize a general solution to calculate and visualize the complex spatio-temporal problems formulated according to Definition 3.1.2.

Figure 3.3 shows the software architecture of the RecIV system. It has three layers, which are described below.

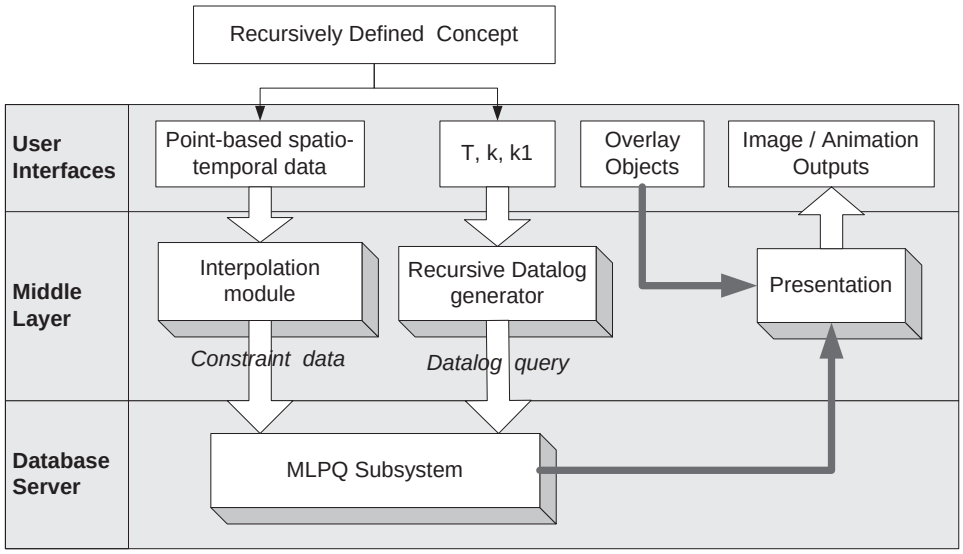

Figure 3.3: The software architecture of the RecIV system.

I. User interface layer accepts input from the user and displays the output to the user. There are five inputs from the user: the point-based spatio-temporal data set, the three arguments k, k_1, and time T, and a set of overlay objects.

II. Middle layer does the internal data processing. It has the following three modules.

1. The *interpolation module* is used to interpolate and translate point-based relational data into constraint data. It implements the 2-D shape function for triangles (Li & Revesz 2004) to interpolate and translate the original relational data set into a constraint data set.

2. The *recursive Datalog Generator* is designed to generate the Datalog query with the input arguments and send the query to the MLPQ constraint database subsystem.

3. The *presentation module* explains and visualizes the output of the database layer. It allows the user to zoom in and zoom out any specific area within the whole map. It can also overlay some input objects on top of the output image

and generate a combined output map. Those objects may include state/county boundaries, highways, or rivers.

III. Database layer: The *MLPQ constraint database subsystem* is used to evaluate the input Datalog queries and visualize the result as animation or image(s). Finally, the *presentation layer* generates a new web page with result image(s) and displays it to the users.

3.3.2 Naive Algorithm and Optimization

Definition 3.2.1 can be translated into the following Datalog *Query 1*:

$$\begin{aligned} A(x,y,t) &:- M(x,y,t,w),\ \ w \geq k,\ \ t \leq T. \\ B(x,y,t) &:- M(x,y,t,w),\ \ k_1 < w < k,\ \ t \leq T. \end{aligned}$$

$$\begin{aligned} P(x,y,t) &:- A(x,y,t). \\ P(x,y,t) &:- B(x,y,t),\ \ P(x,y,t-1). \end{aligned}$$

Datalog *Query 1* is the core program of the RecIV system and can be evaluated by the MLPQ subsystem directly. Comparing the size and complexity of a C++ or Java program needed to solve the same problem based on relational databases, Datalog and constraint databases provide a more concise and manageable approach. A simple and independent query solution makes the program easy to understand and maintain.

Although the code of naive implementation is concise, its efficiency can be further improved.

There are two sources of inefficiency. The first is the difficulty of providing appropriate time boundary conditions for the Datalog query. This is necessary in real implementation, because without a reasonable boundary condition, the recursive process may not terminate.

The second issue is the redundant calculation introduced by the naive implementation. Although the user only needs to know the area with property P in one or several separate time instances, the naive implementation always calculates these areas for every time unit during a time period. That is an extra burden for the system that can be avoided.

For example, suppose a user wants to visualize the areas that have property P at time T. If the user decides to limit the depth of the recursive execution to 10 weeks, then the naive implementation will try to find all such areas at every week from week $T-10$ to week T. The results between week $T-10$ to week $T-1$ are neither necessary nor confident.

In order to improve the efficiency of the algorithm, we modify the naive Datalog solution as follows:

Theorem 3.3.1

$$\begin{aligned} P \;=\; \{(x,y,t) \;\mid\; & A(x,y,t) \;\vee \\ & (\bigvee_{m=1}^{+\infty} (C(x,y,t,m-1) \;\wedge\; A(x,y,t-m)))\} \end{aligned}$$

where

$$\begin{aligned} C = \{(x,y,t,m) \;\mid\; & (B(x,y,t) \wedge m = 0) \;\vee \\ & (B(x,y,t-m) \;\wedge \\ & C(x,y,t,m-1) \;\wedge\; m \geq 1)\} \end{aligned}$$

Proof 3.3.1 *First we can prove that*

$$C(x,y,t,m) = \bigwedge_{i=0}^{m} B(x,y,t-i). \tag{3.1}$$

It is easy to expand $C(x, y, t, m)$ *for any integer* $m \geq 1$ *as follows:*

$$\begin{aligned}
C &= B(x, y, t-m) \wedge C(x, y, t, m-1) \\
&= B(x, y, t-m) \wedge B(x, y, t-m+1) \wedge \\
&\quad C(x, y, t, m-2) \\
&= \ldots \\
&= B(x, y, t-m) \wedge B(x, y, t-m+1) \wedge \cdots \wedge \\
&\quad B(x, y, t-1) \wedge C(x, y, t, 0) \\
&= B(x, y, t-m) \wedge B(x, y, t-m+1) \wedge \cdots \wedge \\
&\quad B(x, y, t-1) \wedge B(x, y, t) \\
&= \bigwedge_{i=0}^{m} B(x, y, t-i)
\end{aligned}$$

Second, by expanding $P(x, y, t-1)$ *in the definition of* P *we get:*

$$\begin{aligned}
P &= \{(x, y, t) \mid A(x, y, t) \vee \\
&\quad (B(x, y, t) \wedge (A(x, y, t-1) \vee \\
&\quad (B(x, y, t-1) \wedge P(x, y, t-2))))\}
\end{aligned}$$

Simplifying the right hand side we get:

$$\begin{aligned}
P &= \{(x, y, t) \mid A(x, y, t) \vee \\
&\quad (B(x, y, t) \wedge A(x, y, t-1)) \vee \\
&\quad (B(x, y, t) \wedge B(x, y, t-1) \wedge P(x, y, t-2))\}
\end{aligned}$$

We can continue expanding $P(x, y, t-2)$ and simplifying it as follows:

$$\begin{aligned} P \;=\; & \{(x,y,t) \mid A(x,y,t) \;\vee \\ & (\bigwedge_{i=0}^{0} B(x,y,t-i)) \wedge A(x,y,t-1) \vee \\ & (\bigwedge_{i=0}^{1} B(x,y,t-i)) \wedge A(x,y,t-2) \vee \\ & \vdots \\ & (\bigwedge_{i=0}^{m-2} B(x,y,t-i)) \wedge A(x,y,t-m+1) \vee \\ & (\bigwedge_{i=0}^{m-1} B(x,y,t-i)) \wedge P(x,y,t-m)\} \end{aligned}$$

Using Equation (3.1), the above can be further simplified as:

$$\begin{aligned} P \;=\; & \{(x,y,t) \mid A(x,y,t) \;\vee \\ & C(x,y,t,0) \wedge A(x,y,t-1) \vee \\ & C(x,y,t,1) \wedge A(x,y,t-2) \vee \\ & \vdots \\ & C(x,y,t,m-1) \wedge P(x,y,t-m)\} \end{aligned}$$

Expanding to infinity, the right hand side of the formula can be also written as:

$$A(x,y,t) \;\vee\; (\bigvee_{m=1}^{+\infty} (C(x,y,t,m) \wedge A(x,y,t-m-1)))$$

which is stated by the theorem. ■

Theorem 3.3.1 allows us to express the problem of finding relation P by an efficient Datalog query as follows.

$$
\begin{aligned}
C(x,y,t,0) \quad &:- \quad B(x,y,t). \\
C(x,y,t,m) \quad &:- \quad B(x,y,t-m), \\
&\qquad\quad C(x,y,t,m-1), \quad m \geq 1.
\end{aligned}
$$

$$
\begin{aligned}
P(x,y,t) \quad &:- \quad A(x,y,t). \\
P(x,y,t) \quad &:- \quad C(x,y,t,m-1), \\
&\qquad\quad A(x,y,t-m), \quad m \geq 1.
\end{aligned}
$$

where relations $A(x,y,t)$ and $B(x,y,t)$ are defined in *Query 1*. Assume $C_k = \{(x,y,t) \mid C(x,y,t,k)\}$, then we have $C_j \subseteq C_i$ for all $1 \leq i < j$. That means for each fixed time t the area of $C(x,y,t,m)$ monotonously decreases as m increases.

Hence the Datalog query evaluation of the P relation should terminate after some finite number of rule applications. The users can easily assign an appropriate constant M as the upper bound of m for their specific application to balance the accuracy and calculation time. The bigger M is, the more accurate the result is but the more calculation is required.

In general, the execution time of the optimized Datalog is much less than the evaluation time of the naive implementation. Section 4.4.2 gives the comparison in the case of visualizing drought areas.

3.3.3 Extension for Linear Combination and Time Delay

Definition 3.1.2 can only deal with one indicator for property P and the time delay of the indicator is fixed to be one time unit. In epidemiology research, one infectious disease commonly has several indicators (i.e. measurable disease carriers) and different indicators may have different effectiveness. The animal indicators also may predict ahead the human infection with different delay times. To consider these extra complications, we extend Definition

Definition 3.3.1 Let $M_i(x, y, t)$ represent the amount of indicator i measured at location (x, y) at time unit t. For each indicator i, let w_i be the effectiveness weight and t_i be the time delay to indicate property P. Then location (x, y) has property P during time unit T if

1. $\sum w_i M_i(x, y, T - t_i) \geq k$ or
2. $k_1 \leq \sum w_i M_i(x, y, T - t_i) \leq k$
   and the location has property P during time unit $T - 1$.

Part (1) of Definition 3.3.1 says that property P holds at time T if the linear combination of measurements of the indicators at the appropriate previous times (i.e., with their respective time delays) is greater than some threshold value k. Part (2) says that P also holds in those areas where the same linear combination is only between k_1 and k but already had property P at time $T - 1$.

Example 3.3.1 The West Nile virus has four major types of disease carriers: wild bird as indicator 1, mosquito as indicator 2, horse as indicator 3 and chicken as indicator 4. Figure 5.1 suggests that the onset of human infections generally occurs three weeks later than the onset of wild bird infections, one week later than the onset of mosquito infections, about six weeks after the onset of chicken infections and almost at the same time as the horse infections. Hence, we can assign the time delay for these four indicators as follows:

$t_1 = 3, \;\; t_2 = 1, \;\; t_3 = 6, \;\; t_4 = 0$

Considering that big animals usually contain more virus than small animals contain, we may assign the effectiveness weight of WNV infection to the four major carriers according to their relative body sizes as follows:

$w_1 = 1, \;\; w_2 = 0.2, \;\; w_3 = 1.5, \;\; w_4 = 5$

We assume that the infected animals reported at time $T - t_i$ are representative

of the entire animal population at the same time and part of the unreported infected animals at that time may continue to be infected at least until time T.

Suppose we would like to find the areas on a map that have a high risk of human WNV infections at time T. Let $k = 8$ and $k_1 = 4$, and $M_i(x, y, t)$ be as in Definition 3.3.1.

First, we compute the linear combination of the measurements of the indicators for each area as follows:

$$\begin{aligned} W &= \sum w_i M_i(x, y, T - t_i) \\ &= M_1(x, y, T - 3) + 0.2M_2(x, y, T - 1) + 1.5M_3(x, y, T - 6) + 5M_4(x, y, T) \end{aligned}$$

The area is at high risk of human WNV infections at week T if during week T it has:

$W \geq 8$, or

$4 \leq W < 8$ *and* it is at high-risk during week $T - 1$.

Let relation $M_i(x, y, t, w)$ store the measurements of the indicator i at location (x, y) during time t. We can express the general Definition 3.3.1 in SQL query language with recursion as follows.

Query 3.3.1 Recursive SQL query for Definition 3.3.1:

$$\begin{array}{ll} Create\,View & A(x, y, t)\ \ As \\ Select & M_1.x,\ \ M_1.y,\ \ t \\ From & M_1, M_2, \ldots, M_n \\ Where & c_1 M_1.w + c_2 M_2.w + \cdots + c_n M_n.w \geq k, \\ & M_1.t = t - t_1, \quad M_2.t = t - t_2,\ \ \ldots,\ \ M_n.t = t - t_n, \\ & M_1.x = M_2.x = \cdots = M_n.x, \\ & M_1.y = M_2.y = \cdots = M_n.y \end{array}$$

Relation A returns the spatiotemporal locations (x, y, t) that satisfy part (1) of

Definition 3.3.1.

$$
\begin{array}{ll}
Create\,View & B(x,y,t)\ \ As \\
Select & M_1.x,\ \ M_1.y,\ \ t \\
From & M_1, M_2, \ldots, M_n \\
Where & k_1 \leq c_1 M_1.w + c_2 M_2.w + \cdots + c_n M_n.w < k, \\
 & M_1.t = t - t_1, \quad M_2.t = t - t_2, \ \ \ldots, \ \ M_n.t = t - t_n, \\
 & M_1.x = M_2.x = \cdots = M_n.x, \\
 & M_1.y = M_2.y = \cdots = M_n.y
\end{array}
$$

Relation B returns the spatiotemporal locations (x, y, t) that satisfy the first condition of part (2) of Definition 3.3.1.

$$
\begin{array}{ll}
Create\,View & With\ Recursive\ \ P(x,y,t)\ \ As \\
(Select & x, y, t \\
From & A) \\
UNION & \\
(Select & B.x,\ B.y,\ B.t \\
From & B,\ \ P \\
Where & P.t = B.t - 1 \\
 & P.x = B.x, \quad P.y = B.y)
\end{array}
$$

The above recursive query returns the spatiotemporal locations (x, y, t) that satisfy either part (1) or part (2) of Definition 3.3.1. Relation P stores all the spatiotemporal locations that have property P, according to Definition 3.3.1.

Chapter 4

Drought Online Analysis Application

Although the concept of drought is well-known for most people as a deficit of precipitation, it is hard to precisely define the beginning and ending time of a drought. Precipitation has to be combined with time and location to represent a drought condition. Meteorologists have developed many drought indices to help the analysis of drought. Standardized Precipitation Index (SPI) (McKee, Doesken & Kleist 1993) is one of the common and simple measures of drought. The original SPI data is calculated by the entire precipitation data stored in a point-based spatiotemporal database sampled from geographically distributed weather stations, that is, only the sample points have SPI values.

We use SPI values to calculate the drought regions. Values of SPI range from 2 and above (extremely wet) to -2 and less (extremely dry) with near normal conditions ranging from 0.99 to -0.99. McKee et al. (McKee et al. 1993) defined the criteria for a *drought event* for any time scale as follows. A drought event occurs at any time the SPI is continuously negative after reaches an intensity of -1 or less. The event ends when the SPI becomes positive. Therefore, each drought event has a duration

defined by its beginning and end, and an intensity for each time unit that the event holds.

We describe the problem in the format of Definition 3.1.2 as follows:

Definition 4.0.2 An area is in *drought* during week T if during T its SPI value is either

(i) -1 or less, or

(ii) between -1 and 0 *and* it's in *drought* during $T-1$.

The recursive definition and spatial-temporal features of the drought problem make it very difficult to solve by most database systems. However, constraint databases (Revesz 2002) provide an efficient way to store the spatiotemporal data, while Datalog query language supports recursion. In this Chapter, we construct the Drought Online Analysis System (DOAS) based on the web-accessible MLPQ system as an efficient solution for the drought spatial extent problem.

Section 4.1 describes the design and software architecture of the DOAS system. Section 4.2 gives the major queries that are implemented in the DOAS system.

4.1 The DOAS System Architecture

The DOAS system is a server-based, thin-client distributed spatiotemporal analysis system. Unlike other relational database based geospatial systems, DOAS uses constraint database to represent its spatiotemporal data and builds the system based on MLPQ constraint database system, which is one of the first constraint database prototype systems that implements rational linear constraint databases and queries. The most significant features of the DOAS system include:

- Support recursive Datalog query on spatiotemporal data.

- Support vivid animation representation of spatiotemporal result.
- Accurate and efficient geo-information interpolation method guarantees the output's reliability.
- Simple, concise and independent query solution makes the program easy to understand and maintainable.
- Web accessible MLPQ constraint database system enables flexible and easy development of web-accessible interfaces.
- Server-based, thin-client three-tier software architecture offers unparalleled flexibility to foster collaboration, exploration, and discovery. Thin clients are small, affordable, easy to maintain, reliable, and secure.

The DOAS system was implemented in a three-tier software architecture shown in Figure 4.1.

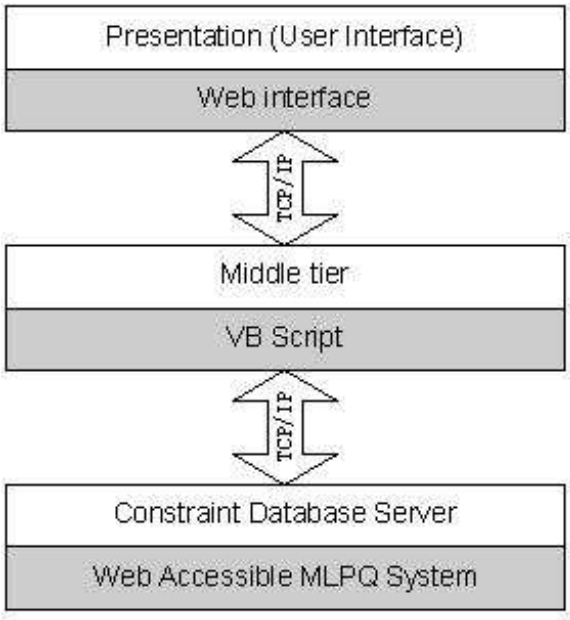

Figure 4.1: The three-tier software architecture based on MLPQ Web Accessible System

The first tier is the MLPQ Constraint Database system v3.0. MLPQ is the abbreviation for *Management of Linear Programming Queries*. It supports both SQL and Datalog queries, minimum and maximum aggregation operators over linear objective functions, and some other operators (Revesz 2002). It is a suitable tool for representing, querying, and managing spatiotemporal constraint database.

The second tier is the software that works between the user interfaces and the MLPQ system. The program in this tier accepts the requests from the user interfaces, translates the request to be the query that the constraint database server is understandable, and sends the query to the server through Socket. When the server generates some outputs, it will explain the outputs and send it back to the third tier.

The third tier is the user interfaces (client) layer. We use HTML language to construct the user interfaces in this paper. Through the page design, we provide a convenient way for people to express the user's requests naturally, access the middle tier remotely and display the output easily. In a B/S application like DOAS, the client is a "thin" client. There are few performance requirements for the client machine. Thin clients are small, affordable, easy to maintain, reliable, and secure. Figure 4.2 shows the homepage of the Drought Online Analysis system. Based on the three-tier software architecture, people can conveniently access DOAS anywhere through the Internet.

4.2 Implementation and Analysis

In GIS applications, (x, y) are usually given as UTM (Universal Transverse Mercator) coordinates (Terry 1996) (Dutch 2003) for easting and northing, respectively. Chomicki (Chomicki 1994) and Toman (Toman 1996) discuss the time representation issue of instances versus intervals. The sample *SPI_Point* relation shown in Figure 2.1 is obtained from the Unified Climate Access Network (UCAN). We use one kilometer

as the unit of the easting and northing attributes.

Our original point-based data consisted of records collected continuously for over one hundred years at 48 main weather stations spread out all over Nebraska. To represent the SPI values beyond these 48 weather station points, it is necessary to do some spatiotemporal interpolation on them. In this paper, we use a 2-D spatial interpolation function for triangles to interpolate and translate the original point-based relational weather information into a constraint relation as described in Li and Revesz (Li & Revesz 2003).

Currently, DOAS implements six queries. Now we give a detailed description of some of the sample queries.

A *color band display* is appropriate when an entire region needs to be displayed. In a color band display each color represents a given range of SPI values.

Query 4.2.1 Display the SPI value of the entire region for each week during the time period T_{start} to T_{end} in a color band.

We use multiple SQL queries for this problem. Each query generate a relation which represents the region where the SPI values falls into a given range.

Next we assign different colors for each relation and display them together. Since the ranges cover all possible SPI values, the result is the whole map of the state in color bands. For example, to represent the band with SPI value ranging from -1 to

+1, we can use following SQL query:

$$
\begin{array}{ll}
CreateView & normal \\
Select & SPI.id, \\
 & SPI.east, \\
 & SPI.north, \\
 & SPI.t \\
From & SPI \\
Where & SPI.s >= -1 \text{ and } SPI.s <= 1
\end{array}
$$

Figure 4.3 is a sample result of color bands in multi-pictures style. If the user choose animation style, then the system will display the pictures one-by-one according to their time stamps.

Values of SPI range from 2 and above (extremely wet) to -2 and less (extremely dry) with near normal conditions ranging from 0.99 to -0.99. McKee et al. (McKee et al. 1993) defined the criteria for a *drought event* for any time scale as follows. A drought event occurs any time the SPI is continuously negative and reaches an intensity of -1 or less. The event ends when the SPI becomes positive. Therefore, each drought event has a duration defined by its beginning and end, and an intensity for each month that the event continues.

Query 4.2.2 Display the map of the drought area at time T.

Using the above definition of drought based on SPI, it is easy to write a recursive

Datalog query that finds the drought places at any week t as follows:

$$\begin{aligned} drought(i,x,y,t) \ :- \ & SPI(i,x,y,t,s), \\ & s \leq -1.0, \\ & SPI(i,x,y,t1,s), \\ & t - t1 = 1, \ \ s \leq 0. \end{aligned}$$

$$\begin{aligned} drought(i,x,y,t) \ :- \ & SPI(i,x,y,t,s), \\ & s > -1, \ \ s \leq 0, \\ & drought(i,x,y,t1), \\ & t - t1 = 1. \end{aligned}$$

The first rule means every place with a SPI value less than or equal to -1 at time t is in drought. The second rule means that a place is also in drought at time t if its SPI value is negative at time t, and it has been in drought at time $t-1$.

The above query calculates the drought area for every week. Since we only need to return the map of drought area at week T, we can make the above query more efficient by modifying it as follows:

$$
\begin{aligned}
dry(i,x,y,t) \quad &:- \quad SPI(i,x,y,t,s),\\
&\qquad\quad s \leq -1.0,\\
&\qquad\quad SPI(i,x,y,t1,s1),\\
&\qquad\quad t - t1 = 1, \;\; s1 \leq 0.
\end{aligned}
$$

$$
\begin{aligned}
possibledry(i,x,y,t) \quad &:- \quad SPI(i,x,y,t,s),\\
&\qquad\quad s > -1, s < 0.
\end{aligned}
$$

$$
\begin{aligned}
candidate(i,x,y,t) \quad &:- \quad possibledry(i,x,y,t),\\
&\qquad\quad t = T.\\
candidate(i,x,y,t) \quad &:- \quad possibledry(i,x,y,t),\\
&\qquad\quad candidate(i,x,y,t1),\\
&\qquad\quad t1 - t = 1.
\end{aligned}
$$

$$
\begin{aligned}
drought(i,x,y) \quad &:- \quad dry(i,x,y,t),\\
&\qquad\quad t = T.\\
drought(i,x,y) \quad &:- \quad candidate(i,x,y,t),\\
&\qquad\quad dry(i,x,y,t1),\\
&\qquad\quad t - t1 = 1.
\end{aligned}
$$

Relation $dry(i,x,y,t)$ is the place that has SPI value less than or equal to -1 at time t and has negative SPI value at time $t-1$.

The $candidate(i,x,y,t)$ relation contains those areas which may be in drought at time T. An area at time $t < T$ is a candidate drought area at time T if it always stays in *possibledry* condition (i.e., has an SPI value between -1 and 0) from time t to time T. The calculation starts from time T and goes backward. The candidate region at time t is the intersection of the candidate region at time $t-1$ and the possible

dry region at time t. It is easy to see that in each iteration of this recursive Datalog query the number of tuples generated monotone decreases and finally ends up with an empty set at some time $t' \leq T$.

Based on the *dry* and *candidate* relations, we can now give the following new Datalog query definition of drought :

An area is in drought at time T if either it is dry at time T, or it belongs to both candidate relation at time t and dry relation at time $t-1$ for each $t \leq T$ in the database.

Figure 4.5 on the next page shows the evaluation process of this recursive Datalog query. Blue area is the area of *candidate* relation. The red area represents the *dry* relation.[1] We can see that the red area increases or decreases randomly, while the blue area monotone decreases. For any time $t_i < t_j$ we have $region\_of(candidate(i, x, y, t_i)) \subseteq region\_of(candidate(i, x, y, t_j))$. Figure 4.4 above is the map of drought region at time 10515, which represents the 27th week of year 2001.

Query 4.2.3 Calculate the number of weeks that location (P_x, P_y) is in drought during time period (T_{start}, T_{end}).

We solve this problem by applying recursive Datalog and aggregation SQL queries together. First, we use the following recursive Datalog query to find the week when the given point is in drought:

[1]This figure looks best if printed in color. In a black-and-white printout, the blue areas will be black and the red areas gray.

$$
\begin{aligned}
dt(id, week) \ :- \ & SPI(i, P_x, P_y, week, s), \\
& id = 0, \ \ s \leq -1.0, \\
& SPI(i, P_x, P_y, week1, s1), \\
& s1 < 0, \ \ week - week1 = 1, \\
& week \geq T_{start}, \ \ week \leq T_{end}.
\end{aligned}
$$

$$
\begin{aligned}
dt(id, week) \ :- \ & possible\_dry(i, P_x, P_y, week), \\
& dt(id, week1), \\
& week - week1 = 1, \\
& week \geq T_{start}, \ \ week \leq T_{end}.
\end{aligned}
$$

Relation *dt* contains the serial number of those weeks when the given location is in drought. Hence we can use an SQL query with aggregation to obtain the total number of weeks in the *dt* relation:

```
View name   drought_weeks(weeks)
Select      count(dt.week)
From        dt
```

The above SQL query returns the number of weeks in relation *dt*. This solution is a good illustration of how SQL and Datalog queries can be used together to solve a difficult problem that requires several steps that need both the recursive power of Datalog and the aggregation power of SQL.

4.3 Comparison

For spatiotemporal applications, many GIS systems (Longley et al. 2001) will not store the interpolation results, and hence need to run a built-in interpolation function

for every query to approximate the data at the requested point and time. On the other hand, constraint databases only need to run the interpolation function once and then can store the interpolation results in a constraint relation. Hence no further computer-intensive interpolation processes are required in executing a sequence of user queries of the above type, as is usually the case.

Recursive queries are *not expressible* using the basic query languages of GIS systems. Some relational database and knowledge-based systems provide recursive queries, but they do not provide spatiotemporal data representation. Hence the drought query cannot be expressed in any previous system in any easy way. They would usually require some special functions to be written in a programming language like C or C++ and added to a library. In contrast, DOAS only needs standard SQL and Datalog queries and calls to the constraint database system MLPQ. Therefore, the DOAS programs are very simple, declarative queries that are high-level and easy to maintain.

This is important, because recursive problems on spatiotemporal data are frequent enough to need a general and simple solution method. For example,

- A city or county is *highly-infected during week* T if (i) it reports during week T ten or more new infected persons or (ii) it is highly-infected during week $T-1$ and reports during week T between three and nine new infected persons.

- An ecosystem is in *danger during month* T if (i) the density of one important plant species decreases more than 10% during month T, or (ii) it is in danger during month T and the density of the same species decreases more than 2% during month T.

DOAS can solve the above and many other similar recursive spatiotemporal problems with equal ease.

Figure 4.5 shows the evaluation process of this recursive Datalog query. The red area represents the A relation. Green area is the B relation and blue area is the area of C relation.[2] For any integer $m_i < m_j$ we have $C_{m_i} \subseteq C_{m_j}$.

4.4 Other Similar Applications

The RecIV system provides a general method to visualize recursively defined concepts. It can be easily applied in many different research areas. In Section 4.4.1, we describe a sample application that are implemented by the RecIV system. In Section 4.4.2, we compare the performance of the naive and the optimized algorithms in the case of the drought visualization problem.

4.4.1 Air Pollution Visualization Problem

Clean air is a basic requirement for human health and well-being. The development of toxicity is a complex function of the interaction between a pollutant concentration and the exposure duration. After peak exposure for a short period, a pollutant may cause acute, damaging effects. Exposure to a lower concentration of a pollutant for a long period of time may cause irreversible chronic effects. It is easier to evaluate the effects of a short-term peak exposure to a chemical than a prolonged exposure to a lower concentration. However, in some cases, the low concentration over a long period can have more impact than the pattern of peak exposure (WHO-Europe 2000).

As stated in (WHO-Europe 2000), a similar situation occurs for effects on vegetation. Plants are generally damaged either by short-term exposures to high concentration or by long- term exposures to low concentration. Therefore, both short-and long-term guidelines to protect plants are proposed in (WHO-Europe 2000). We focus on

[2]This figure looks best if printed in color. In a black-and-white printout, the blue areas will be black, the red areas gray, and the green areas light gray.

the effects of polluted air on plants because the exposure - response relationship between pollutants and plants is more accurate than that on human health. However, the rules can be easily applied on human health if required.

We can evaluate and visualize the air pollution based on the safe and critical level of the pollutants given in the air quality guidelines (WHO-Europe 2000).

When the pollutant concentration exceeds a critical level, the probability of damage is considered to be non-zero. That implies a non-sustainable force. This force can lead to actual damage at any point in time. If the concentration keeps below the safe level, then the pollution is safe for most acceptors.

When the concentration level of the pollutant is between the safe level and the critical level, then there is no immediate damage caused by the pollutant, but the plant may accumulate this chemical in the body and produce adverse effects after a long period of time. The long-term pollution area can be defined as follows:

Definition 4.4.1 An area is *polluted* during week T if during T we measure either

(i) *critical* level or more pollutant in the air or

(ii) between $safe$ and *critical* level of pollutant in the air
and it was *polluted* during week $T-1$.

The critical and safe levels may not be unique in different regions. For example, the critical level of sulfur dioxide (SO_2) in forest and natural vegetation areas are only two-third to one-half of that in the agricultural crop areas. The World Health Organization (WHO-Europe 2000) provides a table of the critical levels for the effects of sulfur dioxide on vegetation, which is listed in Figure 4.6. To handle this problem, we can apply the general solution for each region and combine the results together in the output relation.

We choose sulfur dioxide (SO_2) as the sample pollutant to visualize the air pollution in RecIV system. Applying the general solution for each region and combining

them together, we can get the output image of the polluted regions shown in Figure 4.7. The area we evaluate here has four regions. The green region on the left represents natural vegetation area. The yellow area on the right-bottom of the map represents agricultural crops. The dark green top-right area is a forest. The blue region is a river. The red point in the middle of the map represents a factory which releases SO_2 into the air every day.[3] The legend in Figure 4.7 gives the color of each object in the image. Each of these objects is represented by a spatial constraint relation in the input database. The user can specify them as the overlay objects through the web interface. The image in Figure 4.7 is the combination of the polluted regions, different plant regions, the river object, and the factory object. The user can calculate the total area of each object by selecting it from the legend. The user can also calculate the intersection or difference of the total areas of any two objects.

4.4.2 Performance Analysis

To evaluate the improvement of the optimization, we show in Figure 4.8 the execution result on the drought visualization problem.

In our experiment the input constraint database relations were M, which stored the interpolated SPI data of year 2001 using 4264 tuples, and the corresponding relations A and B with 608 and 2511 tuples, respectively.

The experiments show that the naive method is inefficient in practice and unreliable. In the worst case it even exceeded the available memory of the testing machine, hence it generated no output after running for over three hours or 10800 seconds. The same case was the hardest for the optimized method too, but it generated only $218+99 = 317$ new tuples and completed the evaluation in $2153+118 = 2271$ seconds. Hence the optimized method is more efficient and reliable than the naive method is.

[3] In a black-and-white printout, the legend in Figure 4.7 can be a better explanation for each object.

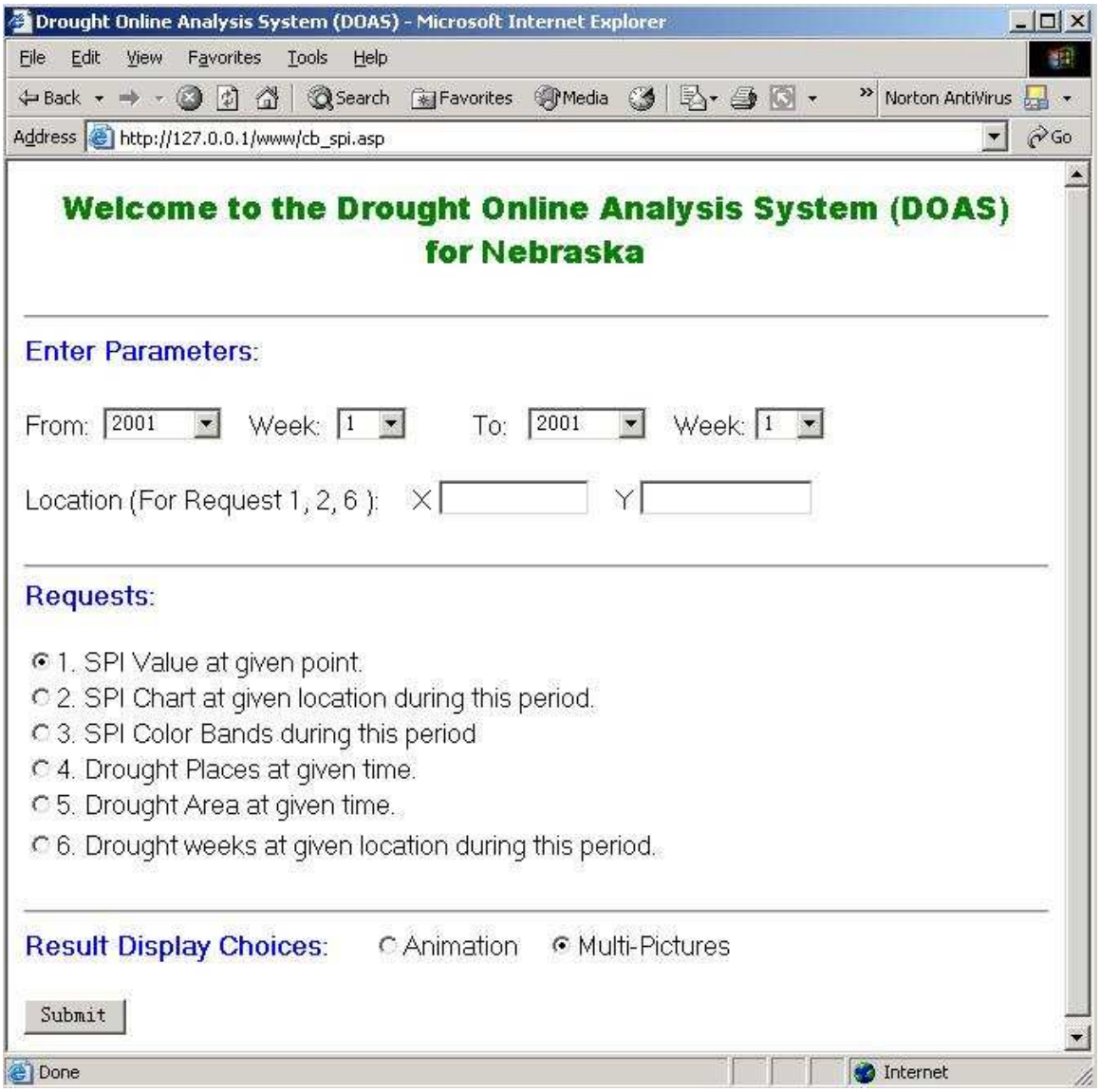

Figure 4.2: Homepage of the DOAS: Drought Online Analysis System

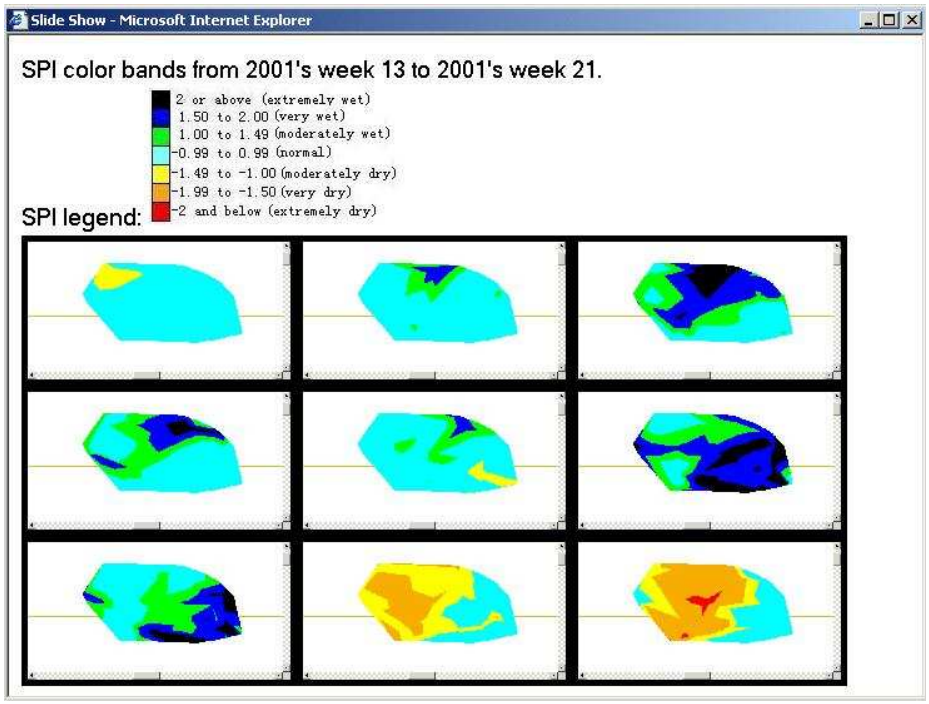

Figure 4.3: Color bands for SPI value

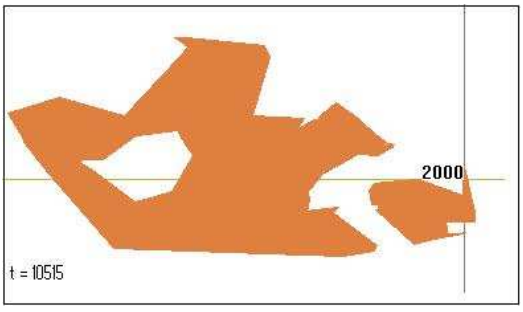

Figure 4.4: Drought region in Nebraska at week 10515 (the 27th week of 2001)

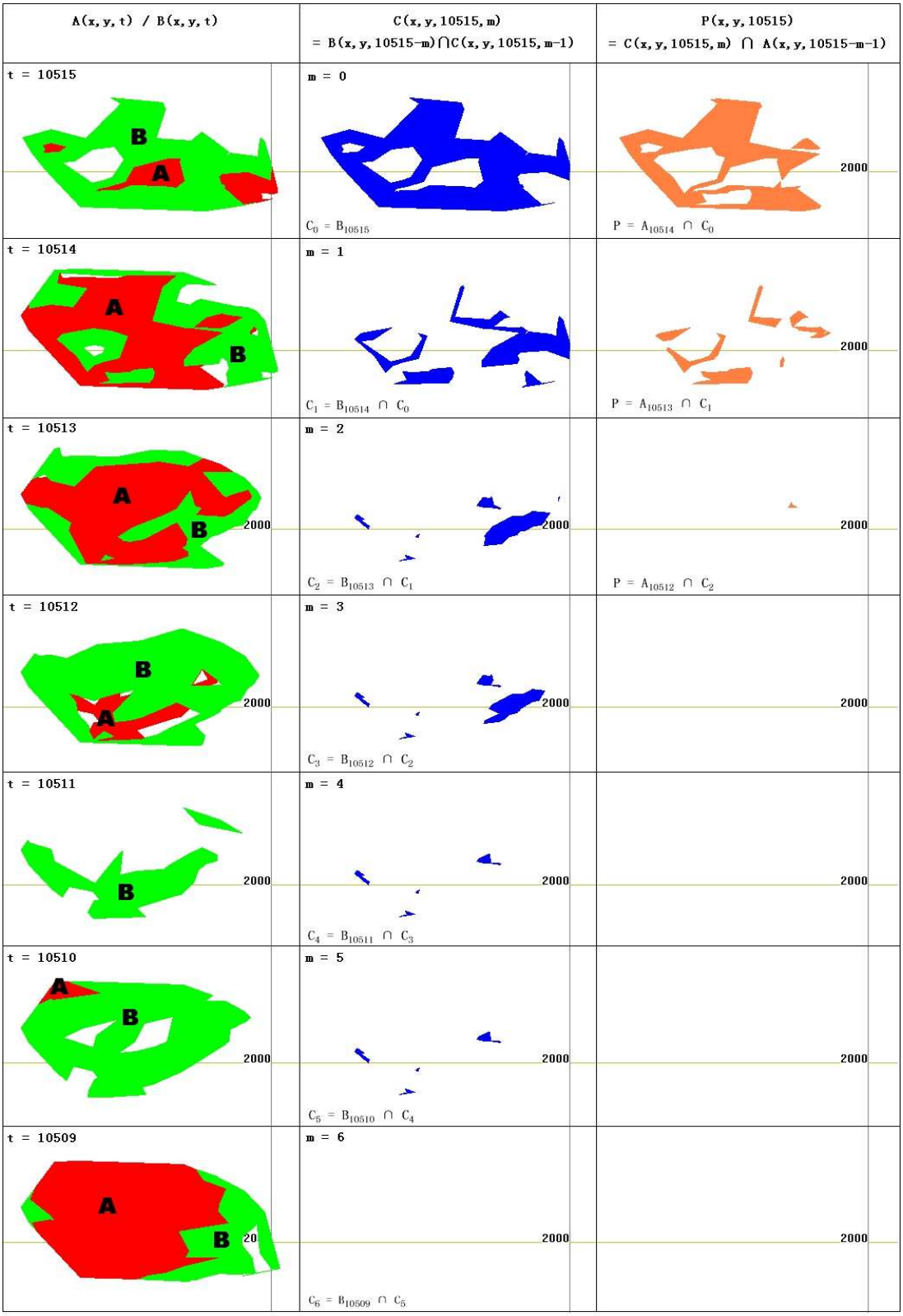

Figure 4.5: Process of finding the drought places at time 10515 (the 27th week of year 2001).

Vegetation Type	$\mu g/m^3$	Time period
Agricultural crops	30	Annual and winter mean
Forests and natural vegetation	20	Annual and winter mean
Forests and natural vegetation (*a*)	15	Annual and winter mean
Lichens	10	Annual mean
Forests (*b*)	1	Annual mean

(a) where accumulated temperature sum above $5\,^{\circ}C$ *is less than* $1000\ ^{\circ}C \cdot$ *days per year.*

(b) where ground level cloud is present at least 10% *time*

Figure 4.6: WHO Guidelines for critical levels of sulfur dioxide (WHO-Europe 2000).

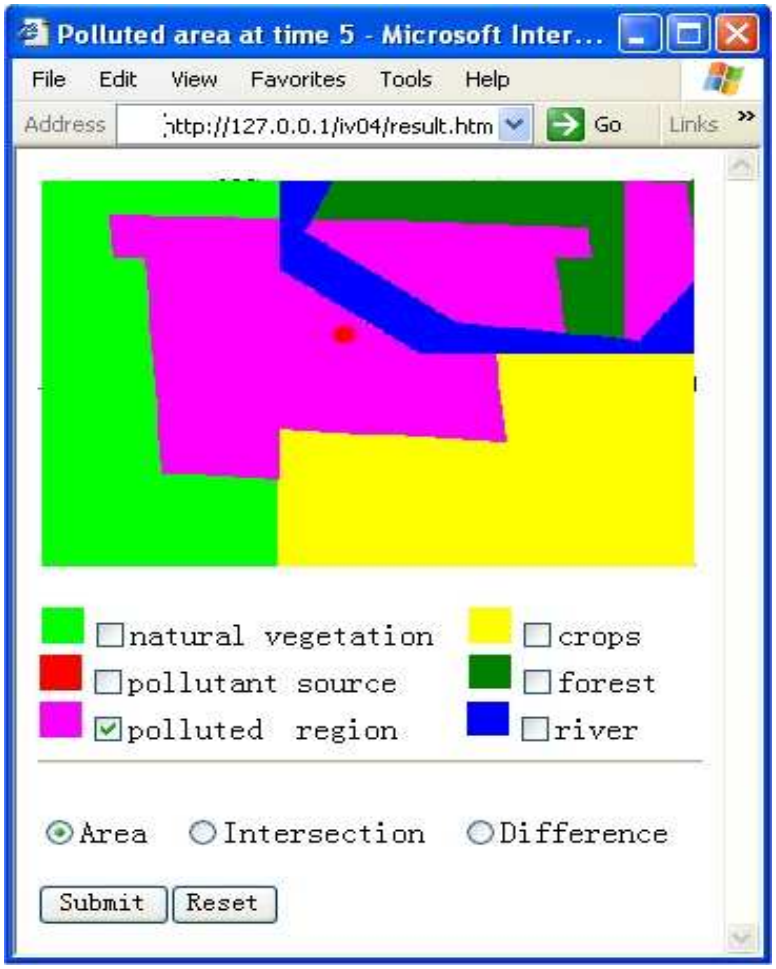

Figure 4.7: Visualization of polluted regions.

Weeks	Measure	Naive	Optimized	
		P	C	P
10489-10499	seconds	1060	669	48
	tuples	80	150	0
10499-10509	seconds	1020	324	28
	tuples	217	86	59
10505-10515	seconds	$\geq$ 10800	2153	118
	tuples	N/A	218	99
10519-10529	seconds	4706	325	18
	tuples	397	40	0
10529-10539	seconds	2791	1183	67
	tuples	97	203	5

Figure 4.8: Comparison of the naive and optimized methods on drought data.

Chapter 5

West Nile Virus Application

This section is organized as follows: Section 5.1 describes the source data we use for the West Nile Virus analysis (in Section 5.1.1) and its interpolation and storage in a constraint database (in Section 5.1.2). Section 5.2 describes the methodology we use in designing the West Nile Virus Information System (WeNiVIS) for tracking and reasoning about epidemics. Section 5.2.1 proposes a general recursive definition for predicting high-risk areas for epidemics. Section 5.2.2 shows the (WeNiVIS) we developed for the WNV analysis. Section 5.3 presents major results and benefits of this project. Finally, Section 5.4 discusses some specific issues about our method and system.

5.1 The West Nile Virus Data

5.1.1 Data Source

West Nile Virus (WNV) was originally discovered in the West Nile district of Uganda in 1937. It has been known to cause infection and fevers in humans in Africa, West Asia, and the Middle East. The first appearance of WNV in the United States occurred in 1999 in New York City. Since then, the disease has spread across the

United States. In 2003, WNV activity occurred in 46 states and caused illness in over 9,800 people (U.S. Geological Survey 2004).

WNV is transmitted to humans through mosquito bites. Mosquitoes become infected when they feed on infected birds that have high levels of WNV in their blood. Infected mosquitoes can then transmit WNV when they feed on humans or other animals (U.S. Geological Survey 2004).

We obtained data on the spread of WNV in Pennsylvania in 2003 from Pennsylvania's West Nile Virus Control Program (Pennsylvania's West Nile Virus Control Program 2004). The data include dead wild birds, mosquitos, sentinel chickens, equine (horse) veterinary and confirmed human cases of WNV as explained below.

- **Dead wild bird:** In Pennsylvania's WNV Control Program, the dead birds were collected by passive surveillance, relying on public reporting through telephone and Internet. Dead birds and infected birds that display erratic behavior are highly visible for casual observers to identify and report in the areas where WNV may be active. When a dead bird is sighted, the information about that bird and its location is recorded. Then samples from each bird are tested for WNV and the results are recored to the database.

- **Mosquito:** Mosquitoes were routinely collected in surveillance locations. All mosquitoes collected in one effort create a sample of mosquitoes. Only adult mosquitoes in the sample are tested for WNV.

- **Sentinel chicken:** Pennsylvania's surveillance system includes sentinel chickens. Several flocks are housed near areas with dense human populations and stagnant water sources in hopes of alerting medical experts to the presence of the virus. Samples from these sentinels are collected weekly and tested for WNV.

- **Equine:** Equine diagnostic blood samples submitted by veterinarians across the state are also tested for WNV.

Since the datasets are not recorded during the weekends, the data is summarized to weekly data to generate a continuous surveillance dataset.

We first compare the onset of several kinds of animal infections hosts with the onset of human infections. To show the time relationship between the number of various animal disease carriers found and the number of human infection cases reported, we show in four separate bar charts in Figure 5.1 the number of the four animal diseases on the top and the number of human cases on the bottom in reverse, i.e., with negative y-axis values. Figure 5.1 shows that there is a time lag of about six weeks between the onset of sentinel chicken infections and human cases. Similarly, there is a time lag of about three, one, and zero weeks between the infected wild bird, mosquito, and equine veterinary cases with respect to the human cases. Hence while Figure 5.1 shows that each of the three types of animal WNV infections are strongly related to the human WNV infections, the various animal cases provide different advance warnings of human WNV epidemic outbreaks.

In addition, Figure 5.1 suggests that the onsets of WNV epidemics have a two weeks cycle in both humans and animals. This time cycle is above the mean incubation period of hospitalized patients, which is 5.3 days (D. Nash, F. Mostashari, A. Fine, et al. 2001), and within the range of human infections, which is 3-15 days (Centers for Disease Control and Prevention (CDC) 2002).

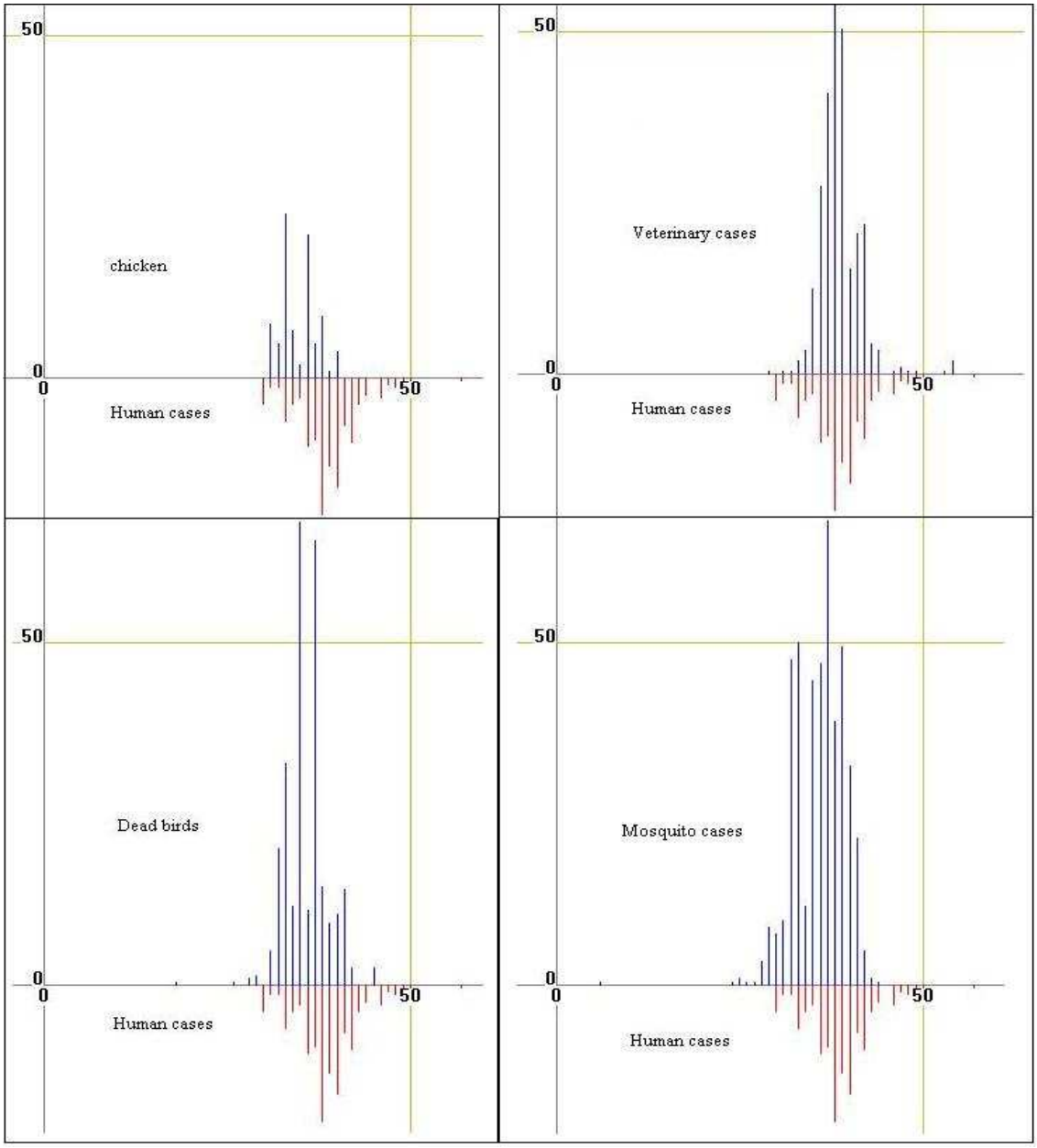

Figure 5.1: The comparison of time shifts between the infections on human and various types of animal hosts.

5.1.2 Epidemiological Data Stored in Constraint Databases

A point-based spatiotemporal data set only stores information of some sample points at some sample times. That is usually what one can obtain as the raw data for infectious diseases. To represent the features beyond those finite spatiotemporal points, it is necessary to do some *spatiotemporal interpolation* on them. Interpolation requires some basic assumptions about the nature of the point data set. Theophilides et al. (Theophilides et al. 2003) makes the following interpolation assumptions:

1. West Nile Virus is a continuous phenomenon across space;
2. Humans are infected at their resident places;
3. Nonrandom space-time interaction of bird deaths is attributed to West Nile Virus infection;
4. Each dead bird has an equal opportunity of being reported;

We make similar assumptions. We differ from (Theophilides et al. 2003) by applying instead of Knox spatiotemporal interpolation a 2-D shape function-based interpolation method, which Li and Revesz (Li & Revesz 2004) found to be both the most reliable among several well-known spatial and spatiotemporal interpolation methods. It also happens to be easily implementable in constraint database systems, as we will see later.

To translate the relational dataset to constraint database, we need to find some data point from each county in Pennsylvania. We have the positions of 102 cities and towns of Pennsylvania. We pick as a sample point the biggest city of each county based on the 1990 city population census in Pennsylvania. If a county is too small to have any cities on the list, we arbitrarily pick the center of the county as the sample point.

Figure 5.2 shows a point-based spatiotemporal data set consisting of the vertices shown there, and its "Delaunay Triangulation" network (Goodman & O'Rourke 1997).

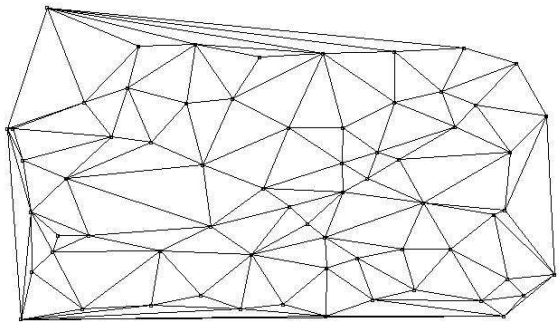

Figure 5.2: The triangulated network of sample points.

Figure 5.3 shows a part of the constraint relation that describes the result of a linear shape function-based interpolation (Li & Revesz 2004). The constraint relation contains many constraint tuples (rows).

Each constraint tuple contains three or four constraints. The first three constraints represent the area of a triangle as the intersection of three linear inequality constraints over x and y. The optional fourth constraint is used only when there is a non-zero number of dead WNV-infected birds in the triangular area. Then the fourth constraint is a linear equation that expresses the relationship between the number n of dead WNV-infected birds and the spatial variables x and y. In each location (x, y) the number of dead WNV-infected birds is an indicator of the danger level of WNV infection to humans. We can predict the number of human WNV infections by some product of the danger level and the total human population in that location.

In each tuple the *week* attribute always represents the time measured in weeks past January 1st, 2001, hence the week starting January 1st, 2001 would be week 1, the week starting January 8th, 2001 would be week 2 etc. Finally, we also give a unique *id* value to each triangular area for easy identification.

id	lat	long	week	WNV birds	
1	x	y	122	0	$x - 0.1373y \leq -85.8305$, $x + 0.3969y \leq -64.1339$, $-x - 0.0960y \leq 76.5405$
2	x	y	122	0	$x + 1314.0605y \leq 52284.1758$, $-x + 126.0875y \leq 5101.4253$, $x - 211.5507y \leq -8504.7412$
⋮	⋮	⋮	⋮	⋮	⋮
65	x	y	130	n	$-x + 1.7315y \leq 147.7618$, $-x - 4.0891y \leq -85.0644$, $x - 0.2581y \leq -88.3867$, $x - 0.2581y + 0.4391n = -88.3867$
⋮	⋮	⋮	⋮	⋮	⋮

Figure 5.3: The constraint-based interpolated weekly WNV infected birds dataset.

5.2 Methodology

In this section, we describe the methods and implementation of the epidemiological information system. Section 5.2.1 describe the methods used in modeling the epidemiological problem and Section 5.2.2 describe the implementation of WeNiViS system.

5.2.1 Recursive Definitions and Implementation

In Chapter 3, we have Definition

In order to implement Query 3.3.1, we also extended the MLPQ system to allow recursive SQL queries. The improvement includes the modification of the MLPQ system's SQL parser and significant C++ programming to make the recursive SQL language suitable as a basis for reasoning about constraint data.

5.2.2 The WeNiVIS System

The West Nile Virus Information System (WeNiVIS) is an epidemiological information system designed to manage the spatiotemporal WNV information. It is built on top of the MLPQ constraint database system and has many unique functionalities.

The WeNiVIS system has four major components as follows:

1. **Recursive reasoning interface** provides a convenient user interface to define the recursive epidemiology concepts.

2. **Visualization window** can accept and display the spatial result of the reasoning in a set of snapshots or animation.

3. **Time navigating bar** can be used to track visually the spread of WNV over time. It is especially helpful when users have to track and compare several maps with different indicators using different time shifts.

4. **Formated SQL query interfaces** are helpful to guide users in generating correct SQL queries.

Figure 5.4 shows the interface of the WeNiVIS system. The user can generate many visualization windows in the frame window. In Figure 5.4, each visualization window represents one indicator of the WNV infections.

As shown in the center of Figure 5.4, the recursive reasoning dialog box provides a simple interface for the user to generate complex recursive queries. It only asks for the values of several critical parameters and can automatically generate a recursive query based on the general format defined in Query 3.3.1. The new recursive query can be sent to the MLPQ system and be evaluated in it.

The WeNiVIS system has two kinds of time navigating bars. The local time navigating bar on top of each visualization window is used to choose the time of the map displayed in that window. The changing of the time in each visualization window

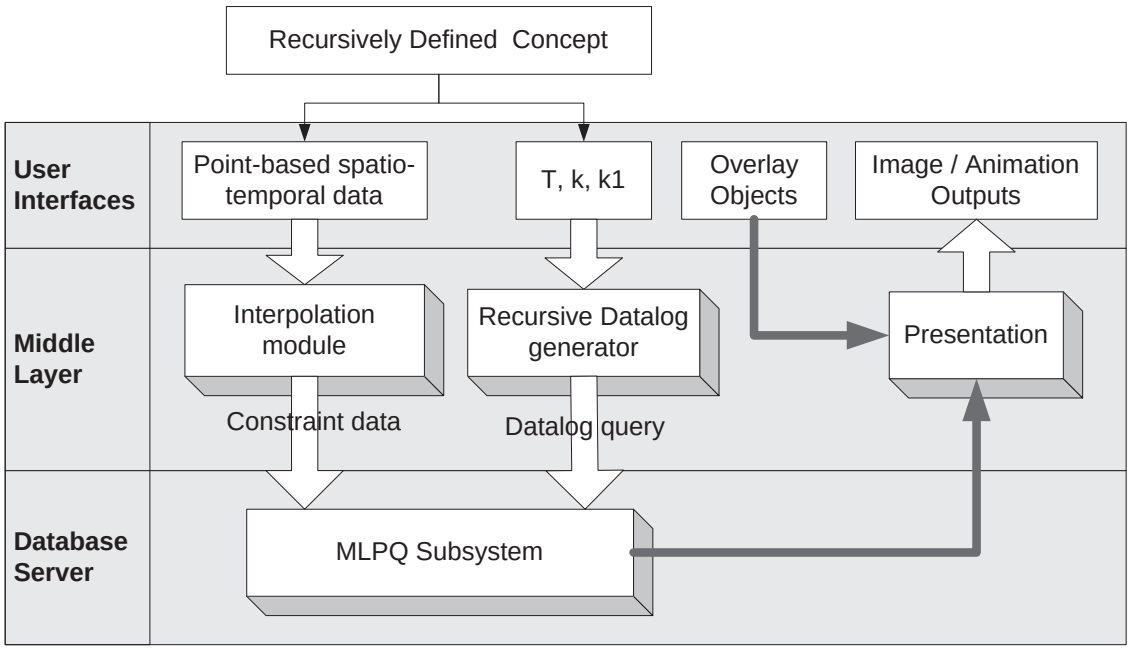

Figure 5.4: The WeNiVIS system in analyzing WNV infections.

by its own time navigating bar is independent of the time in the other windows. On the other hand, the global time navigating bar in the frame window can be used to change the time of all visualization windows within that frame window at the same speed. For example, the user can first set the time of one visualization window as 122 and the time of another window as 125 by their own time navigating bar. Then the user can use the global time navigating bar to browse both of these two windows simultaneously. This function is helpful when the user wants to check visually the effects of the distribution of the several indicators, which may have different time delays with respect to the target property.

The formated SQL query interface has a fixed format for different kind of SQL queries. The WeNiVIS system supports basic, aggregation, set, nested and recursive queries on constraint databases.

5.3 Results

5.3.1 Flexible User Interfaces with a High-Level Language

In the WeNiVIS system, users can do many interesting reasonings based on the spatiotemporal dataset and the recursively defined risk evaluation through the formated SQL query interfaces.

We have three tables defined as follows:

- **Risk(x, y, t, w)** is the result of the recursive risk evaluation based on Definition 3.3.1 and Query 3.3.1. It stores the predicted risk value as w at location (x, y) during time unit t.
- **City(id, x, y, name, pop)** stores information about the cities in Pennsylvania. Attribute *pop* is the population of the city based on the census of year 1990.
- **Event(id, title, organizer, t, cid)** stores information about conferences and other events that are scheduled to be held in city *cid* at time t by the *organizer*.

Some of these queries are described as follows:

Query 5.3.1 Pennsylvania health officials want to find and warn the organizers of the events that are scheduled to be held in a city when it is at a high risk of WNV infection.

$$\begin{array}{ll} Select & E.organizer \\ From & City\ as\ C,\ \ Event\ as\ E,\ \ Risk\ as\ R \\ Where & C.x = R.x,\ \ C.y = R.y,\ \ R.w = 1,\ \ R.t = E.t,\ \ C.id = E.cid \end{array}$$

Query 5.3.2 Find the total area of Pennsylvania at risk of WNV in week 141.

First, we find the (x, y) locations at risk as follows

$$\begin{array}{ll} Select & x, y \\ From & Risk \\ Where & t = 141, \;\; w = 1 \end{array}$$

Then we can call the *area* function in the system to compute the area of the above.

Query 5.3.3 Find the total population of the major cities in the areas defined above.

$$\begin{array}{ll} Select & sum(C.pop) \\ From & City\ as\ C, \;\; Risk\ as\ R \\ Where & C.x = R.x, \;\; C.y = R.y, \;\; R.t = 141, \;\; R.w = 1 \end{array}$$

Query 5.3.4 A big sport event is scheduled to be held in Pennsylvania at week 141. Any city with more than 50,000 population in the state can host the event. People want to know which city will be safe from WNV infections during the event time.

$$\begin{array}{ll} Select & id,\ name \\ From & City \\ Where & pop >= 50000, \;\; id \;\; NOT\ IN \\ & (Select \;\; C.id \\ & From \;\; City\ as\ C, \;\; Risk\ as\ R \\ & Where \;\; C.x = R.x, \;\; C.y = R.y, \;\; R.t = 141, \;\; R.w = 1) \end{array}$$

The above queries are easy to understand and are efficiently evaluated by the WeNiVIS system. Comparing the size and complexity of a C++ or Java program needed to solve the same problem based on relational databases, SQL and constraint databases provide a more concise and manageable approach. A simple and indepen-

dent query solution makes the program easy to understand and maintain.

5.3.2 Enhancement of Tracking and Reasoning about Epidemics

The WeNiVIS system is used in analyzing the spread of the West Nile Virus epidemic in Pennsylvania. Figure 5.5 displays the distribution of infected animals and humans for week 139 using a color band display with darker colors meaning more infections. We have four different small maps. The distribution of infected wild birds is shown in the upper-left, horses in the upper-right, mosquitos in the lower-left, and humans in the lower-right window. The figure shows that the four cases are only weakly correlated with each other during the same week. Hence a time delay is needed in the analysis.

We already mentioned in Section 5.1.1 that the dead wild bird data is followed by the human cases after about three weeks. Figure 5.6 shows on the left side the predicted high-risk areas based on the infected wild bird data alone and on the right side the actual observed human cases. In generating the predictions we used Definition 3.3.1 with values $k_1 = 0.4$ and $k = 1.4$. We found these values by experiments. The smaller values of k_1 and k will generally yield more areas, but they may over-predict the risks and be less accurate.

In this article, the WeNiVIS system is used to visualize and reason about the spread of West Nile Virus in Pennsylvania as a sample application. Beside this particular case, the general methodology used in the implementation of the system is also appropriate for many other applications.

First, with the additional data available, the system can be directly applied to any other states in which West Nile Virus may occur. We only need to enter the new state's data in a constraint database format.

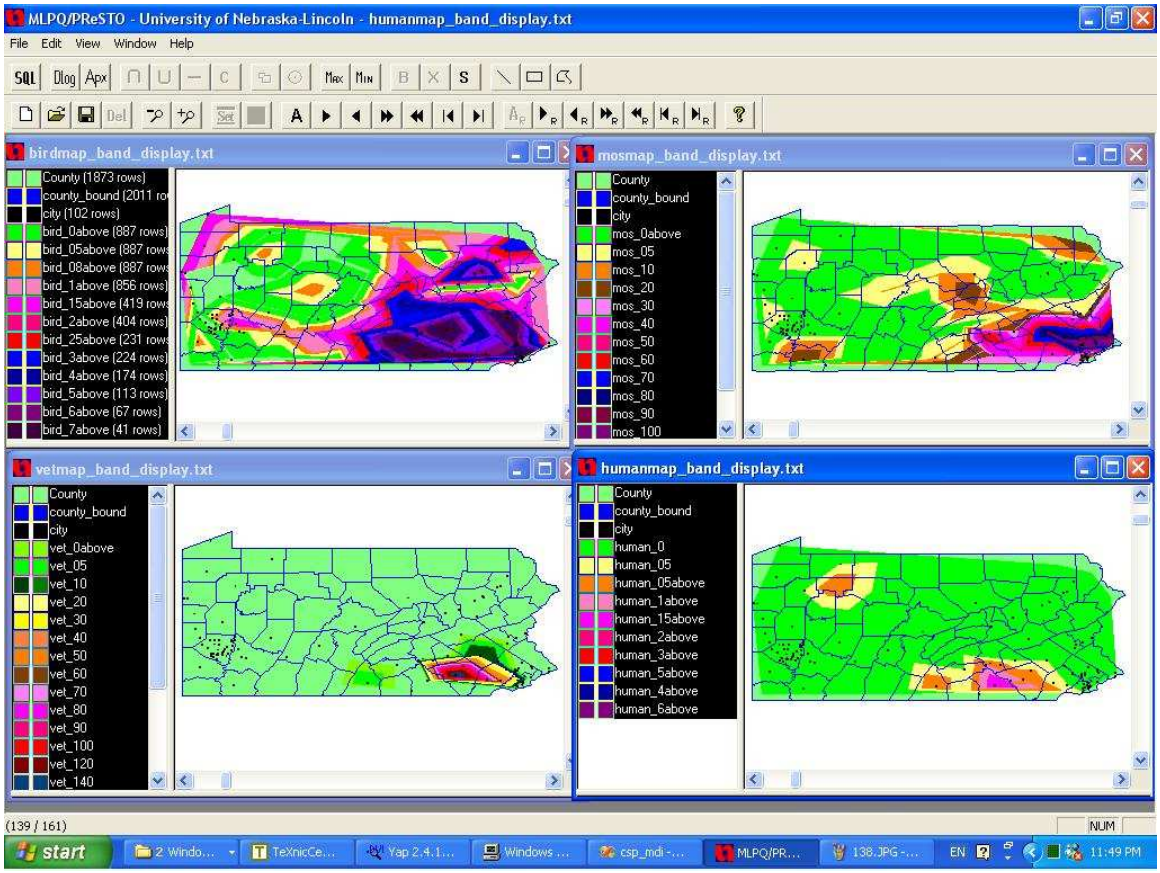

Figure 5.5: The distribution of West Nile Virus infections in week 139.

Second, based on our methodology, similar tracking and reasoning systems can be implemented for other epidemics besides the West Nile Virus epidemic.

Third, there are many other diseases that have recursive spread characteristics that are similar to those of epidemics. For example, some inherited diseases can be naturally defined by recursive definitions. The system can be also used to track and reason about many of those diseases too.

Finally, the recursive definition seems also natural for many other problems in medicine. For example, the current status of a chronic patient is closely related to his/her previous status. Our methods can be adapted to those situations too.

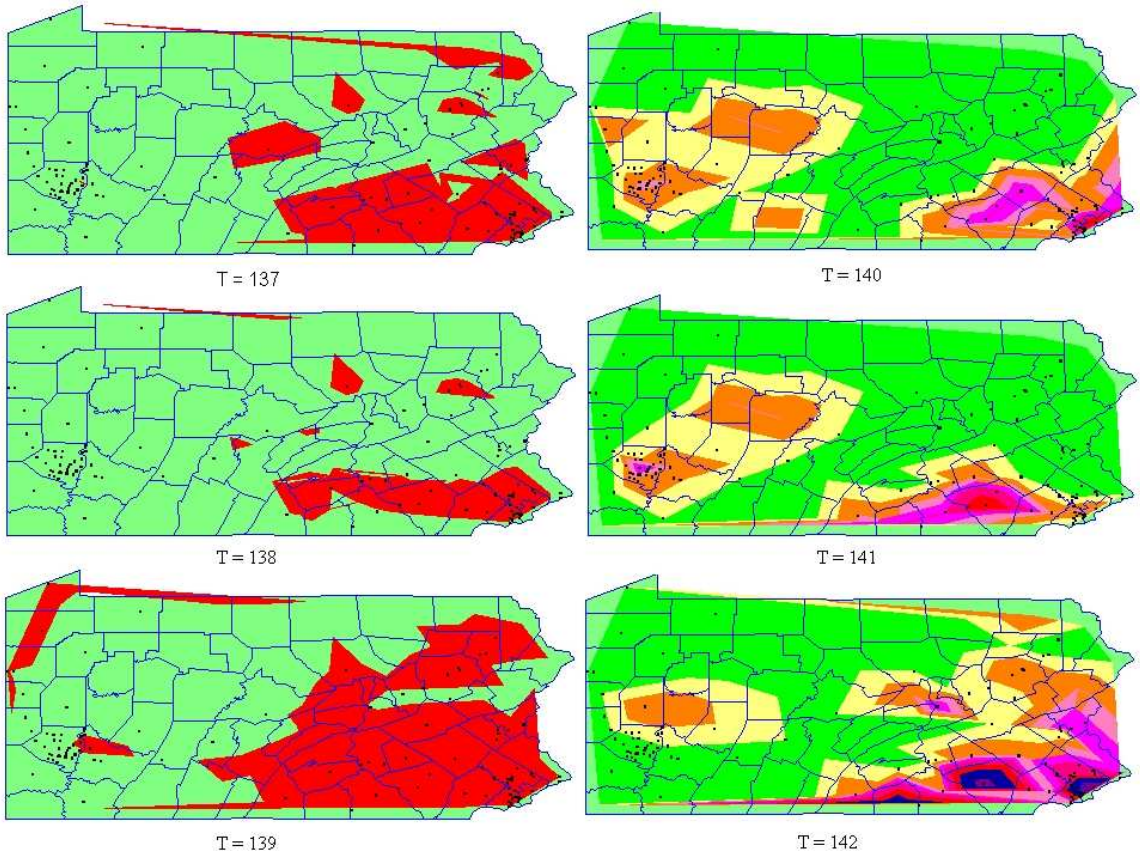

Figure 5.6: Left side: The predicted high-risk areas (dark color areas) based on the infected wild bird data. Right side: The actual distribution of human infections (darker color areas mean more infections) three weeks after the wild bird data to its left.

5.4 Discussion

Although the WeNiVIS system is implemented and tested on the WNV data of Pennsylvania, there are several general issues that need to be mentioned.

First, in the spatiotemporal interpolation process, we made four interpolation assumptions in Section 5.1.2 similar to Theophilides et al. (Theophilides et al. 2003). However, we have to point out the sensitivity of these assumptions to the actual data being interpreted. Theophilides et al. (Theophilides et al. 2003) applied their method within New York City, where each district has a high density. Since New York City

is heavily interconnected with a huge commuter population, assumption (2) may not hold.

We apply our interpolation over the whole state of Pennsylvania with a high variation of population density with the Philadelphia and the Pittsburgh areas having higher and the middle areas lower population densities. In this case, assumption (2) may be more reasonable, but assumption (4) may be less reasonable. Indeed, it is more reasonable to assume that in areas with a higher population density, a dead bird has a higher "opportunity" or chance of being reported than a dead bird in a low density area. In other words, death is unfair.

However, in this paper our intent is not to make a medically valid interpolation but to illustrate some methodologies and leave it to the medical experts to test the methodology with empirical data and decide whether it is applicable to the particular epidemics that they study.

Second, we only consider the relative size of each kind of infected animal in assigning the effectiveness weight for it in Example 3.3.1. Alternatively, the effectiveness weight for a type of animal may be based on how close it generally lives to human beings. Intuitively, the closer an animal lives to human beings, the more likely it is to infect them.

Third, for spatiotemporal applications, recursive queries are not expressible using the basic query languages of GIS systems. Some relational databases and knowledge-base systems provide recursive queries, but they do not provide spatiotemporal data representation. Hence the visualization of recursively defined concepts cannot be easily handled by these systems. They would usually require some special functions to be written in a programming language like C or C++ and added to a library. In contrast, our system only uses standard SQL queries to solve the problem. Therefore, the program is a simple, declarative, and high-level query that is easy to maintain. This feature is important, because the requirement of visualizing recursively defined

concepts on spatiotemporal data is frequent enough to need a general and simple solution method.

Chapter 6

Design Methods for Spatiotemporal Databases

conclusion The previous chapters focused on particular aspects of constraint and spatiotemporal database systems. This chapter considers three more design issue. First, it describes the general problem of designing an entire software system. Second, it applies the general methods to the design of the MLPQ constraint and spatiotemporal database system. Third, it considers sensor networks as the distributed data entry for spatiotemporal database applications.

The third issue is motivated by the fact that geographic information systems and spatiotemporal database systems are often combined with sensor networks in many applications. For example, in weather monitoring and forecasting it is essential to collect continuously information from distributed sensors measuring such variables as temperature, precipitation, wind direction, wind speed, and humidity.

Section 6.1 describes *Software Stability Modeling (SSM)*, which is a recent software design method with the goal of minimizing the software deterioration caused by the changes that result from the evolution of business requirements. The sample application of open-pit mining is based on (Fayad & Wu 2002). Further, the argument

that Software Stability Modeling is helpful for model-based software reuse is based on (Wu, Fayad & Nabavi 2002).

Section 6.2 considers *Design Patterns*, which provide another software design method with the goal of easily reusing the design made for one problem to a similar other problem. This section, based on (Wu, Mahdy & Fayad 2002, Wu et al. 2003), criticizes the current use of design patterns as being hard to follow in actual applications and suggests improving the design pattern methods by combining them with Software Stability Modeling.

Section 6.3, based on (Revesz & Wu 2003, Wu 2003), applies Software Stability Modeling in designing, maintaining and improving with features such as remote accessibility the MLPQ constraint database system.

Section 6.4, based on (Hamza et al. 2005, Hamza & Wu 2004), introduces a new routing algorithm for distributed sensor networks. The new routing algorithm reduces the total energy used for routing the information from the sensors to the central database store.

6.1 Software Modeling

Unlike the products of other engineering fields, there is no physical deterioration to cause software to fail. Most software defects and deterioration are caused by changes in software (Fayad & Altman 2001). Generally, these changes cannot be avoided. Such changes are often the result of the natural evolution of business processes and changes in the underlying requirements of software systems to meet these evolving needs. To achieve software stability is to balance the seemingly contradictory goals of stability over the software life cycle with the need for adaptability and extensibility.

To accomplish this balancing act, we hypothesize that certain refinements can be applied to conventional OO analysis and design techniques. However, such refine-

ments must not overly complicate our conventional approach. In software, we view simplicity and elegance as synonymous. Simplicity is considered an important characteristic of a good model. To demonstrate the simplicity and elegance of the Software Stability Model (SSM), we apply it to a study of open-pit mining (The detailed description of this problem can be found at ww.cse.unl.edu/ fayad/SoftwareStability/OOPSLA01-DesignFest-Final1.doc).

6.1.1 The Study

In an open-pit mine, mechanical shovels load material from depots into dump trucks. These trucks then transport material to various destinations. Ore is transported to mineral processing facilities to be processed and prepared for market. Waste is delivered to dumps.

A scheduling or dispatching system is required to assign the empty trucks to their proper destinations. This system checks the status of each truck and shovel. As soon as a truck is free, it is assigned to a shovel that will be free when the truck arrives. If all the shovels are busy, the dispatcher system selects the shovel with minimum wait time.

A conventional model for this problem is given in Figure 6.1. The model illustrates how the transport system works. Ore is placed into the ore extraction openings, and is then transported to a mineral processing facility. Similarly, waste is placed into waste removal openings and is then transported to waste dumps. Dump trucks are assigned to these openings according to a schedule. Shovels load ore and waste into these trucks.

The project's business objective is to maximize the mines profitability and satisfy production quotas. The project will accomplish this by optimizing the equipment, personnel, and the overall efficiency of the mining operation.

We can readily imagine changes in production techniques or mining conditions

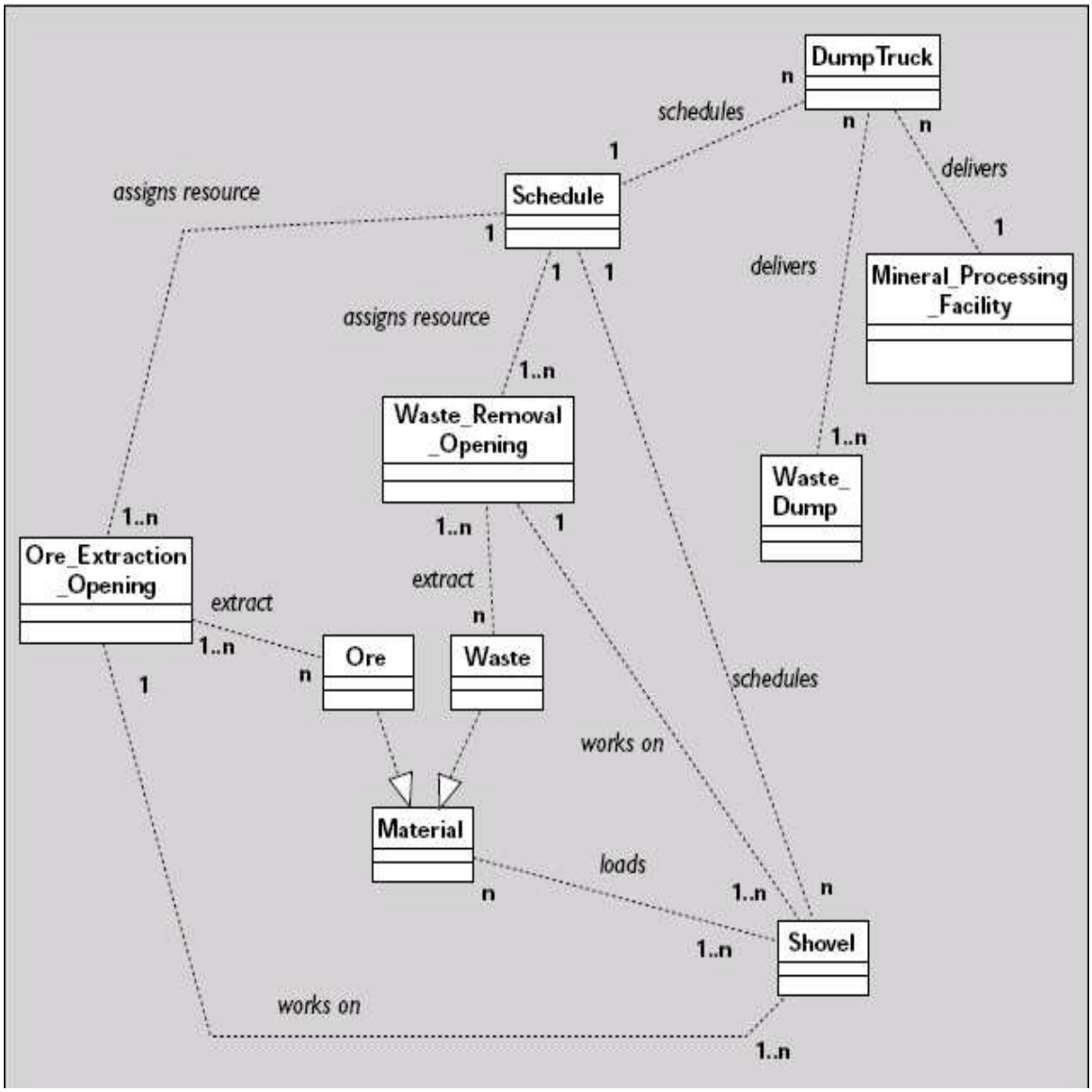

Figure 6.1: Typical class model for open-pit mining.

that will require changes to this software model. Perhaps conveyor belts are used as the main transportation methods instead of trucks. In this method, loaders carry materials from the depot and dump it into feeders. The feeder feeds a crusher at a steady rate and the crusher reduces the size of rocks to a proper size for transporting by conveyor belt. The whole system must be consistent in terms of capacity of materials flow. The loader must handle the required volume of material over time (for example, tons/hour.) The feeder must feed the crusher at the same rate, and the crusher must deliver the same amount of material of the proper size to the conveyor. Conveyors, in turn, must have the proper width and speed to transport the material. If any element of this system fails to maintain the required flow rate, material congestion occurs, and the whole system fails to achieve its goal of transporting the designated quota of material to the destination.

With the introduction of this new technique, the conveyer belt, the old model fails to satisfy the business requirements. We must modify it. The new class diagram might be generated as in Figure 6.2. We note that the new model does not bear a close resemblance to our original model, nor would it satisfy the requirements of our first open-pit mine.

Other locations may use a pipeline as the transportation mode. With this pipeline technique, material is loaded into feeders by loaders. Each feeder passes the material to a crusher and then on to a ball mill. Ball mills reduce the size of materials and prepare them for transportation through pipelines. The milled material is poured into mixers, where water is added to make slurry. The pipeline then carries the material to the destination.

With conventional models, the tendency is to remodel (as in Figure 6.3). Again, the result is a substantial change.

As these examples illustrate, conventional OO modeling is subject to instability. Many changes to the business process most likely to lead to changes in the previous

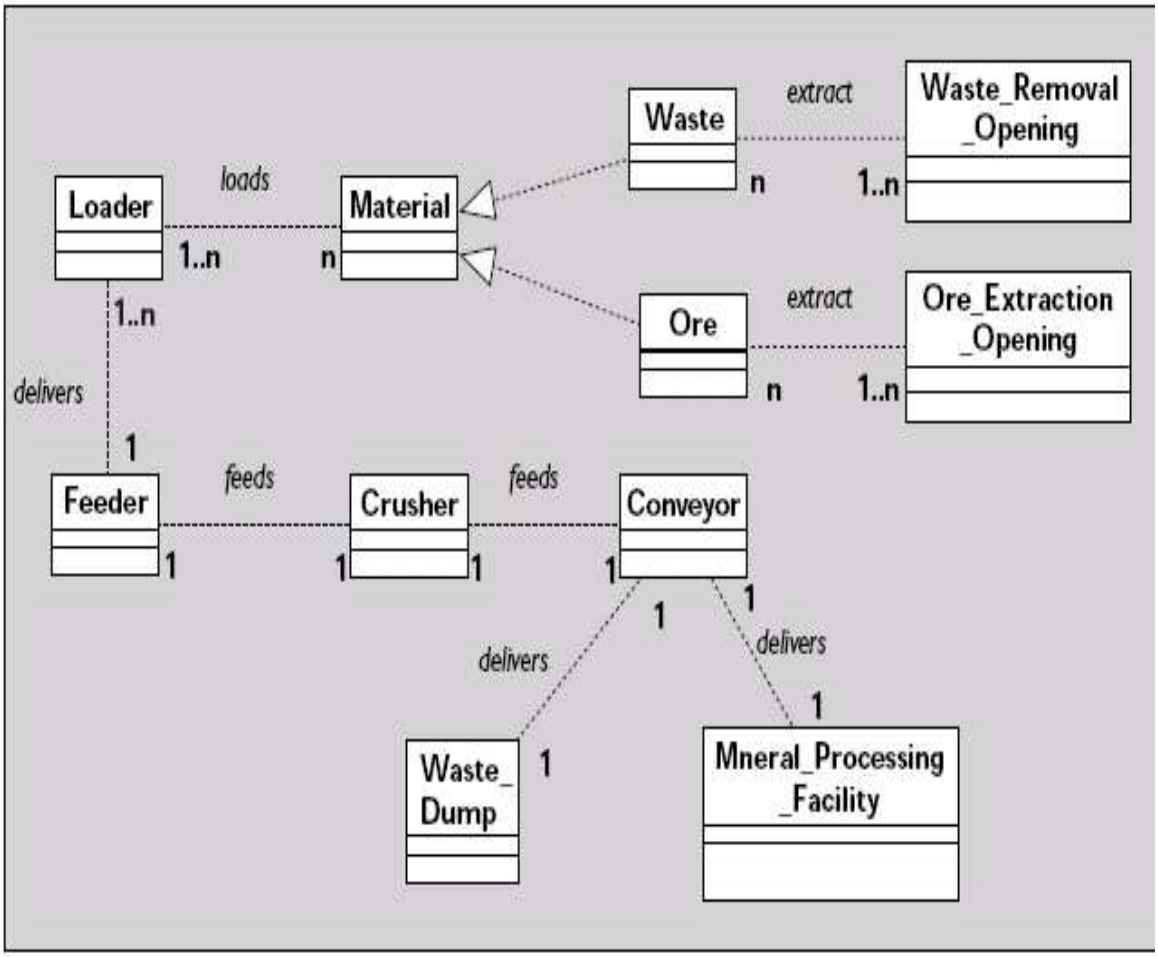

Figure 6.2: Class diagram of conventional model for conveyor belt transportation.

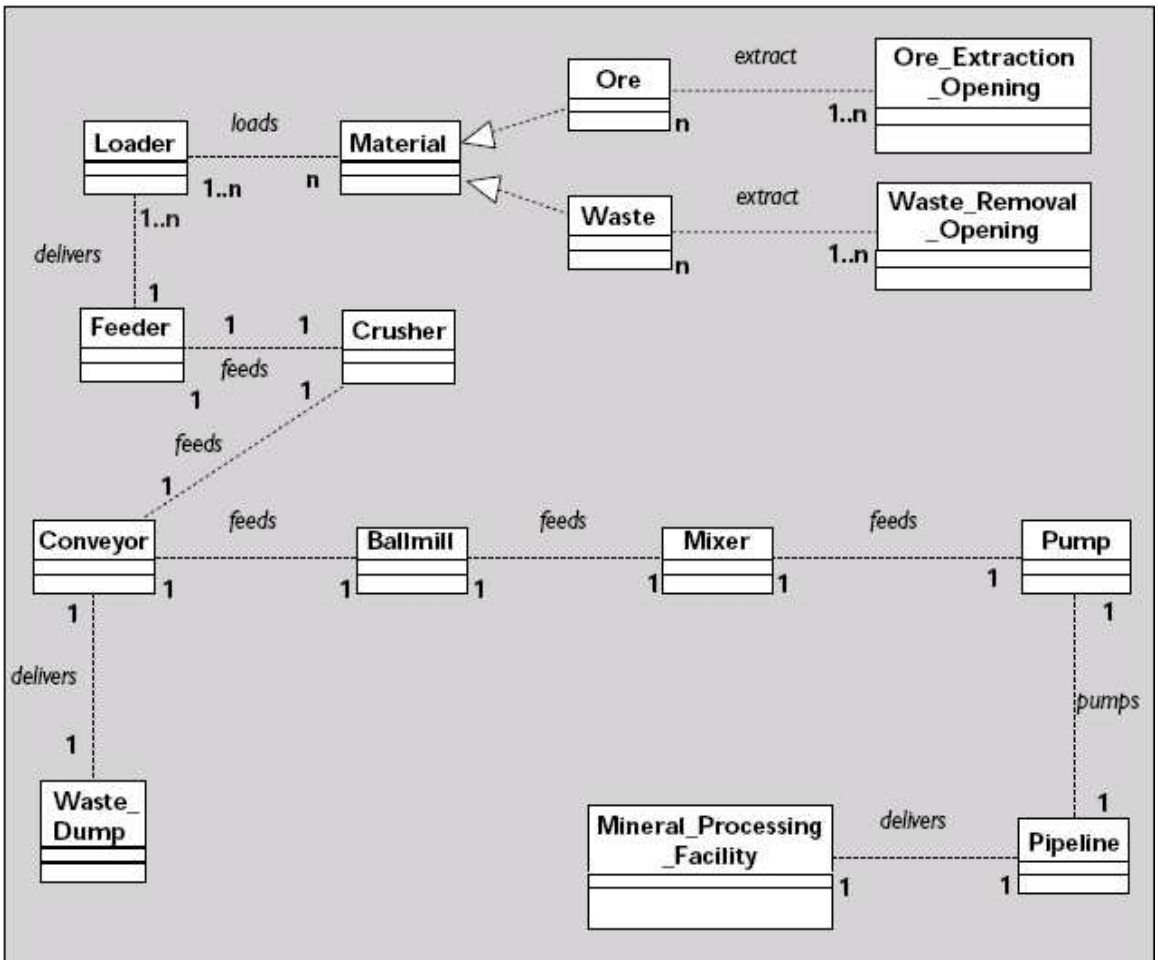

Figure 6.3: Class diagram of conventional model for pipeline transportation.

models, prompting a reengineering process. If the previous work is reused, the tendency is to attach new objects to the existing model similar to the way that barnacles adhere to the hull of a ship.

6.1.2 The Stability Approach

With the stability approach (see Figure 6.4), we first define the purpose of the system and delineate why the system is needed.

Although this aspect of analysis is not unique to the SSM, the model formalizes the analysis and focuses the analysis activities around the concept of Enduring Business Themes (EBTs). EBTs are the core abstractions of a problem domain. These themes are unlikely to change over time. For example, one important theme of the open-pit mining problem is *efficiency*. The concept of efficiency clearly has a role as an EBT, because efficiency is part of the business objective for building the system. When we derive a schedule for trucks, we are actually trying to make the system work more efficiently, hence, more profitably.

Another important theme is *concurrency*. Consider the interaction between different components of the transport system; they are assigned to work together when they are available. Dump trucks are assigned to shovels when they are available, and they only work together when shovels are available at the same time. In other words, they have concurrency. This concept is also for material flow through the conveyor belt and pipeline systems.

All of the components in the system have to pass a specific amount of material to the next component within a specified amount of time. All of these material flows have to be equal in a system and all the components of the system must be available and working at the same time to be concurrent. Without this concurrency we cannot imagine a transport system; the result is congestion and failure. Therefore, concurrency is a very important measurement and one of the core purposes of our

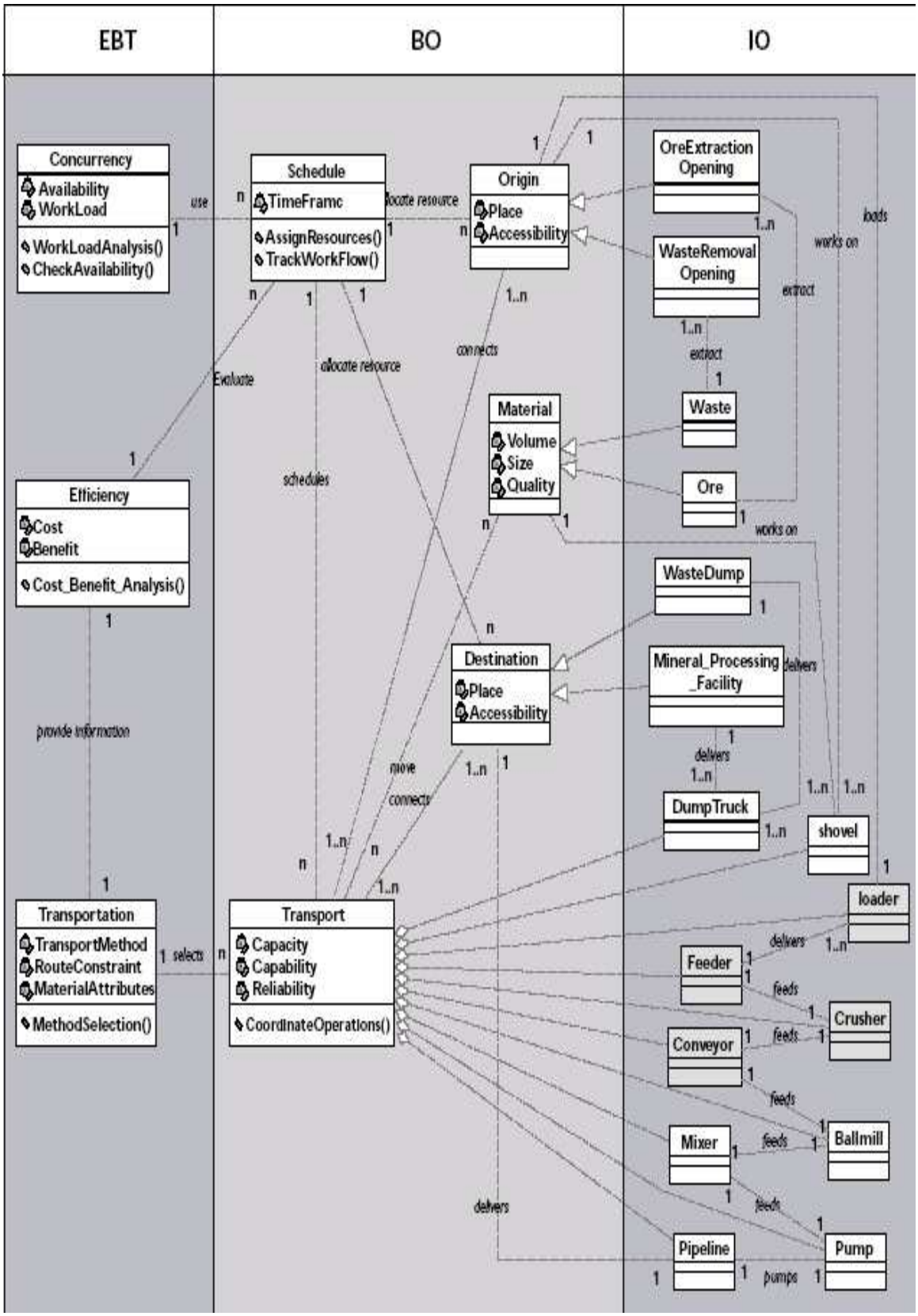

Figure 6.4: Class diagram of stability model for transport system.

scheduling system.

After identifying the EBTs associated with our problem, we can identify and associate those business objects (BOs) that provide the necessary abstractions of the processes underscoring the business operations (such as origin, destination, and transport). The third step of the SSM is to define those Industrial Objects (IOs) that refine (or instantiate) the various BOs. In this way, our model exhibits stability over changes. If we substitute feeders, loaders, crushers, and conveyors for shovels and dump trucks, the core model is unchanged. It is through changes to the peripheral objects that we achieve the balance between changes and our goal of stability. Despite the various modifications to support various instances of the problem, our EBTs and BOs remain stable. In addition, the model remains well organized over the course of many design iterations. Ultimately, we believe this approach has tremendous benefits for software teams. This study shows the SSM is elegant and extensible. The layered approach of the SSM provides an important technique for decomposing a problem space. By adding the focus of identifying EBTs and BOs, the SSM forces software engineers to clearly rationalize the role of each object described within a problem space.

Due to their higher level of formalism, we believe SSMs are more measurable and testable than conventional models. EBTs describe the goal of the system (Fayad 2002, Fayad & Altman 2001). BOs are the plan to reach those goals and IOs are the objects that perform the work to realize the plan (Fayad 2002, Fayad & Altman 2001). The sequences and logical relationships among those objects are clear. By tracing the objects top-down, we can test and verify the use of every class in the diagram systematically. We will address the issues of measurability and testability in future columns.

Although the stability approach requires a high level of discipline in analysis and a high level of rigor, this is inevitable in any approach that tries to reduce maintenance

costs. The three-layer method for identifying EBTs, BOs, and IOs is central to the Stability Approach. System designers can provide a precise, well-structured, adaptable, and maintainable solution for a problem only if they understand why the objects are necessary to the problem.

6.2 Design Patterns

6.2.1 Context

The basic message of this paper is that the implementation of design patterns leaves the programmer with no trace of what patterns were applied and what design decisions were made. Thus, when changes have to be made, the entire design has to be almost entirely reconstructed.

We will explain this with an analogy to driving rules. The pattern instances correspond to actual driving and the problem context can be thought of as different driving situations. The rules are always constant although the actual driving may change under different circumstances. To drive safely and efficiently, people must learn the rules first and then apply them according to the different circumstances. Every time they drive, they must keep the rules in mind in order to adapt to different situations. That is the key to stable driving. Expecting these models to be stable under changes is similar to applying past driving sequence actions to different times and locations.

6.2.2 Problems

"A pattern is a plan, rather than a specific implementation (Coad, North & Mayfield 1995)." They are "descriptions of communicating objects and classes that are customized to solve a general design problem in a particular context (Gamma, Helm,

Johnson & Vlissides 1995)." Consequently, design patterns must be stable, abstract, and common. This makes them unsuitable for detailed implementation on specific problems. Design patterns are usually used only during the initial, conceptual design. They guide the modeling process and detailed design process. By instantiation, the original abstract objects of design patterns are replaced with concrete objects representing instances of a specific domain; plus other objects. Those instances are the actual guide for programming. Figure 6.5 shows the mechanism of the adaptation process of a design pattern.

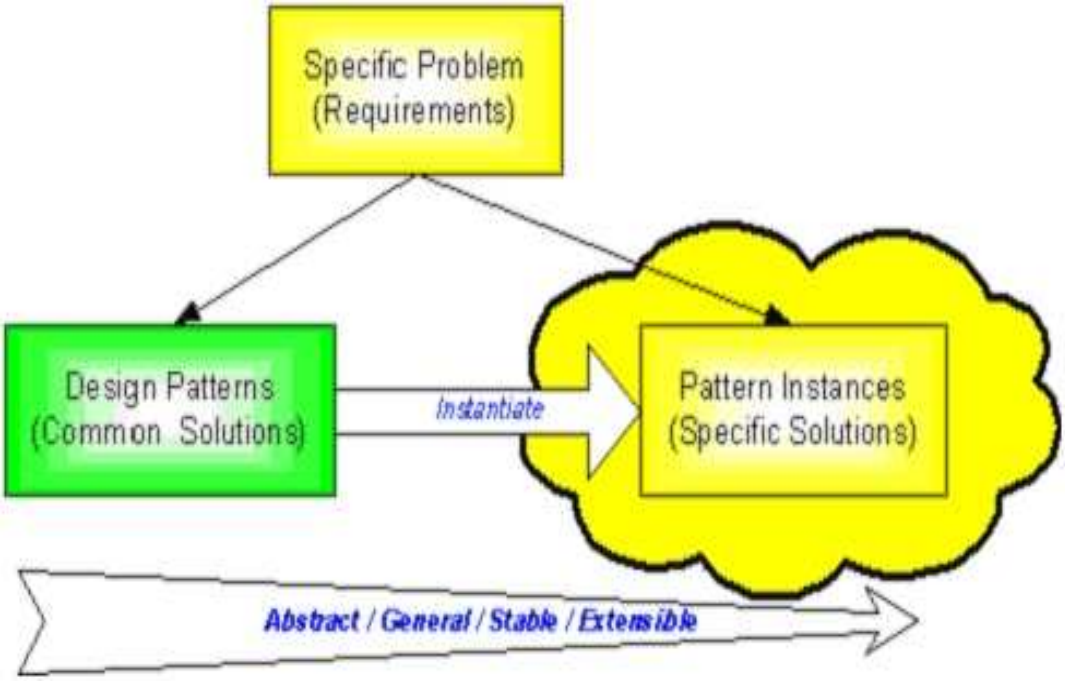

Figure 6.5: Current approach of implementing design patterns.

In current approaches of implementing design patterns, design patterns and their instances are separate models: abstract and concrete domains. After instantiation, the instances are less abstract, less stable and difficult to extend, compared to their design patterns. This is because a pattern gains its quality factors from being a

recurrent design issue and solution in multiple contexts/domains. Design patterns are not traceable from the resulting implementation model. Thus, the solution loses an important quality of patterns: stability in the face of changes.

The instantiation process can be used to solve this discrepancy. But the description of this process is not traceable from the resulting implementation model too. This means a specific piece of code (implementation model) implementing a given solution, lacks the ability to trace back to its blueprint design pattern. This disallows the extension of the model for future changes. Thus, the theme of this paper is that the pattern itself is lost in the implementation. Is there a way to convey the pattern characteristics to its instances to achieve both usability and stability in the resulting models?

6.2.3 Solution

One solution to this problem is proposed in (Soukup 2001). It suggests using Pattern Classes to improve the readability and maintainability of final code. We suggests the use of stable models and Enduring Business Themes (Fayad & Altman 2001) to help restore what would otherwise be lost in the implementation. The model loses its generality and abstraction after instantiation, causing it to be weak in adaptability and extensibility. When later developers want to extend the software, they cannot directly reuse this model because the implementation of pattern does not allow tracing back to the abstract design pattern. They have to reconstruct the whole model from the original design patterns.

The Software Stability approach (Fayad & Altman 2001) has the potential to build such models. “A Software Stability Model (SSM) can be triply partitioned into levels: Enduring Business Themes (EBTs), Business Objects (BOs), and Industrial Objects (IOs). EBTs represent intangible objects that remain stable internally and externally. BOs are objects that are internally adaptable but externally stable, and

IOs are the external interface of the system. In addition to the conceptual differences between EBTs and BOs, a BO can be distinguished from an EBT by tangibility. While EBTs are completely intangible concepts, BOs are partially tangible. These artifacts develop a hierarchal order for the system objects, from totally stable at the EBTs level to unstable at the IOs level, through adaptable though stable at the BOs level. The stable objects of the system are those do not change with time (Mahdy, Fayad, Hamza & Tugnawat 2002)." From an abstractness aspect, the EBTs are completely abstract, the BOs are mostly abstract, and the IOs are not abstract. Hence, the EBTs and BOs are common among applications with similar core, while the IOs are those object differentiate an application from another. Figure 6.6 shows the SSM structure.

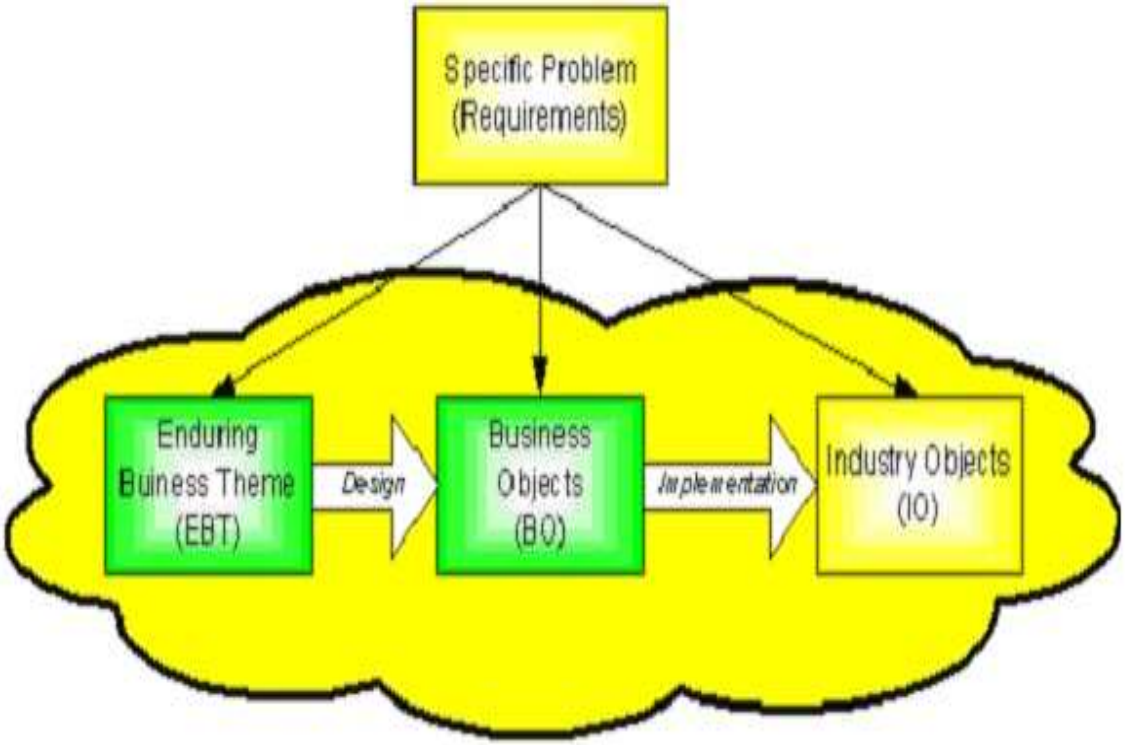

Figure 6.6: Stability Model Structure.

The EBTs and BOs are abstract like design patterns, but they do not disappear

in the implementation. They describe a common solution to the problem. The IOs are the same as the pattern instance. They are concrete, problem-specific and unstable. When we include the original design patterns in the final instance models and provide their collaborations, the model bears a striking resemblance to the SSM. By associating these two parts in the SSM instead of separating them as in the current Design patterns approach, we add a stable core to the resulting model. This keeps the model stable over changes.

Future developers can trace the ideas of the original model designers and extend the model safely, as the core remains stable. Combining the abstraction and generalization from the original design patterns (i.e. EBTs and BOs) and the specification of the instances (i.e. IOs), the SSM achieves stability and usability concurrently. It shoots two birds with one stone.

6.2.4 Case Study

There is no instantiation processes shown in Figure 6.8; the final result of model using the current design patterns implementation approach. The instances were deduced from the design patterns, but the design patterns were discarded in the final model. In Figure 6.7, which is taken from Figure 6-32 in (Coad et al. 1995), it is very difficult and uncertain to induct *SubsequentTransaction* from *Picklist*. Thus a designer cannot go back to the original designs to extend existing models. Each time that the designers want to extend the models, they must go all the way back to the original design patterns and reconstruct the whole model. For example, if a new object named *Planlist* is introduced in another situation, the information displayed in Figure 6.8, which is taken from Figure 6-33 in (Coad et al. 1995), is not sufficient to define the new *Planlist* object.

Figure 6.9 describes the same problem as Figure 6.8 using the Software Stability approach. Using this new model, we can easily change the IOs without worrying about

destroying the whole structure of the model. Suppose for example, the circumstances change and we need to introduce *Planlist* to satisfy a new requirement. Using the SSM we can easily instantiate *Planlist* from *SubsequentTransaction* as an IO to extend the model without modifying the whole structure. The EBTs and BOs remain constant during this process. This efficiently keeps the core structure and design ideas of this model unchanged over time.

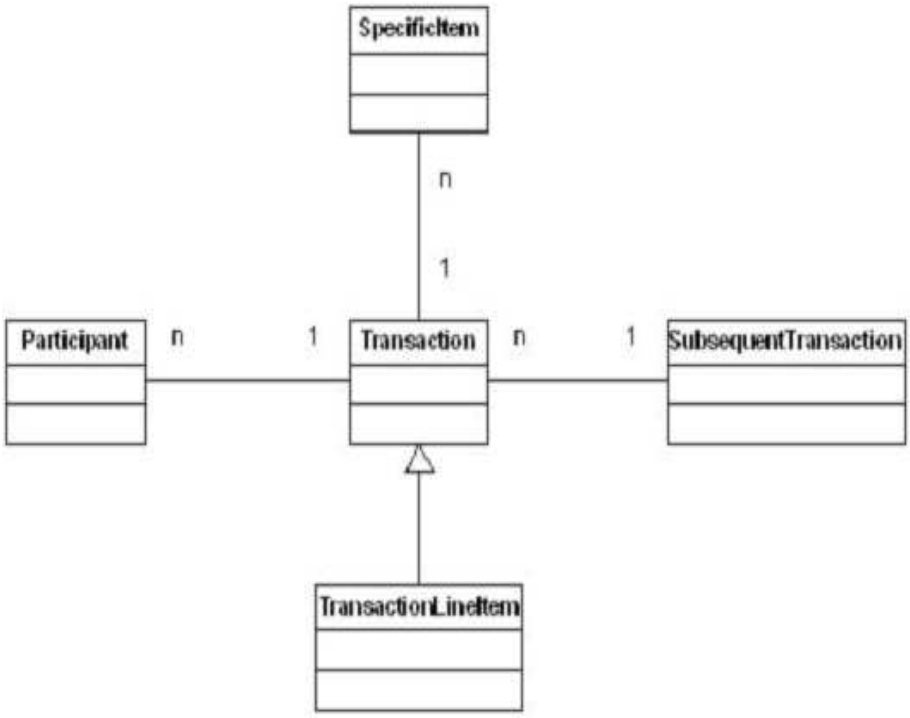

Figure 6.7: Design Patterns for Transaction.

6.3 Improvement and Maintenance of the MLPQ System

This section describes a design experience in the Constraint Database Management System (DBMS) development based on Stability Modeling Approach. It may be

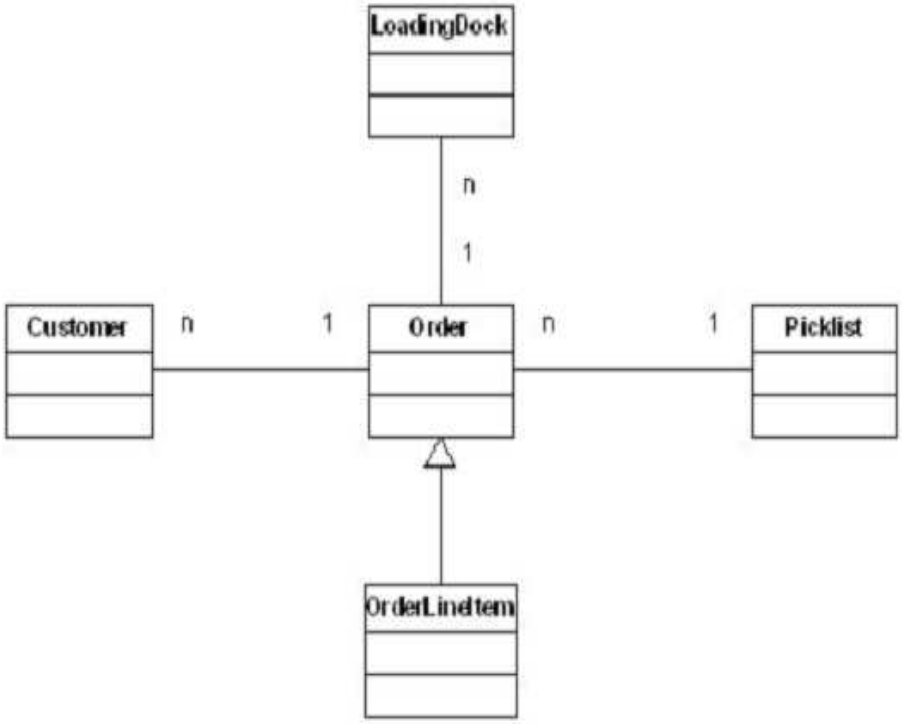

Figure 6.8: Instance of transaction patterns on orders).

helpful for the DBMS system designers and implementers, who are designing and implementing the DBMS modules and interfaces as a software package (Elmasri & Navathe 2003).

A DBMS is a software system designed to efficiently store, retrieve, manipulate, and query large amounts of data. We believe that the core purpose of any DBMS is to provide a tool for people to retrieve meaningful data quickly, accurately, and securely from related databases. To reach this goal, the DBMS has to provide a way for people and the computer to organize the databases so that queries can be executed efficiently and accurately. If people will not retrieve any information from a database, maintaining this database is useless. Therefore, maintaining and organizing a database is primarily a preparation for queries. Bearing this idea and the stable model designing approaches in (Fayad & Altman 2001) and (Fayad & Wu 2002), we can generate Effi-

cient Query, Efficient Modification, Data Consistency, and Security as the Enduring Business Themes (EBTs) in our stability model of constraint database management system. Among them, Data Consistency is critical in maintaining database and generating accurate result. Efficient Query means that the program can execute the user's query efficiently on a database and represent the result clearly. For this purpose, we generate Efficient Search and Efficient Representation as Business Objects (BOs). For Efficient Search, we need a Structured Data which represents a well-organized data format, and a Search Engine to execute the actual queries on the Structured Data. For Efficient Representation, a Query Language and a User Interface are necessary for people to efficiently express their willingness to the Search Engine and clearly display the result generated from the engine.

Until now, we only discussed the general concept of a DBMS design. The objects we have are all abstract and general concepts which can be applied in any database management system. They are stable designs on different kinds of DBMS, whether they are relational databases, constraint databases, or others. For our constraint database management system, we can easily instantiate the model shown in Figure 6.10. For example, the Structured Data now is instantiated as MLPQ and PReSTO which represent two major data formats in our MLPQ/PReSTO system (Revesz 2002). To support client/server communications using TCP/IP, we can just instantiate a new object Socket as an Industry Object (IO) from User Interfaces. By this object, people can easily access the databases in the server through network (Wu 2003). However, the whole structure and design are still constant over this improvement.

Currently, we only implemented Efficient Query part in our MLPQ/PReSTO system because we believe that it is the core of a DBMS, as we discussed above. The executable files and documents can be found at http://csce.unl.edu/ revesz/MLPQ/mlpq.htm. However, security, efficient modification, and data consistency concepts are also very

critical for a real DBMS. How to design and implement them in a Constraint Database system by stability modeling is an interesting future development work.

6.4 Wireless Sensor Networks

Wireless Sensor networks (WSNs) consist of a large number of small nodes networked via wireless links. Because of size limitations, a node can have only limited energy resources, storage capacity, and computational power. Spatiotemporal database applications, such as efficient weather information sharing, disaster aid and risk management, all may benefit from the ability to efficiently deploy a temporary wireless network.

A major concern in such networks is to efficiently route queries and replies between nodes. The limited resources call for efficient routing algorithms to minimize routing overhead and energy consumption.

In this paper, we propose an algorithm called EDIVER-Efficient Distance Vector Routing for routing in sensor networks. EDIVER is a flat routing (Kulik, Heinzelman & Balakrishman 2002) algorithm that adopts good features of two different routing algorithms: the Ad hoc On-demand Distance Vector (AODV) (Perkins & Royer 1999); an ad hoc routing algorithm, and the Rumor routing algorithm (Braginsky & Estrin 2002); a routing algorithm for sensor network. Our objective is to adopt the best features from these two algorithms suitable for sensor networks to develop an algorithm that can establish a one-to-one communication pattern while avoiding the high cost of existing routing algorithms that are used in Mobile Ad hoc Networks (MANETs).

AODV is a relatively mature routing algorithm and it has several features that make it a good candidate to be adapted for sensor networks (He, Stankovic, Lu & Abdelzaher 2003). Simulation results show that EDIVER reduces the average energy consumption, average end-to-end delay and normalized routing overhead compared

to AODV algorithm. However, packet delivery fraction for EDIVER is a little less than that of the AODV algorithm, although it is very close under some scenarios.

6.4.1 The EDIVER Algorithm

In EDIVER, every node in the network maintains two tables: a neighbor table; a table that contains the addresses of the neighbor nodes, and an event table; a table that contains routes to the different events detected in the network. EDIVER adapts the same processes as in AODV for both reverse and forward path setup (See (Braginsky & Estrin 2002)).

However, EDIVER use different path discovery process than that of AODV. The difference is that we eliminate the broadcasting of the request message. Therefore, when a node receives a route request packet, it checks its event table; if it finds the event node in its table it sends a RREP message (similar to AODV). If no entry in the event table exists, unlike in AODV, the node checks a new header field called forward_nb in the request packet that contains the address of the neighbor that should rebroadcast the RREQ. This header field is set by the node that previously forwarded the RREQ. If a node finds its address in this header field, it rebroadcasts the RREQ; otherwise it drops it. Before the specified node rebroadcasts the request, it randomly selects a neighbor other than the one it has received the request packet from, and updates the header filed with its address. The node also decrements the TTL of the request packet before it rebroadcast it. The sequence of events that occurs when a node receives a Request packet is illustrated in Figure 6.11.

Since AODV explores more options in finding a path by broadcasting the RREQ, it is more likely that AODV will establish a path faster than in EDIVER. To improve the path discovery chance, EDIVER adopts the concept of Event Agent packet and event table form Rumor algorithm.

The Event Agent packet contains two tables: an event table and a seen table. The

event table is used to maintain a list of events that the packet has seen while traveling through the network, whereas the seen table records the nodes that an Event Agent packet has visited so far.

When an event node detects an event, it initializes an Event Agent packet and records the event in that packet. The node selects randomly a neighbor from its neighbor table and sends the packet to it. The sequence of events for sending an Event Agent packet is shown in Figure 6.12. When a node receives an Even Agent packet, it first performs a synchronization process (Braginsky & Estrin 2002), where the Event Agent packet and the node both update their event tables with the best route to the known events in the network. Figure 6.13 shows the typical sequence of actions that a node performs when it receives the Event Agent packet. Upon receiving the Event Agent packet, the node first checks the packet's event table, if it sees a new event that it does not have, it adds the event to its event table. If it finds an event in the packet that is in its event table, it checks the hop count. If the node finds that the agent has better route to the event it has in its event table (i.e. a route with less number of hops), it updates its table with the new route.

After the synchronization process, the node will check the Event Agent packet TTL filed, and if it finds it zero it simply drops the packet, i.e. the agent effectively dies. Otherwise, the node checks the packet header, which is similar to that used in the request message above. If this address in the header does not match the node's address; the node drops the Event packet. Otherwise, the node first copies the event table it has to the Event Agent packet (since this table has the current routing information to all events seen so far by both the node and the packet), and then it picks a neighbor from its neighbor table randomly. It compares this neighbor's address with those in the seen table in the Event Agent packet. If it does not match any, the node updates the packet header with this address. In case if all the neighbors have been seen before by the Event packet, the node picks any of its neighbors randomly

and then broadcast the Event Agent packet.

Although in this paper, following the approach in Rumor algorithm (Braginsky & Estrin 2002), we assume that each sensor node has an ID; however, in real-life sensor networks it may not be feasible or desirable to assign node Id's for certain applications.

Finally, we point out some of the differences between Rumor and EDIVER algorithms. First, Rumor algorithm provides a way to route a request to an event node; however, it does not provide a mechanism for reverse and forward path setup (the path that will be used to send the data from the event node to the requester). In contrast, EDIVER establishes a complete connection between the requester node and the event node. Second, in Rumor algorithm, when a node encounters a request packet more than once, it forwards the packet randomly, but not to the same node that it had forwarded to before. On the other hand, in EDIVER the node drops such packets to minimize overhead and reduce energy consumption. We observed that the gain in performance by forwarding packets more than once does not always justify the corresponding increase in energy consumption, and hence we chose to remove this feature. Third, in Rumor algorithm a node may or may not inject an Event Agent packet upon detecting an event in the network. In contrast, in EDIVER when a node detects an event, it always injects an Event packet. By doing this, we hope to increase the packet delivery and to compensate for the negative effect of eliminating the multiple request forward as discussed above, and the request broadcasting in AODV algorithm.

6.4.2 Simulations

We implemented the EDIVER in ns-2 simulator (Project 2003) as follows.

- Simulation Parameters

We simulated an ad hoc network of 200 stationary nodes scattered randomly across a 500m500m area. Each simulation runs for 100 seconds. The traffic was assumed to be CBR.

- Methodology and Metrics

  To evaluate the performance of EDIVER algorithm we fixed the number of events in the network to 12 events, and varied the number of requests in each simulation run. The number of requests used was: 13, 15, 20, 29, 36, and 50. We also simulated a scenario of 114 events and 135 requests, the results followed the same trends to those presented here, and hence were eliminated. We used four metrics to compare the performance of the three algorithms:

  - Average Energy Level: Measures the average energy per node at different instances of time.
  - Normalized routing load (Gupta 2003): Measures the total number of routing packets transmitted per data packet arriving at the destination.
  - Average end-to-end delay of data packet (Gupta 2003): Measures the total time it takes for a data packet to arrive from the source to the destination. It takes into account delays such as those caused by queuing, propagation delay.
  - Packet delivery fraction (Gupta 2003): Gives the ratio of the data packets that were successfully delivered to the destinations to those generated by sources.

  As pointed out earlier, Rumor algorithm does not provide a mechanism to perform an end-to-end routing. That means the algorithm does not establish a connection between two nodes. Therefore, the above evaluation metrics do not

apply, and thus, comparing the performance of our algorithm to that of Rumor is not meaningful.

6.4.3 Main Results

In this Section, we present the simulation of the AODV and EDIVER algorithms. Figure 6.14 gives the normalized routing load. As shown, EDIVER has smaller normalized routing load compared to AODV. This result is intuitive as EDIVER does not flood the route request packet in the path discovery phase as AODV does. We observe that for AODV, as the number of quires increases, the routing overhead may also increase, this is due to the increasing overhead in path discovery phase that is associated with every request. However, when the number of quires was 36, the normalized routing overhead was less than that for 29 quires, which may be due to the fact that some requests may establish a request to a node that has been recently accessed, and hence, the routing information in the intermediate routers were still valid. For EDIVER, the increase in overhead is proportional to the increase in the number of requests. This increase is due to the fact that EDIVER does not flood the request, and hence, the path discovery process may expand larger number of nodes before the path can be found. Notice that even thought the path discovery may be longer in EDIVER compared to AODV; EDIVER does not flood the request packet and thus the routing overhead does not increase vastly.

Figure 6.15 gives the average end-to-end delay values for different number of requests. We observe that EDIVER has lower average end-to-end delay for all cases. We do not have enough data to explain the alternating behavior of AODV. In AODV, the increase in number of requests may benefit from the routing information maintained in the intermediate nodes.

Figure 6.16 depicts the result of packet delivery fraction. The best performance is achieved by AODV. However, we find that EDIVER performance is almost as good as

that of AODV when number of queries is between 15 and 36, but outside this window, the EDIVER performance slightly drops compared to that of AODV. These results suggest that there exists a window within which EDIVER performance is close to AODV. We believe that EDIVER will perform much better in networks where some redundancy in the number of sensor nodes exists.

We plot the change of the energy level per node observed over a simulation time of 100 sec. Figure 6.17 depicts the average energy level per node observed over the simulation time of 100 sec. Eliminating the flood of the request message in EDIVER has improved the energy consumption compared to AODV.

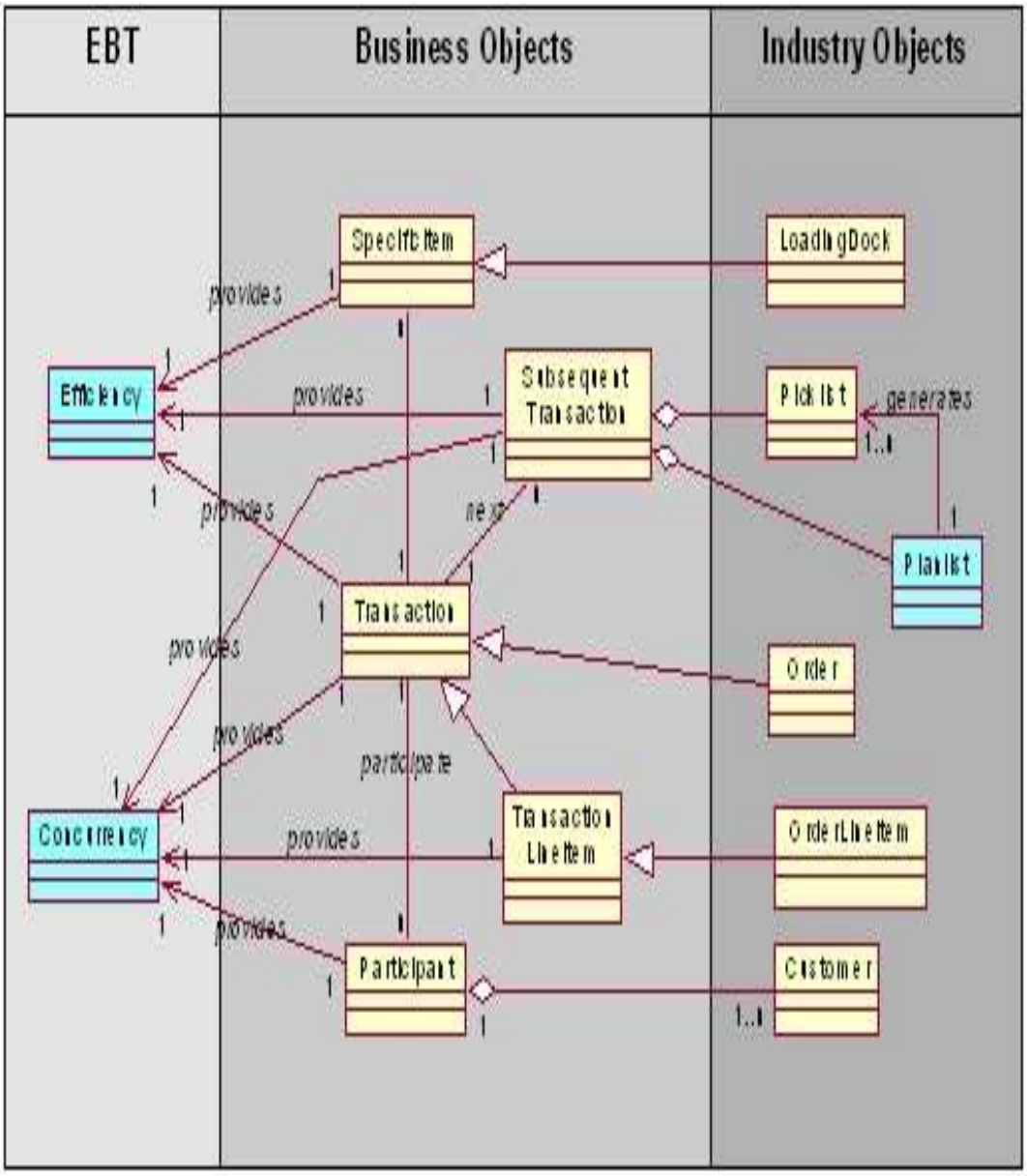

Figure 6.9: Stability model for Transaction Design Patterns on order instance.

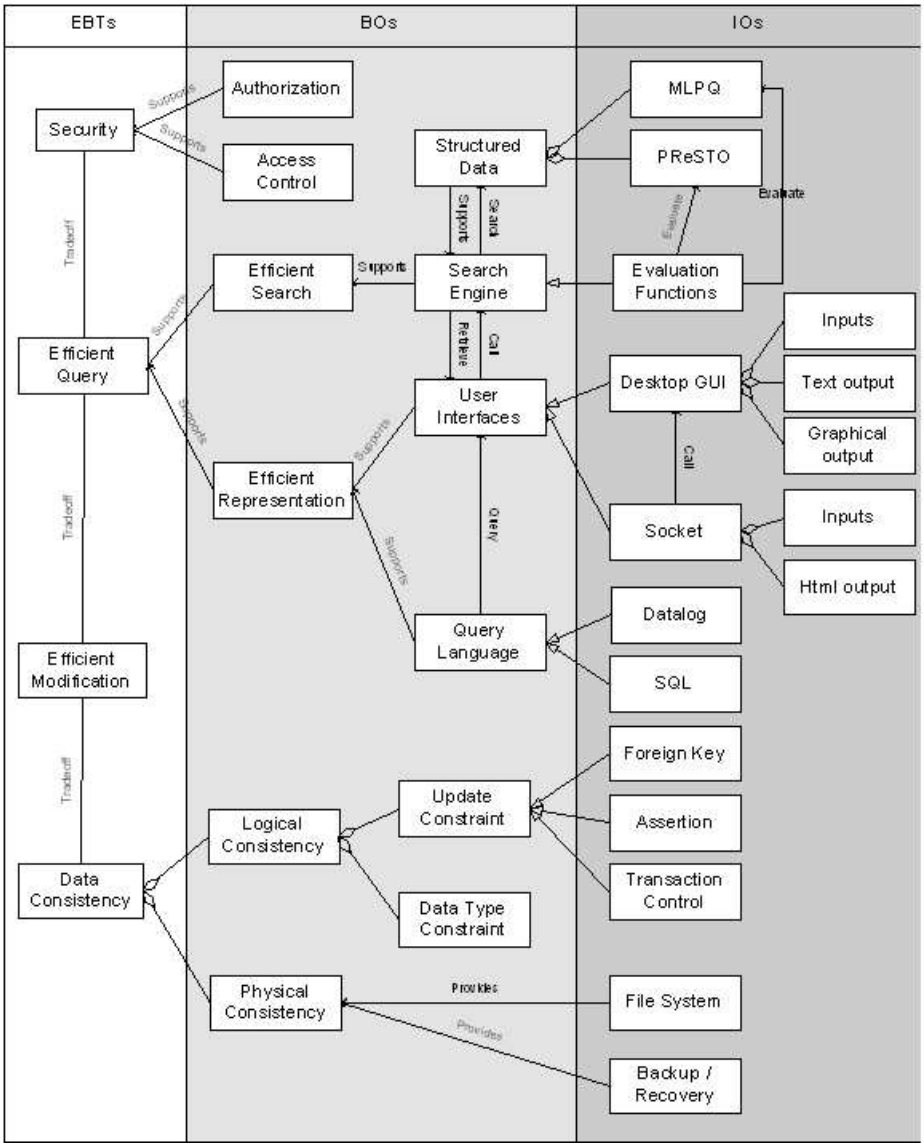

Figure 6.10: A Stability Model for Constraint Database Management Systems.

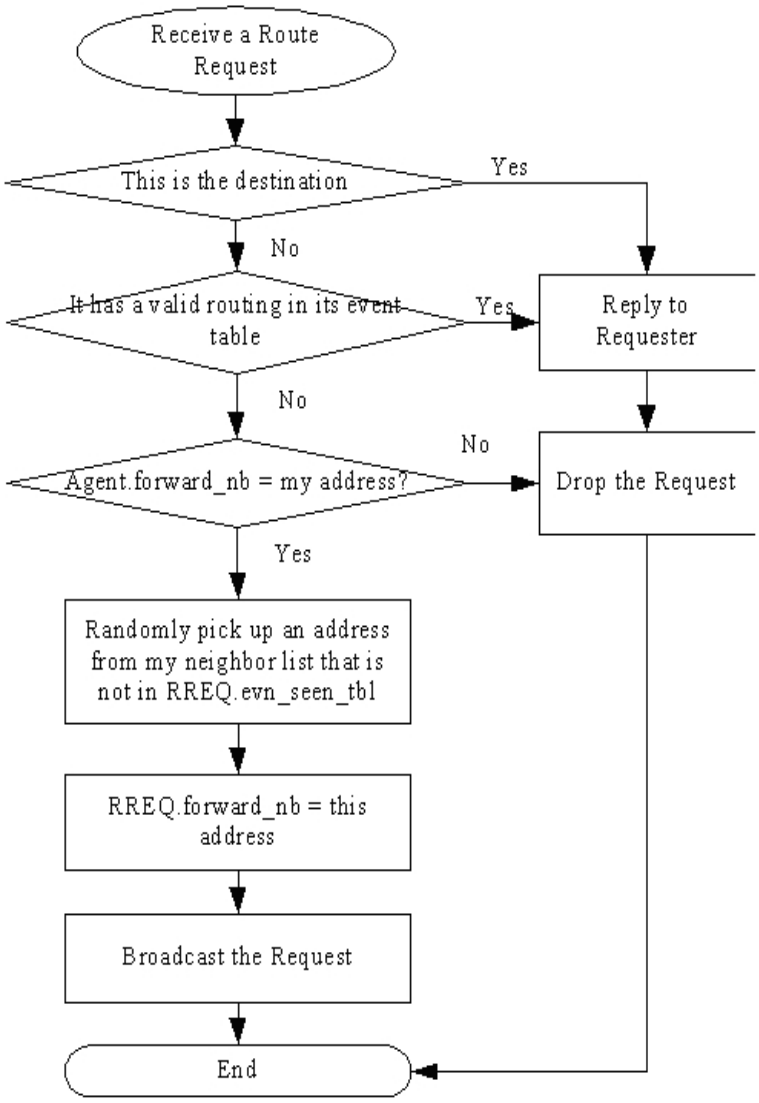

Figure 6.11: The sequence of events when a node receives a Request packet.

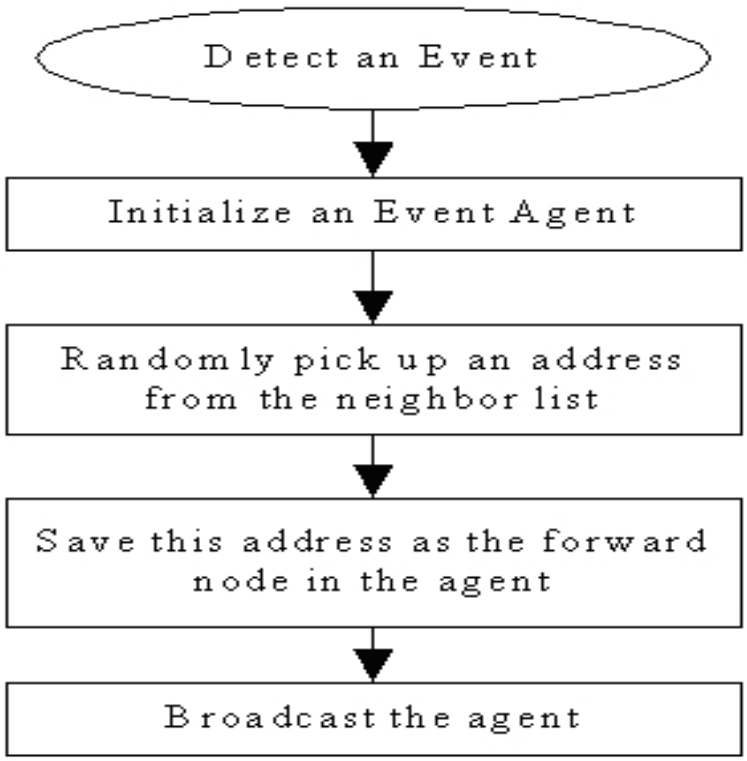

Figure 6.12: The sequence of events when a node detects an event.

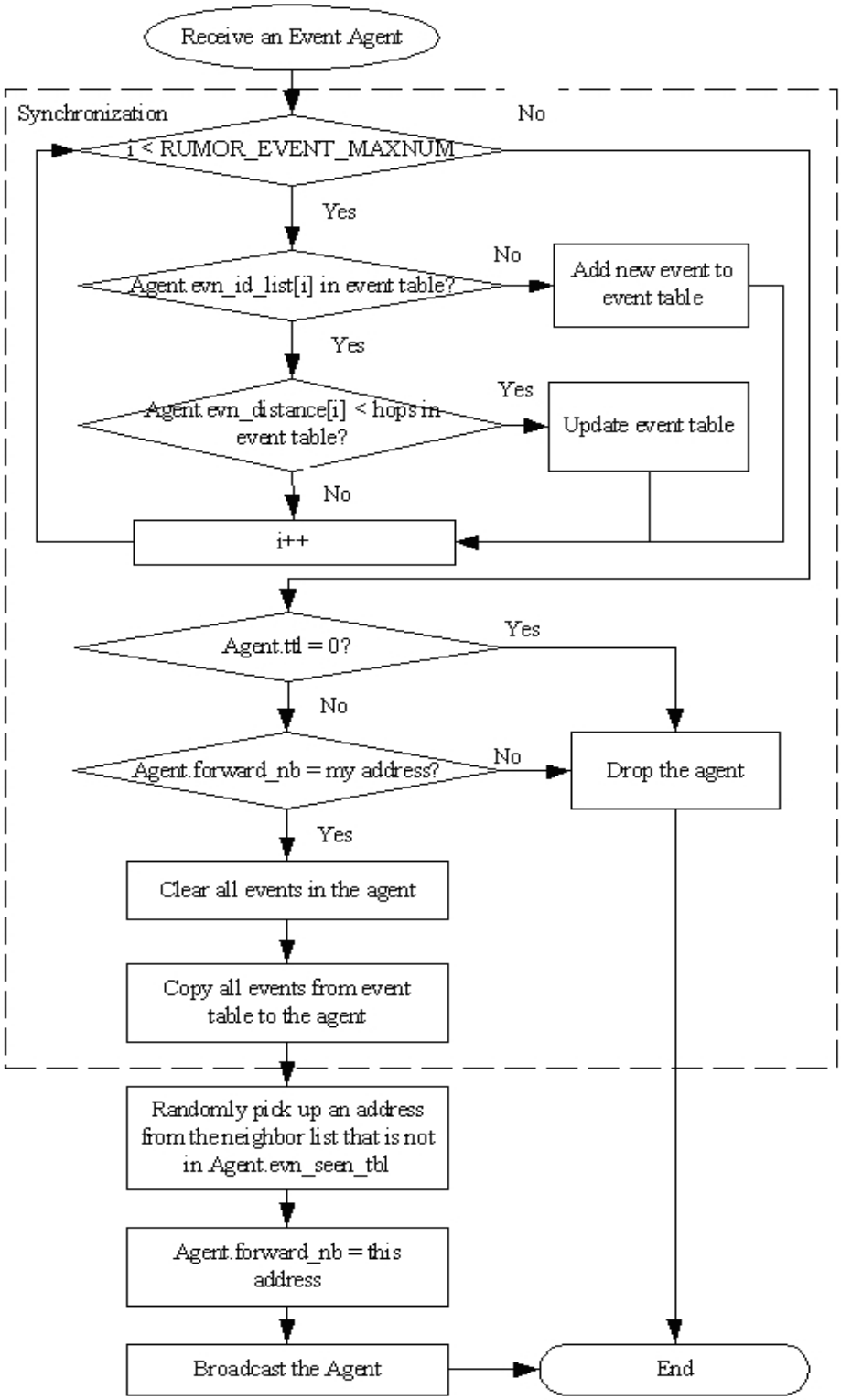

Figure 6.13: The sequence of events when a node receives an Event Agent packet.

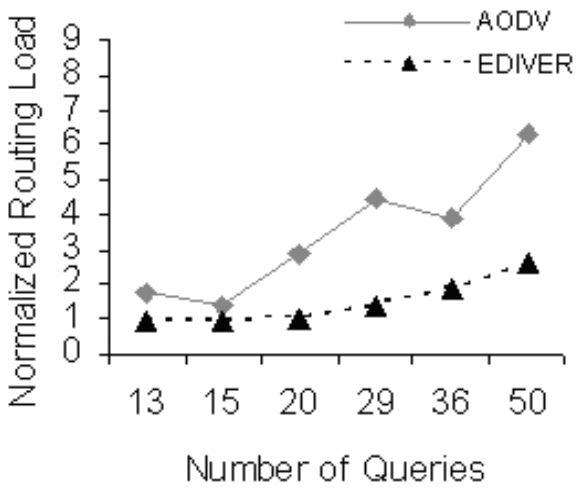

Figure 6.14: Normalized routing load for AODV and EDIVER.

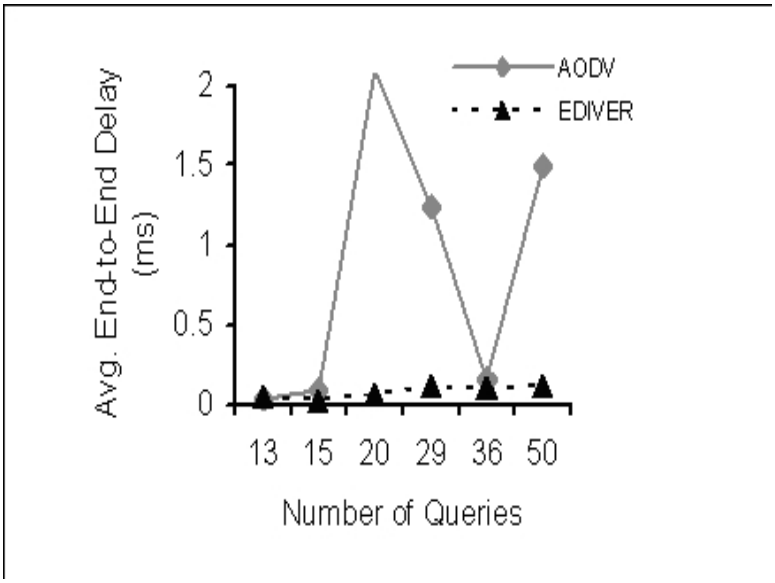

Figure 6.15: Average End-to-End Delay for different number of quires.

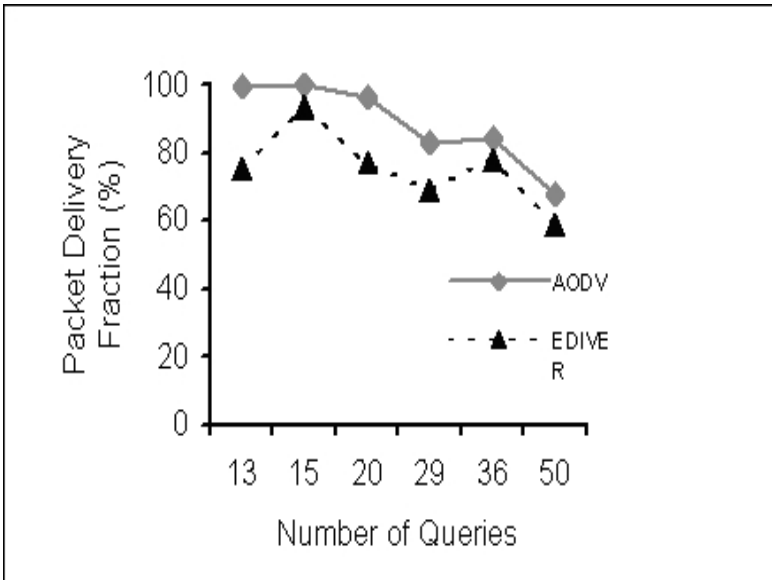

Figure 6.16: Packet delivery fraction for AODV and EDIVER.

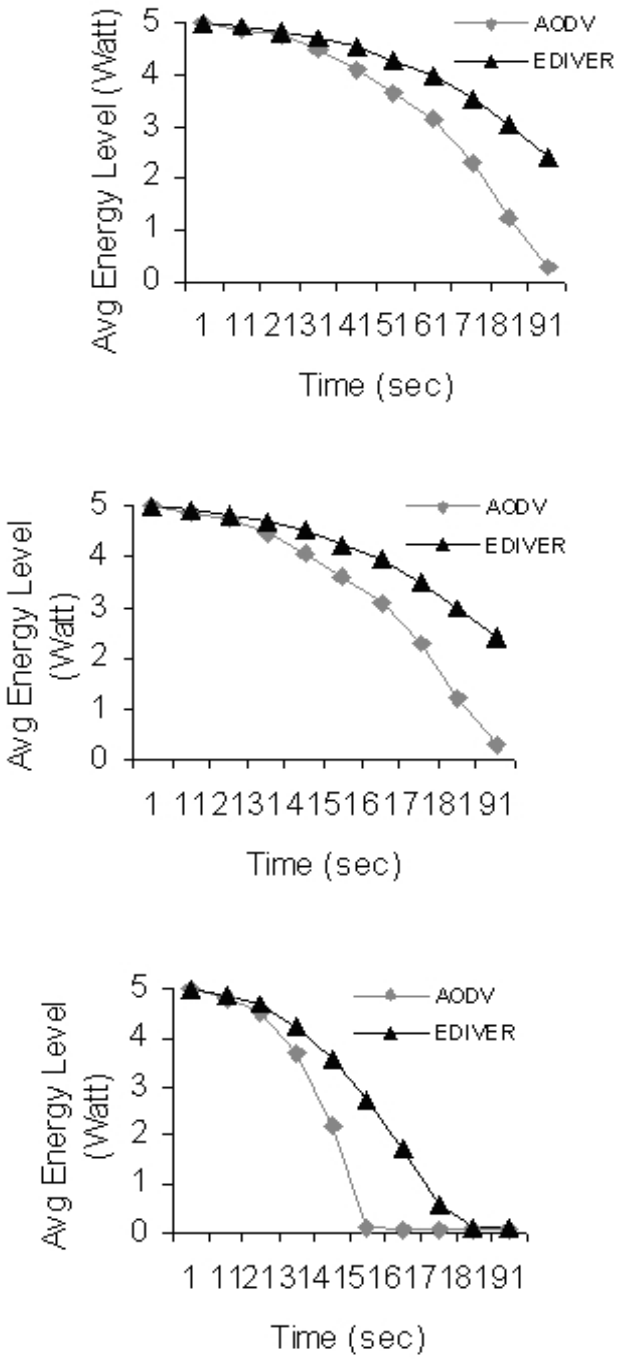

Figure 6.17: Avg. Energy Level (a) 13 Requests (b) 20 Requests and (c) 50 Requests.

Chapter 7

Conclusions and Future Work

For spatiotemporal applications, recursive queries are not expressible using the basic query languages of GIS systems. Some relational database and knowledge-based systems provide recursive queries, but they do not provide spatiotemporal data representation. Hence the visualization of recursively defined concepts cannot be handled by these systems in an easy way. They would usually require some special functions to be written in a programming language like C or C++ and added to a library. In contrast, the RecIV system only uses standard SQL and Datalog queries to solve the problem. Therefore, the program is a simple, declarative, and high-level query that is easy to maintain.

This feature is important, because the requirement of visualizing recursively defined concepts on spatio-temporal data is frequent enough to need a general and simple solution method. We give just one more example:

Example 7.0.1 *An ecosystem is in* danger during month T *if during month T the density of one important plant species either*

(i) decreased 10% or more or

(ii) decreased between 2% and 10%

and *it was already in danger during month* $T-1$.

ex spatio-temporal problems (Revesz & Li 2003). Bas Constraint databases integrate database technology with constraint solving methods to visualize compiled on the recursive Datalog query language and the MLPQ system, we can visualize some novel queries in a simple and efficient way, which would be difficult or impossible to do with other systems.

We are currently extending the usage of the RecIV system. For example, we are improving the user interface and allow the user to specify the color of each overlay object. We are expanding the *recursive Datalog Generator* so that the user can select an object and calculate the growth in its area between two different time instances. This would add a time parameter to the web interface in Figure 4.7. Besides the improvement of the RecIV system, we are also branching out to other applications that require displaying recursively defined spatio-temporal concepts.

Our general solution for reasoning about epidemics and related spatiotemporal phenomena enables one to solve many problems similar to WNV without much modification. The user does not need to search for different ad hoc solutions for each specific epidemiological case.

There are still many interesting directions for future work. For example, we are currently extending the WeNiVIS system by allowing the user to specify in a convenient way the color of each overlay object. We are also expanding the *recursive SQL generator* so that the user can select an object and calculate the growth in its area between two different time instances. Besides improving WeNiVIS, we are also branching out to other epidemic applications.

These and other issues remain interesting research topics for the future. We provide free copies of the WeNiVIS system to other researchers and potential users who would like to try it out on WNV or other similar infectious disease data.

The common ideas of conventional models try to describe the problems and so-

lutions using real world objects, while a stability model try to use EBTs and BOs to answer the questions such as: "Why do we build this system?" and "How does the system achieve its goal?" The key to stability modeling is to identify aspects of the environment in which the software will operate without change, and to cater the software to these areas (Fayad 2002).

Although stability models demand a greater investment in analysis, our practical experience in Fayad02 has shown that the savings in development and maintenance costs can more than make up for the additional time spent during analysis. We also show that the stability approach has potential to reduce or eliminate the cost of the re-engineering cycles commonplace in software engineering projects. We view the software stability approach as an essential refinement to existing Object-Oriented analysis and design processes. Moreover, we do not note any added complexity in SSMs. Instead, our study suggests SSMs are concise, elegant, and inherently adaptable and extensible.

"Patterns explicitly capture expert knowledge and design tradeoffs, and make this expertise more widely available (Schmidt, Fayad & Johnson 1996)." However, we point out in (Wu, Mahdy & Fayad 2002) that the implementation of design patterns has difficulty constructing stable software products because much of the design abstractions are lost in the implementation, with no traceability back to the design patterns. In (Wu, Mahdy & Fayad 2002), we also propose software stability approach as a practical method of explicitly describing the two-way mapping relationship between design patterns and their implementations/instances.

We proposed the EDIVER routing algorithm, which combines different features from both AODV and Rumor algorithms, for wireless sensor networks. Extensive simulations show an improvement in energy consumption, average end-to-end delay and normalized routing overhead over the AODV. However, packet delivery fraction for EDIVER was lower than that of the other two algorithms, although it was very

close in some scenarios.

For future work, investigating different approaches that can be used to improve the packet delivery fraction in EDIVER is an interesting problem. In addition, one may study the scalability issues in the EDIVER. Moreover, extending EDIVER to accommodate the case when sensor nodes are mobile is an important research issue.

Bibliography

Braginsky, D. & Estrin, D. (2002), Rumor routing algorithm for sensor networks, *in '1st ACM Workshop on Sensor Networks and Applications (WSNA)'*, ACM press, pp. 22–31.

Brodsky, A., Segal, V. E., Chen, J. & Exarkhopoulo, P. A. (1999), The CCUBE constraint object-oriented database system., *in* 'SIGMOD Conference', pp. 577–579.

Centers for Disease Control and Prevention (CDC) (2002), 'West Nile virus website: http://www.cdc.gov/ncidod/dvbid/westnile/'.

Chomicki, J. (1994), Temporal query languages: a survey, *in* D. M. Gabbay & H. J. Ohlbach, eds, 'Temporal Logic: ICTL'94', Vol. 827 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 506–534.

Chomicki, J., Haesevoets, S., Kuijpers, B. & Revesz, P. (2003), 'Classes of spatio-temporal objects and their closure properties', *Annals of Mathematics and Artificial Intelligence* **39**, 431–461.

Chomicki, J. & Revesz, P. (1999), 'Constraint-based interoperability of spatiotemporal databases', *Geoinformatica* **3**(3), 211–43.

Coad, P., North, D. & Mayfield, M. (1995), *Object Models Strategies, Patterns, & Applications*, Yourdon Press, Prentice-Hall, Inc. New Jersey.

Codd, E. F. (1970), 'A relational model for large shared data banks', *Communications of the ACM* **13**(6), 377–87.

D. Nash, F. Mostashari, A. Fine, et al. (2001), 'The outbreak of West Nile virus infection in the New York City area in 1999', *The New England Journal of Medicine* **344**, 1807–14.

Damianos, L., Ponte, J., Wohlever, S., Reeder, F., Day, D., Wilson, G. & Hirschman, L. (2002), 'MiTAP for bio-security: a case study', *AI Magazine* **23**(4), 13–29.

Demers, M. N. (2000), *Fundamentals of Geographic Information Systems, 2nd edn*, Jhon Wiley & Sons, New York.

Dutch, S. (2003), The universal transverse mercator system, Natural and Applied Sciences, University of Wisconsin - Green Bay.
URL: *http://www.uwgb.edu/dutchs/FieldMethods/UTMSystem.htm*

Elmasri, R. & Navathe, S. B. (2003), *Fundamentals of Database Systems - Fourth Edition*, Addison-Wesley.

Fayad, M. E. (2002), 'Accomplishing software stability', *Communication of the ACM* **45**(1), 95–98.

Fayad, M. E. & Altman, A. (2001), 'An introduction to software stability', *Communications of the ACM* **44**(9).

Fayad, M. E. & Wu, S. (2002), 'Merging multiple traditional models in one stable model.', *Communications of the ACM* **45**(9), 102–106.

Gamma, E., Helm, R., Johnson, R. & Vlissides, J. (1995), *Design Patterns: elements of reusable object-oriented software*, Addison-Wesley.

Goldin, D. Q., Kutlu, A., Song, M. & Yang, F. (2003), The constraint database framework: lessons learned from CQA/CDB., *in* 'International Conference on Data Engineering', pp. 735–737.

Goodman, J. E. & O'Rourke, J. (1997), *Handbook of Discrete and Computational Geometry*, CRC Press.

Grumbach, S., Rigaux, P. & Segoufin, L. (1998), The DEDALE system for complex spatial queries, *in* 'ACM SIGMOD International Conference on Management of Data', pp. 213–24.

Gupta, S. (2003), 'Performance evaluation of ad hoc routing protocols using ns2 simulations'.
URL: *http://www. cs.utk.edu/ gupta*

Hamza, H. & Wu, S. (2004), Raodv: An efficient routing algorithm for sensor networks, *in* '*Proceedings of the 2nd international conference on Embedded networked sensor systems*', ACM Press, pp. 297–298.

Hamza, H., Wu, S. & Deogun, J. (2005), Ediver: An effecinet distance vector routing algorithm for wireless sensor networks, *in* '*Proceedings of the IASTED International Conference on Communication Systems and Applications (CSA 2005)*'.

He, T., Stankovic, J. A., Lu, C. & Abdelzaher, T. F. (2003), Speed: A stateless protocol for real-time communication in sensor networks, *in* '*Proceedings of the 23rd International Conference on Distributed Computing Systems (ICDCS)*', pp. 46–.

Kanellakis, P. C., Kuper, G. M. & Revesz, P. Z. (1995), 'Constraint query languages', *Journal of Computer and System Sciences* **51**(1), 26–52.
URL: *http://www.cse.unl.edu/ revesz/JCSS95.ps*

Kulik, J., Heinzelman, W. & Balakrishman, H. (2002), 'Negotiation-based protocols for disseminating information in wireless sensor networks', *Wireless Networks* **8**(2-3).

Kuper, G. M., Libkin, L. & Paredaens, J., eds (2000), *Constraint Databases*, Springer-Verlag.

Langran, G. (1992), *Time in Geographical Information Systems*, Taylor & Francis.

Li, L., Li, Y. & Piltner, R. (2004), "A New Shape Function Based Spatiotemporal Interpolation Method", *in* '1st International Symposium on the Applications of Constraint Databases', number 3074 *in* 'Lecture Notes in Computer Science', Springer, pp. 25–40.

Li, L. & Revesz, P. (2002), A comparison of spatio-temporal interpolation methods, *in* 'Proceedings of the 2nd International Conference on Geographic Information Science', Lecture Notes in Computer Science, Springer-Verlag, pp. 145–160.

Li, L. & Revesz, P. (2003), "The relationship among GIS-oriented spatiotemporal databases", *in* 'Third National Conference on Digital Government Research', pp. 375–378.

Li, L. & Revesz, P. (2004), 'Interpolation methods for spatio-temporal geographic data', *Computers, Environment and Urban Systems* **28**(3), 201–227.

Longley, P. A., Goodchild, M. F., Maguire, D. J. & Rhind, D. W. (2001), *Geographic Information Systems and Science*, Wiley.

Mahdy, A., Fayad, M., Hamza, H. & Tugnawat, P. (2002), Stable and reusable model-based architectures, *in* '*Proceedings of 12th Workshop on Model-based Software Reuse, 16th ECOOP*'.

McKee, T. B., Doesken, N. J. & Kleist, J. (1993), The relationship of drought frequency and duration to time scales, *in* '8th Conference on Applied Climatology', pp. 179–184.

Pennsylvania's West Nile Virus Control Program (2004), 'West Nile virus surveillance maps: http://www.westnile.state.pa.us/surv.htm'.

Perkins, C. & Royer, E. (1999), 'Ad-hoc on-demand distance vector routing'.

Project, M. (2003), Monarch extension to the ns-2 simulator.
URL: *http://www.monarch.cs.rice.edu/cmu-ns.html*

Raffaetà, A., Renso, C. & Turini, F. (2002), Enhancing GISs for spatio-temporal reasoning, *in* '10th ACM International Symposium on Advances in Geographic Information Systems', ACM Press, pp. 42–48.

Ramakrishnan, R. (1998), *Database Management Systems*, McGraw-Hill.

Revesz, P. (2002), *Introduction to Constraint Databases*, Springer.

Revesz, P., Chen, R., Kanjamala, P., Li, Y., Liu, Y. & Wang, Y. (2000), The MLPQ/GIS constraint database system, *in* 'ACM SIGMOD International Conference on Management of Data'.

Revesz, P. & Li, L. (2002), Constraint-based visualization of spatial interpolation data, *in* '6th International Conference on Information Visualization', IEEE Press, pp. 563–569.

Revesz, P. & Li, L. (2003), "Constraint-Based visualization of spatiotemporal databases", *in* M. Sarfraz, ed., 'Advances in Geometric Modeling', John Wiley Inc., pp. 263–276.

Revesz, P. & Wu, S. (2003), A stability model for constraint database management system, *in* '*Proceedings of the Workshop on Stable Analysis Patterns: A True Problem Understanding with UML*', pp. 29–31.

Revesz, P. & Wu, S. (2004), Visualization of recursively defined concepts, *in* '8th International Conference on Information Visualization', IEEE Press, pp. 613–621.

Revesz, P. & Wu, S. (2005), 'Spatiotemporal reasoning about epidemiological data', *Artificial Intelligence in Medicine* p. submitted.

Schmidt, D., Fayad, M. & Johnson, R. (1996), 'Software patterns', *The Special Issues in Communications of the ACM* **39**(10).

Shewchuk, J. R. (1996), "Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator", *in* 'Workshop on Applied Computational Geometry', number 1148 *in* 'Lecture Notes in Computer Science', Springer.

Silberschatz, A., Korth, H. & Sudarshan, S. (2005), *Database System Concepts*, McGraw-Hill, 5th edition.

Soukup, J. (2001), 'Implementing patterns'.
URL: *http://www.codefarms.com/publications/papers/patterns.html*

Terry, N. G. (1996), How to read the universal transverse mercator (UTM) grid, *in* 'Advanstar Communications', p. 32.
URL: *http://www.nps.gov/prwi/readutm.htm*

Theophilides, C. N., Ahearn, S. C., Grady, S. & Merlino, M. (2003), 'Identifying West Nile virus risk areas: The dynamic continuous-area space-time system', *American Journal of Epidemiology* **157**(9), 843–854.

Toman, D. (1996), Point vs. Interval-based Query Languages for Temporal Databases, *in* 'ACM Symposium on Principles of Database Systems', pp. 58–67.

U.S. Geological Survey (2004), 'West Nile virus maps: http://westnilemaps.usgs.gov/'.

WHO-Europe (2000), *Air quality guidelines for Europe, Second Edition*, number 91 *in* 'European Series', World Health Organization Regional Publications.

Worboys, M. F. (1995), *GIS: A Computing Perspective*, Taylor & Francis.

Wu, S. (2003), User interface improvement for mlpq system, Master's thesis, University of Nebraska - Lincoln.

Wu, S., Fayad, M. E. & Nabavi, M. (2002), Stable model-based software reuse., *in* '*Proceedings of the 16th European Conference on Object-Oriented Programming (ECOOP)*'.

Wu, S., Hamza, H. & Fayad, M. E. (2003), Implementing pattern languages using stability concepts., *in* '*Proceedings of the ChiliPLoP workshop*'.

Wu, S., Mahdy, A. & Fayad, M. E. (2002), The impact of stability on design patterns implementation., *in* '*Proceedings of the Pattern Languages of Programs (PLoP) Conference*'.

Wu, S. & Revesz, P. (2004), Doas: A drought online analysis system with constraint databases., *in* '*Proceedings of the 4th National Conference on Digital Government Research*', pp. 417–418.