Representing and Manipulating Spatial Data in Interoperable Systems and its Industrial Applications

by

Marat Zborovskiy

B.S. Computer Science (1999)

Florida International University

Submitted to the System Design and Management Program
in Partial Fulfillment of the Requirements for the Degree of

Master of Science in Engineering and Management

at the

Massachusetts Institute of Technology

May 2006

Signature of Author [illegible]
Marat Zborovskiy
System Design and Management Program
February 2002

Certified by [illegible]
David Brock
Thesis Supervisor
Laboratory for Manufacturing and Productivity

Certified by [illegible]
Patrick Hale
Director
System Design and Management Program

Representing and Manipulating Spatial Data in Interoperable Systems and its Industrial Applications

by

Marat Zborovskiy

Submitted to the Engineering Systems Division

on 5/01/2006 in Partial Fulfillment of the Requirements for the Degree of

Master of Science in Engineering and Management

ABSTRACT

Multiple ways to represent spatial data in information systems were researched. The conclusion was made that only the semantic approach can ensure seamless interoperability among heterogeneous spatial information systems. Modern research was found to concentrate on lower levels of interoperability without much concern for the power of semantics. A new spatial data standard was developed that is compliant with the M-language designed with data interoperability. The new standard allows for the semantic interoperability of spatial data. The spatial data is represented by a set of images along with functions to operate on the set. A Service-Oriented Architecture is proposed as a preferred approach for deploying the standard. A prototype was built to demonstrate the standard and the adopted architecture.

Thesis Supervisor: David L. Brock, Ph. D.
Title: Principal Research Scientist

Acknowledgements

The research that led to this thesis would not be possible without the support and dedication of the members of the MIT Data Center. I would like to specifically thank my thesis advisor and the MIT Data Center Director and founder, David Brock, for the opportunity that he provided me to work under his tutelage. My gratitude also goes to Edmund Schuster, the Data Center co-director who was always there to provide his support and expert guidance.

My special thanks go to my wife Anna and daughter Alice for helping me keep my spirits up and encouraging me to pursue my MIT dream.

TABLE OF CONTENTS

M Spatial Data XML File **150**

TABLE OF FIGURES

Chapter 1 Introduction

Motivation

The amount of information available nowadays is staggering and increases exponentially. Making sense of this data has become increasingly difficult because of the two factors:

- The sheer volume of data
- The lack of interoperability between disparate data sources and models

While one can do little about the former factor, the latter one can be mitigated by advancing solutions that make data easy to work with and ensure the interoperability among data sources and models in intelligent networks. One way to achieve interoperability is to force every entity involved in the data exchange to adopt the same standard. However, organizations have heavily invested in proprietary data standards and are unlikely to replace their existing standards with a new one. Therefore, another solution is to create a standard, through which organizations can translate their data sources and share them with their customers or general community. The MIT Data Center[1] is spearheading an initiative to create M – a language that is capable to provide the much needed interoperability between divergent data sources and models with an ultimate goal of creating a new intelligent information infrastructure (Brock, Schuster and Kutz 2006).

Much of the information available today comes in the form of spatial data. For instance, in agriculture weather conditions around the crop fields, concentrations of crop insects, etc are often given in the spatial form. Ensuring the interoperability of spatial data is, therefore, important and will lead to greater efficiencies and new opportunities in data analysis in many

[1] www.mitdatacenter.org

fields, including marketing, agriculture, geographical information systems, and defense. However, the use of spatial data in business decision making is currently underutilized because of interoperability issues.

It is worth noting that although some authors use the term "spatial" and "geospatial" interchangeably, this thesis clearly delineates the two. Spatial data and systems encompass entities in the abstract n-dimensional space and are generic enough to describe physical multidimensional phenomena. Geospatial systems and related data manipulate geographical information and data that can be overlaid on top of various types of geographical maps. Geospatial systems have many practical applications, and most of the today's research concentrates on the interoperability within geospatial systems only. However, while geospatial systems are an important subset of spatial systems, this thesis' goal is to use a holistic approach in analyzing spatial systems and to develop and analyze methods, algorithms, and standards that are generic enough to be used across many disciplines.

Research Objectives

The purpose of the thesis is to extend the M-Language with additions of rules for spatial data representation and examine various data manipulation techniques that such a representation can afford. Primary research objectives include:

- Determining optimal rules for a spatial data representation standard
- Exploring various ways that data can be manipulated
- Examining how models can be applied to such representations to gain useful insights.

Research Methods and Approaches

The thesis investigates existing methods for representing spatial data and then uses decision analysis to select an optimal specification. In addition, software tools and prototypes are developed to demonstrate the chosen specification

Chapter 2 Literature Review

Issues in Spatial Data Interoperability

Digital information has achieved its prominence as a result of the computer revolution that has been ongoing for the last few decades. Because of the nature of the fast development and change in information technologies, significant progress has been achieved, and many competing technologies and systems have been built. However, the flip side of the fast progress was a proliferation of proprietary and incompatible standards and systems. Organizations and individuals that use computer technologies have long understood the need for data interoperability to efficiently utilize existing resources and take maximum advantage of all the available technologies and data sources. The rise of Internet over the last decade facilitated the notion that system and data interoperability is a necessity rather than a luxury.

Spatial data systems like other computer information systems have been built first without interoperability in mind, but the need for interoperability has become apparent in the last 10 years. Andrej Vckovski (1998) notes that spatial information systems have been able to integrate disparate information sources and that some researchers want to establish interoperability as a defining principle for spatial information systems. However, Vckovski pragmatically concedes that modern systems still lack compatibility in data formats, quality standards, models of the world, etc.

To make things more complicated, modern spatial information systems differ in the way they handle the data. For example, there exist vector and raster Geographic Information Systems (GIS), Earth imaging systems, navigation systems, systems, providing location-based services, cartography, etc. (Reichardt, 2004). Vector GIS depict images with vectors that are comprised of lines and arcs. Raster GIS, on the other hand, displays images as an ordered grid structure with pixels formed by intersections of rows and columns.

Current State of the Art in Spatial Interoperability

Open Geospatial Consortium

The last several years saw the increasing popularity of the Internet and Web-related technologies to bring interoperability to the world of geospatial data. The Open Geospatial Consortium (OGC)[2] is an umbrella organization that comprises 310 companies at the time of this writing and is tasked with developing public standards and interfaces that achieve interoperability of geospatial data. OGC supports an Interoperability Program that pursues the following initiatives[3]:

Test beds are supposed to provide a quick and efficient way to define, develop, and test various multi-vendor specifications and interfaces.

Pilot Projects provide an opportunity for proof-of-concept of proposed specifications and interfaces. These projects efficiently adhere to best practices in code reuse and the usage of commercial off-the-shelf software that implement OpenGIS specifications[4].

Interoperability Support Services provide consortium members with expertise in developing architecture based on open standards.

Interoperability Experiments comprise of short-termed projects whose goals are to attain a specific technical objective in an effort to advance OGC Technical Baseline.

The Interoperability Program initiatives and its global reach are depicted in Figure 1.

[2] http://www.opengeospatial.org/
[3] http://www.opengeospatial.org/initiatives/
[4] http://www.opengeospatial.org/specs/?page=specs

The Interoperability Program achieved a number of milestones since its inception in 1999. Among the most noteworthy are the following (Harrison, 2002, Reichardt, 2005):

- Web Mapping Testbeds
- Military Pilot Project Testbed
- Object Domain Modeling Support Initiative
- OGC Web Services Initiative
- Sensor Web Enablement

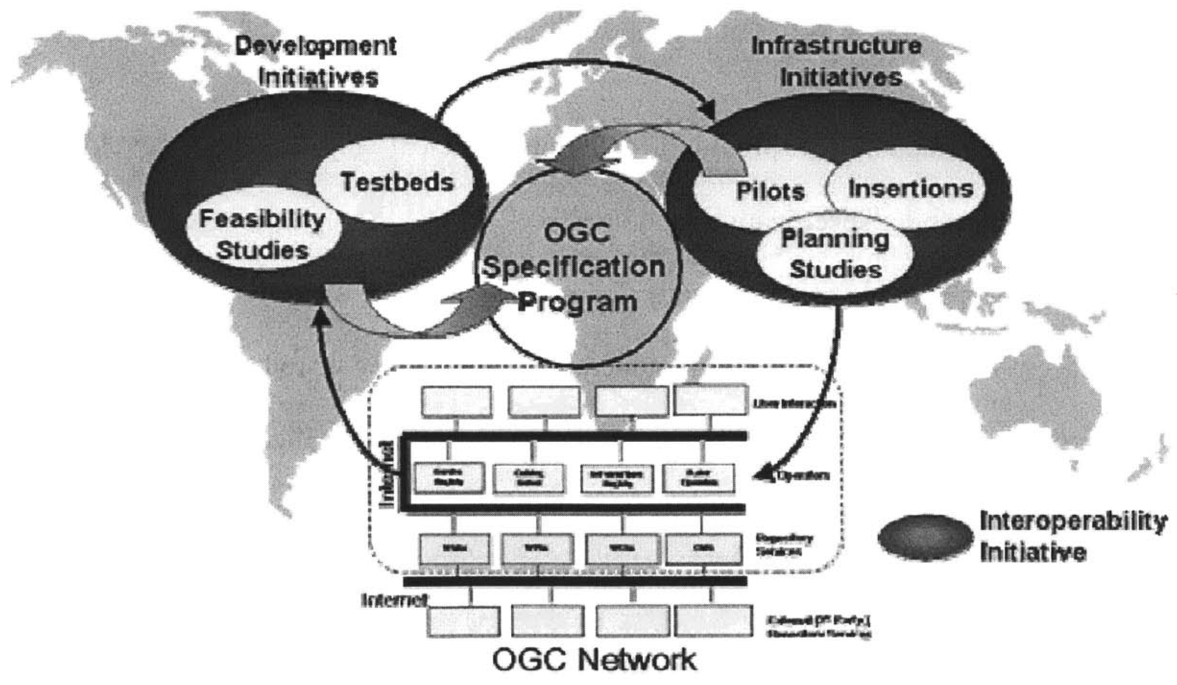


Figure 1: OGC Interoperability Initiatives
Source: (Harrison, 2002)

The OGC philosophy is to promote open standards, protocols, and interfaces, and it operates by consensus. A weak link in OGC's approach to geospatial data interoperability is relatively insignificant attention to semantic issues surrounding interoperability and integration efforts. The importance of semantics is stressed in **Interoperability on Semantic Level** sub-chapter, and semantics forms the backbone of the spatial data interoperability approach proposed in this thesis. OGC has not been totally oblivious to the semantic issues, and it invested resources in exploring and building semantic-aware prototypes developed under the aforementioned Object Domain Modeling Support Initiative. However, these efforts have not achieved significant progress as of the time of this writing.

Interoperability on Semantic Level

It is difficult to provide a definition of interoperability with respect to spatial information systems. Interoperability may be defined differently for various systems, deployed in disparate organizations. For example, in the military systems spatial data interoperability may mean a set of capabilities to seamlessly integrate intelligence information in the forms of satellite images, positions of friendly and enemy forces along with short- and long-term operational plans. On the other hand, in GIS spatial data interoperability may be defined less broadly to mean the capabilities to interlay images on the top of each other in a meaningful manner.

Yaser Bishr (1998) solves this problem by defining six levels of spatial information interoperability, in the order of the increased level of capabilities and sophistication:

- Network protocols
- Hardware and operating systems (OS)
- Spatial data files
- Database Management Systems (DBMS)

- Data model
- Application semantics

Figure 2 depicts these levels of interoperability between two independent spatial systems A and B:

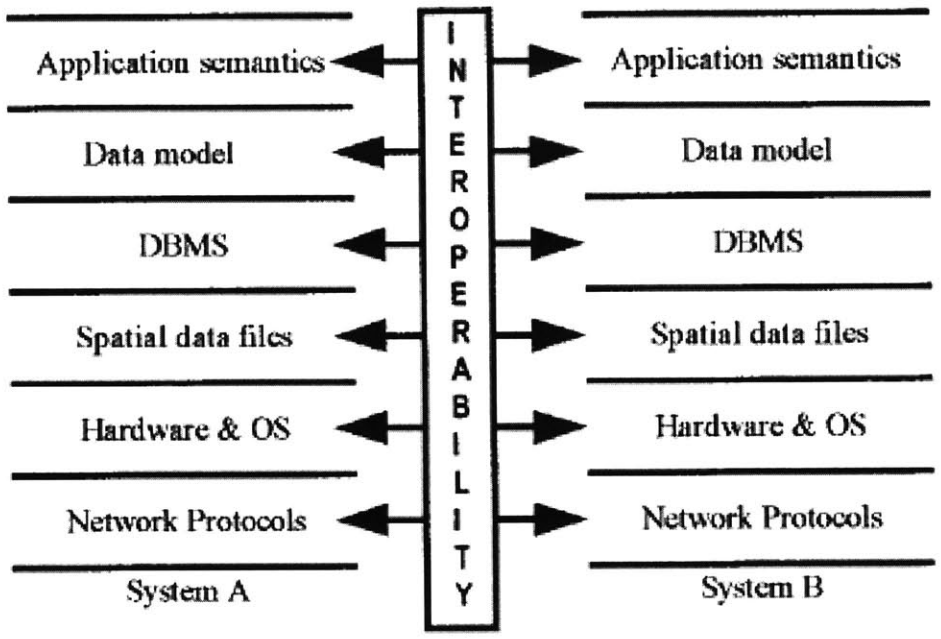


Figure 2: Levels of Interoperability
Source: (Bishr 1998)

A short description of each level is in order (Bishr 1998):

Network Protocols
This is the lowest level of interoperability and is present in any modern networking system. An example of such interoperability is Internet itself where remote machines can talk to each other using the ubiquitous TCP/IP protocol[5].

[5] http://en.wikipedia.org/wiki/TCP/IP

Hardware and OS

This level allows systems to interact without the knowledge of the underlying hardware or operating systems. For example, File Transfer Protocol (FTP)[6] uses its own proprietary commands (get, put, etc.) that are abstracting away the underlying operating system. At this level users need to understand the format of files that they exchange.

Spatial Data Files

At this level systems can exchange standard files and convert them into internal formats if needed. This approach is widespread nowadays, and I describe two common file standards in Chapter 3.

DBMS

At this level users can query systems in their local query language, and the interoperability is carried through via standard databases access methods, such as Open Database Connectivity (ODBC)[7]. However, the systems need to know respective data models and semantics to be able to interoperate with each other.

Data Model

Remote systems at this level are abstracted out by a virtual data model that all systems use for the data access and information interchange. This level of interoperability is where much of the current research is concentrated. A mediator approach to Data Model interoperability is described in (Bishr 1998). Even though systems have the same virtual data model at this level, systems' semantics may differ. Therefore, the knowledge of participating actors' semantics is still required.

[6] http://en.wikipedia.org/wiki/File_Transfer_Protocol
[7] http://en.wikipedia.org/wiki/ODBC

Application Semantics

The following definition of semantics will be used in this thesis (Brock et al., 2005, p. 99):

"Semantics refers to the precise meaning of data or models in a machine and human understandable way. Once a semantic is established, it can be used as a descriptor."

Application Semantics level is the most powerful and provides the most interoperability between spatial information systems. However, since there are no viable semantics-aware systems, either in the research or commercial domain, it is instructive to examine prospective approaches and systems. The bottom two levels, Network Protocols and Hardware/OS, are of little interest for this thesis.. These are inherently low-level and are concerned with infrastructure, or "plumbing," that allow computer information systems to communicate with each other. The reader is advised to refer to excellent texts by Andrew Tanenbaum, (2002) and (2001), for a thorough treatment of Network and Hardware/OS levels, respectively.

Interoperability through the DBMS level is still widespread. First, there are generic standards, such as ODBC and JDBC.[8] Using these standards, one is capable to connect to various databases in wide use, such as Oracle, Microsoft (MS) SQL Server, and MySQL.[9] Using similar computer code to connect to disparate databases avoids the issue of incompatible SQL statements.[10] One of the major disadvantages of database connectivity standards is the difficulty in maintaining database consistency.

[8] A competitor to ODBC for the Java world, see http://en.wikipedia.org/wiki/JDBC

[9] Oracle and MS SQL Server are commercial closed-source relational DBMS' while MySQL is an open-source alternative. See www.oracle.com, http://www.microsoft.com/sql/prodinfo/overview/what-is-sql-server.mspx, and www.mysql.com for the information on Oracle, MS SQL Server, and MySQL, respectively.

[10] See http://en.wikipedia.org/wiki/SQL for an SQL primer. Although SQL is an ANSI/ISO standard, each major RDBMS vendor includes proprietary extensions to the language, thus rendering it largely incompatible among competitors.

Federated Databases

There is active research in the subject of federated databases that aims to solve the problem of maintaining the data consistency (Devogele et al. 1998). Federated Databases technology allows database administrators to choose which parts of their databases to make public and which parts to keep local. Then users can access the combined parts from participating databases as a single virtual database. The virtual database has only a schema defined, and database instances are created dynamically, when needed. This whole process is managed by the federated system.

For the federated database technology to become successful, there are two main problems to solve: design and operation. Design issues encompass database integration, evolution of schema, and intricacies of running multiple databases in parallel, whereas operational issues focus on object and method exchanges, security, and open architecture among other things.

According to Parent and Spaccapietra (1998, p. 166), "database integration is the process which takes as input a set of databases, and produces as output a single unified description of the input schemas (the integrated schema) and the associated mapping information supporting integrated access to existing data through the integrated schema." Oftentimes, database integration has to be done manually, wasting valuable time and resources. Automating this process would enable database integration to play a larger role in the spatial data integration in the future (Devogele et al. 1998). Database integration involves 3 steps (Parent and Spaccapietra 1998):

1) Pre-integration, where transformations are applied to input schemas to make them syntactically and semantically aligned with each other.
2) Correspondence identification, where interschema relationships are identified and described.
3) Integration, where all schemas are combined while resolving interschema conflicts.

This process is depicted in Figure 3.

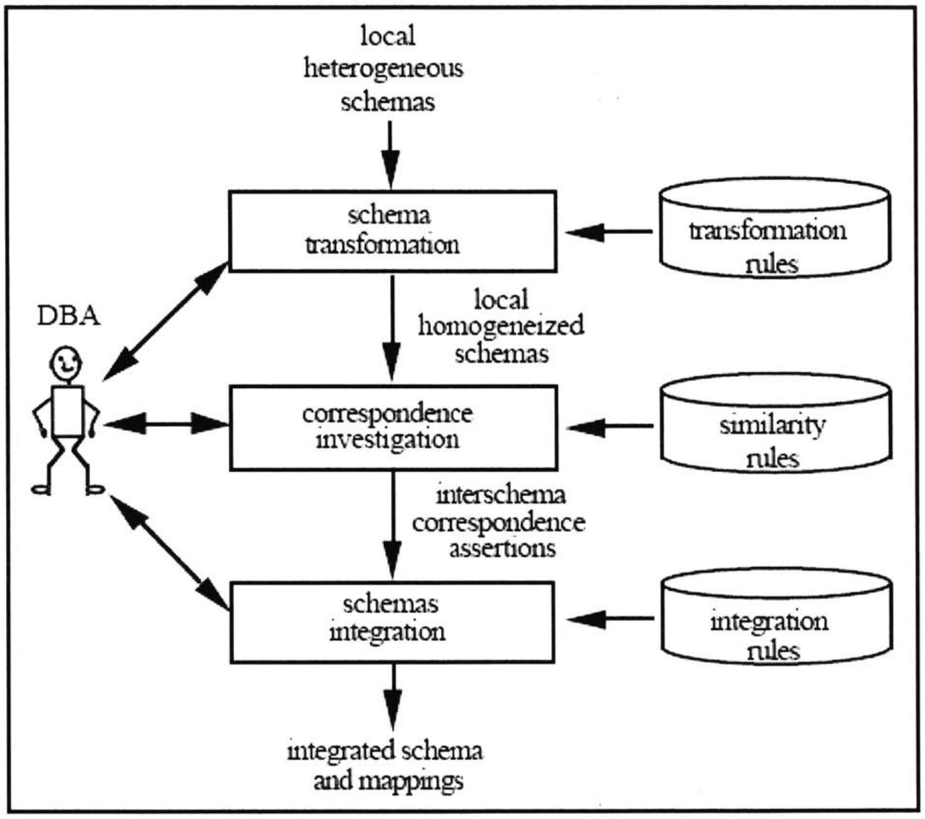

local heterogeneous schemas
schema transformation
transformation rules
DBA
local homogeneized schemas
correspondence investigation
similarity rules
interschema correspondence assertions
schemas integration
integration rules
integrated schema and mappings

Figure 3: Database Integration Process
Source: (Parent and Spaccapietra 1998)

While the DBMS interoperability level is commonly used to exchange information in virtually all fields of knowledge, there are specific issues that are pertinent to spatial information systems. These issues include transformations between raster and vector data (Piwowar et al. 1990), coordinates transformation (Bugayevskiy and Snyder 1995), geometry matching using overlay techniques (Frank 1987, Dougenik 1980, Pullar 1993, Demirkesen and Schaffrin 1996) or rubber-sheeting techniques (Fagan and Soehngen 1987, Laurini 1994), surface aggregation (Flowerdew 1992), and automated generalization in multi-scale databases (Ruas and Plazanet 1996) (as cited in Parent and Spaccapietra 1998).

In practice, the federated databases technology has gained traction. IBM has included the technology, known as "Garlic" (Haas and Lin 2002) in its flagship DB2 RDBMS.[11] "Garlic" technology has the following characteristics:

- Transparency – the underlying data sources are abstracted away from the user.
- Heterogeneity – data sources can be disparate, even being originated from non-DB2 databases.
- A high degree of function – a federated database provides DB2 SQL capabilities for all data sources. In addition, unique data sources' functions are retained or substituted by DB2 functions, if required.
- Extensibility and openness of the federation – new data sources can be easily added when needed.
- Autonomy of data sources – operations on data sources are not disturbed when these data sources join the federation.
- Optimized performance – SQL queries execution variants are judged based on the cost of execution, and the path of execution with the minimum cost is chosen.

[11] See http://www-306.ibm.com/software/data/db2/ for IBM DB2 description.

The architecture of an IBM federated system is presented in Figure 4. It is interesting to note that various data sources are included into the federated system via wrappers, thus allowing data sources to be transparent and of heterogeneous nature.

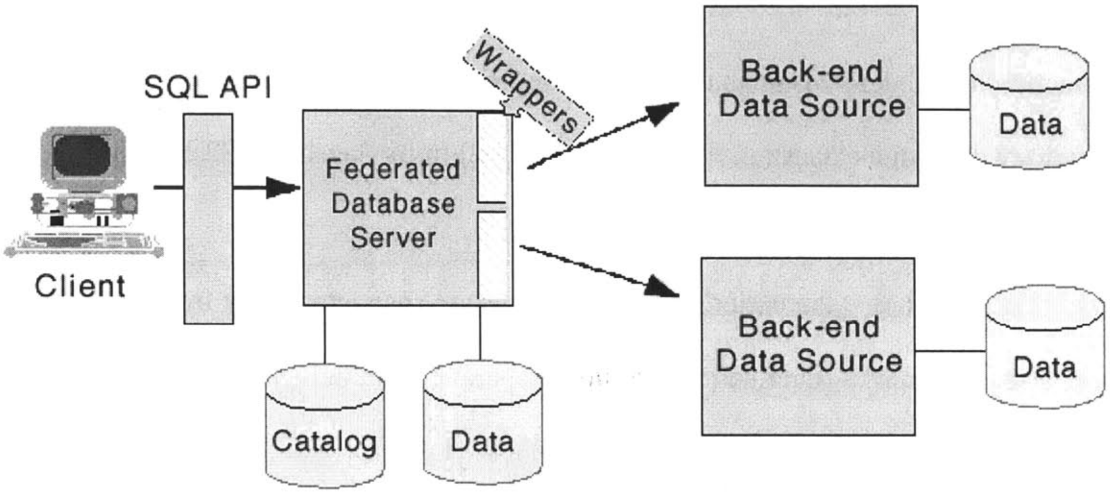


Figure 4: Architecture of an IBM federated system
Source: (Haas and Lin 2002)

Before further exploring the application of semantics in the field of spatial data interoperability, it is useful to briefly look at the subject of ontology that can be applied in computer information systems to automatically reason about various entities.

Ontologies in Computer Information Systems

In the field of computing, "ontology is a data model that represents a domain and is used to reason about the objects in that domain and the relations between them" (Ontology, 2006, para.1). According to this definition, the use of ontologies corresponds to the Data Model level of the interoperability hierarchy (see Figure 2). As such, ontologies exhibit all the interoperability problems that are inherent to data modeling level (Brock et al., 2005). First, ontologies suffer from rigidity and are inflexible in nature. If one decides to create and interoperate models based on ontologies, then one needs to define canonical forms of models, including the exhaustive set of rules for interoperability. As a result, a slight change in models will render interoperability between models incomplete at best or entirely break the interoperability at worst. Second, ontologies are not capable to adopt basing on inductive reasoning, which only adds to their inflexibility. Finally, it is hard to integrate disparate ontologies into a coherent whole. This creates a major dilemma in development of computational systems that are capable of taking advantage of disparate and independent components, logically and geographically distributed over the Internet.

At the highest level of spatial data interoperability, systems can seamlessly integrate with each other and exchange information without the knowledge of underlying semantics. Bishr (1998) developed a theory on the semantic interoperability in GIS. That theory can be extended to cover the general spatial data interoperability and introduces the concepts of Discipline Perception World and Project World. The former encompasses the mental model of users of the discipline (spatial data domain in this case), whereas the latter includes assumptions, axioms, and rules that govern spatial data entities. Discipline Perception World and Project World are shown in Figure 5 along with their mutual interactions.

The relationship between Discipline Perception World and Project World is not immutable when data from two different disciplines need to interoperate. In order to maintain this relationship, one can do one of the following:

- Provide the data consumer with provider's Discipline Perception World information

- Enable an ability to automatically translate provider's Discipline Perception World into consumer's one

The former approach is difficult for users to implement because they would have to interpret the provider's Discipline Perception World and then translate it in their own. The latter approach, on the other hand, is more viable since because it affords a relatively effortless translation, assuming the mechanism exists to translate semantics from the Discipline Perception World into the Project World. When implementing this semantics translation mechanism, the main problem to resolve is the heterogeneity of spatial data environment. Bishr (1998) highlights three heterogeneity subsets: semantic, schematic, and syntactic. These subsets are considered in order, based on Figure 5:

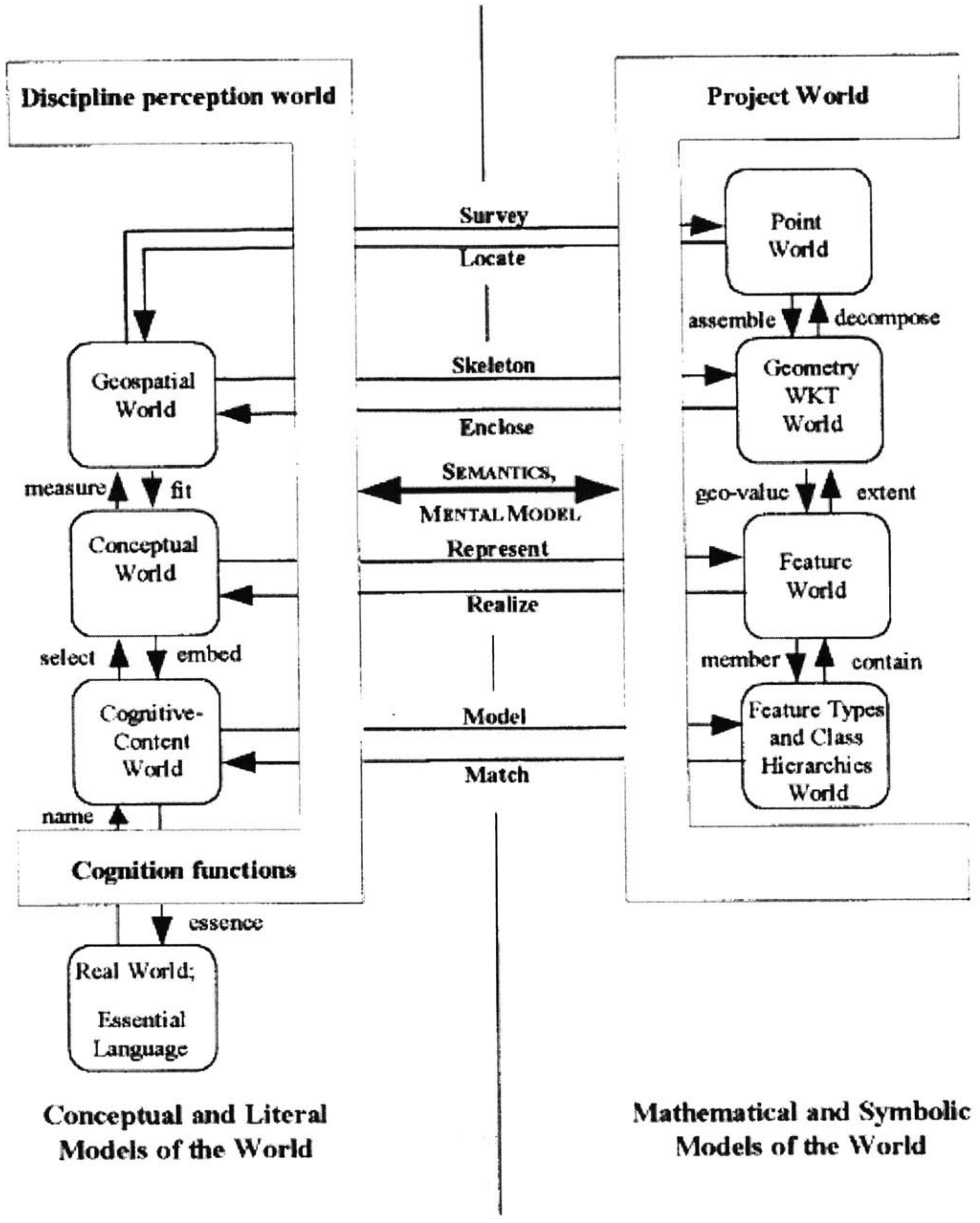


Figure 5: Semantic Interoperability Model
Source: (Bishr 1998)

Semantic Heterogeneity

Semantic heterogeneity is the source of most interoperability problems. Problems related to semantic heterogeneity have their origins in the Discipline Perception World since mental models in different disciplines (domains) differ. For example, the road network means different things in different domains. Logistics and supply chain professionals view roads as transportation arteries through which goods and services are delivered. Transportation departments view the same roads as objects that must be repaired and maintained. Marketing professionals may view the roads as parts of customer addresses. To make things more complicated, the same words may have different semantic meanings, thus making semantic models difficult to interoperate without non-trivial and laborious translation efforts. The MIT Data Center and its M-Language solve this problem by using words with precise meanings given by the M-Dictionary. For example, the word "cell" has different meanings: it may mean a biological cell, a prison cell, a wireless telephony cell, etc. (Brock et al. 2006) The M-Language uses words "cell.1" to define a biological cell, "cell.3" to define a prison cell, and "cell.6" to define a wireless telephony cell.[12] Using such words resolves the main semantic problem stemmed from the inherent ambiguity of human languages. For detailed introduction into the M-Language, see (Brock et al. 2006). Also, a short language description is provided in this thesis; see **Introduction into the M-Language**.

Schematic Heterogeneity

The mental model's schema that encompasses the hierarchy and classification of Real World objects and categories differs from one domain to another. One way to solve this problem is to come up with an integrated schema that serves as an umbrella for all primary heterogeneous schemas. This has been done on the Database Level as presented in this thesis in **Federated Databases**.

[12] Based on M-Dictionary version 0.03, http://18.78.2.98/dictionary003/index.php

Syntactic Heterogeneity

Syntactic heterogeneity can be classified into two types: the representation of the logical data model and the representation of the spatial objects in the database. The logical data model does not have to depend on the underlying database type (relational, object-oriented, semantic, etc). For example, a semantic modeling diagram (Rishe 1992) can be translated directly into objects and relations in a semantic database or transformed into relational database tables.

In addition, the syntactic heterogeneity must be dealt with when considering two distinct ways to represent spatial data in databases:

- Raster data structure: a collection of points that makes up the spatial grid.
- Object data structure: an object identifier connects thematic data and geometric data.

Yaser Bishr (1998) notes that a proper way out of this conundrum is the adoption of common syntax for spatial data representations. The author also claims that resolving the problem of semantic heterogeneity is the primary objective since the other two heterogeneities, schematic and syntactic, will follow the suit. This thesis provides a way to unify disparate spatial data representations at the highest, semantic level.

Operating on the semantics level provides benefits not only for spatial data interoperability but also for the host of other applications. For example, Brock et al. applied principles of semantic modeling to logistical systems (2005). That work emphasizes the importance of a distributed network of models that can be interoperable in a seamless manner using semantic principles. The idea of having the network of models can be applied beyond the field of logistics in various fields where spatial information systems are utilized.

In order for the network of models to be accessible, a semantic modeling language is required. This language implements (Brock et al., 2005, p. 104) “the idea of having data and models defined and linked in a way that can be used by machines not just for display purposes, but also for automation, integration, and reuse across various applications.”

For such a network to be operational, the following components are envisioned (Brock et al., 2005, p. 105):

- Data Modeling Language (DML) – provides formalism for description of modular components via their inputs, outputs, and data elements.
- Data Modeling Protocol (DMP) – facilitates grid computing when multiple models are running on remote machines in parallel.
- Automated Control Language (ACL) – facilitates connections between models and data
- Automated Control Protocol (ACP) -- provides discovery services to match models’ inputs and outputs.
- The Dictionary – a collection of words with multiple meanings. An existing dictionary such as WordNet may be used.[13]

These components form an architecture for a semantic-aware distributed network-centric system. The DML describes models’ inputs and outputs; the ACP finds matching models in the network. The ACL then connects the appropriate models, and the DMP manages

13 It is worth mentioning that further research conducted at MIT Datacenter produced the M-Dictionary, http://18.78.2.98/dictionary/index.php that can be used in this context.

operation in the parallel computing grid. The described system architecture is presented in Figure 6.

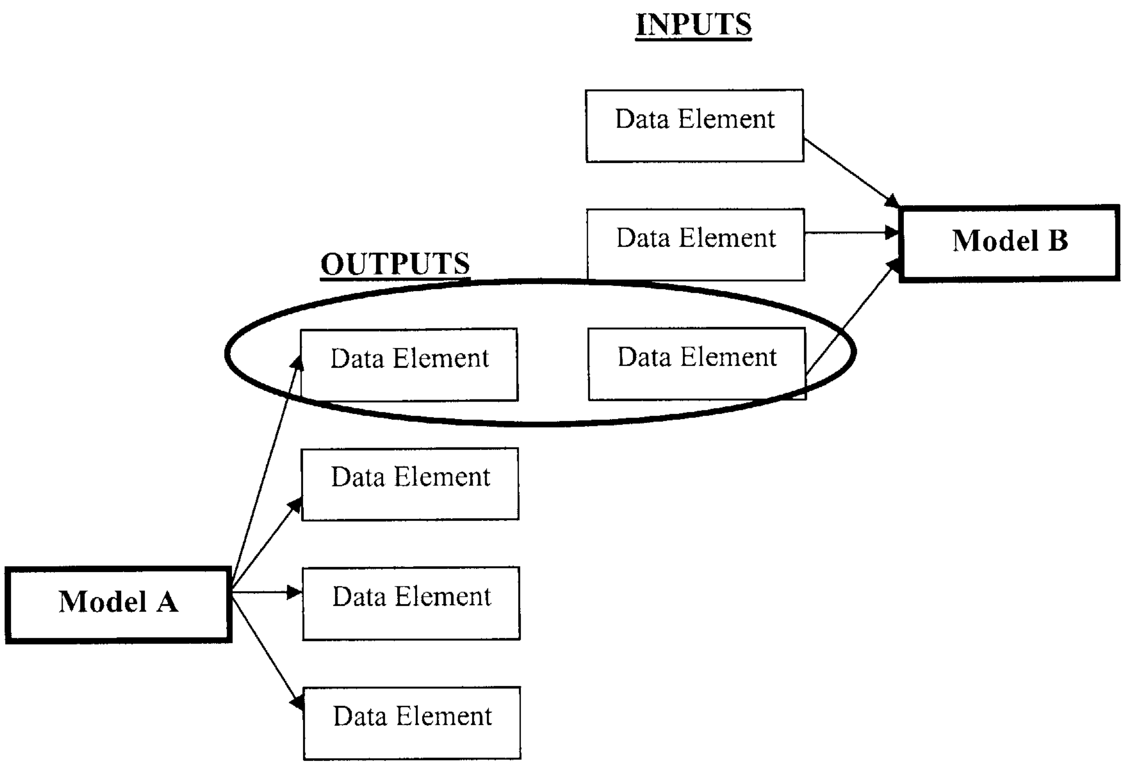


Figure 6: Connecting Models through Data Outputs and Inputs
Source: (Brock et al., 2005)

Chapter 3 Existing Spatial Data Standards

The purpose of this chapter is to review the existing spatial data standards that have been adopted in the industry and academia in the last two decades. Adopting a common standard usually meant selecting a binary file standard that various programs can read and write to. Using such standards represents the interoperability on the Spatial Data Files level only (see Figure 2). While using such a standard is clearly inferior to providing the interoperability on the higher levels, these standards are in wide use nowadays. Therefore, examining them serves the dual purpose of:

- Understanding historical trade-offs and requirements that went into the process of designing a standard
- Making a unified spatial data representation, proposed in this thesis, seamlessly interoperable with existing standards.

ESRI Shapefile

The shapefile standard was created by ESRI[14] for the use in its GIS products. ESRI's flagship product, ArcGIS,[15] provides an integrated approach to serving organization's geospatial data needs and uses shapefiles to interoperate with external data. Because of shapefile's wide use, other makers of GIS systems support the shapefile standard.

Simply put, "A shapefile stores nontopological geometry and attribute information for the spatial features in a data set. The geometry for a feature is stored as a shape comprising a set of vector coordinates." (ESRI Shapefile Technical Description 1998, p.5). Shapefiles do not

[14] http://www.esri.com
[15] http://www.esri.com/software/arcgis/index.html

support topological structures, and while this is clearly a disadvantage, it affords a faster rendering and editing speed. In addition, shapefiles are easier to read and write and require less disk space. While the latter point may be moot in today's era of abundant and cheap disk storage, a small footprint was an important consideration at the time the standard was born.

Basic geometric elements, namely a point, a line, and an area, are supported by shapefiles. Shapefiles represent area features as closed loop, digitized polygons with the double precision. Because of its inherent ability to support these basic geometric primitives, a shapefile standard is a powerful way to represent spatial data. These geometric primitives are augmented by attributes that are stored in a table of records with one-to-one shape-to-attribute mappings (ESRI shapefiles 2006). This table is, in turn, stored in dBASE format files.[16]

Shapefiles can be created using four methods (ESRI Shapefile Technical Description 1998):

- Exporting from software packages that support shapefiles (such as ArcView GIS)
- Digitizing shapes using ArcView built-in tools
- Using a macro language, such as Avenue, ARC Macro Language (AML), or Simple Macro Language (SML)
- Programming to shapefile specifications that are available publicly from ESRI.

Commercial software packages provide support for creating shapefiles for some common use cases, such as presenting Global Position Satellites (GPS) data as shapefiles.

[16] See http://www.dbase.com

A shapefile consists of three entities: a main file, an index file, and a dBASE table. A main file contains shape-describing records. The index file contains offsets from the beginning of the main file for each record. The dBASE table contains attributes for each record.

The physical organization of the main file comprises a fixed-length header and a variable number of pairs where a pair consists of a record header and record contents. This organization is depicted in Figure 7.

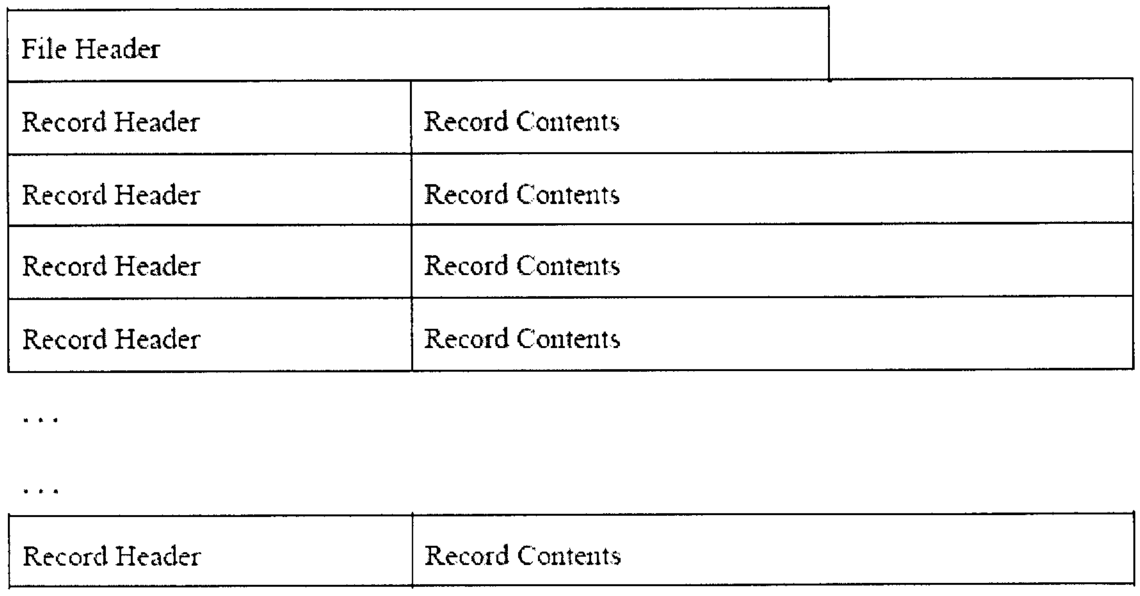


Figure 7: Organization of the Main File
Source: (ESRI Shapefile Technical Description 1998)

The shapefile can be broadly divided into two categories:

- Data related
- File Management related

The shapefile standard exhibits a peculiar byproduct of "Endian wars"[17] with some data composed in the little endian order while other data composed in the big endian order. While

[17] See http://en.wikipedia.org/wiki/Endian for an explanation of endian issues

of obvious interest to computer historians, this eclectic data representation only adds complexity to software development efforts to implement the standard.

While it is beyond the scope of this thesis to explore the representation of every shape in the shapefile standard, an example of specifying a polygon would give readers a glimpse of the shapefile internals. The reader is referred to (ESRI Shapefile Technical Description 1998) for more technical details on the shapefile standard.

A sample shape – polygon

Simply put (ESRI Shapefile Technical Description 1998, p.12),

> A polygon consists of one or more rings. A ring is a connected sequence of four or more points that form a closed, non-self-intersecting loop. A polygon may contain multiple outer rings. The order of vertices or orientation for a ring indicates which side of the ring is the interior of the polygon. The neighborhood to the right of an observer walking along the ring in vertex order is the neighborhood inside the polygon. Vertices of rings defining holes in polygons are in a counterclockwise direction. Vertices for a single, ringed polygon are, therefore, always in clockwise order. The rings of a polygon are referred to as its parts.

The following C-like[18] data structure completely describes polygons (ESRI Shapefile Technical Description 1998, p.12):

[18] C programming language. See (Weiss 1995) for an excellent tutorial on C.

```
Polygon
{
        Double[4] Box               // Bounding Box
        Integer NumParts            // Number of rings in polygon
        Integer NumPoints           // Total Number of Points for all rings
        Integer[NumParts] Parts     // Index to First Point in Part
        Point[NumPoints] Points     // Points for All Parts
        }
```

Figure 8: Polygon Data Structure

For readers unfamiliar with C-like syntax, square brackets [] indicate an array of elements, with a number of elements specified inside square brackets. Double refers to double precision.[19] An example of the polygon shape is given in Figure 9. This polygon has a hole and eight vertices. Note that the rings are closed (the first and last vertex are the same), and ring segments do not intersect each other.

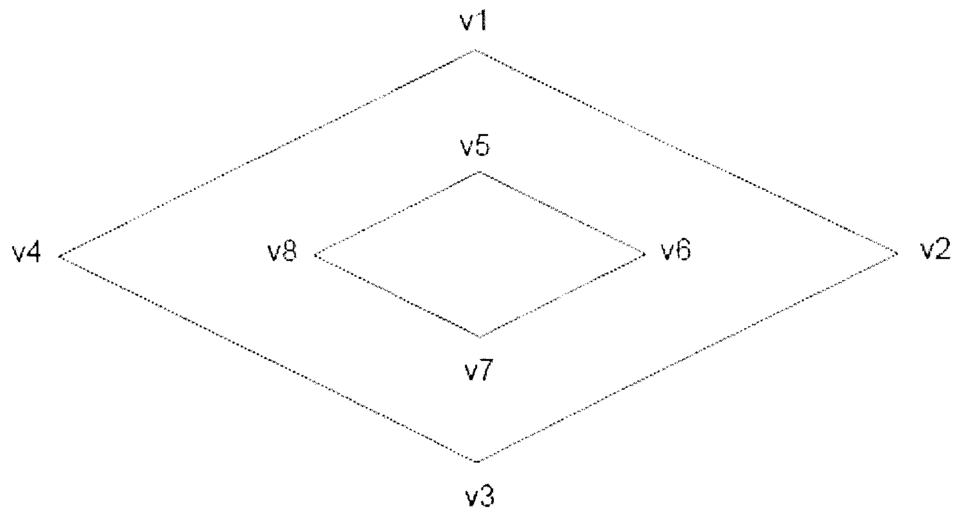


Figure 9: Example of Polygon shape
Source: (ESRI Shapefile Technical Description 1998)

[19] See http://www.psc.edu/general/software/packages/ieee/ieee.html

Let us see (Figure 10) how the data structure in Figure 8 can be filled based on the example polygon presented in Figure 9. Parts and Points are arrays. Points contain all the vertices for two rings, and Parts point to the first vertex in the ring. The first element of Parts contains the index of the first vertex of the first ring (v1), while the second element of Parts contains the index of the first vertex of the second ring (v5).

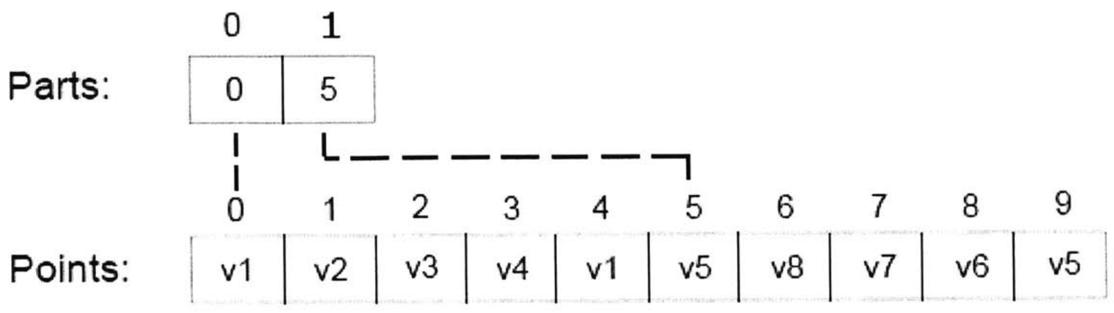


Figure 10: Polygon Parts and Points Arrays
Source: (ESRI Shapefile Technical Description 1998)

The organization of the index file consists of a 100-byte header followed by the contiguous list of records, with their index offsets and content sizes. This organization is depicted in Figure 11. Finally, the organization of dBASE file follows the standard set for .dbf files.[20]

While the shapefile standard is popular in the industry, there is a plethora of other standards that are not directly compatible with each other. Since users' demand for spatial data interoperability is strong, ESRI conducts extensive research in this area. Currently, the Institute makes it easy to translate the shapefile data format into other popular data formats. However, their research (ArcGIS: Engineered for Interoperability 2006) indicates that the

[20] See http://www.dbase.com/ for more details.

answer to the interoperability lies in the adaptation of Web Services and Service Oriented Architecture (SOA).

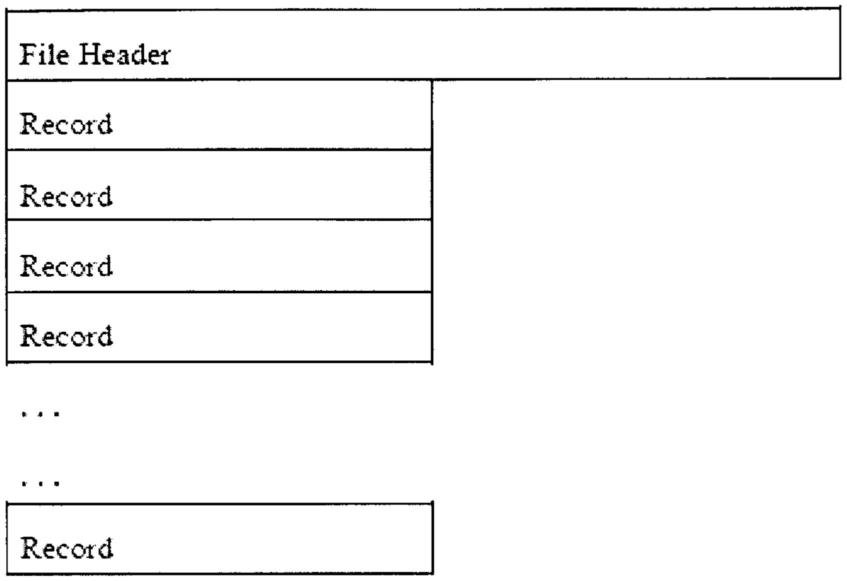


Figure 11: Index File Organization
Source: (ESRI Shapefile Technical Description 1998)

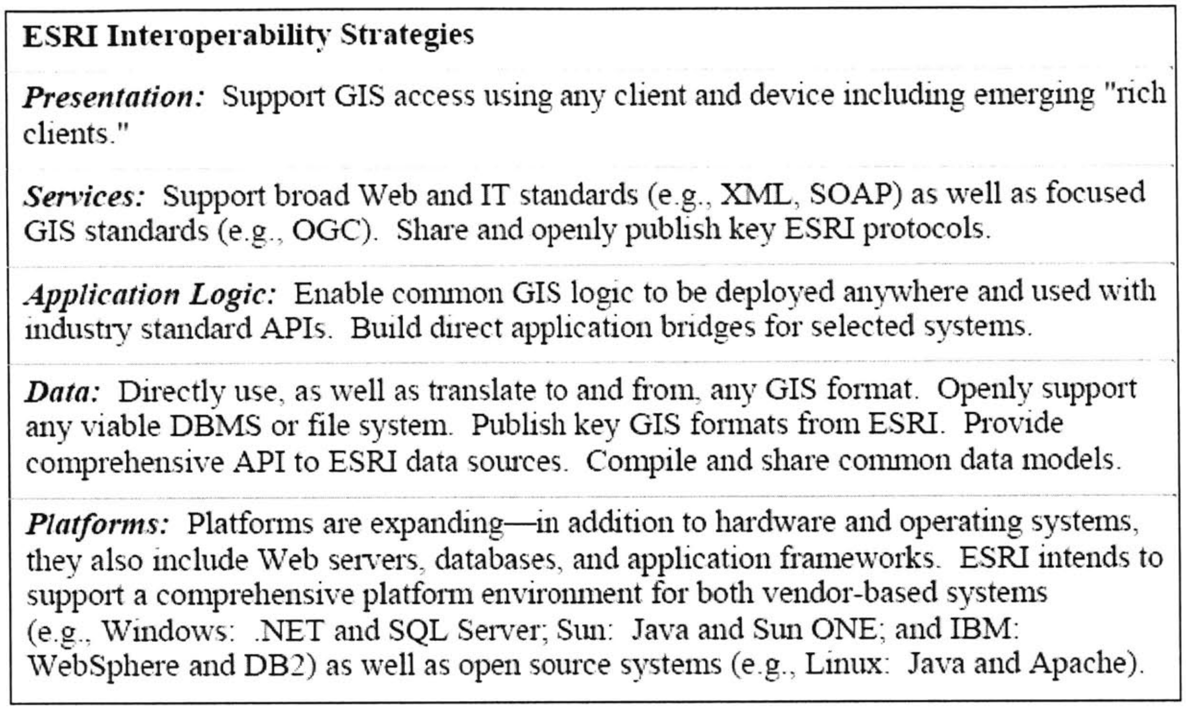
The Institute is advancing the federated databases technology as applied to the GIS domain. The following three interoperability initiatives would allow ESRI to be at the forefront of the spatial data interoperability advancements:

Metadata: describes documents located at various GIS portals worldwide. GIS portals contain catalogs that can gather data from participating sites and make them available in the SOA fashion.

XML and Web Services: ESRI has created its own ArcXML standard and makes geospatial data available via Web Services. See also **Service Oriented Architecture (SOA)** for a short introduction to SOA concepts.

Location services: ESRI took part in the OpenGIS initiative in the Location Services area. See **Open Geospatial Consortium** in this thesis for more information on other OpenGIS interoperability initiatives.

ESRI interoperability strategies are summarized in Figure 12.



ESRI Interoperability Strategies
Presentation: Support GIS access using any client and device including emerging "rich clients."
Services: Support broad Web and IT standards (e.g., XML, SOAP) as well as focused GIS standards (e.g., OGC). Share and openly publish key ESRI protocols.
Application Logic: Enable common GIS logic to be deployed anywhere and used with industry standard APIs. Build direct application bridges for selected systems.
Data: Directly use, as well as translate to and from, any GIS format. Openly support any viable DBMS or file system. Publish key GIS formats from ESRI. Provide comprehensive API to ESRI data sources. Compile and share common data models.
Platforms: Platforms are expanding—in addition to hardware and operating systems, they also include Web servers, databases, and application frameworks. ESRI intends to support a comprehensive platform environment for both vendor-based systems (e.g., Windows: .NET and SQL Server; Sun: Java and Sun ONE; and IBM: WebSphere and DB2) as well as open source systems (e.g., Linux: Java and Apache).

Figure 12: ESRI Interoperability Strategies
Source: (ArcGIS: Engineered for Interoperability 2006, p. 13)

Critique of ESRI Shapefile Standard

ESRI Shapefile standard has been devised at the time when GIS systems were mostly local and lack interconnection capabilities other than through standard network protocols. Therefore, a common data format standard was a necessity to provide any interoperability level between disparate GIS systems. According to Bishr classification (see Figure 2), sharing data via ESRI shapefiles corresponds to the Level 2 interoperability, which is low by modern standards.

The advantages of shapefiles are their portability, open interface, low footprint, and quick edit capabilities without the need for sophisticated and expensive software packages. However, shapefiles do not support topological data. Moreover, shapefiles have not been designed to interoperate on higher interoperability levels (DBMS, Data Model, and Application Semantics). ESRI has made strides to do just that, and their interoperability research (using federated databases, SOA, and web services) aims in the right direction. However, it still means that multiple existing standards have to be translated into each other and that resulting interoperability levels will cater to the lowest common denominator. Finally, these strategies will at most provide the interoperability at the Data Model level and will not be capable at accomplishing the true semantic interoperability that this thesis advocates.

National Imagery Transmission Format

National Imagery Transmission Format (NITF) standard was developed in 1994 with the purpose of providing interoperability in spatial data among various government agencies. Government agencies as well as military often used incompatible standards that prevented efficient sharing of information. Each organization had to develop translation software to be able to understand various formats. This process was laborious and inefficient.

The NITF standard (MIL-STD-2500B1998) provides for efficient transmission and storage of electronic imagery among Department of Defense (DOD) and Intelligence Community

(IC). The standard allows packaging of disparate data (images, text, and stand-alone graphics) into a single file along with the transmission, decoding, and archiving of such information. The standard assumes that receiving organizations will translate the NITF file into their internal standards. The NITF file consumers do not need to translate all the data from the file; organizations can pick and choose data that they need. Obviously, an organization can adopt NITF as its internal standard, so that all the needs for translation will be eliminated. The NITF functional architecture is presented below:

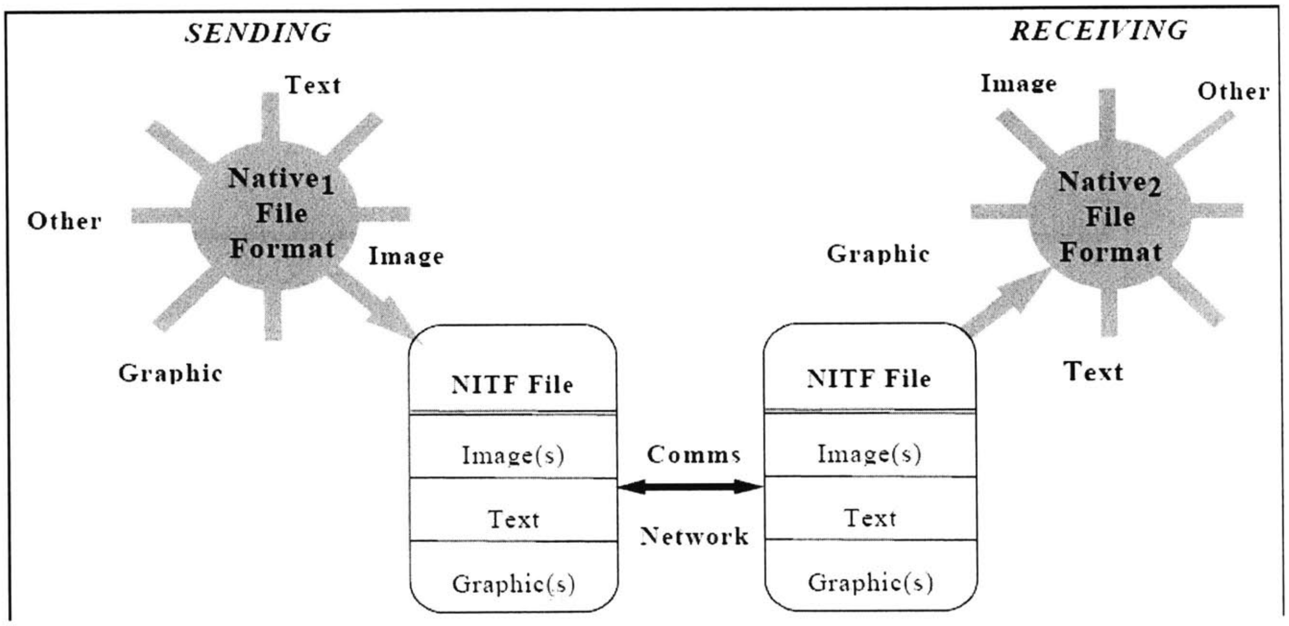


Figure 13: NITF Functional Architecture
Source : (MIL-STD-2500B1998)

NITF presents images in the Common Coordinate System (CCS) that is a virtual Cartesian coordinate system. The origin is located in the upper left corner, and points located in the CCS have positive coordinate values. An interesting feature of CCS is the introduction of Complexity Level that is composed of lower and right boundaries in the coordinate plane. The following figure exemplifies CCS and its features:

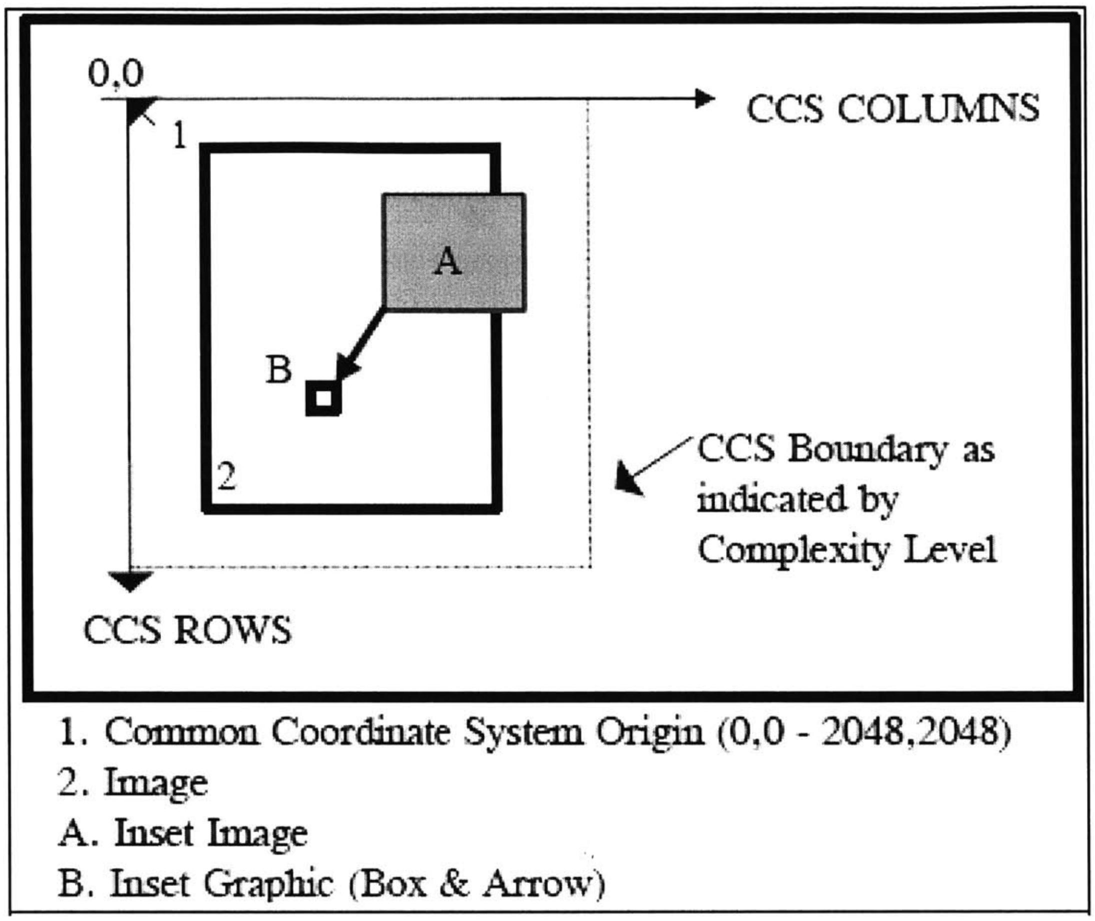

0,0
CCS COLUMNS
1
A
B
2
CCS Boundary as indicated by Complexity Level
CCS ROWS
1. Common Coordinate System Origin (0,0 - 2048,2048)
2. Image
A. Inset Image
B. Inset Graphic (Box & Arrow)

Figure 14: Common Coordinate System (CCS)
Source: (MIL-STD-2500B1998)

Limitations of CCS become apparent when an image must be rotated or otherwise geometrically transformed. The standard then requires an addition of empty and padded blocks to make sure that the resulting image in CCS remains rectangular with borders parallel to coordinate axis. The following figure illustrates the point:

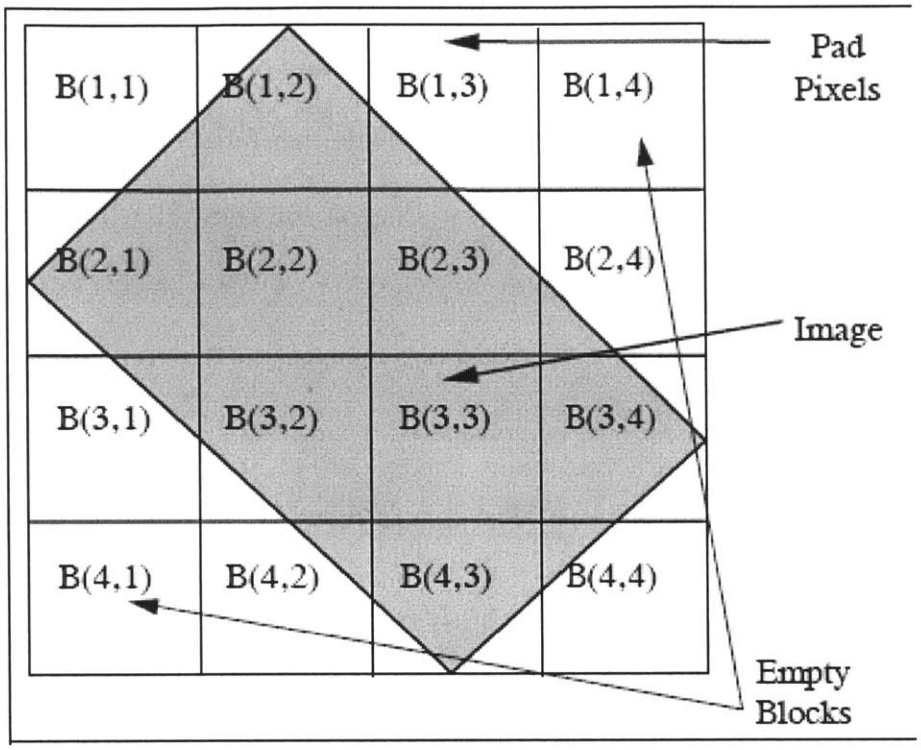


Figure 15: Image Rotation in CCS
Source: (MIL-STD-2500B1998)

NITF Critique

A disadvantage of NITF standard consists of the requirement to package all the existing information into a single file. While this may look convenient on the surface, disparate internal image standards adopted in various organizations at DoD and IC necessitated the creation of a complex standard that attempts to integrate various requirements from all stakeholders. Organizations have to build complicated converters to translate their internal data into NITF standard to distribute data externally and, conversely, build additional translators to handle NITF

The approach advocated in this thesis (using the M-compliant spatial data representation) calls for the Service-Oriented Architecture in the distribution and analysis of spatial data.

Instead of imposing the burden of translation on both consumers and producers of NITF files, the M-compliant representation asks producers only to provide two functions, f and g, that help consumers interpret data files unambiguously (see Chapter 6). These functions can be accessed remotely in accordance to SOA principles, as DCOM or CORBA objects as well as Web Services. The latter approach is preferred in situations when spatial data is distributed to 3rd parties, and network security issues must be considered. Using SOA principles provides the benefits of easier extensibility and deployment. For example, NITF standard is designed to work with 2-dimensional data; an extension into 3 dimensions will be problematic. In addition, the notion of time as the fourth dimension is often required to analyze temporal systems. NITF standard treats time statically, while the temporal analysis can be performed on M-compliant spatial data simply by employing remotely-accessed models.

3-D Standards

Similar to 2-D standards, the 3-D standards also proliferate. For example, these standards play an important role in Computer-Aided Design (CAD). There is a multitude of 3D CAD standards. One of the better known ones is ISO 1101:2004[21] that specifies geometrical tolerancing (tolerances of form, location, orientation, and run-out. In addition to the field of mechanical engineering, 3D standards become important in GIS, which is the subject of the next section.

Google Earth and KLM Standard

[21] http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=1147

One of the first movers with sizable marketing clout is Google with its Google Earth product[22]. The basic version of this product is available free of charge, but in order to obtain sophisticated features found in GIS software packages, one needs to consider Google Earth Enterprise.[23] This solution can provide spatial data from Google Earth or interface with organization's own spatial dataset. Google Earth can interoperate with other existing GIS systems. If an organization has invested into its own datasets, Google Earth Fusion (Google Earth Enterprise n.d.) integrates the datasets into Google Earth Enterprise. The support for interoperability is substantial, with both popular raster and vector data formats supported. It is interesting to note that 3-D shapefiles (.shp) are also supported. Google Earth supports a client-server computing paradigm with its Google Earth Server[24] and Google Earth Enterprise Client (EC).[25] The external data formats are converted into KML for the internal use.

Google Earth Industry Applications

Geospatial data is used extensively in many industries. The following industries, including commercial, non-profit, and governmental organizations, can benefit from the adoption of geospatial technology (Google Earth Enterprise n.d.)[26]:

Table 1: 3-D Geospatial Data Usage

Industry	Usage of Geospatial Data
Commercial Real Estate	Site analysis Presentations Teamwork

[22] http://earth.google.com/
[23] http://earth.google.com/earth_enterprise.html
[24] http://earth.google.com/earth_server.html
[25] http://earth.google.com/earth_ec.html
[26] http://earth.google.com/industries.html

Residential Real Estate	Obtaining market properties Finding homes Customer presentations
Architecture/Engineering	Site location feasibility studies Land use impact analysis Customer presentations
Media	Visual displays of information Key data, such as information on cities, road networks, relevant statistics Visual data overlays
State and Local Governments	Emergency planning and first response Zoning and land use management Economic development, marketing Access to data by citizens
Defense/Intelligence	Mission planning, training, and simulation Distribution of aerial imagery to geographically dispersed personnel Field assets and personnel visualization
Insurance	Risk identification by using RIMS[27] overlays Risk management
Federal Government and Non-Governmental Organizations	Environment monitoring and planning Space research

[27] Risk and Insurance Management Society (RIMS), see http://www.rims.org/

	Geological/Land management Visualization of economic and political conditions
Homeland Security	Vulnerability assessment Critical infrastructure protection Familiarization with sites by first responders Search for spatial patterns in surveillance data
Asset Tracking	Integration with RFID Integration with GPS
Security Systems	Getting location details

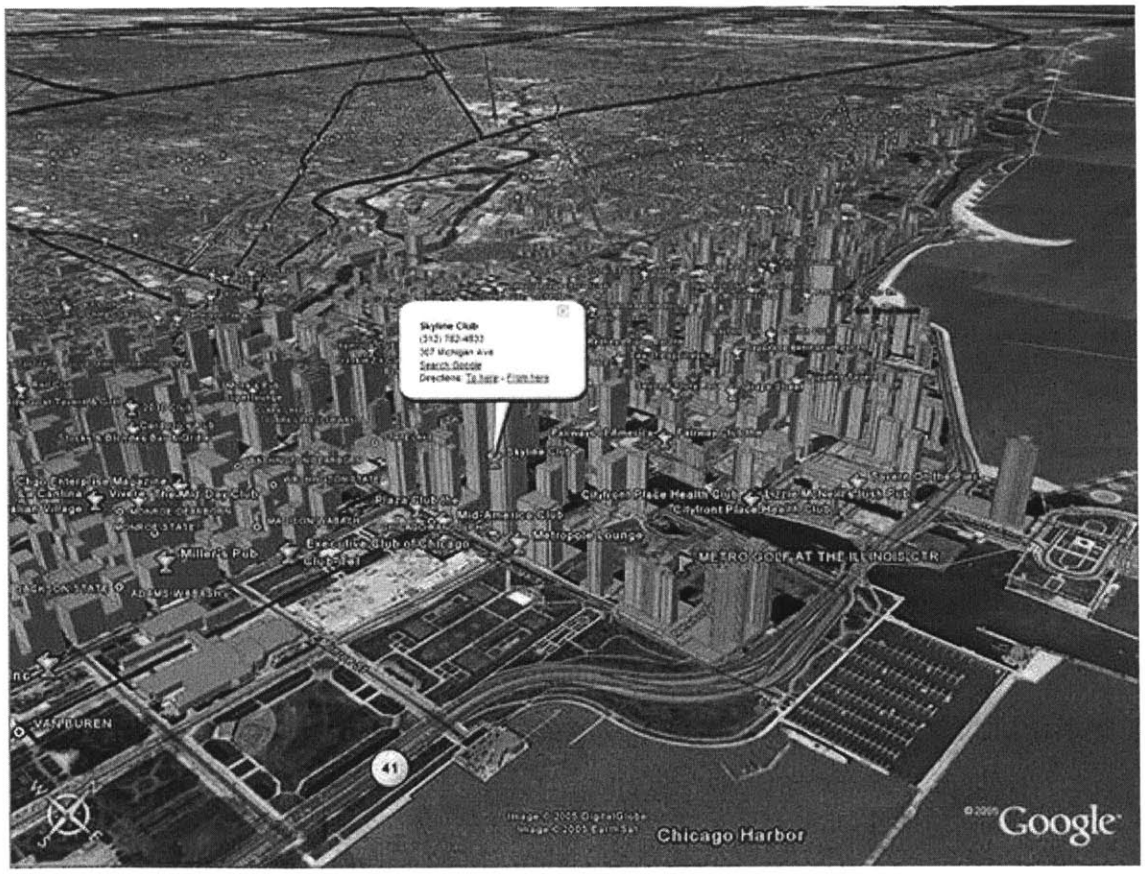


Figure 16: Google Earth Geospatial Data Example
Source: Google

KML Integration

Google Earth uses Keyhole Markup language (KML) as its external spatial data format. KML is XML-based and has a schema. Using KML, organizations can import geospatial data into Google Earth and, for instance, overlay it on top of the spatial data provided by Google. A sample KML code is presented in Figure 18. This code places a tethered placemark in a predetermined location. As seen here, KML has a tag-based structure, and Google Earth acts as a browser of KML files (Google Earth KML 2.0 2006).

Besides an ability to situate placemarks, KML has capabilities to manipulate the following 2-dimensional geometric shapes (Google Earth KML 2.0 2006):

- Points: can be visually represented on the Earth's surface by either icons or labels, or both and positioned at different altitudes.
- Lines: can be positioned at various altitudes similar to points. The support for polylines is included.
- Polygons: can be created in Google Earth, in 2- or 3-D and can be of solid form or with inner boundaries. As a result, complex 3-D shapes can be rendered at various altitudes.

The appearance of these geometric shapes can be manipulated using the following techniques (Google Earth KML 2.0 2006): defining coordinates, extruding to make 3-D shapes, and grouping into collections. Defining coordinates technique is used by entering coordinates and an elevation above the seal level. Extruding works by positioning an element at the specified position and then using the <extrude> tag. Grouping into collection is done using <MultiGeometry> tag and is used for organization purposes.

In order to enhance the software package's displaying capabilities, image overlays are also supported (Google Earth KML 2.0 2006). Using this technique, it is possible, for example, to overlay a 3-D blueprint of a construction site over the location site where the construction is planned to occur. Image overlays can be divided into two parts: ground overlays and screen overlays. Ground overlays are images that are static with respect to the Earth's surface whereas screen overlays are fixed to the computer screen and are independent of the underlying geography.

Moreover, KML allows one to define one's own schema if the default schema does not satisfy all needs. The schema requires the following elements (Google Earth KML 2.0 2006):

- Parent element: a base KML schema for inheritance. Uses <parent> tag.
- Schema name: uses <name> tag.
- Field declarations: define individual elements.

The following example in Figure 17 illustrates the concept of defining a new schema:

```
<Schema>
    <name>States</name>
    <parent>Placemark</parent>
    <SimpleField>
        <name>FIPS</name>
        <type>wstring</type>
    </SimpleField>
    <SimpleField>
        <name>STATE</name>
        <type>wstring</type>
    </SimpleField>
</Schema>
```

Figure 17: KML Schema Definition
Source: Google Earth KML 2.0 2006

In this example two fields are declared: FIPS and STATE, both of type wstring (a UNICODE 16-bit string). The schema itself inherits from Placemark element and is, therefore, capable of working correctly with all the Placemark attributes.

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://earth.google.com/kml/2.0">
<Placemark>
<description>Tethered to the ground by a customizable tail
</description>
        <name>Tethethed placemark</name>
 <LookAt>
  <longitude>-122.0856375356631</longitude>
  <latitude>37.42240551227282</latitude>
  <range>305.8880792294568</range>
  <tilt>46.72425699662645</tilt>
  <heading>49.06133439171233</heading>
 </LookAt>
 <visibility>0</visibility>
 <Style>
        <IconStyle>
                <Icon>
                 <href>root://icons/palette-3.png</href>
                <x>96</x>
                <y>160</y>
                <w>32</w>
                <h>32</h>
                </Icon>
        </IconStyle>
 </Style>
 <Point>
         <extrude>1</extrude>
        <altitudeMode>relativeToGround</altitudeMode>
        <coordinates>-       122.0856204541786,37.42244015321688,50
        </coordinates>
 </Point>
</Placemark>
</kml>
```

Figure 18: KML Sample Code
Source: (Google Earth KML Tutorial n.d.)

KML Critique

KML is a powerful XML-type language used to extend the applicability and usefulness of Google Earth package. However, its disadvantage stems from the fact that it is based on the rigid schema that KML documents must conform to in order to work correctly in Google Earth. In addition, tag names are ambiguous because of the inherent vagueness of a human language. Therefore, KML is ill-suited to provide spatial data interoperability on the semantic level. M-Language, on the other hand, solves the ambiguity problem and is well positioned to offer spatial data interoperability solutions for disparate data sources.

Chapter 4 Technology Applications for Spatial Data Interoperability

The spatial data interoperability allows organizations to solve various problems in many fields of human endeavor. This chapter considers several applications that are important in today's society, but the list is by no means exhaustive.

Marketing

The applications of the spatial data interoperability to Marketing can be broadly divided into two camps:

- Using geospatial data to as means to predict customer's behavior, sales, plan and fine-tune a marketing campaign, conduct marketing research, etc.
- Applying spatial data in a more abstract manner to conduct both statistical and empirical analysis of marketing data.

The former camp is represented by the concept of spatial diffusion while the latter is described with an example of correspondence analysis.

Spatial Diffusion[28]

A new product introduction is a demanding task that all product companies have to face. Marketing plays an undoubtedly important role in this process. Because of the resource constraints, marketing budgets are limited, and performing market segmentation and targeted advertisement are instrumental in achieving competitive returns on marketing investment (Schuster et. al 2006).

[28] The description of spatial diffusion is based on the unpublished work of Ed Schuster et al. (2006).

It is widely known that product's market penetration is often a function of geography. Products may sell briskly in one place and be poor sellers in another. The geographic factor by itself encompasses various sub-factors, such as climate, income distribution, cultural norms, demographics, distances to retail outlets, etc. (Schuster et. al 2006). An area of study in Marketing Science that tasks itself with the study of the aforementioned factors is called *spatial diffusion*. Spatial diffusion is also concerned with building mathematical models that can predict the rate of customer acceptance of new products, and its dependence on time and various other factors.

The knowledge of spatial diffusion can greatly aid product managers and marketing professionals in understanding, predicting, and distribution of marketing resources for new products. The authors quote academic researchers on importance of understanding of spatial diffusion: "Managers who understand the geography of the processes by which consumers change their behaviors can be much more successful in launching new initiatives and can make much better use of their resources while doing so" (Allaway 2003).

The authors further note that even if a new product is successful technically and reasonably matches customer needs, there is still a significant chance of failure if near-real-time data concerning spatial diffusion is unavailable. At most, companies have point estimates of aggregate demand obtained from barcode scans, focus groups, and other marketing research activities. Moreover, these data sources do not include geographical information that is detailed enough to be an input in spatial diffusion mathematical modeling. With an advent of RFID technologies, detailed geographical information can be gathered so that mathematical modeling can be performed to aid in decision making.

To make matters more complicated, there is little support in current managerial information systems to visualize spatial diffusion and make connections to mathematical models in an open, network-centric way. Therefore, even if the raw data is there, its utilization and analysis are handicapped. Spatial diffusion has a significant potential to gain insights into the dynamics of the product penetration into various markets, especially for high-tech consumer goods. The advent of new tools to analyze spatial data (discussed in Chapter 2 and Chapter 3) gives more value to spatial diffusion because of the ease of spatial data analysis, both systematic (using statistical tools) and ad-hoc by viewing spatial data in browsers or purpose-built programs.

The diffusion has been made popular in the late 1960's by Frank Bass who constructed a model that predicted the number of consumers buying a product (Bass 1969). Bass' diffusion model did not make the use of spatial data, however. The introduction of spatial data on the individual level (tracking by an individual product unit) requires the use of the unique identification, afforded by EPC Global Network and RFID technologies. Having this capability would allow one to analyze the effect of geography on the rate of product adoption (Schuster et. al 2006).

While the concepts of spatial diffusion have been successfully applied to many fields, ranging from ecology to economy, the research on spatial diffusion in the area of new product introduction has been scarce (Schuster et. al 2006). Schuster et al. identifies 3 stages of product adaptation:

- Lead adaptors

- Neighborhood effects
- Consolidation of adoption

The integral characteristic of the 3 stages is the classic S-shape curve (see Figure 19).

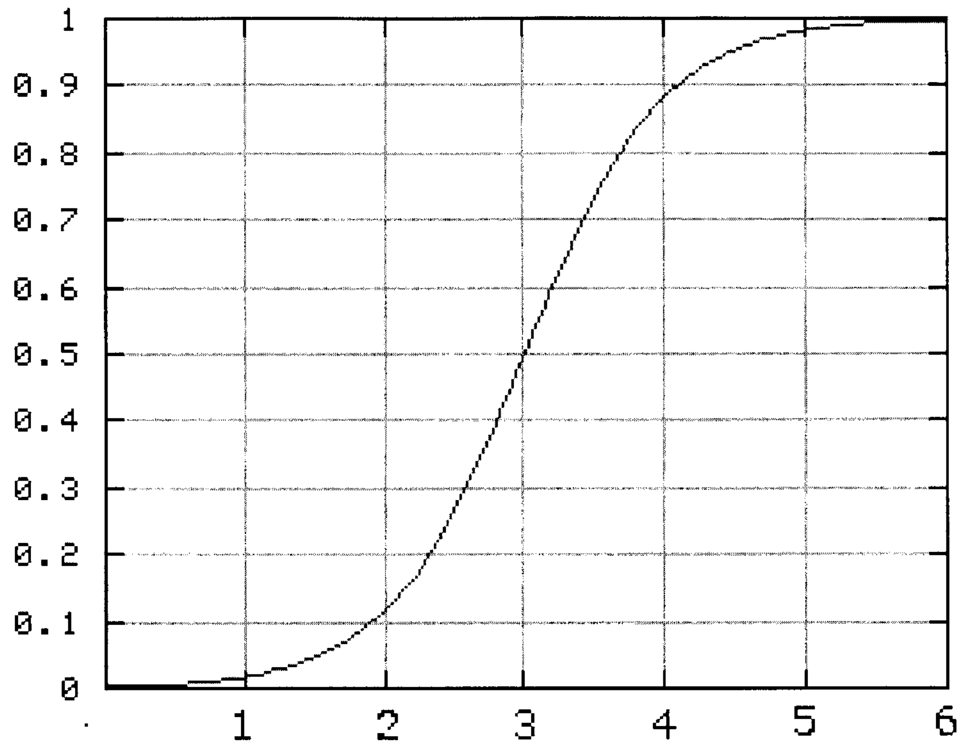


Figure 19: S-Curve

It should be noted that this model has been extended to the start-up marketing in the high-tech industry. Geoffrey Moore popularized the notion of the Technology Adoption Life Cycle (TALC) that consists of 5 stages (Moore 2002):

- Innovators
- Early Adopters
- Early Majority
- Late Majority
- Laggards

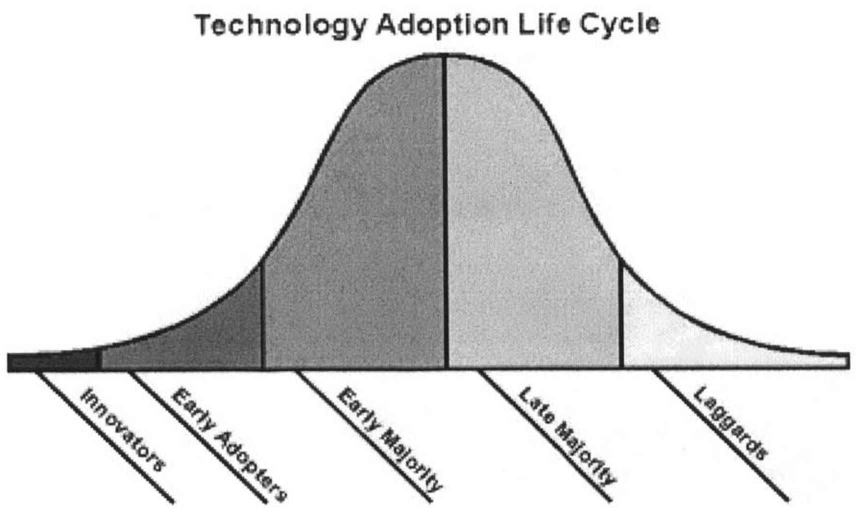


Figure 20: Technology Adoption Life Cycle
Source: (Moore 2002)

The S-curve and the bell-shape curve essentially represent the same phenomena; the S-curve can be obtained by integrating the Gaussian function over product's lifecycle time. Shuster's three stages can be mapped to Moore's five stages in the following manner:

Lead adopters → Innovators and Early Adopters

Neighborhood effects → Early Majority and leading part of Late Majority

Consolidation of adoption → trailing part of Late Majority and Laggards

Because these models specifically include time as the main independent variable, they are valuable in analyzing dynamics of customer behavior, which is an under-researched subject in marketing (Schuster 2006). Figure 21 illustrates the dynamics of spatial diffusion for a successful and failed product launch. Note that in the failed case the product never passes the stage 1, which is consistent with the Moore's view (2002) that a gap ("chasm") exists between Early Adopters and Early Majority. Those interested in techniques that allow one to "cross the chasm" should refer to Moore's work.

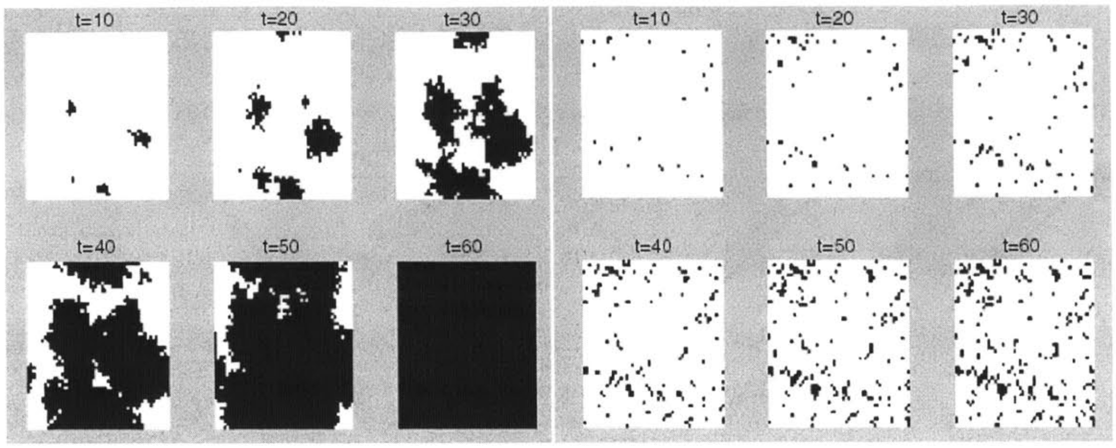


Figure 21: Illustration of Spatial Diffusion: Successful and Failed Product Launches
Source: (Garber et al. 2004)

Schuster et al. notes several implications that are of interest to marketing and supply chain managers. First, the rate of diffusion influences the overall pattern of product adaptation. Knowing the current pattern may help predicting future growth patterns. It is noteworthy that research on human communication patterns is applicable here. Allen (1971) has shown that human communication networks in Research and Development (R&D) organizations exhibit certain patterns based on the way knowledge is acquired and disseminated. In an R&D facility there exist so called knowledge gatekeepers through whom the knowledge in the organization is dispersed. These gatekeepers form clusters where the knowledge is rapidly disseminated. Since during the new product introduction, word-of-mouth marketing plays a large role (Schuster et al. 2006), understanding the concept of clustering in human communication is important.

Moreover, some research indicates that the probability of success of a new product launch can be estimated using spatial diffusion at stage 1 (Schuster et al. 2006). According to that research, spatial diffusion can improve the demand forecasting compared to a traditional time-series approach.

Finally, spatial diffusion can be applied to optimize spending on advertisement. Having real-time geospatial data and patterns of buying customers transforms into decisions to increase advertisement where there is trouble in cluster formations or decrease it when clusters become self-sustained because of the word-of-mouth phenomenon.

Correspondence Analysis

In marketing, there is a need to interpret relationships and find statistical correlations with data in rectangular format, represented graphically as a matrix. Correspondence analysis is a way to accomplish just that. In correspondence analysis each row and column receives a numerical score in corresponding units, facilitating the task to transform variables from an n-dimensional space to two dimensions for the ease of analysis and visual interpretation (Hoffman and Franke 1986).

Correspondence analysis can be useful in marketing research because the method employs the multivariate handling of data through a possibility of exhaustive pair-wise comparisons of data in rows and columns. In addition, not only pair-wise comparisons show that variables are related, but they also facilitate the calculation of coefficients of variation. The authors also note that correspondence analysis is flexible in incorporating marketing knowledge and performing rigorous mathematical analysis on categorical data.

The introduced standard for the spatial data representation fits well with correspondence analysis since function f can unambiguously translate n-dimensional space into two dimensions, while function g can perform the correspondence analysis on pairs of variables.

GIS Interoperability

Given the widespread installations of GIS systems, the importance of spatial data interoperability in these systems can hardly be overestimated. This area is, probably, the most likely beneficiary of improved ways to represent spatial data. Several issues concerning spatial data interoperability has been considered in **Issues in Spatial Data Interoperability** chapter. It is important to note, though, that a common interoperability deficiency in both existing systems and ongoing research is resulted from the lack of semantic support. In other words, these systems do not truly operate on the Semantic Application interoperability level (see Figure 2).

This thesis, on the other hand, proposes a spatial data interoperability standard that uses semantic concepts for superior interoperability results. But before introducing the standard, it is worthwhile to explore several important topics that are relevant to the geospatial data interoperability.

Spatial Data Representations in Information Systems

First, let us take a look at the ways spatial data can be represented in an information system. According to Laurini and Thompson (1992), there are two ways to accomplish this goal:

- Table or matrices

- Maps

Tables or matrices have no connotation of spatial position while maps by definition do. In tables the order of rows and columns may not be important, but it is so for maps. The authors introduce the term "map model" as an alternative way of representing spatial data. Maps in this view are not a final but rather an intermediate form of displaying reality, with inherent limitations caused by cartographic conventions, map projections, etc. Because of these limitations, the authors choose to operate with the term "data model." Data models constitute a comprehensive suite of tools, algorithms, data structures, and rules for organizing and presenting spatial data. In addition, spatial data systems have to deal with several aspects of data representation:

- Two or three dimensions
- Planar or non-planar situations
- Continuous or discrete referencing
- Isotropic or anisotropic conditions
- Forms and distances that change over time
- Measurements of qualitative or quantitative properties of space
- Smooth or rough objects

Map Projections

Because the Earth has a curved shape (a geodic, to be precise), representations of geographic fixtures on the flat surface are inherently imprecise. Humans have been trying to solve this problem since the dawn of map making, but various planar Earth representations (called projections) exhibit trade-offs in their attempts to represent one or more features accurately at the expense of others. These features are (Map projection 2006):

- Area
- Shape
- Direction
- Bearing
- Distance
- Scale

Planar maps cannot accurately represent all these features using the same projection.

There is a number of map projections in use nowadays, and they differ by which features are preserved. Map projections can be classified as follows (Map projection 2006):

- Projections by surface
    - Cylindrical: Meridians are represented by equidistant vertical lines while parallels are mapped to equidistant horizontal lines. In most cases the resulting projection is a finite rectangle.
    - Pseudocylindrical: Only central meridian and all parallels are straight lines while the rest are curved based on a certain mathematical function.
    - Conical: same as cylindrical, but the surface to map to is a cone.
    - Pseudoconical
    - Azimuthal: distances from a central point are preserved
- Projections by preservation of a metric property
    - Conformal: angles are preserved
    - Equal-area: area is preserved
    - Equidistant: distances from a standard point or line are preserved

    - Gnomonic: great circles[29] are represented by straight lines
    - Retroazimuthal: the bearing for the shortest route from point A to point B translates into the direction from A to B on the map.
- Compromises: no specific metric property is preserved, but the balance is maintained to make the projection "look right."

It is clear that in order to provide the spatial data interoperability, the support for all widely used projections is needed. This can be accomplished by utilizing a network of mathematical models, being run in a distributed manner according to SOA principles. The spatial data interoperability standard proposed in this thesis relies on such a network to achieve its goals.

Time in Spatial Data Information Systems

Time has been a subject of many philosophical teachings and is indeed an interesting subject. While philosophers have been fascinated with the concept of time since the very beginning, issues of representing time in informational systems are technical in nature (Langran et al. 1993).

The authors introduce the notion of cartographic time to represent spatiotemporal characteristics in a general way. Treating time as a fourth dimension allows one to achieve uniformity in treating both space and time as dimensions. However, certain temporal corollaries have to be added to time constructs to allow this. While in Einstein's Special Theory of Relativity space and time do interact, cartographic time behaves under the laws of Newtonian physics.

[29] Great circle: a circle on the surface of a sphere having the circumference equal to that of the sphere.

Since much of the information nowadays is stored in relational databases, the authors extensively treat problems of temporality in information systems. There are two ways to model temporality in information systems:

- Process modeling
- Time modeling

The former modeling allows one to construct analysis of trends, statistical correlations, change management, etc, while the latter one is optimized for setting temporal triggers that cause certain actions to occur if a time condition is met (sending a reminder letter if a customer has not responded to an offer, for instance). This work uses process modeling principles for its power in analyzing trends and statistical correlations.

The classic relational database theory does not support the temporality very well except for date-time stamp fields and transaction logs. Even semantic databases (Rishe 1992) treat date-time stamps as attributes. This is not a trivial issue to solve, and is a part of the proposed further research (see Chapter 8).

Network-Centric Warfare

The concept of the network-centric warfare (NCW) or network-centric operations (NCO) is a current chapter in the ongoing evolution in the military research of the developed nations, especially the United States. Simply put,

> The vision for NCW/NCO is to provide seamless access to timely information to all warfighters and decision-makers at every echelon in the military hierarchy. The goal is to provide ubiquitous connectivity throughout the organization, including individual infantry soldiers and ground vehicles, command centers, aircraft and naval vessels, and spacecraft. This enables all elements to share the information collected to be combined into a coherent, accurate picture of the battlefield, which is made available to all units. Every unit "sees" the sum of what all other units "see" and thus enjoy a greatly increased situation awareness. It is envisioned that having rapid access

to reliable situational awareness and a relevant common operational picture will result in faster strategic planning and more effective tactical decisions (Network-centric warfare 2006).

The architecture for NCW potentially includes three elements (Murdock 2002):

- Sensor grid
- Information grid (or Global Information Grid)
- Transaction grid

The Department of Defense (DoD) has put force a requirement that the Global Information Grid be used in conjunction with a technical implementation of NCW concepts. "The Global Information Grid (GIG) is defined as the globally interconnected, end-to-end set of information capabilities, associated processes, and personnel for collecting, processing, storing, disseminating, and managing information on demand to warfighters, policymakers, and support personnel" (Global Information Grid 2006). The GIG high-level architecture is depicted in Figure 22.

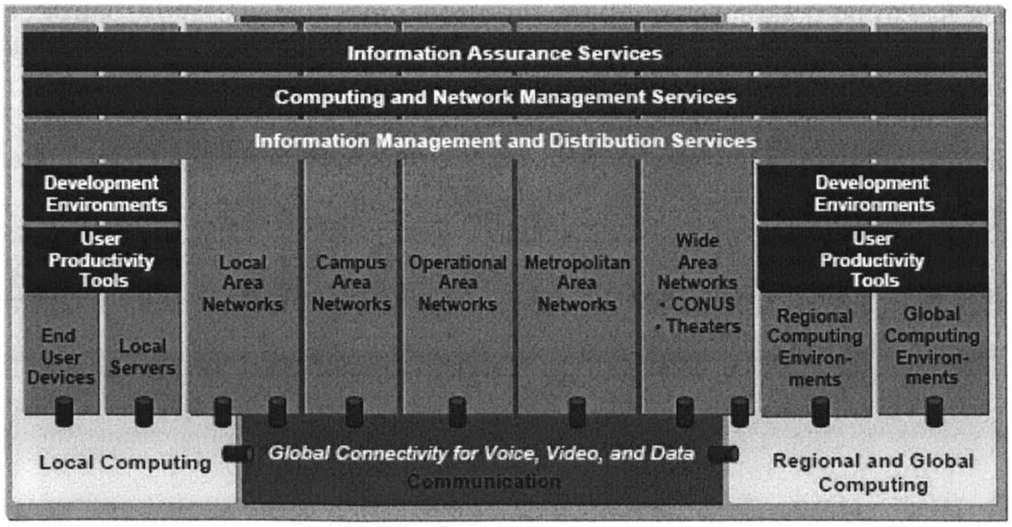

Information Assurance Services
Computing and Network Management Services
Information Management and Distribution Services
Development Environments
User Productivity Tools
End User Devices
Local Servers
Local Area Networks
Campus Area Networks
Operational Area Networks
Metropolitan Area Networks
Wide Area Networks • CONUS • Theaters
Development Environments
User Productivity Tools
Regional Computing Environ-ments
Global Computing Environ-ments
Local Computing
Global Connectivity for Voice, Video, and Data
Communication
Regional and Global Computing

Figure 22: GIG Architecture
Source: (Network Centric Warfare 2001)

The Intelligence Community (IC) is also an actor in NCW and GIG, and proactive, relentless, and real-time gathering of battlefield intelligence is a key to a successful implementation of the NCW vision. However, even now there are issues related to the interoperability of data gathered through the sensor grid. The DoD realizes that the interoperability of various components comprising the GIG is the key to its successful deployment (Network Centric Warfare 2001). Alberts et al. (2001) also agree that the interoperability of disparate and heterogeneous assets must be significantly improved in order for NCW to become operational. The authors note that DoD promoted a Joint Technical Architecture as a means to enhance the interoperability among DoD and IC.

Much of the intelligence collected from a battlefield is carried in the form of spatial data. Therefore, the interoperability of spatial data becomes especially important for the successful NCW operations. According to open sources[30], the integration of spatial data is of concern since it often has to be done manually by the personnel in Command and Control centers. Improved interoperability and integration of data, especially on the semantic level, would alleviate interoperability and integration problems and make NWC initiatives an operational reality. The proposed M Language-based standard for the spatial data integration is well suited for this problem because of its inherent capabilities to interoperate on the semantic level and its native support for SOA that plays an important role in the GIG.

[30] Presentation by Rudolf F. Peksens the Director, Strategic Pursuits, Command and Control Systems, Raytheon Company at "The M Language: Update and Applications" event hosted by the MIT Data Center.

Chapter 5 M-Compliant Standard – an Approach

The literature search and review of existing practices in data integration and interoperability has revealed that only the interoperability on the semantic level can ensure success in integrating and interoperating disparate and heterogeneous data sources. While the ongoing research pays attention to the topic of semantic interoperability, there are no working systems that truly support this approach. This chapter introduces the reader to the M Language – a general approach to providing data interoperability on the semantic level. The M Language is being developed at the MIT Data Center, and this chapter contains a short introduction into the center's activities. The spatial data interoperability, proposed in the thesis, is a natural extension to the M Language.

MIT Data Center

The Data Center at Massachusetts Institute of Technology[31] is an organization that aims to research ways for bringing together divergent and heterogeneous data sources to create synergies and make sense of the data. The Datacenter envisions a future where heterogeneous databases can be combined without *a priori* knowledge of their schemas. The volumes of information that every organization has amassed can be mined intelligently and joined together to create value. The disparate data sources can be fed into mathematical models that are themselves run in parallel and distributed manner, thus enabling decision makers to understand the meaning behind the data. Finally, the mathematical models can be combined into composite models to bring even more power to data processing and attain the goal of creating an intelligent information network.

[31] See www.mitdatacenter.org for the main page.

Although this vision is lofty, the path to it is feasible because the existing technologies can be leveraged to accomplish the goal in small increments. First, solutions to the short-term problems in data integration can be obtained, using existing technologies. The center also works in parallel on defining new protocols, interfaces and languages (M-Language) that allow organizations to achieve seamless data interoperability. In the longer term, advanced technologies, such as parallel computing, immersive visualization, and multi-modal user interfaces, will be utilized to provide means to manage and visualize information and algorithms. These approaches will always be measured against real-life business goals and use cases. In the end, the center will be able to create a foundation for intelligent information management and interoperable analytic modeling (Brock 2004).

Introduction into the M-Language

M-Language mechanics

This introduction is based on (Brock et al. 2006) work. The M-Language consists of two parts – words and rules. Words are regular human-understandable terms that can be found in any dictionary, with one important caveat. Human languages are imprecise. For example, the word "cell" in English means a biological cell, a prison cell, a fuel cell, among other meanings. In order to avoid the ambiguity that is a characteristic of any human language, words in the M-Language have qualifiers represented by a dot followed by a number. Continuing with the current example of "cell," this word in the M-Language is represented as follows:

cell.1

The M-Dictionary, which is an integral part of the M-Language, defines the word "cell.1" in the following way[32]:

> cell.1 -- (biology) the basic structural and functional unit of all organisms; they may exist as independent units of life (as in monads) or may form colonies or tissues as in higher plants and animals.

Similarly, other meaning for the word "cell" are supported by adding extension 2, 3, etc after the dot. This way the word "cell.x" where $x \in \mathbb{N}, x > 0$ uniquely defines the term. If multiple parties agree on the meaning of the word (and they agree by using the common M-Dictionary), communications amongst involved parties can become semantically unambiguous.

Besides storing the definition for a word, the M-Dictionary also contains auxiliary information that is nevertheless important in semantic interoperability and integration. The data is comprised of the following sections:

Word relations: synonyms, antonyms, types and parts. Synonyms and antonyms have the same meaning as in English language. Types refer to the generalizations or inheritance (for example, cell.1 is a *type of* living_thing.1. Parts are defined as words that constitute other, "composite" words. For instance, cell.1 is a *part of* organism.1 and, in turn, contains the following *parts*: cell_membrane.1, cell_nucleus.1, cell_organ.1, etc.

Data formats: this section defines formats, patterns, often in terms of regular expressions that help make sense of a given word. This is especially useful in machine-machine interactions where parsing of text files may be ambiguous. For example, a zip code in the

[32] Based on the version 0.03, see http://18.78.2.98/dictionary/index.php

United States contains 5 digits, while a postal code in Russia contains 6 digits. Such formats can be useful in dealing with localization issues when data sources are bound to different locales.

Language translations: simply a translation of a word in various human languages. Since words cell.1 and cell.2 may be translated differently from English into other languages, the localization of every definition helps in translation efforts.

The development of the M-Dictionary itself does not resemble the usual process through which dictionaries are created. Instead of relying on a small number of professional linguists, the creation and maintenance of the dictionary is opened to the public similar to the way Wikipedia operates[33]. There are safeguards, though, to make sure that malicious or spurious information is not entered in. For example, all users must register. In addition, the M-Dictionary has a concept of core words that are closed to editing by users with regular privileges.

So far, the concepts of words in the M-Language as well as the related topic of the M-Dictionary have been covered. In order for any language to be operational, there must be *rules* to connect words together so that the result can be unambiguously interpreted by communicating entities. The initial version of the M-Language contains three rules that are sufficient to cover most of the use cases: phrases, key-value pairs, and tables.

[33] See http://en.wikipedia.org/wiki/Wikipedia:About for a description of a collaborative way Wikipedia operates.

Simply put, in the M-Language "a phrase is a sequence of machine-understandable words representing a single idea" (Brock et al. 2006, p. 5). The syntax dictates the last word to be the root while other words are modifiers. For instance, the phrase "relational database management system" is represented in the M-language as:

relational.1_database.1_management_system.1

Here the root is the word system.1 whereas relational.1, database.1, management.1 are modifiers. The words are customarily connected into phrases by an underscore sign '_'. Phrases in the M-Language form a separate semantic unit for the ease of information processing by machines.

Key-value pairs (maps or associative arrays in computer science parlance) are pair of words with corresponding data. Reports, financial statements and bills are a good example of information presented in the key-value pair format. For example, the following is a key-value pair in the M-Language:

Name.1 – "John Doe"

Key-value pairs facilitate the data interoperability both internally and externally by making it easy to split and combine various forms and reports.

The rule for representing tables in the M-Language is different from other common ways of table representation. Here the table is represented as set of key-value pairs with identical keys and repeating patterns. The M-Language automatically recognizes the patterns and manipulates the data accordingly. The following is an example of a table in the M-language:

```
customer.1
        Social_security_number.1          –          “123-45-6789”
        name.1                           --        “John Doe”
customer.1
        Social_security_number.1          –          “111-11-1111”
        name.1                           --        “Jane Doe”
```

Here the repeating pattern is the set of customers with their social security numbers and names. Note that the indentation of a social security number and name key-value pairs is for readability only and does not affect the parsing.[34]

The M-Language presented here is in its initial formation and lacks rules for equations, mathematical models, and spatial data. This thesis proposes rules for the spatial data representation, and it is the subject of Chapter 6.

Applications of M-Language

The M-Language is designed with the interoperability in mind; therefore, solving the data interoperability problems is its natural application. The current business world is full of

[34] While most machine languages ignore white spaces and tabs in the source code, in some languages, notably Python, tabs have a syntactic meaning.

organizations with proprietary schemas and data models. The M-Language can act as an intermediary to translate one schema into another. The following example (see Figure 23) shows how data from one database is translated into the M-Language and then sent over the Internet to another database where it is translated to conform to B's schema.

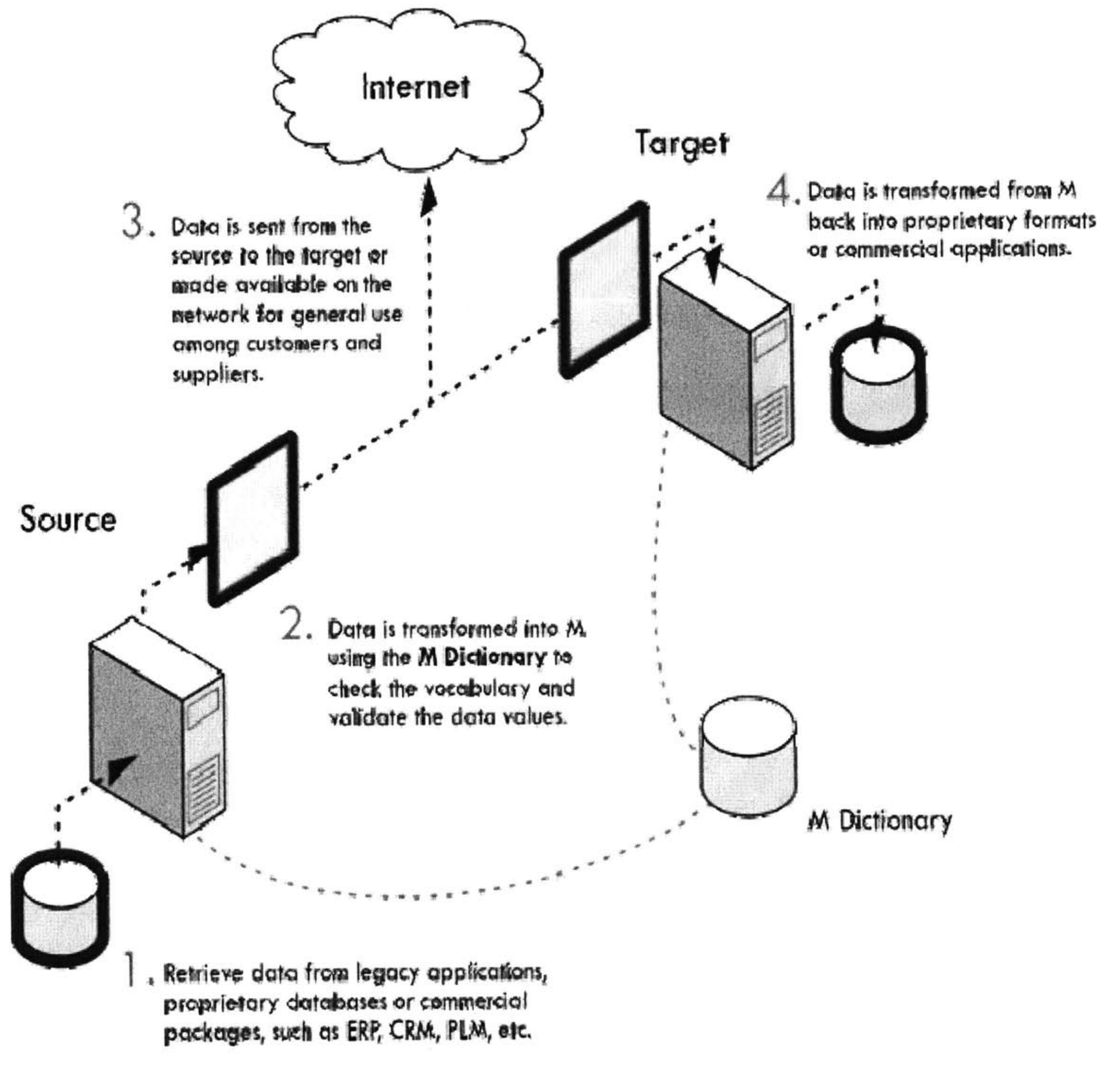

Internet
Target
4. Data is transformed from M back into proprietary formats or commercial applications.
3. Data is sent from the source to the target or made available on the network for general use among customers and suppliers.
Source
2. Data is transformed into M using the M Dictionary to check the vocabulary and validate the data values.
M Dictionary
1. Retrieve data from legacy applications, proprietary databases or commercial packages, such as ERP, CRM, PLM, etc.

Figure 23: Interoperability with M-Language
Source: (Brock et al. 2006)

Simply put, "the M-Language serves as a common transport between distributed, incompatible data systems" (Brock et al. 2006, p. 7). This way the translation happens at the network edge, thus avoiding multiple translation efforts by consumers and at the same time offloading the translation load from centralized servers. In the M-compliant world, providers expose their data sources as M-feeds, and interested parties can subscribe to these feeds statically and discover them dynamically when needed. These M-feeds can be plugged in to the network of models, creating an intelligent information grid where data is interoperable.

M-compliant feeds, forms, reports, etc. can be distributed in an XML format[35]. XML stands for Extensible Markup Language and is a de-facto standard of the data exchange on the Internet and other communication networks. Organizations usually define proprietary XML standards that include proprietary dictionaries and XML schemas, which dictate the structure of documents to exchange. There are many XML standards and schemas in use today, and they are mostly incompatible with each other. In order to provide data interoperability in the world of proprietary XML schemas, organizations build translation programs. This is clearly inefficient since the translation effort has to be made every time a schema ought to be translated into a new proprietary one.

An important part of the M-Language is that M-feeds do not require a rigid XML schema. Since the M-Language is based on the semantic approach, individual words, key-value pairs, and tables can be pulled from data sources and merged to produce a desirable document. The following demonstration, produced by the MIT Datacenter and available on-line[36], shows how two disparate data sources (customer orders) are translated into M-feeds and then integrated to produce a combined list of customers, extracted from these two orders.

[35] See www.w3c.org/XML for an introduction into XML
[36] http://18.78.2.98/Demo/DataMergeDemo.html, valid as of April 5, 2006

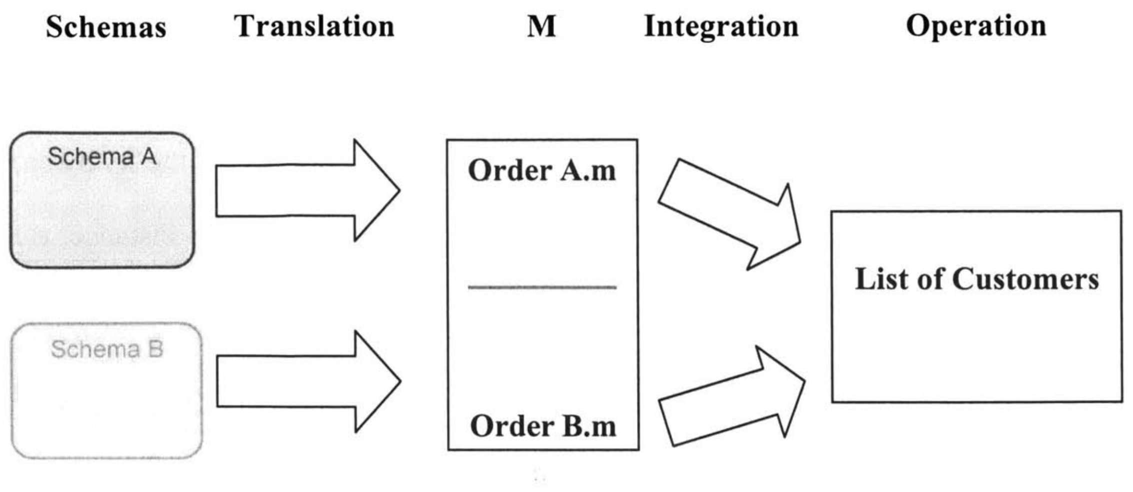


Figure 24: Demonstration of M-Language
Source: http://18.78.2.98/Demo/DataMergeDemo.html

Two customer orders, A and B, are XML files with incompatible schemas. XSLT[37] translations are applied to both schemas to produce Order A.m and Order B.m that are M-compliant with words from the M-Dictionary. Since the semantics of the orders are now similar, it is straightforward to derive a combined list of customers from both orders.

In order to make the M-Language easier to use, the MIT Data Center has developed the M Browser to create, edit, and manipulate data in the M-Language (Brock et al. 2006). Within the browser, the following useful features are available now:

Definition-based search: thanks to the M-Dictionary, words in M-Language have only one meaning. Therefore, performing a keyword search on such words yields search results that are unavailable to conventional search algorithms. For example, continuing with the previous example of "cell," a query for *type of* living_thing.1 will currently return the following

[37] http://www.w3.org/TR/xslt

results: being.1, biont.1, cell.1, life.1, organism.1. This affords interesting applications in browsing and searching through inheritance trees as an example.

Data validation and quality: the browser performs the data validation on the fly by ensuring that words belong to the M-Dictionary and that patterns are correct (for instance, a Social Security Number does have 9 digits). As a result, data quality and integrity improve within an organization and at communicating partners.

Language translation: Since words in the M-Language have exact semantics, their translation into various human languages is easy since translators do not have to look for a context in which the word is used. The M-Dictionary provides word translations into many languages.

Model integration: perhaps, the most important feature of the M-Language and its browser is the ability to interoperate not only data but also mathematical models across the network. Because the M-Language provides the semantic precision, inputs of one model can be connected to the outputs of another model semantically, without the need for a laborious translation. This enables the vision of distributed mathematical models combined together as the need arises to form complex systems. The user will be able to search for compatible models within the browser, combine them semantically, and start the simulation without importing the models first and resolving input and output discrepancies. The M-compliant spatial data representation, presented in this thesis, relies on the power of semantic modeling to perform much of its analytics.

In addition, the M-Language provides the following capabilities:

- Interoperable data and translation at the edge of computing systems

- Data aggregation
- A standard for sensors
- A standard for spatial data (subject of this thesis)
- Visualization of data

Chapter 6 M-Compliant Spatial Data Standard Internals

Introduction

Since there is a clear need for interoperability between various data sources, simple and efficient ways to communicate and operate spatial data are in order. The following proposal relies on the power of images to encode, transmit, decode, and analyze spatial data in a universal and unambiguous manner.

In order to make the spatial standard usable for a wide range of business, scientific, and technological applications, this thesis considers an n-dimensional space. The results can then be scaled down to the appropriate number of dimensions for any specific application or use case. The following section, **Problem Statement,** introduces the n-dimensional space mathematically

Problem Statement

In general, let us consider a Euclidean n-dimensional space where each point can be represented by an n-tuple of numbers $(x_1, x_2, \ldots, x_n)$. Restricting this discussion to real numbers, this n-tuple is a part of n-dimensional real coordinate space $\mathbb{R}^n$. If we assign a datum to each of these points (and this data point can also be an n-tuple), then we effectively obtain a relationship that can be represented by a topological mapping. If A and B are topological spaces, and $A \in \mathbb{R}^n$, $B \in \mathbb{R}^n$, then $f : A \to B$ is a function that maps A into B.

We also need to visit the concept of an inverse function. If f is a function with domain X and f is a bijection (one-to-one and onto), then f^{-1} is considered its inverse function if and only if for every $x \in X$ the following is true[38]:

$$f^{-1}(f(x)) = f(f^{-1}(x)) = x.$$

Furthermore, a point in the n-dimensional space may be a vector. In this case, we are dealing with a vector field. The following is the definition of a vector field[39]: If **S** is a subset of $\mathbb{R}^n$, then a vector field can be represented by a vector-valued function $V : S \rightarrow \mathbb{R}^n$ in standard Euclidean coordinates $(x_1, x_2, \ldots, x_n)$.

The following (Figure 25) depicts an n-dimensional vector field. Variables $x_1 \ldots x_n$ are independent whereas vectors, such as $(y \ldots y_m)$ are dependent variables.

[38] The definition is taken from http://en.wikipedia.org/wiki/Inverse_%28functions%29
[39] http://en.wikipedia.org/wiki/Vector_field

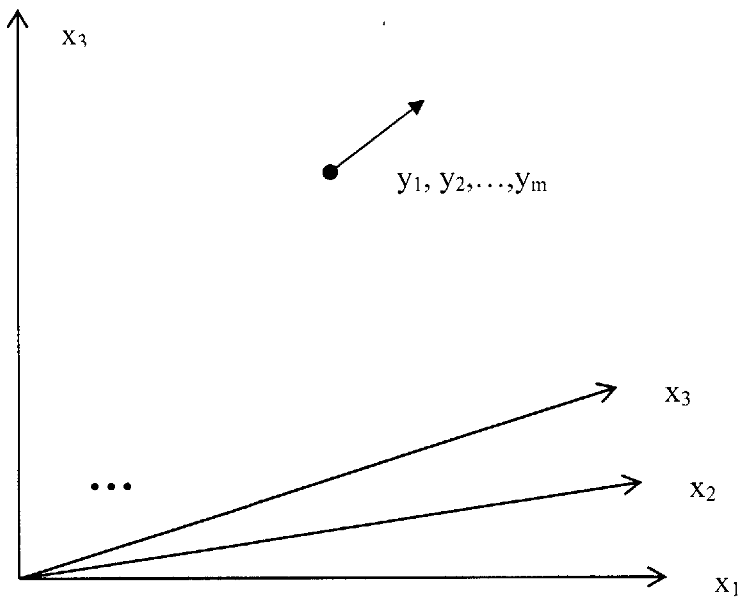


Figure 25: N-dimensional vector field

The following examples should help the reader to understand the value and power of an abstract mathematical concept of vector and scalar fields to concrete applications in business, science, and technology.

2-dimensional road map

This is an example of a 2-dimensinal space, or scalar field, that shows the road network in Houston area.

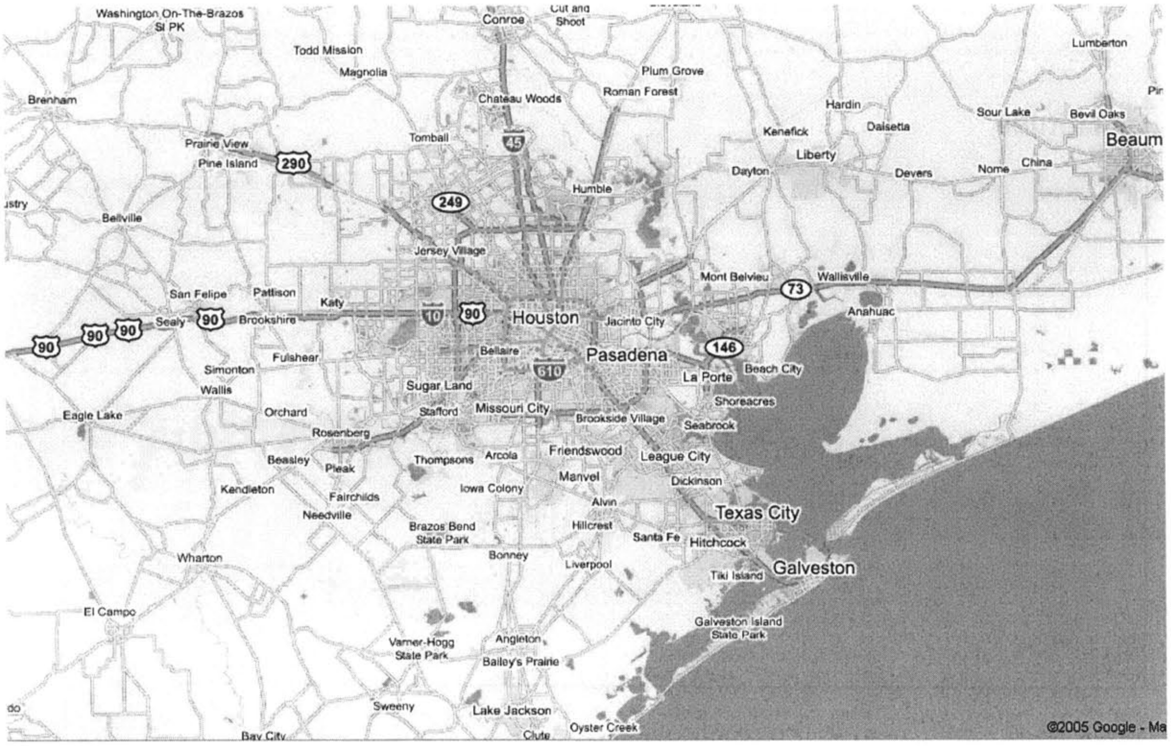


Figure 26: Houston Area Road Network

Note that if one desires to track dominant traffic flows during rush hours, as an example, then the scalar field would become a 2-dimensional vector field. A vector's magnitude here corresponds to the average traffic speed. A few hypothetical points of this vector field are shown in Figure 27. Note that in the morning rush hour the dominant traffic flows to Houston and the average speed decreases as the traffic approaches the downtown. Also note the vector pattern in the downtown showing a circular traffic direction in the city center. Here latitude and longitude are independent variable x = (latitude, longitude), and y = $(\mathbf{V}_1 \ldots \mathbf{V}_k)$ are dependent variables.

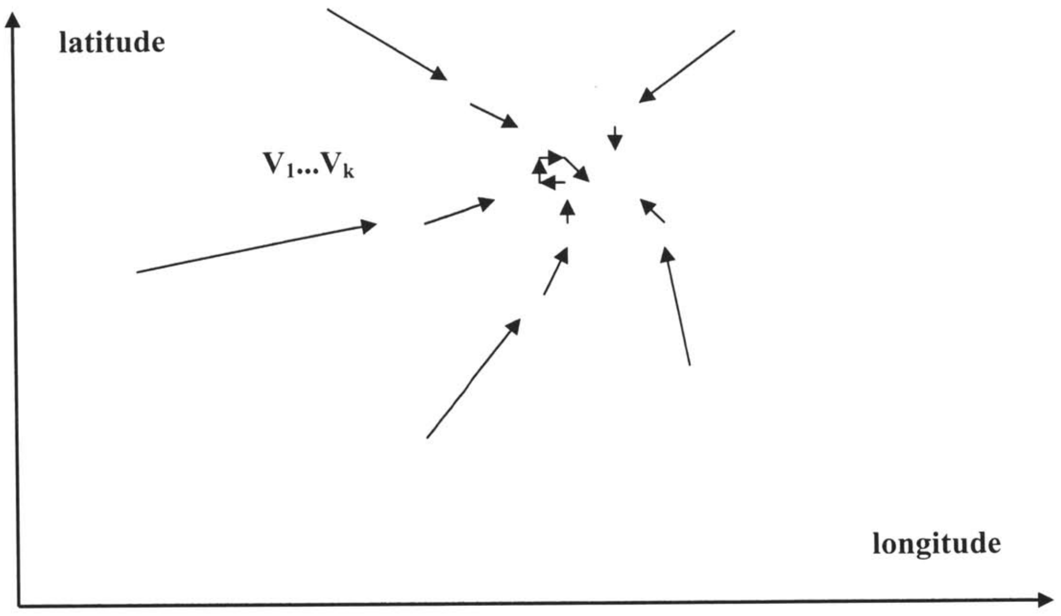


Figure 27: Traffic Flow in Houston Area in the Morning Rush Hour shown as a vector field

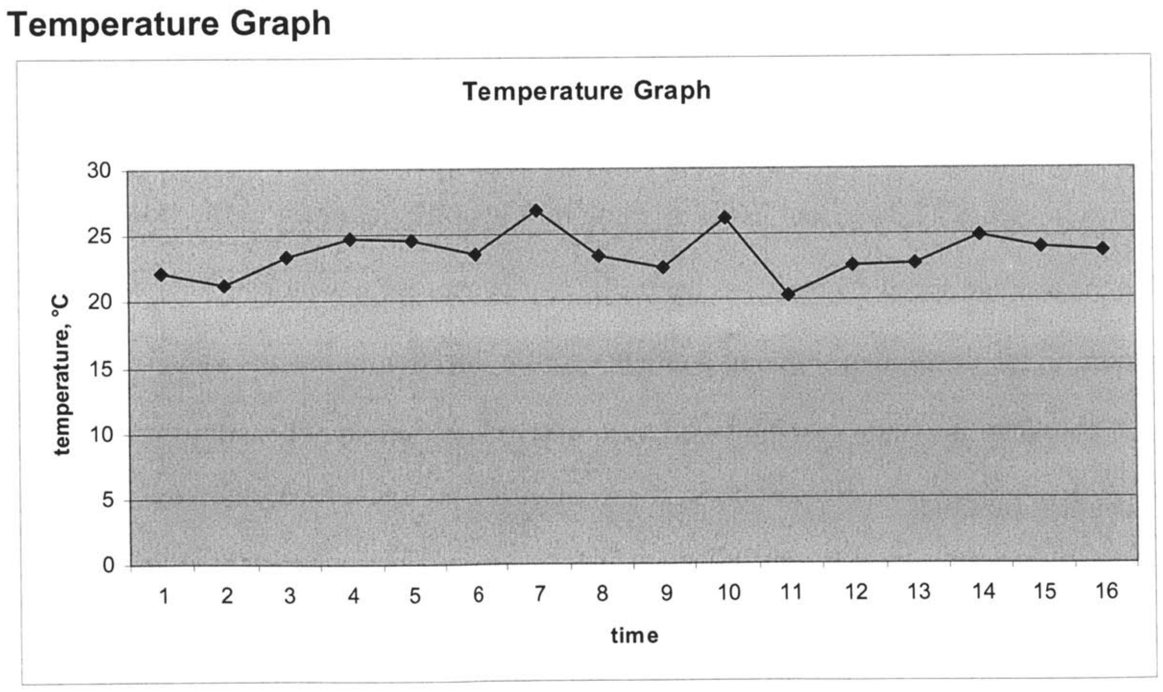


Figure 28: Temperature Graph

This example shows a graph of temperature vs. time. Time is an independent variable, whereas temperature is a dependent one. This graph can be represented by a 2-dimensional vector field where a vector's magnitude is equal to the value of temperature and direction is chosen so that all vectors are collinear. This vector field is depicted in Figure 29. Here time is an independent variable (x = (t)), while vectors $\mathbf{V}_1 \ldots \mathbf{V}_k$ are dependent variables y = ($\mathbf{V}_1 \ldots \mathbf{V}_k$).

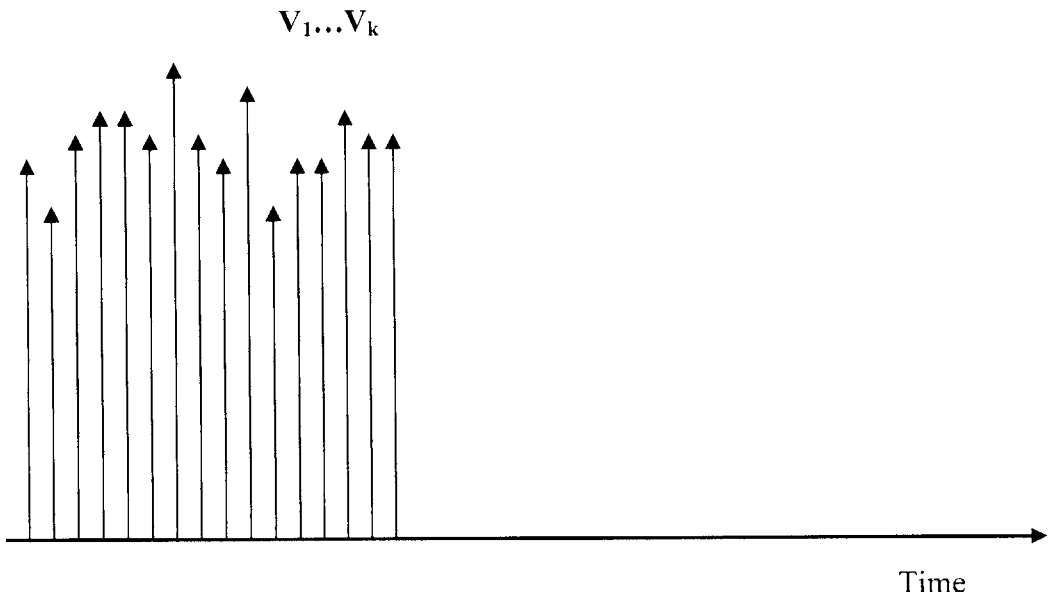


Figure 29: Vector Field as Representation of Temperature Graph

Magnetic Field

Magnetic field is a classic example of a vector field from an elementary course of physics. Figure 30 depicts the field lines and field vectors for a static magnetic field generated by a magnet with North and South poles. Here Cartesian coordinates (not shown in the picture) x = (x_1, x_2, x3) are independent variables, and vectors are dependent variables y = ($\mathbf{H}_1 \ldots \mathbf{H}_n$).

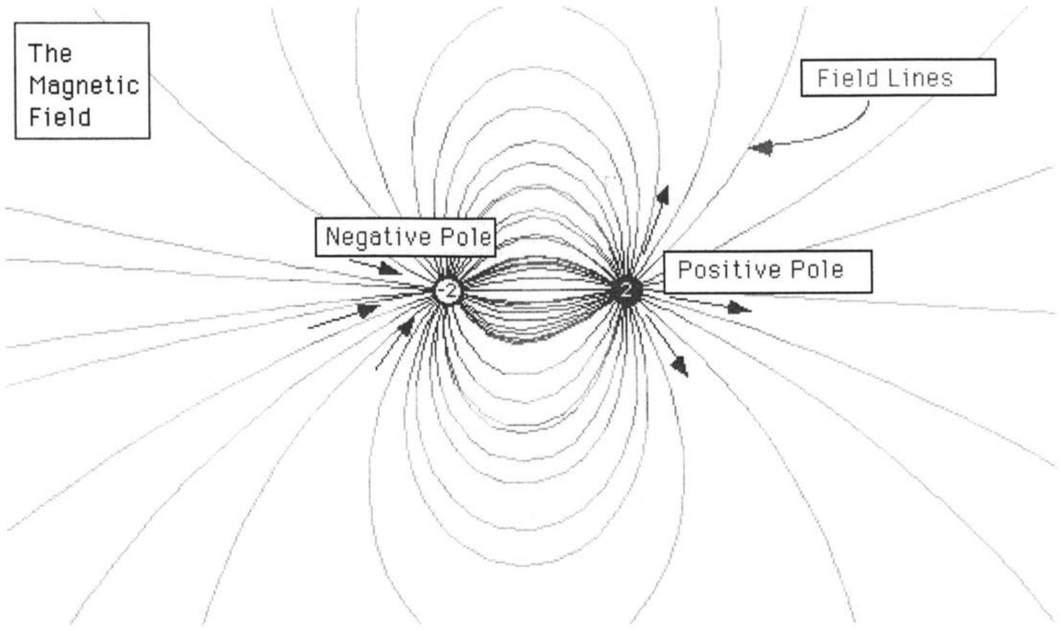


Figure 30: Magnetic Vector Field
Source: http://www.windows.ucar.edu/windows.html

Electromagnetic Induction

Also known from the elementary course of physics, an electromagnetic wave propagates itself in the space-time continuum by electric and magnetic fields Both of these fields vary in time, so both magnitudes and directions of vectors **E** (electric field) and **H** (magnetic field) are functions of time. The propagation of electromagnetic induction wave is depicted in Figure 31. This is an example of a 4-dimensional vector field, with 3 Euclidian dimensions and time. Here Cartesian coordinates and time t are independent variables $x = (x_1, x_2, x_3, t)$ whereas **E** and **H** are dependent ones ($y = (\mathbf{E},\mathbf{H})$).

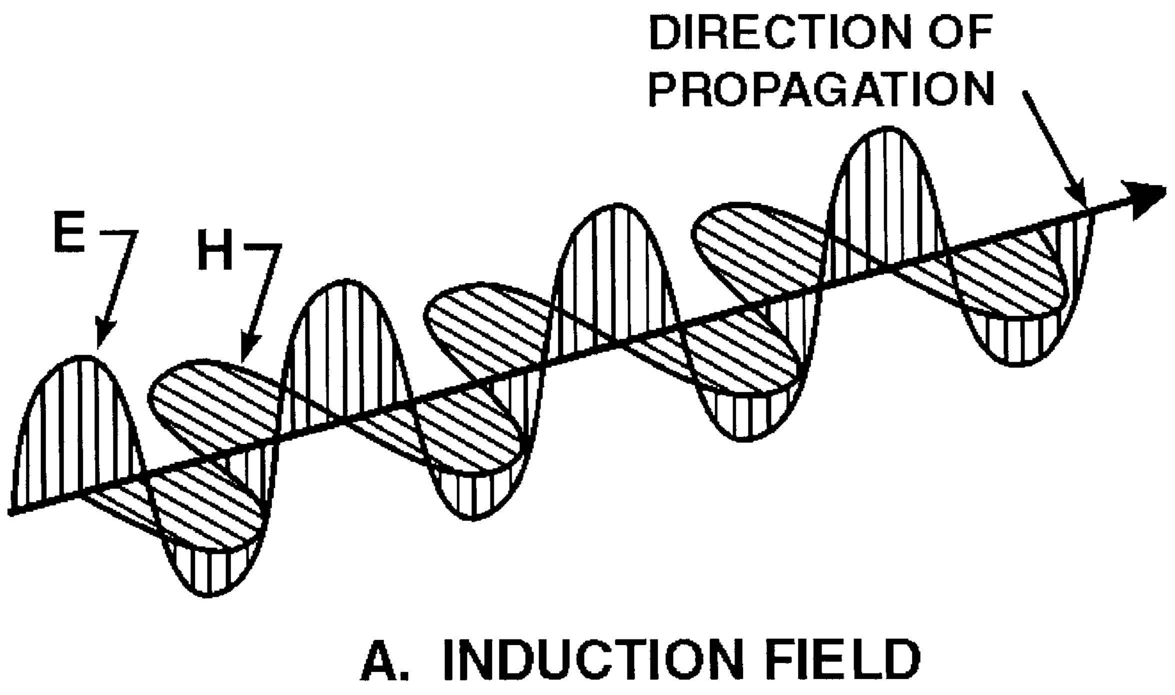


Figure 31: Electromagnetic Induction Field
Source: US Army

These examples illustrate various spatial data representations in many fields of human endeavor. The problem that needs to be solved is to create a standard that allows one to encode, transmit, decode, and interpret these spatial representations in a uniform and unambiguous manner. This thesis proposes the use of images for this purpose.

Introducing Images as Means to Communicate Spatial Data

While Literature Review and Existing Spatial Data Standards chapters presented a number of spatial data standards, none of them was capable of operating in an n-dimensional space and, thus, be suitable for a wide range of scientific, technological and business problems. This thesis proposes the idea of using images for spatial data communication and integration. There are three main reasons to choose images as the backbone of the proposed spatial data standard:

1) Images allow an efficient way to encode data in a binary format
2) Images are natural for spatial data visualization
3) Existing tools, technologies, and image standards can be leveraged for image manipulation

There are many graphic file formats, such as BMP, GIF, JPEG, PNG, PBM, PS, etc.[40] These formats can allow images to be uncompressed or provide either lossy or lossless compression. They are divided into two general categories: vector and raster graphics. Simply put, "vector data is comprised of lines and arcs, defined by beginning and end points, which meet at nodes. … Features are defined by their boundaries only and curved lines are represented as a series of connecting arcs." (Introduction to GIS n.d., vector-based GIS) Raster format, on the other hand, relies on a regular grid structure to display spatial data. "Each cell must be rectangular in shape, but not necessarily square. Each cell within this matrix contains location coordinates as well as an attribute value." (Introduction to GIS n.d., raster-based GIS). Vector and raster graphics have advantages and disadvantages over each other. I chose to use a raster graphics format because of the following advantages (Introduction to GIS n.d.):

- Computational speed
- Ease of update
- Use of continuous space
- Ease of integration

[40] For an extensive list of formats, see http://en.wikipedia.org/wiki/Graphics_file_format

Among raster graphics there is a number of graphics formats to choose from. We consider several representative ones here and then make a decision on which graphics format to choose for the spatial data standard.

Portable Bitmap Format (PBM): A black-and-white format with 1 bit per pixel. The advantage of the format is its simplicity. The main disadvantage is the scarcity of information that can be encoded in this format.

Portable Graymap Format (PGM): A gray scale format with 8 bits per pixel to render different shades of gray. While this format allows for better information density, the lack of color support prevents this format to become a top choice.

Portable Pixmap Format (PPM): A color format that allocates 24 bits per pixel, 8 bits for Red, Green, and Blue each. This format is a step forward because of the color support and more information packing capabilities (24 bits per pixel vs. 8 vs. 1). However, this format has several disadvantages. First, it is a text-based format and thus lacks the efficiency of binary formats. Second, it does not support bit transparency, thus limiting data visualization capabilities and decreasing the information density per pixel.

Portable Network Graphics Format (PNG): A format that allows a lossless compression and up to 48 bit depth. It supports bit transparency through an alpha channel. It is an open standard and is widely supported by major operating systems and graphics packages. In its 32-bit format it assigns 8 bits per each channel – alpha, red, green, and blue.

This thesis chooses PNG as its preferred format because of its binary efficiency, open architecture, support for bit transparency, colors and high information density.

While PNG format is compressed, the actual image is easier to manipulate in the information system in an uncompressed form to have an easier access to rows and columns. Let us look closer at the Alpha-Red-Green-Blue (ARGB) format that is important in compressed color formats, such as PNG, and uncompressed formats, such as BMP (used internally in Windows OS). In this context, an image is a n x m matrix where each value, i.e., the intersection of i^{th} row and j^{th} column, represents a color in Alpha-Red-Green-Blue (ARGB) format. This row and column intersection is also called a pixel. While the ARGB representation of a color is not unique, it is widely used and convenient from the information processing standpoint.

Historically, Red-Green-Blue color representation has been with us for a number of years since the ascent of color television. Recently, it has been standardized as sRGB by International Color Consortium (ICC) using the joint Microsoft and Hewlett-Packard proposal (Stokes et al. 1996). Red, green, and blue are primary colors, and mixing them can produce any color that is discernible by a human eye.

The ARGB format is an extension of the RGB concept by adding an alpha channel to three primary color channels. An alpha channel controls the image transparency on a pixel-by-pixel basis. A pixel transparency may vary from 0 to $2^{bitdepth}$-1 where 0 represents full transparency and $2^{bitdepth}$-1 represents total opaqueness (W3C PNG Specification 1996). The inclusion of alpha channel allows one to stack images and be able to see an image through another image. This image characteristic is very important since interlaying images allows one to visualize various dependencies and correlations of disparate data.

From n-dimensional vector field to images – introduction of function *f*

To quickly summarize previous points, there is a need to operate on an n-dimensional vector and scalar spaces in an efficient and interoperable manner, and that this thesis proposes the use of images to accomplish that. It is clear that a single image is not capable of accurately and precisely representing an arbitrary n-dimensional space. For example, if a satellite image is taken by the intelligence community, it is often useful to compare that image with others taken previously to determine the dynamics of enemy forces, changes in target characteristics, etc. Therefore, a set of k images is required to represent an n-dimensional space. Often, the order of images is important, and an k-tuple of images $\left(I_1, I_2, ..., I_k\right)$ must be introduced to solve the problem.

In order to map an n-dimensional space to a k-tuple of images, we introduce a function $f\left(x_1, x_2, ...x_n\right) = \left(y_1, y_2, ...y_m\right)$, where $m = 2k, k \geq 1, k \in \mathbb{N}$, that transforms n independent variables into an m-tuple. Note that when k=1, the m-tuple is reduced to the simplest case of a two-dimensional Cartesian plane that can be represented by a single image as in the preceding example. It follows then the number of images necessary to represent the n-dimensional space is equal to k and comprises a k-tuple $\left(I_1, I_2, ..., I_k\right)$.

From image to data – an introduction of function *g*

Once the k-tuple of images is available, it is necessary to map pixel data (colors and transparency) from the images (vectors or scalars) to vectors or scalars. This can be accomplished by a function

$$g(ARGB_{i,i+1}) = \begin{cases} \overrightarrow{R_j}, & \text{if mapping to a single value} \\ S, & \text{if mapping to a range of values} \end{cases}$$

, where $\overrightarrow{R_j}$ is a vector value and ***S*** is a geometric shape. S is used to show the range of vector values. S can be of a canonical shape if only direction of a vector differs or can be of irregular shape if both magnitude and direction vary. S can degenerate into a line segment if the resulting field is scalar or the vector's direction is constant. An example of S shape is shown in Figure 32 where S takes the form of a sphere segment. This represents a case when the vector's magnitude is constant but the direction varies within the spatial angle ά.

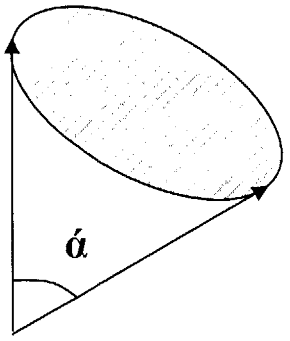


Figure 32: An example of shape S

R, G, and B bytes can be used for value's scalar part whereas the alpha channel is reserved for an indication of direction if transparency of images is not needed. In addition, if spatial data are only scalars, the alpha channel can also be used as an additional byte to represent scalar values.

The following simple example illustrates a concept of exchanging spatial data via an image file and the usage of functions *f* and *g*. This image (Figure 33) represents average surface temperatures in the United States during one day.

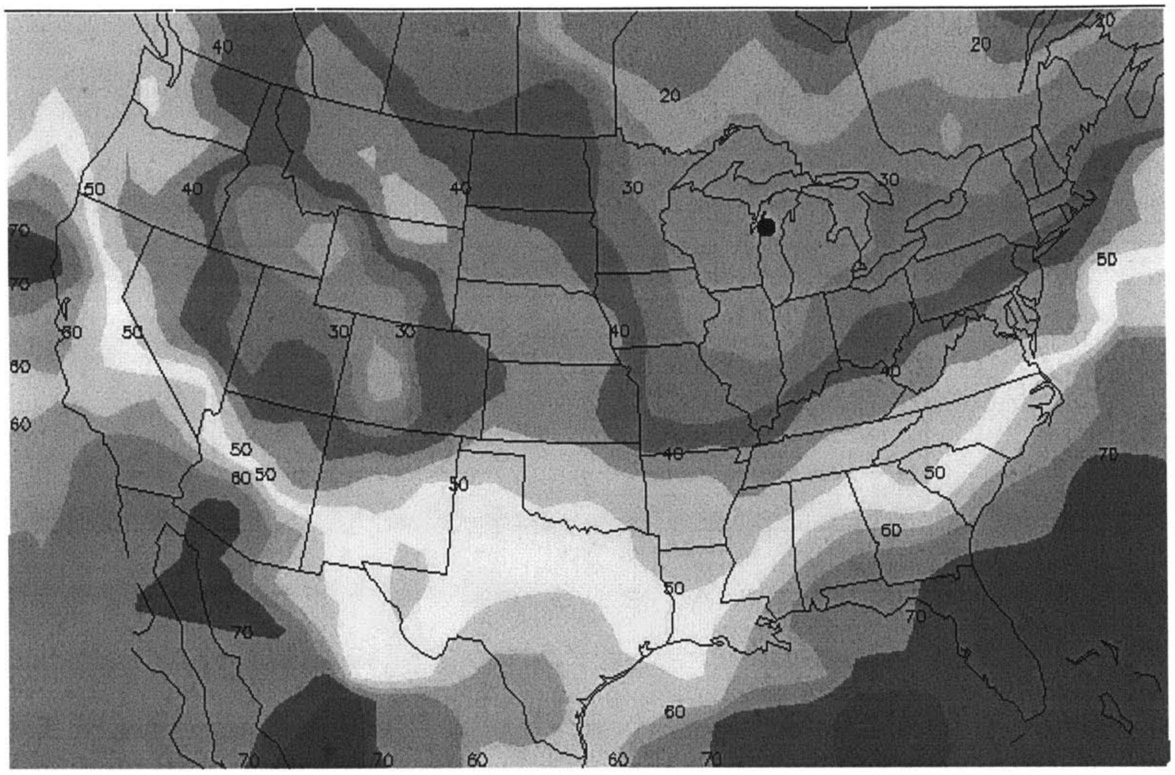


Figure 33: US Surafce Temperatures, Optimized
Source: Unisys

The image's width and height are 640 and 425 pixels, respectively. Therefore, the image can be represented by a 425 x 640 matrix where each 4-byte value is of ARGB type.

Figure 34 shows the steps that spatial data processing software would take to interpret the image.

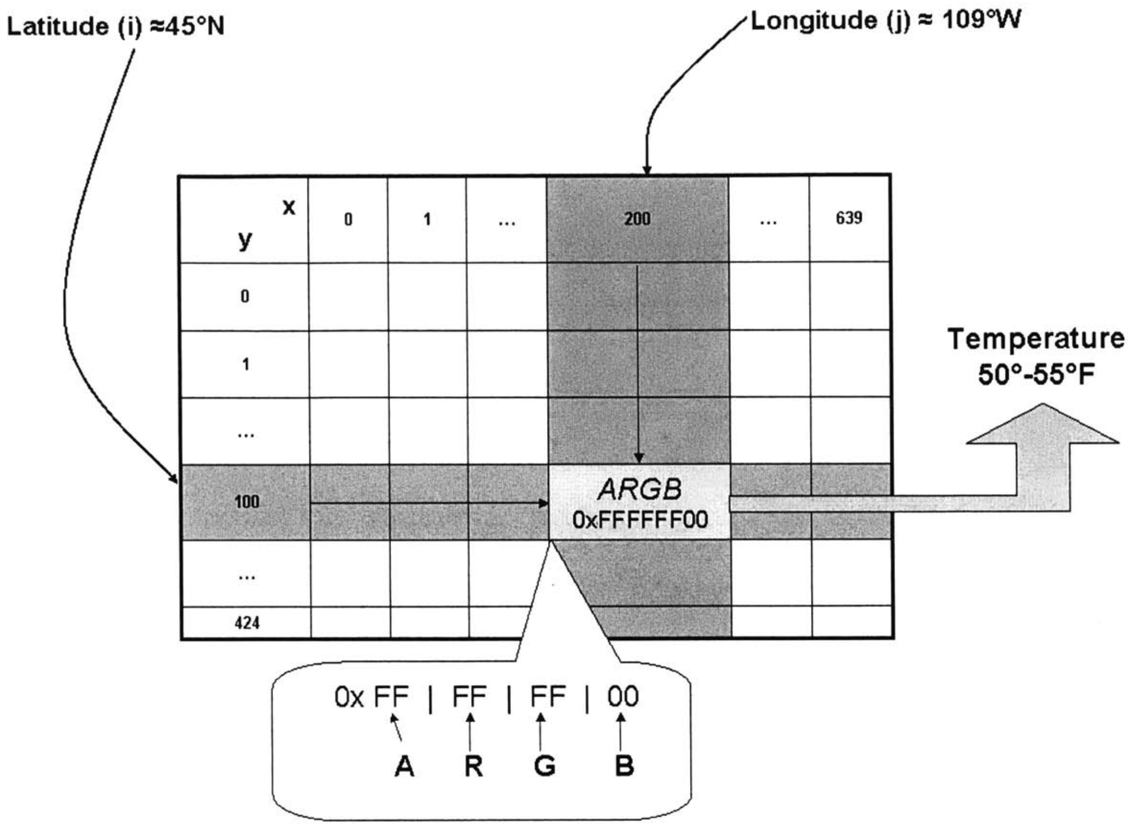


Figure 34: Image Interpretation

These steps include:

1) Translation of spatial coordinates (in this case, latitude (i) and longitude (j)) into pixel coordinates (x and y)
2) Conversion of x and y coordinates into pixel's color
3) Translation of pixel's color into temperature

Note that an ARGB color is customarily presented in a hexadecimal format.

Summary

An n-dimensional vector or scalar field can be represented via images. In order to map an n-dimensional field into the k-tuple of images, functions f and g work in concert. Function f maps a value from the vector or scalar field to the coordinates x and y of the virtual coordinate system of the m-th image from the k-tuple. These coordinates unambiguously determine the ARGB value of the corresponding pixel.Function g, in turn, maps the ARGB value into the vector or scalar in case of a single value or the shape S in case of a range of values.

Also, it is often necessary to find the original coordinate of an n-dimensional space from the coordinates of the Virtual Coordinate System. This can be accomplished by using the inverse function $\boldsymbol{f^{1}}$.

Since an ARGB value is 32 bits, it may not be enough to store unsigned integers larger than 2^{31} -1 or floating point numbers with high precision. To solve this problem, more than one image can be used to increase both range and precision of values. As a side note, this illustrates the need to use an ordered k-tuple of images rather than a set since ordering of images is important. Furthermore, images in the Virtual Coordinate System may not be aligned in the same manner. In this case a rotation is required similar to the rotation shown in Figure 15.

In a nutshell, n-dimensional spatial data can be represented by a k-tuple of images and two functions, f and g that specify the mapping of spatial coordinates into values.

Examples of M-files for spatial data

The following example is from the prototype **Temperature Contour Maps** presented in Chapter 7. There the temperature scalar field over the United States during various days is given via images. The temperatures are single values rather than ranges. See the prototype description for details. The partial M-file is presented here for convenience. It is also available in Figure 40 and in full form in Appendix A.

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Color-Temperature map -->
<spatial.1_data.1_descriptor.1>
      <image.2_function.7_url.1>
      http://localhost/TemperatureMapF/Service1.asmx?WSDL
      </image.2_function.7_url.1>

      <color.4_temperature.1_map.1>
            <color.4_temperature.1_pair.3>
                  <color.1>FFFF00FF</color.1>
                  <temperature.1>-12.5</temperature.1>
            </color.4_temperature.1_pair.3>
            <color.4_temperature.1_pair.3>
                  <color.1>FFBF00FF</color.1>
                  <temperature.1>-7.5</temperature.1>
            </color.4_temperature.1_pair.3>

                              ...

      </color.4_temperature.1_map.1>
      <Image.2>

      <url.1>http://localhost/temperatureMapF/other/sfc_con_temp_mod.gif
      </url.1>
            <width.1>640</width.1>
            <height.1>425</height.1>
            <begin.7_latitude.2>48.07</begin.7_latitude.2>
            <begin.7_longitude.1>-133.34</begin.7_longitude.1>
            <end.8_latitude.2>21.35</end.8_latitude.2>
            <end.8_longitude.1>-72.32</end.8_longitude.1>
            <date.5_time.6>03/04/2006:17.0.0.0</date.5_time.6>
            <map.1_projection.4>Mercator</map.1_projection.4>
      </Image.2>
                              ...

</spatial.1_data.1_descriptor.1>
```

Figure 35: An example of M-file for Temperature Scalar Field

This example is from the prototype **Vector Data** and shows the electrostatic vector field generated by a single proton. See **Vector Data** for more details.

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Color-Temperature map -->
<spatial.1_data.1_descriptor.1>
      <image.2_function.7_url.1>
            http://localhost/Vector_Data/Service1.asmx?WSDL
      </image.2_function.7_url.1>

      <!-- SI system is used -->
      <charge.13>1e-19</charge.13>
      <scale.10>
            <meter.4_pixel.1>0.01</meter.4_pixel.1>
      </scale.10>
      <color.4_electric_field.1_map.1>
            <color.4_electric_field.1_pair.3>
                  <color.1>FFFF00FF</color.1>

      <low.1_electric_field.1>0</low.1_electric_field.1>
                  <high.1_electric_field.1>1.0e-
20</high.1_electric_field.1>
            </color.4_electric_field.1_pair.3>
            <color.4_electric_field.1_pair.3>
                  <color.1>FFBF00FF</color.1>
                  <low.1_electric_field.1>1.0e-
20</low.1_electric_field.1>
                  <high.1_electric_field.1>1.0e-
19</high.1_electric_field.1>
            </color.4_electric_field.1_pair.3>

...

<color.4_electric_field.1_pair.3>
                  <color.1>FF990000</color.1>
                  <low.1_electric_field.1>1.0e-
2</low.1_electric_field.1>
                  <high.1_electric_field.1>1.0e-
1</high.1_electric_field.1>
            </color.4_electric_field.1_pair.3>
      </color.4_electric_field.1_map.1>
</spatial.1_data.1_descriptor.1>
```

Figure 36: M-file for Electrostatic Vector Field

Note that functions *f* and *g* are encapsulated in the web services, WSDL description of which is available under `<image.2_function.7_url.1>` tag. This is just an example of the power of mathematical models that this standard leverages in order to aid building the infrastructure for interoperable spatial data.

Data Interoperability

Representing spatial data as a k-tuple of images and two functions is a powerful concept because it lends itself well to the problem of data interoperability. Creating interoperable data standards is not an easy task because various organizations have vested interests in promoting their ways of handling data as a standard. The prototype **Integrated Circuits Yield** presented in Chapter 7 demonstrates the interoperability of disparate and heterogeneous data sources.

One of the important tasks in developing a standard is choosing what *not to control.* Over-control may fail because of the lack of authority (organizations may choose to ignore the standard) while under-control may deprive the system from it integration qualities, depriving integrated products from system qualities. For example, Motion Picture Experts Group (MPEG) standardized video stream decompression only, while letting implementers choose their own techniques for video stream encoding. As a result, organizations that collaborate on the standard can roll out proprietary data compression schemes, thus retaining a source of competitive advantage (for example, resolving an engineering trade-off of time to compress vs. compression ratio vs. video quality) (Maier and Rechtin 2002).

Data interoperability is an important facet of collaborative systems, and standards facilitate the interoperability as well as growth and usage of collaborative systems. It is advisable to let independent implementations to compete, thus preserving and enhancing the ecosystem that is developed around standards and collaborative systems. Maier and Rechtin (2002) state that standards are network goods, and thus benefit from their wider adoption. Therefore, the authors advocate the minimization of barriers to entry and setting clear conformance criteria.

The standard that was developed in the thesis takes cues from the book's view on standards in a sense that it attempts to accommodate as many potential stakeholders as possible. Since enterprises have already invested significant monetary and human resources into their often proprietary standards, a path to the spatial data interoperability should be easy to take. This issue is explored further in Chapter 1 when architecture choices are compared and contrasted.

Chapter 7 Spatial Data Standard Implementation

This chapter is concerned with the reference implementation of the M-compliant spatial data standard. As a part of the thesis research, architecture was developed and a prototype built to demonstrate some of the standard's capabilities.

Choice of Architecture

As pointed out in Chapter 6, a choice of architecture is driven by both technical and political considerations. A technically superior standard may not take hold because it did not appeal to the stakeholders. A case in point is a struggle of Betamax vs. VHS where Betamax, considered technically superior to VHS, lost to its adversary.[41] Therefore, potential architectures are considered from all points of view during the decision analysis.

Universal local implementation of M

The main idea in this approach is for every stakeholder to use M as its internal standard. Then the interoperability becomes automatic because of superior capabilities built in M. This is the easiest technical solution to develop but is virtually impossible to deploy in practice. The reasons for this dilemma are the legacy proprietary standards into which all companies have made major investments. Rewriting and redeploying a major spatial information system is a significant undertaking. Since standards are network goods (Maier and Rechtin 2002), there must be a critical mass of stakeholders adopting a standard for a company to have a business case to switch to the new standard. In the current business climate, it is virtually impossible to persuade all major spatial data stakeholders to switch all at once. Therefore, this approach is infeasible.

[41] See http://en.wikipedia.org/wiki/Betamax for more information on Betamax format and its competition with VHS

Client-Server Architecture

In this approach enterprises continue maintaining whatever internal standards they use, but for spatial data exchange, they send data from clients to centralized servers where data is translated into M. Then if an information recipient is able to accept M files, the data is sent in the M format to that client. Otherwise, the data is translated into the internal standard the recipient uses and then sent to the client. This process is visually depicted in Figure 37.

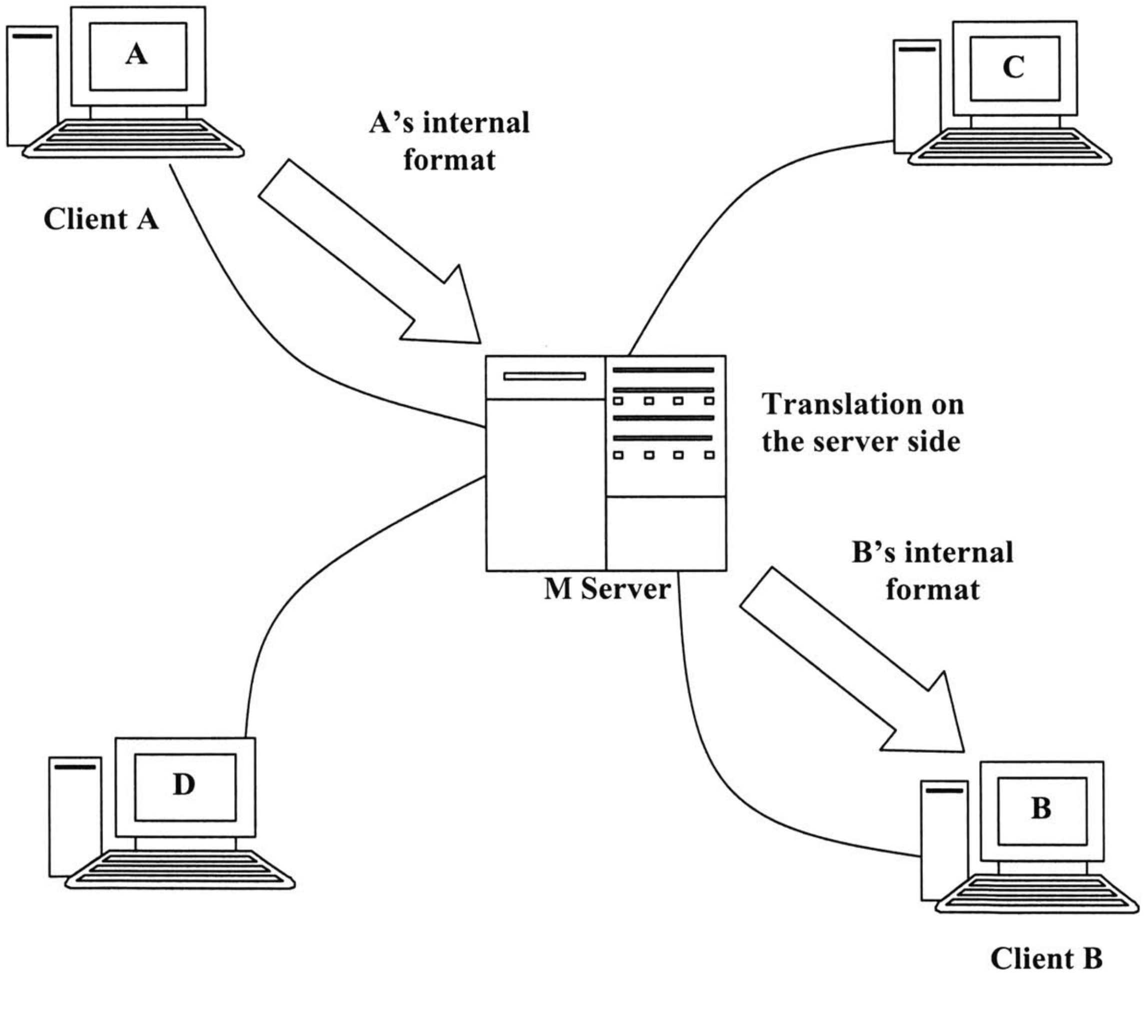


Figure 37: M Client-Server Architecture

This architecture is more efficient than that of converting A's format into any possible internal format of the recipient. This way an organization needs to provide the conversion from its internal format to M and vice versa, thus keeping any proprietary details out of view. The servers are shared amongst the stakeholders that are interested in the interoperability of their spatial data. A disadvantage of this architecture is that all M-related processing is performed on the server side, creating a potential bottleneck in the communication interchange.

Service Oriented Architecture (SOA)

The last considered approach to the spatial data standard architecture is the use of SOA. Generally speaking, "SOA is an architectural style whose goal is to achieve loose coupling among interacting software agents. A service is a unit of work done by a service provider to achieve desired end results for a service consumer. Both provider and consumer are roles played by software agents on behalf of their owners" (He 2003, para. 5). SOA suits well the task of implementing the proposed M standard for spatial data since functions f and g (defined in Chapter 6) can be represented as services by a computer.

Using Web Services within SOA

While employing web services is not a requirement for a successful implementation of SOA, their use is widespread because of support for open standards, such as SOAP, HTTP, XML, etc. There are two types of web services: SOAP RPC (Remote Procedure Call) and REST (He 2003). REST web services are conceptually simpler while SOAP RPC ones are more complex because of the introduction of an additional transport layer (SOAP) on top of HTTP. SOAP allows one to build more sophisticated web services because of the RPC concept. However, He notes that SOAP RPC is ill-suited for interoperability because of the firm prescription of system behavior and application semantics. By removing RPC, a SOAP web service becomes a better match for interoperability tasks, and SOAP web services are used to build a prototype for the thesis research. The architecture for a generic web service is depicted in Figure 38.

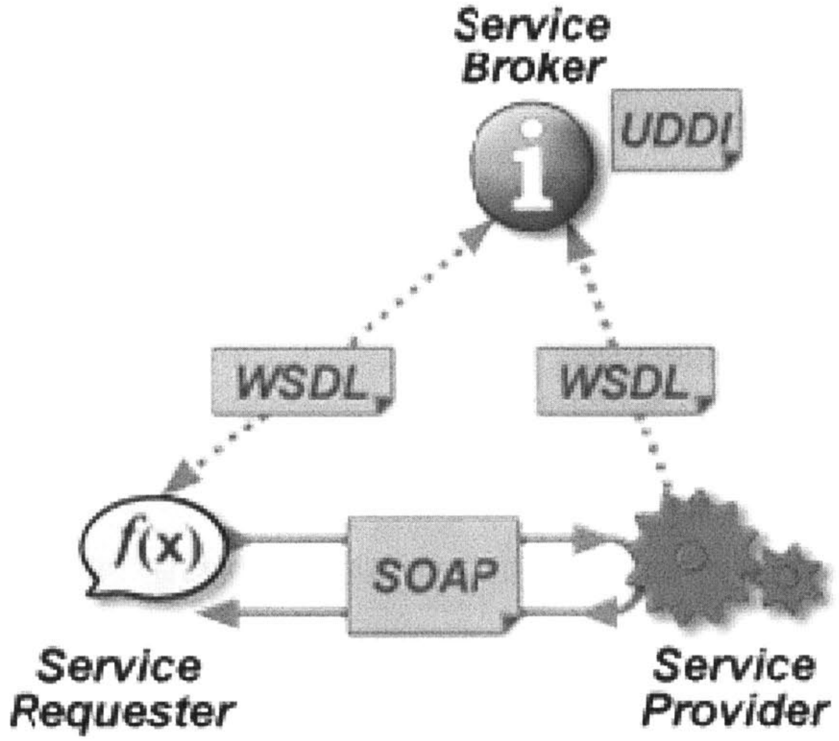


Figure 38: Web Services Architecture
Source: www.wikipedia.org

As seen here, the service requester and service provider exchange SOAP messages to handle data interchange. The description of the web service is published in the Universal Description, Discovery, and Integration (UDDI) registry that is maintained by the Service Broker. Service Requesters can browse this information registry and determine if the web service meets their needs. The last key piece of the architecture is Web Services Description

Language (WSDL) that defines a public interface to the service. Service requesters use the interface to communicate with the service.

Images themselves can be stored and transmitted in any standard format. Portable Network Graphics (PNG) and Graphics Interchange Format (GIF) are two popular formats featuring lossless compression. Using compressed image files speeds up the transmission because of a smaller size. PNG format is preferred since it provides good alpha channel support, which is important for the proposed standard. [42]

After comparing the architecture choices described herein, SOA emerged as the clear winner because of the following three reasons:

1) Built-in support for interoperability
2) Better security when accessing 3rd party infrastructure
3) Open standards and good support from the research community and commercial vendors

Prototype Description

In order to demonstrate the technical feasibility of the proposed standard, a series of prototypes was developed as part of the thesis research. This section describes the prototypes. Selected source code from the prototypes is available in Appendix A.

Temperature Contour Maps

The prototype demonstrates delivery and manipulation of spatial data in the form of temperature contour maps. While a map is delivered as an image to the user (client), its manipulation is accomplished using web services that run on the server side. These web

[42] See http://www.libpng.org/pub/png/ for the official guide to PNG.

services implement functions *f, g,* f^1, and g^{-1}. The spatial data in this prototype has three dimensions: latitude, longitude, and time. Each image represents a spatial temperature contour map for a particular date and time. For this prototype, the spatial data was chosen in the Mercator projection, but in general, any projection can be supported. In fact, using a concept of the *intelligent information network*, presented in this thesis, projections can be transformed one into another by dedicated mathematical models.

The high-level architecture of the prototype generally follows SOA as presented in Figure 38. The prototype was built using .NET technology.[43] The Service Broker and Service Provider run under Internet Information Server (IIS) – a web server from Microsoft. The Service Requester is a fat client (written using .NET Windows Forms technology) and communicates with the Service Broker and Service Requester over the Internet. The Service Broker is provided by Microsoft and built automatically when a .NET web service is deployed to the IIS.

The description of the high-level design for the web service can be aided by the following UML[44] class diagram that shows the relationships among the service's objects (Figure 39). TemperatureData class is the main class, the object for which (a singleton) is instantiated the first time the web service is used.

[43] See http://www.microsoft.com/net/default.mspx for a .NET introduction
[44] Universal Modeling Language, see http://www.uml.org

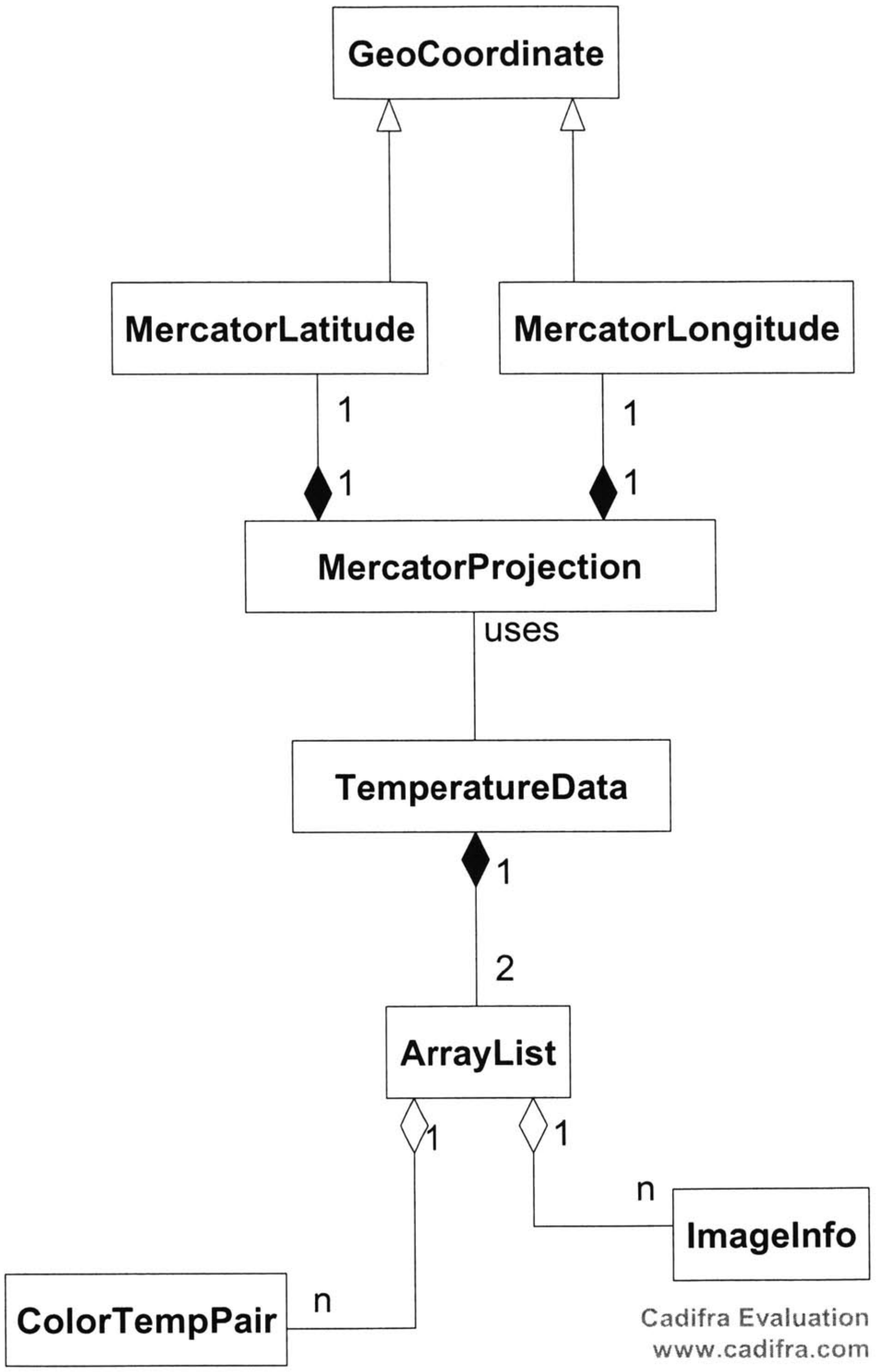


Figure 39: Web Service UML Class Diagram

The class contains two arrays (objects of class ArrayList that is a part of .NET Framework). One array stores the information about images (object of ImageInfo class), and the other one stores objects of class ColorTempPair (a correspondence of a temperature and color) to facilitate the implementation of function *g*.

TemperatureData class works with class MercatorProjection that encapsulates the data and logic to represent spatial maps in Mercator projection and facilitate the implementation of function *f*. MercatorProjection class contains two classes: MercatorLatitude and MercatorLongitude, both of which derived from a base class, GeoCoordinate.

Spatial data is stored in image files and displayed on monitors in a Cartesian-type coordinate system with the origin in the upper left corner, x axis extending to the right and y axis pointing downward. This idea is standard in computer graphics and also used in the NITF standard (MIL-STD-2500B 1998) in the form of the Virtual Coordinate System. The standard is summarized in this thesis; see **National Imagery Transmission Format.**

The translation from the Mercator projection to a Cartesian coordinate system and vice versa to support functions f and f^1 is not a standard one since normally this translation is performed when a Cartesian coordinate system has the origin in the center of the plane. The following shows the derivation of formulae used for translation from the Virtual Coordinate System to Mercator projection coordinates.

Let λ_B and λ_E be the longitudes of the image's leftmost and rightmost points, respectively. Then the longitude of the center of the map, λ_0, can be calculated as:

$$\lambda_0 = \frac{\lambda_B + \lambda_E}{2} \tag{7.1}$$

Then x and y coordinates of a point on the map that has latitude φ and longitude λ can be calculated as follows (Mercator projection 2006):

$$x = \lambda - \lambda_0 \tag{7.2}$$

$$y = \ln\left(\tan\varphi + \sec\varphi\right) \tag{7.3}$$

However, these formulas are valid only for a map whose Cartesian coordinate system has the origin in the center of the map. Since the Virtual Coordinate System has the origin in the upper left corner and the image is a *scaled* version of the map, a linear transformation is needed to map a point from the Virtual Coordinate System into the point in the regular Cartesian system. For convenience, let coordinates with a sign ' denote points in the regular Cartesian plane. Then the following transformation must be defined:

$$(x, y) \longrightarrow (x', y') \tag{7.4}$$

Since it is a linear transformation, it can be defined by the following equations:

$$\begin{aligned} x' &= ax + b \\ y' &= cy + d \end{aligned} \tag{7.5}$$

Since in both equations in (7.5) two coefficients are unknown, a system of two equations with two variables is used to solve for unknown coefficients. Therefore, it is sufficient to have only two points from the map with known latitude and longitude to compose and solve the systems of equations. These systems of equations are presented below:

$$\begin{cases} x_1' = ax_1 + b \\ x_2' = ax_2 + b \end{cases} \qquad (7.6)$$

$$\begin{cases} y_1' = cy_1 + d \\ y_2' = cy_2 + d \end{cases} \qquad (7.7)$$

While (7.7) and (7.8) can be solved for any two points, choosing these points as the origin (x_1,y_1) and the bottom rightmost point (x_2,y_2) will yield the following simplifications: $x_1 = 0$; $y_1 = 0$; x_2 = image width (w); and y_2 = image height (h). Then (7.7) and (7.8) can be simplified as follows to determine **a, b, c, and d**:

$$\begin{cases} x_1' = b \\ x_2' = aw + b \end{cases} \Rightarrow \begin{cases} b = x_1' \\ a = \dfrac{x_2' - x_1'}{w} \end{cases} \qquad (7.8)$$

$$\begin{cases} y_1' = d \\ y_2' = ch + d \end{cases} \Rightarrow \begin{cases} d = y_1' \\ c = \dfrac{y_2' - y_1'}{h} \end{cases} \qquad (7.9)$$

Having found a, b, c, and d, any point (x, y) in the Virtual Coordinate System can be translated into the Mercator latitude and longitude by executing the following steps:

Use (7.7) and (7.8) to find (x', y') in the Cartesian plane

Use (7.2) and (7.3) to find longitude and latitude, respectively

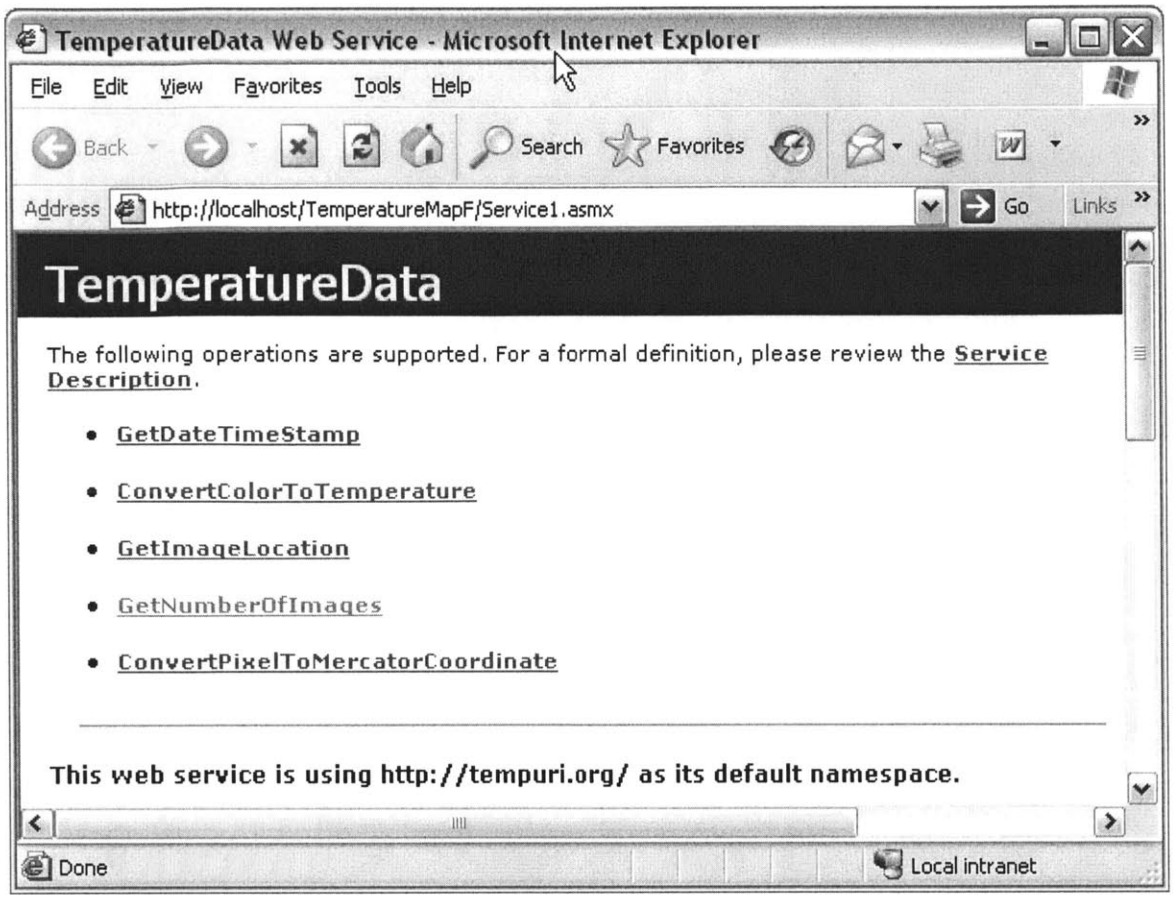
This logic is implemented in `CreateMercatorCoordinate function` (see Appendix A).

The images and their descriptions are encoded in an M-compliant XML file. An excerpt from this file is presented in Figure 40. The entire file is available in Appendix A.

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Color-Temperature map -->
<spatial.1_data.1_descriptor.1>
      <image.2_function.7_url.1>
      http://localhost/TemperatureMapF/Service1.asmx?WSDL
      </image.2_function.7_url.1>

      <color.4_temperature.1_map.1>
            <color.4_temperature.1_pair.3>
                  <color.1>FFFF00FF</color.1>
                  <temperature.1>-12.5</temperature.1>
            </color.4_temperature.1_pair.3>
            <color.4_temperature.1_pair.3>
                  <color.1>FFBF00FF</color.1>
                  <temperature.1>-7.5</temperature.1>
            </color.4_temperature.1_pair.3>

                              ...

      </color.4_temperature.1_map.1>
      <Image.2>

      <url.1>http://localhost/temperatureMapF/other/sfc_con_temp_mod.gif
      </url.1>
            <width.1>640</width.1>
            <height.1>425</height.1>
            <begin.7_latitude.2>48.07</begin.7_latitude.2>
            <begin.7_longitude.1>-133.34</begin.7_longitude.1>
            <end.8_latitude.2>21.35</end.8_latitude.2>
            <end.8_longitude.1>-72.32</end.8_longitude.1>
            <date.5_time.6>03/04/2006:17.0.0.0</date.5_time.6>
            <map.1_projection.4>Mercator</map.1_projection.4>
      </Image.2>
```

Figure 40: M-Compliant Spatial Data XML File

While web services do not normally have User Interface (UI) components, Figure 41 shows the prototype's web service opened directly from the web server. The web site presents functions (web methods) that form service's public interface.


TemperatureData Web Service - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/TemperatureMapF/Service1.asmx
Go Links
Done
Local intranet

TemperatureData

The following operations are supported. For a formal definition, please review the Service Description.

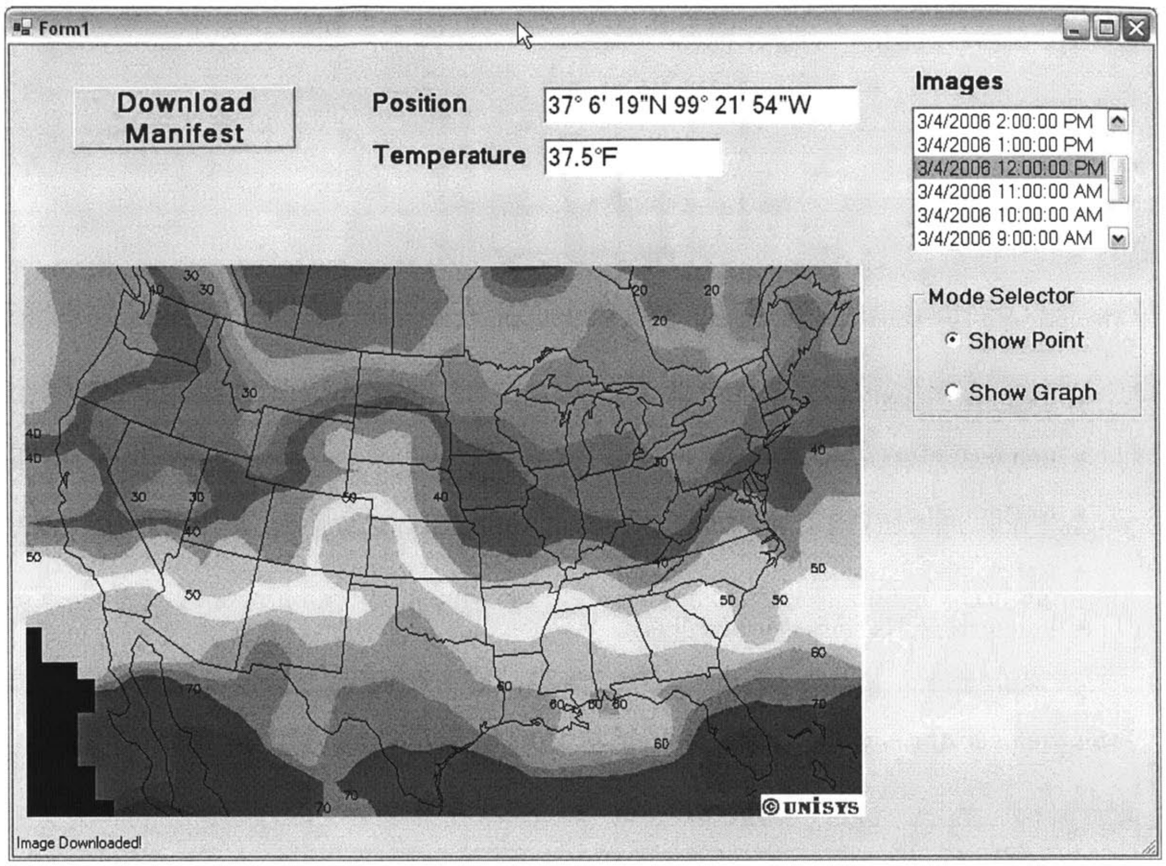
- GetDateTimeStamp
- ConvertColorToTemperature
- GetImageLocation
- GetNumberOfImages
- ConvertPixelToMercatorCoordinate

This web service is using http://tempuri.org/ as its default namespace.

Figure 41: Web Service Test Run

The client part is a .NET application that is able to download an M-compliant spatial data file, parse it, and act on the obtained information. When the M-compliant spatial data standard is integrated in the M-browser, some of these functions, such as downloading and verifying the M-file, will be done automatically.

The following screen shot illustrates the client's UI (Figure 42):



Position: 37° 6' 19"N 99° 21' 54"W

Temperature: 37.5°F

3/4/2006 2:00:00 PM
3/4/2006 1:00:00 PM
3/4/2006 12:00:00 PM
3/4/2006 11:00:00 AM
3/4/2006 10:00:00 AM
3/4/2006 9:00:00 AM

Image Downloaded!

Figure 42: Prototype's Client UI

Clicking **"Download Manifest"** downloads the M spatial data file to the local client. The list of images with their Date-Time stamps appears then in the **Images** listbox. Choosing an image from the listbox displays this image on the form. When the **Mode Selector** is in **Show Point**, clicking on the image will show the position coordinates (latitude and longitude) as well as the corresponding temperatures. Both parameters are obtained over the Internet via the web service. If the **Mode Selector** is in **Show Graph**, clicking on the image produces a graph of Temperature vs. Time across all available images. A sample graph is shown in Figure 43. Note that time is discrete and relative here, with 0 being the latest time of the day (3/4/2006 5 pm) and 11 being the earliest (3/4/2006 6 am).

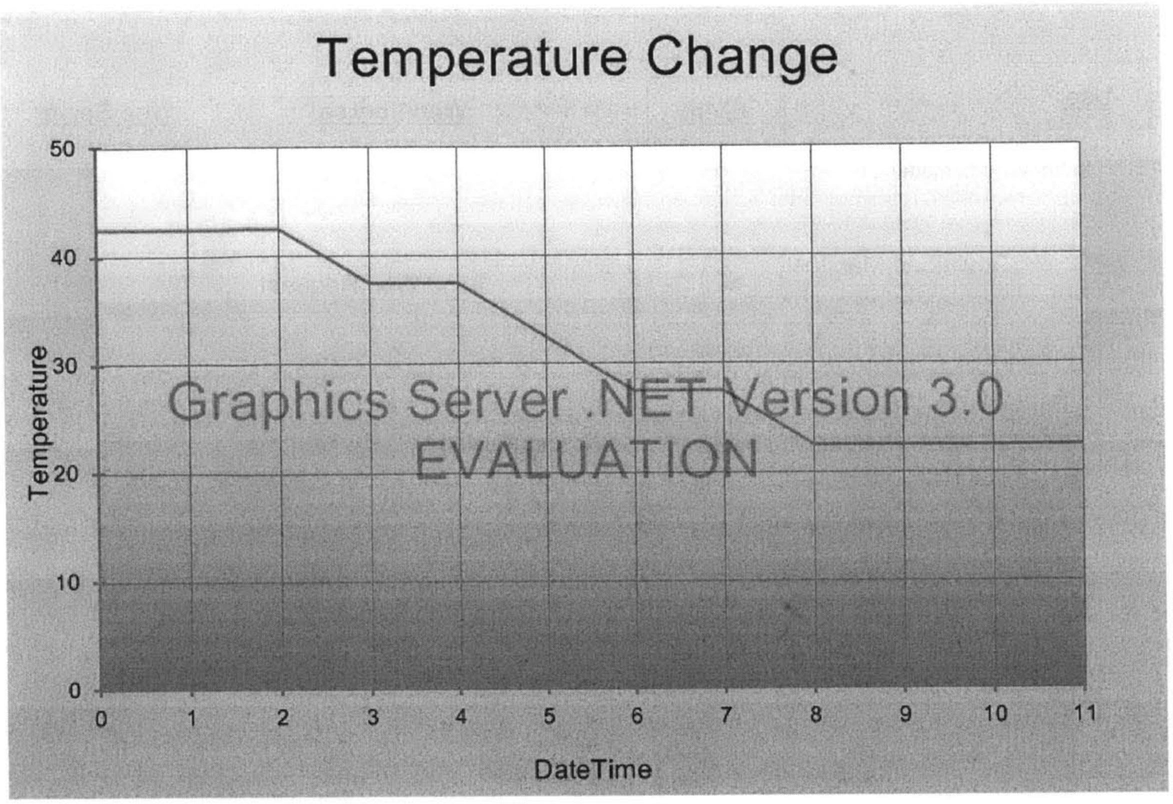


Figure 43: Temperature vs. Time in Prototype

The description of the web service and the client along with their mutual interactions (the high-level sequence diagram) is documented in the UML notation in Figure 44.

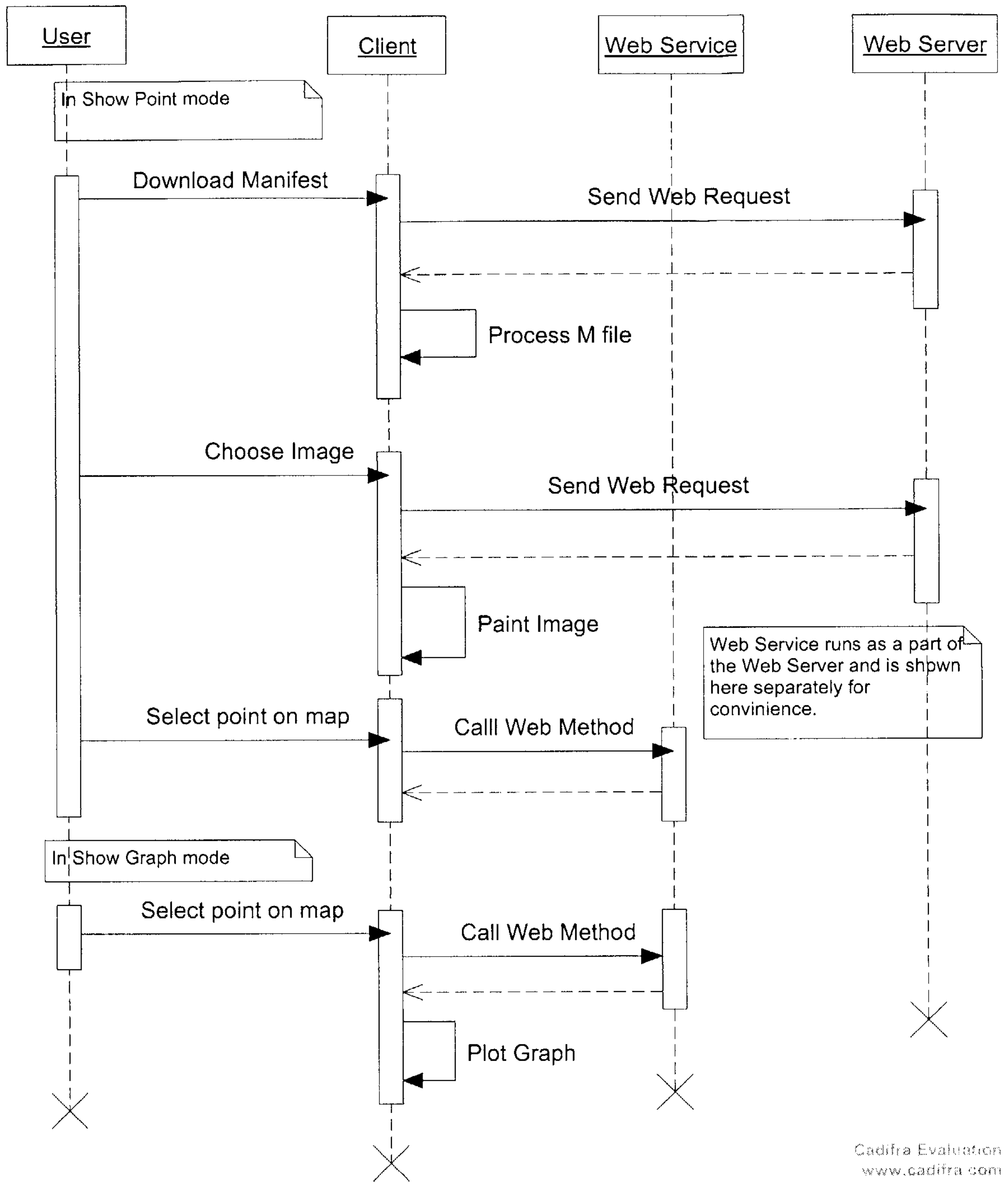


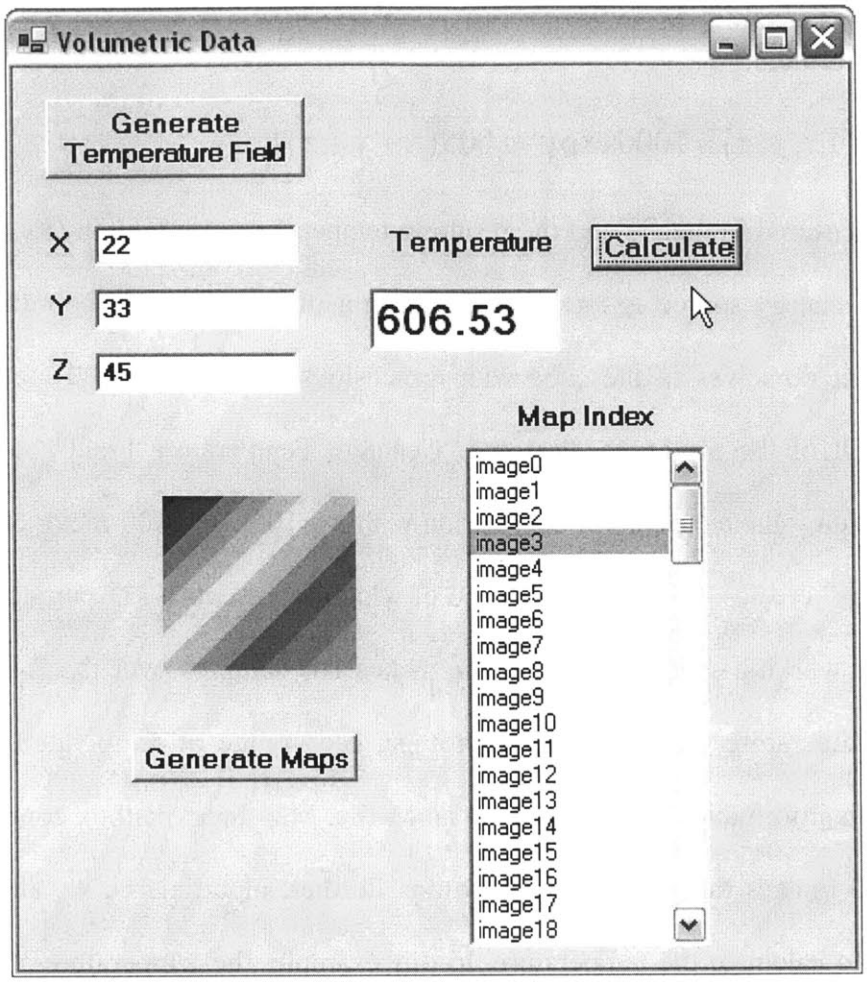
Figure 44: Prototype's High-Level UML Sequence Diagram

Volumetric Data

This prototype demonstrates spatial data standard capabilities in 3 traditional spatial dimensions (3D). The use case includes the visualization, transmission, and analysis of a temperature field in 3D. Let us suppose that temperature in a 3D-space varies according to the following empirical formula:

$$T(x,y,z) = 1000\exp\left(-0.005\left(x+y+z\right)\right),$$

where x, y, and z are measured in meters and the resulting temperature – in Kelvin (K). We further assume that the energy source is located in the origin of the Cartesian coordinate system. Also, we restrict ourselves to the cube with dimensions of 100x100x100 meters. Figure 45 depicts the UI of the prototype. Pressing "Generate Temperature Field" button creates the volumetric data – the temperature distribution within a 100x100x100 meter cube. Pressing "Generate Maps" creates a set of images each of which represents a 2D sample cut across the Z-axis of the working space. The prototype makes 100 samples over the Z-axis. Then similar to the Temperature Contour Maps prototype, each range of temperatures is assigned a color, and this information is delivered via an M-file. Map Index listbox contains the list of resulting 100 images that the user can browse. Further, inputting x-, y-, and z-values allows the user to calculate the temperature. In this example, the temperature at the point (22, 33, 45) is 606.53K.



X 22

Y 33

Z 45

Temperature 606.53

Map Index: image0, image1, image2, image3, image4, image5, image6, image7, image8, image9, image10, image11, image12, image13, image14, image15, image16, image17, image18

Figure 45: Volumetric Data Prototype

Vector Data

The spatial data standard also allows the representation of vector data in addition to scalars. The following prototype demonstrates this capability by displaying a static electric field of a charge equal to that of a proton.

As known from an elementary course of physics, the static electric field in vacuum can be calculated as follows:

$$\mathbf{E} = \frac{q}{4\pi\varepsilon_0 d^2}\mathbf{r},$$

where q is a charge, d – distance from the charge to the point where the force is calculated, ε_0 is permittivity of free space. **E and r** are vector variables. **r** points away from the charge if the charge is positive, and points toward the charge if the charge is negative.

Figure 46 depicts the UI of the prototype. Pressing "Generate Field" creates the vector data distribution. As suggested, each 32-bit color value reserves the upper byte for the direction while the lower 3 bytes represent the magnitude. Direction is represented by an angle in radians calculated from the x-axis. Again, necessary data, including the charge, colors and pixel resolution are supplied by an M-file.

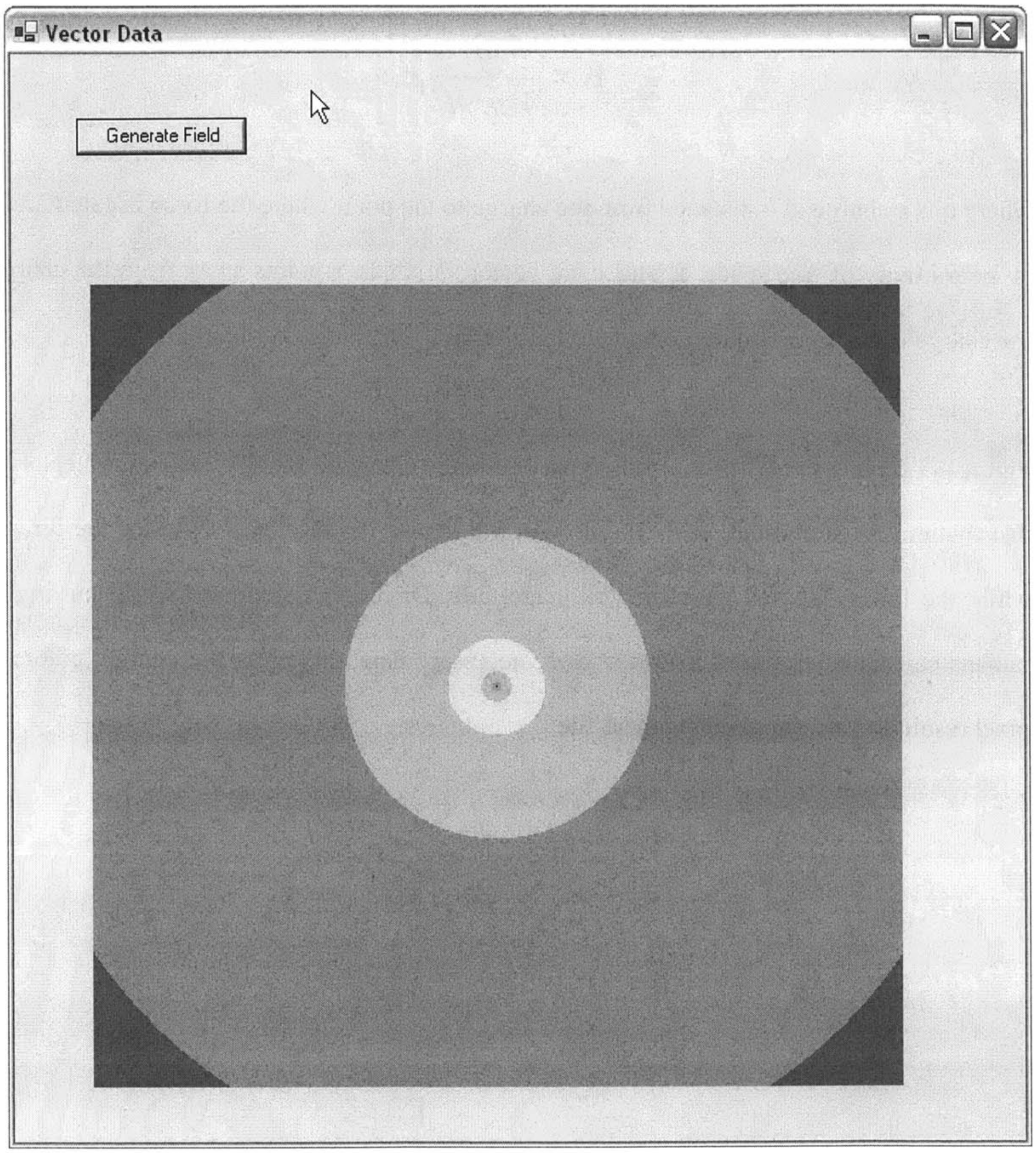
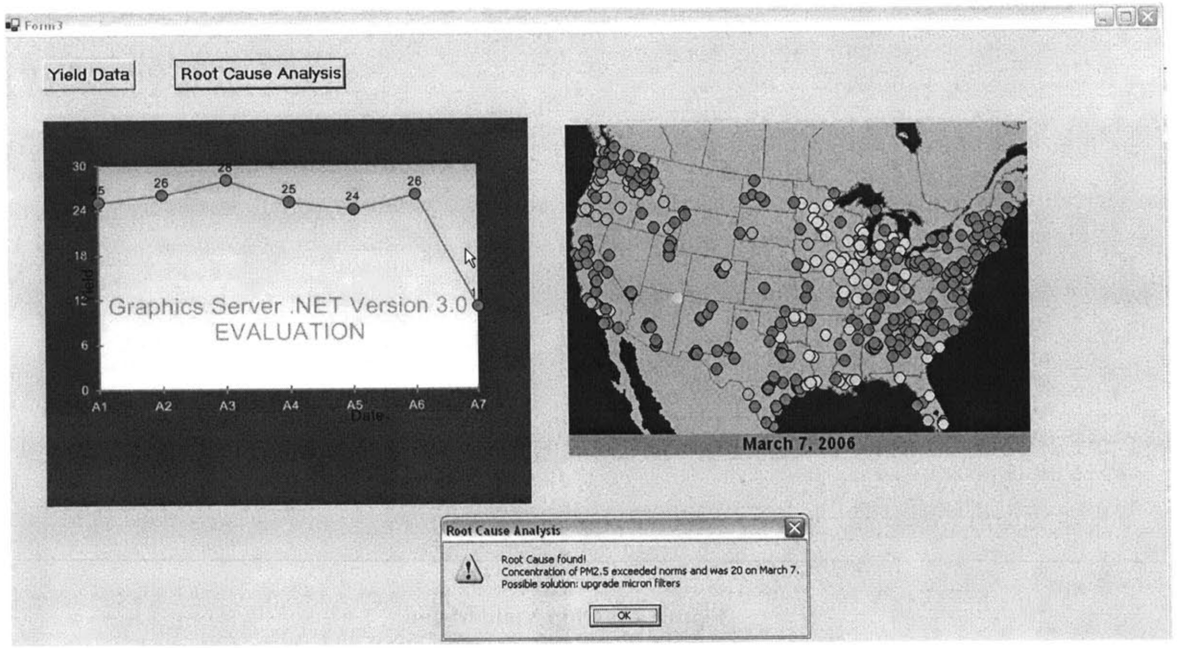


Figure 46: Vector Data Prototype

Integrated Circuits Yield

This prototype demonstrates the advanced capabilities of M-Language, such as joint interpretation and correlation of disparate and heterogeneous data sources and the use of distributed mathematical models to accomplish the task. Although based on artificial data, the described scenario may occur in the semiconductor industry. The prototype is based on the following scenario:

A semiconductor plant, based in Arizona, produces a certain integrated circuit. The normal yield is hovering around 25%. However, on March 6, 2006 the yield drops to just 11%. Engineers are assigned to conduct the Root Cause Analysis of the yield drop.



Root Cause found!
Concentration of PM2.5 exceeded norms and was 20 on March 7.
Possible solution: upgrade micron filters

Figure 47: Integrated Circuits Yield Prototype

Figure 47 shows the UI for the prototype. Pressing "Yield Data" connects the client to the Web Service that distributes the yield data in the form of the M-file. This M-file is presented in Figure 48. The chip yield data is given as a set of key-value pairs and is not of spatial data form. Note that the plant location is given in Universal Transverse Mercator coordinates.

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- semiconductor plant's yield of integrated circuits -->
<yield.1_curve.4>
      <plant.2_location.1>
            <!-- In UTM -->
            <X.2>346633.68</X.2>
            <Y.2>3871005.10</Y.2>
            <zone.3>11</zone.3>
            <hemisphere.2>North</hemisphere.2>

      <map.1_projection.4>Universal_Transverse_Mercator.1</map.1_proj
ection.4>
      </plant.2_location.1>

      <date.5>04/01/2006</date.5>
      <yield.1>0.25</yield.1>
      <date.5>04/02/2006</date.5>
      <yield.1>0.26</yield.1>
      <date.5>04/03/2006</date.5>
      <yield.1>0.28</yield.1>
      <date.5>04/04/2006</date.5>
      <yield.1>0.25</yield.1>
      <date.5>04/05/2006</date.5>
      <yield.1>0.24</yield.1>
      <date.5>04/06/2006</date.5>
      <yield.1>0.26</yield.1>
      <date.5>04/07/2006</date.5>
      <yield.1>0.11</yield.1>
</yield.1_curve.4>
```

Figure 48: Chip Yield M-file

Noting the precipitous yield drop, an engineer will initiate the Root Cause Analysis. The plant management system will determine that the cause for unusually high chip failures is the increased concentration of impurities in the silicon wafers. Continuing looking for the root cause, the system determines that the concentration of PM2.5 particles[45] has been unusually high on March 6. The system is then directed to look for external environmental factors that can cause the high PM2.5 concentration. Connecting to the M-feed, the system is able to find PM2.5 concentrations in the United States for the given range of dates. This spatial data M-feed is presented in Figure 49.

[45] http://www.epa.gov/region4/sesd/pm25/p2.htm

```
<?xml version="1.0" encoding="utf-8" ?>
<spatial.1_data.1_descriptor.1>
      <color.4_pollution.1_map.1>
            <color.4_pollution.1_pair.3>
                  <color.1>FF00E400</color.1>

      <particle.3_concentration.5>10</particle.3_concentration.5>
                  <color.1>FFFFFF00</color.1>

      <particle.3_concentration.5>20</particle.3_concentration.5>
                  <color.1>FF00E400</color.1>

      <particle.3_concentration.5>30</particle.3_concentration.5>
            </color.4_pollution.1_pair.3>
                     </color.4_pollution.1_map.1>
      <Image.2>
            <url.1>C:\Documents and Settings\Marat\My
Documents\thesis\pollution\pm25-24p-super1.gif</url.1>
            <width.1>483</width.1>
            <height.1>350</height.1>
            <begin.7_latitude.2>48.07</begin.7_latitude.2>
            <begin.7_longitude.1>-133.34</begin.7_longitude.1>
            <end.8_latitude.2>21.35</end.8_latitude.2>
            <end.8_longitude.1>-72.32</end.8_longitude.1>
            <date.5>04/01/2006</date.5>
            <map.1_projection.4>Mercator.1</map.1_projection.4>
      </Image.2>
      <Image.2>
            <url.1>C:\Documents and Settings\Marat\My
Documents\thesis\pollution\pm25-24p-super2.gif</url.1>
            <width.1>483</width.1>
            <height.1>350</height.1>
            <begin.7_latitude.2>48.07</begin.7_latitude.2>
            <begin.7_longitude.1>-133.34</begin.7_longitude.1>
            <end.8_latitude.2>21.35</end.8_latitude.2>
            <end.8_longitude.1>-72.32</end.8_longitude.1>
            <date.5>04/02/2006</date.5>
            <map.1_projection.4>Mercator.1</map.1_projection.4>
      </Image.2>
      …
      <Image.2>
            <url.1>C:\Documents and Settings\Marat\My
Documents\thesis\pollution\pm25-24p-super7.gif</url.1>
            <width.1>483</width.1>
            <height.1>350</height.1>
            <begin.7_latitude.2>48.07</begin.7_latitude.2>
            <begin.7_longitude.1>-133.34</begin.7_longitude.1>
            <end.8_latitude.2>21.35</end.8_latitude.2>
            <end.8_longitude.1>-72.32</end.8_longitude.1>
            <date.5>04/07/2006</date.5>
            <map.1_projection.4>Mercator.1</map.1_projection.4>
      </Image.2>
</spatial.1_data.1_descriptor.1>
```

Figure 49: PM2.5 Spatial M-feed

Note that spatial images are presented in Mercator projection. In order to correlate these two heterogeneous M-feeds, we need a mathematical model to translate the Mercator coordinates into UTM coordinates. This model is made available via a web service.[46] It is usable since inputs and outputs are defined semantically with the help of the M-Dictionary.

After converting the plant location to Mercator coordinates and finding the location on the corresponding image, the system determines that the concentration of PM2.5 was twice as high on March 6 than in the previous 6 days. The engineer with the help of the plant management system has determined the root cause and provided the recommendation to upgrade the sub-micron filters that the plant uses to provide clean rooms with air.

[46] The actual translation logic is ported from JavaScript code made available at http://home.hiwaay.net/~taylorc/toolbox/geography/geoutm.html

Chapter 8 Further Research

The research on the spatial data representation for the M initiative is by no means complete. The following topics need to be explored further:

- Using multiple images to represent arbitrary floating point precision
- Developing mathematical models for spatial data analysis
- Using artificial intelligence and semantic principles to automatically discover and connect to interesting spatial data sources
- Developing case studies of applications of the M spatial data standard in various industries

These topics are briefly explained below:

Arbitrary floating point precision

An ARGB value is a 32-bit number, and it limits the floating point precision if the number is used in floating-point arithmetic. To solve this problem, multiple images can be used to increase the number of available bits to any number that may be reasonably used in scientific applications. Algorithms have to be developed and implemented to combine ARGB values from multiple pixels to make a floating point number with given precision and vice versa.

Mathematical Models for Spatial Data Analysis

M-compliant spatial data standard implies a significant role of the intelligent network of models. Since the standard uses SOA as an architecture of choice, integrating the standard with the model network is feasible. Many models need to be developed and added to the model library. For example, an interpolation model to substitute unknown colors with interpolated ones is needed to enhance the robustness of the spatial standard. The research into creating such a network, while not directly a descendant of this thesis, would still be quite beneficial as a complement.

Automatic Discovery and Connection to Spatial Data Sources

The prototype to demonstrate the standard uses a static connection to a spatial data source. In other words, a spatial data source must be known *a priori*. While technically not part of the standard, an ability to dynamically search for interesting data sources in an automatic manner and based on sources' semantics, feed the sources into models, interlace the sources to produce synthetic spatial entities is of great importance. This is a long-term vision of the M-initiative and the MIT Data Center. Naturally, this topic is closely related with the research on intelligent model networks.

Industrial Case Studies

Proving the standard in various industries is an important task and is consistent with the overall goal of the MIT research community to bridge the divide between industry and academia. One application where initial steps have already been taken is the agriculture industry. A case in point is the prediction of crop yields based on various spatial data such as weather, distribution of crop parasites, application of fertilizers, etc.

Time in Spatial Systems

As noted in **Time in Spatial Data Information Systems,** modern information systems represent time as a Date-Time field or an attribute. Since arbitrary n-dimensional systems may include time as an independent variable, a better support for time in M spatial data is needed. In addition, when analyzing astrophysical phenomena, there is a need to support Einstein's Special Theory of Relativity and its notion of time.

In a nutshell, there is a significant amount of work to continue developing the M spatial standard and integrating it with the rest of the M initiatives.

Chapter 9 Conclusion

The research on developing representations of spatial data for the M initiative proved to be an interesting, engaging, and challenging activity. It required a cross-disciplinary knowledge set and a system approach that is a hallmark of MIT in general and the SDM program and MIT Data Center in particular. The result is the definition of the M-compliant rules on the spatial data representation and a working software proof of concept.

The MIT Data Center vision to create infrastructure where disparate data sources, including spatial ones, can be combined with each other on semantic level and fed into mathematical models that are, in turn, capable of being combined into larger, synthetic models for advanced data processing is a lofty one. The research that resulted in this thesis is a stepping stone toward realizing this vision. I am glad that I was given an opportunity to be a part of this initiative and produce work that will help organizations solve the information overload problem and make sense of their data.

Chapter 10 Bibliography

Alberts, D.S. & Garstka, J.J. & Hayes, R.E. & Signori, D.A. (2001). *Understanding Information Age Warfare.* Washington: CCRP.

Allaway, A. W. & Berkowitz D. & D'Souza G. (2003). "Spatial Diffusion of a New Loyalty Program Through a Retail Market," *Journal of Retailing, Vol. 79,* 137 – 151.

ArcGIS: Engineered for Interoperability. (2006). Retrieved March 27, 2006 from http://www.esri.com/library/whitepapers/pdfs/arcgis-engineered-for-interoperability.pdf.

Bass, F. M. (1969). A new product growth model for consumer durables. *Management Science Vol.15, Issue 1,*. 215–227.

Bishr, Y. (1998). Overcoming semantic and other barriers to GIS interoperability. *International Journal of Geographical Information Science, Vol. 12, Issue 4,* 299-314.

Brock, D.L. (2004). *The Data Center Vision: Making Sense of the Data.* Retrieved April 4, 2006 from http://mitdatacenter.org/MIT-DATACENTER-WH-002.pdf.

Brock, D.L, & Schuster, E.W., & Allen, S.J., & Kar, P. (2005). An Introduction to semantic modeling for logistical systems. *Journal of Business Logistics, Vol. 26, No. 2,* 97-117.

Brock, D.L., & Schuster, E.W., & Kutz, T.J. (2006). *An Overview of the M-Language.* Retrieved March 21, 2006, from http://mitdatacenter.org/MIT-DATACENTER-WH-009.pdf.

Bugayevskiy, L.M., & Snyder, J. P. (1995). Map Projections - A Reference Manual (London: Taylor & Francis).

Demirkesen, A. & Schaffrin, B. (1996). Map conflation: spatial point data merging and transformation. *In Proceedings of GIS/L IS'96* (Washington: AAG/ACSM/AMFM/ ASPRS/URISA), 393-404.

Devogele, T., & Parent, C., & Spaccapietra, S. (1998). On Spatial Database Integration. *International Journal of Geographical Information Science, Vol. 12,Issue 4,* 335-352.

Dougenik, J. (1980). WHIRLPOOL : A geometric processor for polygon coverage data. *In Proceedings of Auto-Carto 4* (Bethesda: American Congress on Surveying and Mapping), 304- 311.

ESRI Shapefile Technical Description. (1998). Retrieved March 27, 2006 from http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf.

ESRI shapefiles. (2006). *Wikipedia.* Retrieved March 27, 2006 from http://en.wikipedia.org/wiki/Shapefile.

Fagan, G. & Soehngen, H. (1987). Improvement of GBF/DIME file coordinates in a Geobased Information System by various transformation methods and rubbersheeting based on triangulation. *In Proceedings of Auto-Carto 7* (Bethesda: American Congress on Surveying and Mapping), 481-491.

Flowerdew, R. (1992). Spatial data integration. In Geographic Information Systems: Principles and Applications, edited by D. J.Maguire, M. F. Goodchild, and D.W. Rhind (London: Longman), 337-358.

Frank, A. (1987). Overlay processing in spatial information systems. *In Proceedings of Auto-Carto 8* (Bethesda: American Congress on Surveying and Mapping).

Global Information Grid. (2006). *Wikipedia.* Retrieved April 3, 2006 from http://en.wikipedia.org/wiki/Global_Information_Grid.

Google Earth Enterprise (n.d.). Retrieved March 28, 2006 from http://earth.google.com/earth_enterprise.html.

Google Earth KML 2.0 (2006). Retrieved March 30, 2006 from http://earth.google.com/kml/kml_toc.html.

Google Earth KML Tutorial (n.d.). Retrieved March 28, 2006 from http://www.keyhole.com/kml/kml_tut.html.

Harrison, J. (2002). *Overview of OGC Interoperability Program.* Retrieved March 21, 2006, from http://portal.opengeospatial.org/files/?artifact_id=6196&version=1&format=pdf.

He, H. (2003). *What is Service-Oriented Architecture.* Retrieved April 6, 2006 from http://webservices.xml.com/pub/a/ws/2003/09/30/soa.html

Hoffman, D.L. & Franke, G.R. (1986). Correspondence Analysis: Graphical Representation of Categorical Data in Marketing Research. *Journal of Marketing Research, Vol. 23, Issue 3,* 213-227.

Introduction to GIS. (n.d.). Retrieved May 7, 2006 from http://www.sli.unimelb.edu.au/gisweb/GISModule/GISTheory.htm.

Laurini, R. (1994). Sharing geographic information in distributed databases. *In Proceedings of 32nd Annual URISA Conference* (Washington: Urban and Regional Information System Association), 441-455.

Laurini, R. & Thompson, D. (1992). *Fundamentals of Spatial Information Systems.* San

Diego: Academic Press.

Maier M. & Rechtin, E. (2002). *The Art of System Architecture.* Boca Raton: CRC Press.

Mercator projection. (2006). *Wikipedia.* Retrieved April 9, 2006 from http://en.wikipedia.org/wiki/Mercator_projection.

MIL-STD-2500B. (1998). *National Imagery Transmission Format v.2.1.* Retrieved March 27, 2006 from http://164.214.2.51/ntb/baseline/docs/2500b/2500b.pdf.

Moore, G.A. (2002). *Crossing the Chasm.* New York: Harper Business Essentials.

Murdock P. (2002). Principles of War on the Network-Centric Battlefield: Mass and Economy of Force. *Parameters, Vol. 32, Issue 1,* 86-96.

Network-centric warfare. (2006). *Wikipedia.* Retrieved April 3, 2006 from http://en.wikipedia.org/wiki/Network_Centric_Warfare.

Network Centric Warfare. (2001). *Department of Defense Report to Congress.* Retrieved 4/3/2006 from http://www.dod.mil/nii/NCW/ncw_main.pdf.

Ontology (Computer Science). (2006). *Wikipedia.* Retrieved March 23, 2006, from http://en.wikipedia.org/wiki/Ontology_(computer_science).

Parent, C., & Spaccapietra, S. (1998). Issues and Approaches of Database Integration. *Communications of the ACM, Vol. 41, Issue 5,* 166-178.

Piwowar, J., LeDrew, E., and Dudycha, D. (1990). Integration of spatial data in vector and raster formats in a geographic information system environment. *International Journal of Geographical Information Systems, 4*, 429-444.

Pullar, D. (1993). Consequences of using a tolerance paradigm in spatial overlay. *In Proceedings of Auto-Carto 11* (Bethesda: American Congress on Surveying and Mapping), 288-296.

Reichardt, M. (2004). *The Havoc of Non-Interoperability.* Retrieved March 21, 2006, from http://portal.opengeospatial.org/files/?artifact_id=5097&version=3&format=pdf.

Reichardt, M. (2005). *Sensor Web Enablement.* Retrieved March 21, 2006, from http://portal.opengeospatial.org/files/?artifact_id=11661&version=2&format=pdf.

Rishe N. (1992). *Database Design: The Semantic Modeling Approach.* New York: McGraw-Hill.

Ruas, A. & Plazanet, C. (1996). Strategies for automated generalization. *In Advances in GIS II: Proceedings of 7th International Symposium on Spatial Data Handling*, edited by M. Molennar and M.-J. Kraak (London: Taylor & Francis), 23.

Schuster, E.W., & Allen, S.J., & Brock, D.L. (2006). *Raising the Bar[Code].* Manuscript submitted for publication to Springer Verlag.

Stokes, M., & Andersen, M., & Chandrasekar, S., & Motta, R. (1996). *A Standard Default Color Space for the Internet – sRGB.* Retrieved April 11, 2006 from http://www.w3.org/Graphics/Color/sRGB.

Tanenbaum, A. (2001). *Modern Operating Systems.* New Jersey: Prentice Hall.

Tanenbaum, A. (2002). *Computer Networks.* New Jersey: Prentice Hall.

Vckovski, A. (1998). Special Issue: Interoperability in GIS. *International Journal of Geographical Information Science, Vol. 12, Issue 4,* 297-298.

W3C PNG Specification. (1996). Retrieved April 11, 2006 from http://www.w3.org/TR/PNG-DataRep.html#DR.Alpha-channel

Weiss, M.A. (1995). *Efficient C Programming.* New Jersey: Prentice Hall.

Chapter 11 Appendix A

Selected Web Service Source Code

```
using System;
using System.Collections;
using System.Collections.Specialized;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Web;
using System.Web.Services;
using System.Configuration;
using System.Globalization;
using System.Xml;
using System.Text;

namespace TemperatureMap
{
      public enum ProjectionType {Mercator, Gnomonic, Equirectangular,
GallPeters};

      /// <summary>
      /// Web Service to provide temperature temporo-spatial information
      /// </summary>
      public class TemperatureData : System.Web.Services.WebService
      {
            public const float ABSOLUTE_ZERO = -459.7f; // 0 K

            private ArrayList images = new ArrayList();
            private ArrayList colorTempMap = new ArrayList();

            private void ParseXmlFile(string xmlFileName)
            {
                  int pxWidth=0, pxHeight=0;
                  double degBeginLatitude=0, degEndLatitude=0,
degBeginLongitude=0, degEndLongitude=0;
                  string dateTime = null;
                  string url = null;
                  ProjectionType projection = 0;
                  int color = 0;
                  float temperature = 0;

                  XmlTextReader reader = new XmlTextReader(xmlFileName);
                  reader.WhitespaceHandling = WhitespaceHandling.None;

                  string currElement = null;

                  // Parse the file and get the image
                  while (reader.Read())
                  {
                        switch (reader.NodeType)
```

```
                              {
                                    case XmlNodeType.Element:
                                          currElement = reader.Name;
                                          break;
                                    case XmlNodeType.Text:
                                          switch (currElement)
                                          {
                                                case "url.1":
                                                      url = reader.Value;
                                                      break;

                                                case "width.1":
                                                      pxWidth =
int.Parse(reader.Value);
                                                      break;

                                                case "height.1":
                                                      pxHeight =
int.Parse(reader.Value);
                                                      break;

                                                case "begin.7_latitude.2":
                                                      degBeginLatitude =
double.Parse(reader.Value);
                                                      break;

                                                case "end.8_latitude.2":
                                                      degEndLatitude =
double.Parse(reader.Value);
                                                      break;

                                                case "begin.7_longitude.1":
                                                      degBeginLongitude =
double.Parse(reader.Value);
                                                      break;

                                                case "end.8_longitude.1":
                                                      degEndLongitude =
double.Parse(reader.Value);
                                                      break;

                                                case "date.5_time.6":
                                                      dateTime = reader.Value;
                                                      break;

                                                case "map.1_projection.4":
                                                      if (reader.Value ==
"Mercator")
                                                      {
                                                            projection =
ProjectionType.Mercator;
                                                            // populate imageInfo
                                                            ImageInfo imageInfo =
new ImageInfo(pxWidth, pxHeight, degBeginLatitude, degBeginLongitude,
degEndLatitude, degEndLongitude,
                                                            dateTime, url,
projection);
                                                       images.Add(imageInfo);
                                                 }
                                                 break;

                                          case "color.1":
                                                 // begin new color-temp pair
                                                 color =
int.Parse(reader.Value, NumberStyles.AllowHexSpecifier);
                                                 break;

                                          case "temperature.1":
                                                 temperature =
float.Parse(reader.Value);
                                                 // finished collecting info
for new color-temperature pair,
                                                 // create it and put it in
the array
                                                 ColorTempPair newPair = new
ColorTempPair(color, temperature);
                                                 colorTempMap.Add(newPair);
                                                 break;

                                          default:
                                                 break;
                                   }
                                   break;

                            default:
                                   break;
                     }
               }

               // binary search will be used
               colorTempMap.Sort();

          }

          public TemperatureData()
          {
               //CODEGEN: This call is required by the ASP.NET Web
Services Designer
               InitializeComponent();

               // Get images details

     ParseXmlFile("C:\\inetpub\\wwwroot\\TemperatureMapF\\other\\image_info_
local_m_2.xml");
          }


          #region Component Designer generated code

          //Required by the Web Services Designer
```

```
            private IContainer components = null;

            /// <summary>
            /// Required method for Designer support - do not modify
            /// the contents of this method with the code editor.
            /// </summary>
            private void InitializeComponent()
            {
            }

            /// <summary>
            /// Clean up any resources being used.
            /// </summary>
            protected override void Dispose( bool disposing )
            {
                  if(disposing && components != null)
                  {
                        components.Dispose();
                  }
                  base.Dispose(disposing);
            }

                                          #endregion

            [WebMethod]
            public int GetNumberOfImages()
            {
                  // stubbed for now
                  return this.images.Count;
            }

            [WebMethod]
            public string GetImageLocation(int index)
            {
                  ImageInfo imageInfo = this.images[index] as ImageInfo;
                  return imageInfo.Url;
            }

            [WebMethod]
            public string ConvertPixelToMercatorCoordinate(double x, double
y, int index)
            {
                  ImageInfo imageInfo = this.images[index] as ImageInfo;
                  MercatorProjection coordinate =
MercatorProjection.CreateMercatorCoordinate(x, y, imageInfo);
                  return coordinate.ToString();
            }

            [WebMethod]
            public float ConvertColorToTemperature(int color)
            {
                  ColorTempPair dummy = new ColorTempPair(color, 0);
                  int i = this.colorTempMap.BinarySearch(dummy);

                  if (i < 0)
```

```
                {
                        // color not found
                        return ABSOLUTE_ZERO;
                }
                                                else
                {
                        ColorTempPair pair = this.colorTempMap[i] as
ColorTempPair;
                        return pair.Temperature;
                }
            }

            [WebMethod]
            public DateTime GetDateTimeStamp(int index)
            {
                ImageInfo imageInfo = this.images[index] as ImageInfo;
                return imageInfo.DateTimeStamp;
            }
        }

        public class MercatorProjection
        {
            private MercatorLatitude  latitude;
            private MercatorLongitude longitude;

            public MercatorProjection(MercatorLatitude latitude,
MercatorLongitude longitude)
            {
                this.latitude = latitude;
                this.longitude = longitude;
            }

            // default constructor
            public MercatorProjection(){}

            public MercatorLatitude Latitude
            {
                                                get
                {
                        return latitude;
                }
                                                set
                {
                        if (value != null)
                        {
                            latitude = value;
                        }
                }
            }

            public MercatorLongitude Longitude
            {
                                                get
                {
                        return longitude;
```

```
			}
							set
			{
				if (value != null)
				{
					longitude = value;
				}
			}
		}

		public override string ToString()
		{
			// TODO:  Add MercatorProjection.ToString implementation
			return latitude.ToString() + " " + longitude.ToString();
		}

		public static MercatorProjection CreateMercatorCoordinate(double
x, double y, ImageInfo image)
		{
			double degLambda0 = (image.BeginLongitude +
image.EndLongitude) / 2;
			double radX1Prime =
GeoCoordinate.ToRadians(image.BeginLongitude - degLambda0);
			double radX2Prime =
GeoCoordinate.ToRadians(image.EndLongitude - degLambda0);
			double a = (radX2Prime - radX1Prime) / image.Width;
			double b = radX1Prime;

			double phi = GeoCoordinate.ToRadians(image.BeginLatitude);
			double temp = Math.Tan(phi) + 1/Math.Cos(phi);
			double radY1Prime = Math.Log(temp, Math.E);

			phi = GeoCoordinate.ToRadians(image.EndLatitude);
			temp = Math.Tan(phi) + 1/Math.Cos(phi);
			double radY2Prime = Math.Log(temp, Math.E);
			double c = (radY2Prime - radY1Prime) / image.Height;
			double d = radY1Prime;

			double radXPrime = a * x + b;
			double radYPrime = c * y + d;

			// now use transformation to Mercator latitude and
longitude
			double radLambda = radXPrime +
GeoCoordinate.ToRadians(degLambda0);
			double radPhi = Math.Atan(Math.Sinh(radYPrime));

			MercatorLatitude  latitude    = new
MercatorLatitude(radPhi);
			MercatorLongitude longitude   = new
MercatorLongitude(radLambda);
			MercatorProjection      coordinate   = new
MercatorProjection(latitude, longitude);

			return coordinate;
```

```
            }
      }

      public class GeoCoordinate
      {
            #region Protected members
          protected const int MINUTES_IN_DEGREE = 60;
            protected const int SECONDS_IN_MINUTE     = 60;
            protected const double RADIANS_IN_DEGREE = Math.PI / 180;
            protected const double DEGREES_IN_RADIAN = 180 / Math.PI;
            protected readonly int MaxDegrees;

            protected int degree;
            protected int minute;
            protected int second;
            protected double radCoordinate; // in radians

            protected void ToDecimalForm()
            {
                  radCoordinate = ToRadians((double)degree +
(double)minute/MINUTES_IN_DEGREE + (double)second/(MINUTES_IN_DEGREE *
SECONDS_IN_MINUTE));
            }

            protected void ToCanonicalForm()
            {
                  // ignore fractions of second
                  double degCoordinate = ToDegrees(this.radCoordinate);
                  this.degree = (int)Math.Floor(degCoordinate);

                  double degRemainder = degCoordinate - this.degree; //
remaining minutes and seconds
                  this.minute = (int)Math.Floor(degRemainder *
MINUTES_IN_DEGREE);
                  this.second = (int)Math.Floor((degRemainder *
MINUTES_IN_DEGREE - this.minute) * SECONDS_IN_MINUTE);

#if DEBUG
                  TestInvariant();
#endif
            }

            public static double ToRadians(double angle)
            {
                  return angle * RADIANS_IN_DEGREE;
            }

            public static double ToDegrees(double angle)
            {
                  return angle * DEGREES_IN_RADIAN;
            }

#if DEBUG
            protected void TestInvariant()
            {
```

```
                        string param = null;
                        int val = 0;

                        if (Math.Abs(this.degree) > MaxDegrees)
                        {
                              param = "degree";
                              val = degree;
                        }
                        else if (Math.Abs(this.minute) > MINUTES_IN_DEGREE)
                        {
                              param = "minute";
                              val = minute;
                        }
                        else if (Math.Abs(this.second) > SECONDS_IN_MINUTE)
                        {
                              param = "second";
                              val = second;
                        }

                        if (param != null)
                        {
                              throw new ArgumentOutOfRangeException(param, val,
"Invalid Coordinate");
                        }
                  }
#endif
                                                #endregion

                                                #region Public members
                  public GeoCoordinate(int degree, int minute, int second, int
MaxDegrees)
                  {
                        this.degree = degree;
                        this.minute = minute;
                        this.second = second;
                        this.MaxDegrees = MaxDegrees;

#if DEBUG
                        TestInvariant();
#endif

                        ToDecimalForm();
                  }

                  public GeoCoordinate(double coordinate, int MaxDegrees)
                  {
                        this.radCoordinate = coordinate;
                        this.MaxDegrees = MaxDegrees;
                        ToCanonicalForm();
#if DEBUG
                        TestInvariant();
#endif
                  }

                  public int Degree
```

```
            {
                                                get
                {
                    return this.degree;
                }
                                                set
                {
                    this.degree = value;

#if DEBUG
                    TestInvariant();
#endif
                    ToDecimalForm();
                }
            }

            public int Minute
            {
                                                get
                {
                    return this.minute;
                }
                                                set
                {
                    this.minute = value;
#if DEBUG
                    TestInvariant();
#endif
                    ToDecimalForm();
                }
            }

            public int Second
            {
                                                get
                {
                    return this.second;
                }
                                                set
                {
                    this.second = value;
#if DEBUG
                    TestInvariant();
#endif
                    ToDecimalForm();
                }
            }

            public double Coordinate
            {
                                                get
                {
                    return this.radCoordinate;
                }
                                                set
```

```
				{
					this.radCoordinate = value;
					ToCanonicalForm();
#if DEBUG
					TestInvariant();
#endif
				}
			}
									#endregion
		}

		public class MercatorLatitude : GeoCoordinate
		{
			private const int MAX_LATITUDE = 90;

			public MercatorLatitude(int degree, int minute, int second) :
base(degree, minute, second, MAX_LATITUDE)
			{
			}

			public MercatorLatitude(double latitude) : base(latitude,
MAX_LATITUDE)
			{
			}

			// default constructor
			public MercatorLatitude() : base(0, 360)
			{
			}

			public override string ToString()
			{
				StringBuilder str = new StringBuilder();
				str.AppendFormat("{0}\u00B0 {1}\u0027 {2}\u0022",
Math.Abs(this.degree), Math.Abs(this.minute), Math.Abs(this.second));
				if (this.radCoordinate < 0)
				{
					str.Append("S");
				}
										else
				{
					str.Append("N");
				}
				return str.ToString();
			}
		}

		public class MercatorLongitude : GeoCoordinate
		{
			private const int MAX_LONGITUDE = 180;

			public MercatorLongitude(int degree, int minute, int second) :
base(degree, minute, second, MAX_LONGITUDE)
			{
			}
```

```
            public MercatorLongitude(double longitude) : base(longitude,
MAX_LONGITUDE)
            {
            }

            // default constructor
            public MercatorLongitude() : base(0, 360)
            {
            }

            public override string ToString()
            {
                  StringBuilder str = new StringBuilder();
                  str.AppendFormat("{0}\u00B0 {1}\u0027 {2}\u0022",
Math.Abs(this.degree), Math.Abs(this.minute), Math.Abs(this.second));
                  if (this.radCoordinate < 0)
                  {
                        str.Append("W");
                  }
                                                      else
                  {
                        str.Append("E");
                  }
                  return str.ToString();
            }
      }

      public struct CartesianPair
      {
            public double x;
            public double y;

            public static CartesianPair TransformToImageCentric(CartesianPair
pair, int width, int height)
            {
                  CartesianPair result = new CartesianPair();
                  result.x = pair.x + width/2;
                  result.y = pair.y - height/2;

                  return result;
            }

            public static CartesianPair TransformToMapCentric(CartesianPair
pair, int width, int height)
            {
                  CartesianPair result = new CartesianPair();
                  result.x = pair.x - width/2;
                  result.y = pair.y + height/2;

                  return result;
            }
      }

      public class ImageInfo
```

```
	{
		private readonly string			url;
		private readonly int			pxWidth;
		private readonly int			pxHeight;
		private readonly double			degBeginLatitude;
		private readonly double			degBeginLongitude;
		private readonly double			degEndLatitude;
		private readonly double			degEndLongitude;
		private readonly DateTime		dateTimeStamp;
		private readonly ProjectionType	projection;

		public ImageInfo(int pxWidth, int pxHeight, double
degBeginLatitude, double degBeginLongitude,
			double degEndLatitude, double degEndLongitude, string
dateTime, string url, ProjectionType projection)
		{
			this.pxWidth			= pxWidth;
			this.pxHeight			= pxHeight;
			this.degBeginLatitude	= degBeginLatitude;
			this.degEndLatitude		= degEndLatitude;
			this.degBeginLongitude	= degBeginLongitude;
			this.degEndLongitude	= degEndLongitude;
			this.projection			= projection;
			this.url				= url;

			// separate date and time
			if (dateTime == null)
			{
				throw new ArgumentNullException("dateTime");
			}

			string [] split = dateTime.Split(':');
			if (split != null)
			{
				// process date
				string [] dateSplit = split[0].Split('/');

				// process time
				string [] timeSplit = null;
				if (split[1] != null)
				{
					timeSplit = split[1].Split('.');
				}
								else
				{
					throw new ArgumentException("Malformatted date-
time string", "dateTime");
				}

				dateTimeStamp = new DateTime(int.Parse(dateSplit[2]),
int.Parse(dateSplit[0]), int.Parse(dateSplit[1]),
					int.Parse(timeSplit[0]),
int.Parse(timeSplit[1]), int.Parse(timeSplit[2]));
			}
							else
```

```
                {
                        throw new ArgumentException("Malformatted date-time
string", "dateTime");
                }

        }


        // default constructor
        public ImageInfo(){}

        public int Width
        {
                                                get
                {
                        return pxWidth;
                }
        }

        public int Height
        {
                                                get
                {
                        return pxHeight;
                }
        }

        public double BeginLatitude
        {
                                                get
                {
                        return degBeginLatitude;
                }
        }

        public double EndLatitude
        {
                                                get
                {
                        return degEndLatitude;
                }
        }

        public double BeginLongitude
        {
                                                get
                {
                        return degBeginLongitude;
                }
        }

        public double EndLongitude
        {
                                                get
                {
```

```
				return degEndLongitude;
			}
		}

		public DateTime DateTimeStamp
		{
										get
			{
				return dateTimeStamp;
			}
		}

		public ProjectionType Projection
		{
										get
			{
				return projection;
			}
		}

		public string Url
		{
										get
			{
				return url;
			}
		}
	}

	public class ColorTempPair : IComparable
	{
		private int color; // in ARGB
		private float	temperature;

		public ColorTempPair(int color, float temperature)
		{
			this.color = color;
			this.temperature = temperature;
		}

		// default
		public ColorTempPair(){}

		public int CurrentColor
		{
										get
			{
				return color;
			}
		}

		public float Temperature
		{
										get
			{
```

```
                return temperature;
            }
        }

        #region IComparable Members

        public int CompareTo(object obj)
        {
            // TODO:  Add ColorTempMap.CompareTo implementation
            if (obj == null)
            {
                return 0;
            }

            // types must be uniform
            if (obj.GetType() != this.GetType())
            {
                throw new ArgumentException( );
            }

            ColorTempPair rhs = (ColorTempPair)obj;
            // sort by ARGB
            if (this.color == rhs.color)
            {
                return 0;
            }
            else if (this.color > rhs.color)
            {
                return 1;
            }
                                            else
            {
                return -1;
            }
        }

                                        #endregion
    }

    public class TemperatureComparer : IComparer
    {
        int IComparer.Compare(object x, object y)
        {
            if (x is ColorTempPair && y is ColorTempPair)
            {
                ColorTempPair pair1 = x as ColorTempPair;
                ColorTempPair pair2 = y as ColorTempPair;

                if (pair1.Temperature == pair2.Temperature)
                {
                    return 0;
                }
                else if (pair1.Temperature > pair2.Temperature)
                {
                    return 1;
```

```
                        }
                                                        else
                        {
                              return -1;
                        }
                  }
                                                  else
                  {
                        throw new InvalidCastException("Expected
ColorTempPair");
                  }
            }
      }
}
```

Selected Client Source Code

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.Net;
using System.Text;
using GraphicsServer.GSNet.Charting;
using GraphicsServer.GSNet.SeriesData;
using System.IO;
using System.Threading;

namespace ImageDemo
{
      /// <summary>
      /// Summary description for Form1.
      /// </summary>
      public class Form1 : System.Windows.Forms.Form
      {
            private System.Windows.Forms.Button btnDownloadManifest;
            private System.Windows.Forms.TextBox tbTemperature;
            private System.Windows.Forms.ListBox lbImages;
            private System.Windows.Forms.StatusBar statusBar;

            private ImageDemo.TemperatureData.TemperatureData td = null;
            private Image [] images = null;
            private int currImage = -1;
            private Bitmap memBmp = new Bitmap(800, 600);
            private Rectangle rect = new Rectangle(10, 170, 800, 600);
            private System.Windows.Forms.TextBox tbPosition;
            private System.Windows.Forms.RadioButton rbShowPoint;
            private System.Windows.Forms.GroupBox groupBox1;
            private System.Windows.Forms.RadioButton rbShowGraph;
            private System.Windows.Forms.Label label3;
            private System.Windows.Forms.Label lbTemperature;
```

```
		private System.Windows.Forms.Label lbPosition;

		private Form2 chartHolder = null;


		/// <summary>
		/// Required designer variable.
		/// </summary>
		private System.ComponentModel.Container components = null;

		public Form1()
		{
			//
			// Required for Windows Form Designer support
			//
			InitializeComponent();

			//
			// TODO: Add any constructor code after InitializeComponent
call
			//
		}

		/// <summary>
		/// Clean up any resources being used.
		/// </summary>
		protected override void Dispose( bool disposing )
		{
			if( disposing )
			{
				if (components != null)
				{
					components.Dispose();
				}
			}
			base.Dispose( disposing );
		}

		/// <summary>
		/// The main entry point for the application.
		/// </summary>
		[STAThread]
		static void Main()
		{
			Application.Run(new Form1());
		}

		private void DownloadFile(string remoteUri, string fileName)
		{
			// download file specified in text boxes
			WebClient myWebClient = new WebClient();

			// Download XML file and save it into the current
filesystem folder.
			try
```

```
                {
                        myWebClient.DownloadFile(remoteUri, fileName);
                }
                catch (WebException exception)
                {
                        MessageBox.Show(exception.Message, "Error",
MessageBoxButtons.OK,MessageBoxIcon.Exclamation);
                }
        }

        private void DrawImage()
        {
                //    Rectangle rect = new Rectangle(10,,800,600);

                // for double buffering
                Graphics memGr = Graphics.FromImage(memBmp);

                if(this.images != null && this.images[currImage] != null)
                {
                        memGr.DrawImageUnscaled(this.images[currImage], 0,
0);
                }

                // do bit blit
                Graphics g = CreateGraphics();
                g.DrawImageUnscaled(memBmp, this.rect);
                g.Dispose();
        }

        private void Form1_Load(object sender, System.EventArgs e)
        {
                td = new ImageDemo.TemperatureData.TemperatureData();

                this.rbShowPoint.Checked = true;
        }

        private void btnDownloadManifest_Click(object sender,
System.EventArgs e)
        {
                int count = td.GetNumberOfImages();
                images = new Image[count];

                this.lbImages.BeginUpdate();
                for (int i = 0; i < count; i++)
                {
                        DateTime dt = td.GetDateTimeStamp(i);
                        this.lbImages.Items.Add(dt);
                }
                this.lbImages.EndUpdate();
        }

        private void lbImages_MouseUp(object sender,
System.Windows.Forms.MouseEventArgs e)
        {
                // download selected image if not already downloaded
```

```
			int index = this.lbImages.SelectedIndex;
			if (images[index] == null)
			{
				GetImage(index);
			}

			// cause image to paint
			currImage = index;
			this.Invalidate();
		}

		private void GetImage(int index)
		{
			string url = td.GetImageLocation(index);
			this.statusBar.Text = "Downloading" + url + "...";
			string fileName = String.Format("Image{0}.gif", index);
			DownloadFile(url, fileName);
			this.statusBar.Text = "Image Downloaded!";

			// open image
			images[index] = Image.FromFile(Application.StartupPath +
"\\" + fileName);
		}

		protected override void OnPaint(PaintEventArgs e)
		{
			// TODO:  Add Form1.OnPaint implementation
			base.OnPaint (e);
			DrawImage();
		}

		private void Form1_MouseDown(object sender,
System.Windows.Forms.MouseEventArgs e)
		{
			// if image is clicked
			if (currImage > -1  && e.X >= rect.Left && e.X <
(images[currImage].Width+rect.Left) &&
				e.X >= rect.Left && e.X <
(images[currImage].Width+rect.Left) &&
				e.Y >= rect.Top  && e.Y <
(images[currImage].Height+rect.Top))
			{
				if (e.Button == MouseButtons.Left)
				{
					// calculate position
					this.tbPosition.Text =
td.ConvertPixelToMercatorCoordinate(e.X-rect.Left, e.Y-rect.Top,
this.currImage);

					if (this.rbShowPoint.Checked)
					{
						// calculate temperature
						Bitmap bmp = images[currImage] as Bitmap;
						Color color = bmp.GetPixel(e.X-rect.Left,
e.Y-rect.Top);
```

```
                                        float temperature =
td.ConvertColorToTemperature(color.ToArgb());
                                        tbTemperature.Text =
temperature.ToString() + "\u00B0" + "F";
                                }
                                else
                                {
                                        // calculate temperature graph
                                        float [] temps = new
float[images.Length];
                                        for (int i = 0; i < images.Length; i++)
                                        {
                                                if (images[i] == null)
                                                {
                                                        GetImage(i);
                                                }

                                                Bitmap bmp = images[i] as Bitmap;
                                                Color color = bmp.GetPixel(e.X-
rect.Left, e.Y-rect.Top);
                                                temps[i] =
td.ConvertColorToTemperature(color.ToArgb());
                                        }

                                        if (chartHolder == null)
                                        {
                                                chartHolder = new Form2();
                                                chartHolder.Show();
                                        }

                                        // use an ArrayDataProvider to bind data
to series
                                        ArrayDataProvider prov = new
ArrayDataProvider(temps);

                                        // bind to Y
                                        Series ser = new Series();
                                        ser.BindComponent(SeriesComponent.Y,
prov, string.Empty);

                                        Chart chart =
chartHolder.gsNetWinChart1.Chart;

                                        // remove old data from chart
                                        chart.RemoveAllSeries();
                                        // add new (array) series to chart
                                        chart.AddSeries(ser);

                                        // refresh chart
                                        chart.RecalcLayout();
                                        chartHolder.BringToFront();
                                }
                        }
                }
        }
        private void rbShowPoint_Click(object sender, System.EventArgs e)
        {
              if (this.rbShowPoint.Checked)
              {
                    this.lbTemperature.Visible = true;
                    this.tbTemperature.Visible = true;
              }
        }

        private void rbShowGraph_Click(object sender, System.EventArgs e)
        {
              if (this.rbShowGraph.Checked)
              {
                    this.lbTemperature.Visible = false;
                    this.tbTemperature.Visible = false;
              }
        }
    }
}
```

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;

namespace ImageDemo
{
      /// <summary>
      /// Summary description for Form2.
      /// </summary>
      public class Form2 : System.Windows.Forms.Form
      {
            public GraphicsServer.GSNet.Charting.GSNetWinChart
gsNetWinChart1;
            /// <summary>
            /// Required designer variable.
            /// </summary>
            private System.ComponentModel.Container components = null;

            public Form2()
            {
                  //
                  // Required for Windows Form Designer support
                  //
                  InitializeComponent();

                  //
                  // TODO: Add any constructor code after InitializeComponent
call
                  //
            }

            /// <summary>
            /// Clean up any resources being used.
            /// </summary>
            protected override void Dispose( bool disposing )
            {
                  if( disposing )
                  {
                        if(components != null)
                        {
                              components.Dispose();
                        }
                  }
                  base.Dispose( disposing );
            }

            private void Form2_Load(object sender, System.EventArgs e)
            {
                  this.gsNetWinChart1.Chart.RemoveAllSeries();
                  // refresh chart
                  this.gsNetWinChart1.Chart.RecalcLayout();
            }
      }
}
```

M Spatial Data XML File

```
<?xml version="1.0" encoding="utf-8" ?>
<!-- Color-Temperature map -->
<spatial.1_data.1_descriptor.1>
      <color.4_temperature.1_map.1>
            <color.4_temperature.1_pair.3>
                  <color.1>FFFF00FF</color.1>
                  <temperature.1>-12.5</temperature.1>
            </color.4_temperature.1_pair.3>
            <color.4_temperature.1_pair.3>
                  <color.1>FFBF00FF</color.1>
                  <temperature.1>-7.5</temperature.1>
            </color.4_temperature.1_pair.3>
            <color.4_temperature.1_pair.3>
                  <color.1>FF7F00FF</color.1>
                  <temperature.1>-2.5</temperature.1>
            </color.4_temperature.1_pair.3>
            <color.4_temperature.1_pair.3>
                  <color.1>FF0000FF</color.1>
                  <temperature.1>2.5</temperature.1>
            </color.4_temperature.1_pair.3>
            <color.4_temperature.1_pair.3>
                  <color.1>FF0059FF</color.1>
                  <temperature.1>7.5</temperature.1>
            </color.4_temperature.1_pair.3>
            <color.4_temperature.1_pair.3>
                  <color.1>FF008CFF</color.1>
                  <temperature.1>12.5</temperature.1>
            </color.4_temperature.1_pair.3>
            <color.4_temperature.1_pair.3>
                  <color.1>FF00BFFF</color.1>
                  <temperature.1>17.5</temperature.1>
            </color.4_temperature.1_pair.3>
            <color.4_temperature.1_pair.3>
                  <color.1>FF00FFFF</color.1>
                  <temperature.1>22.5</temperature.1>
            </color.4_temperature.1_pair.3>
            <color.4_temperature.1_pair.3>
                  <color.1>FF00E6CC</color.1>
                  <temperature.1>27.5</temperature.1>
            </color.4_temperature.1_pair.3>
            <color.4_temperature.1_pair.3>
                  <color.1>FF00CC7F</color.1>
                  <temperature.1>32.5</temperature.1>
            </color.4_temperature.1_pair.3>
            <color.4_temperature.1_pair.3>
                  <color.1>FF00B300</color.1>
                  <temperature.1>37.5</temperature.1>
            </color.4_temperature.1_pair.3>
            <color.4_temperature.1_pair.3>
                  <color.1>FF7FCC00</color.1>
```

```
                <temperature.1>42.5</temperature.1>
          </color.4_temperature.1_pair.3>
          <color.4_temperature.1_pair.3>
                <color.1>FFCCE600</color.1>
                <temperature.1>47.5</temperature.1>
          </color.4_temperature.1_pair.3>
          <color.4_temperature.1_pair.3>
                <color.1>FFFFFF00</color.1>
                <temperature.1>52.5</temperature.1>
          </color.4_temperature.1_pair.3>
          <color.4_temperature.1_pair.3>
                <color.1>FFFFCC00</color.1>
                <temperature.1>57.5</temperature.1>
          </color.4_temperature.1_pair.3>
          <color.4_temperature.1_pair.3>
                <color.1>FFFF9900</color.1>
                <temperature.1>62.5</temperature.1>
          </color.4_temperature.1_pair.3>
          <color.4_temperature.1_pair.3>
                <color.1>FFFF6600</color.1>
                <temperature.1>67.5</temperature.1>
          </color.4_temperature.1_pair.3>
          <color.4_temperature.1_pair.3>
                <color.1>FFFF0000</color.1>
                <temperature.1>72.5</temperature.1>
          </color.4_temperature.1_pair.3>
          <color.4_temperature.1_pair.3>
                <color.1>FFCC0000</color.1>
                <temperature.1>77.5</temperature.1>
          </color.4_temperature.1_pair.3>
          <color.4_temperature.1_pair.3>
                <color.1>FF990000</color.1>
                <temperature.1>82.5</temperature.1>
          </color.4_temperature.1_pair.3>
          <color.4_temperature.1_pair.3>
                <color.1>FF660000</color.1>
                <temperature.1>87.5</temperature.1>
          </color.4_temperature.1_pair.3>
    </color.4_temperature.1_map.1>
    <Image.2>

    <url.1>http://localhost/temperatureMapF/other/sfc_con_temp_mod.gif</url
.1>
          <width.1>640</width.1>
          <height.1>425</height.1>
          <begin.7_latitude.2>48.07</begin.7_latitude.2>
          <begin.7_longitude.1>-133.34</begin.7_longitude.1>
          <end.8_latitude.2>21.35</end.8_latitude.2>
          <end.8_longitude.1>-72.32</end.8_longitude.1>
          <date.5_time.6>03/04/2006:17.0.0.0</date.5_time.6>
          <map.1_projection.4>Mercator</map.1_projection.4>
    </Image.2>
    <Image.2>
          <url.1>http://localhost/temperatureMapF/other/sfc_con_temp-
1_mod.gif</url.1>
```

```
            <width.1>640</width.1>
            <height.1>425</height.1>
            <begin.7_latitude.2>48.07</begin.7_latitude.2>
            <begin.7_longitude.1>-133.34</begin.7_longitude.1>
            <end.8_latitude.2>21.35</end.8_latitude.2>
            <end.8_longitude.1>-72.32</end.8_longitude.1>
            <date.5_time.6>03/04/2006:16.0.0.0</date.5_time.6>
            <map.1_projection.4>Mercator</map.1_projection.4>
      </Image.2>
      <Image.2>
            <url.1>http://localhost/temperatureMapF/other/sfc_con_temp-
2_mod.gif</url.1>
            <width.1>640</width.1>
            <height.1>425</height.1>
            <begin.7_latitude.2>48.07</begin.7_latitude.2>
            <begin.7_longitude.1>-133.34</begin.7_longitude.1>
            <end.8_latitude.2>21.35</end.8_latitude.2>
            <end.8_longitude.1>-72.32</end.8_longitude.1>
            <date.5_time.6>03/04/2006:15.0.0.0</date.5_time.6>
            <map.1_projection.4>Mercator</map.1_projection.4>
      </Image.2>
      <Image.2>
            <url.1>http://localhost/temperatureMapF/other/sfc_con_temp-
3_mod.gif</url.1>
            <width.1>640</width.1>
            <height.1>425</height.1>
            <begin.7_latitude.2>48.07</begin.7_latitude.2>
            <begin.7_longitude.1>-133.34</begin.7_longitude.1>
            <end.8_latitude.2>21.35</end.8_latitude.2>
            <end.8_longitude.1>-72.32</end.8_longitude.1>
            <date.5_time.6>03/04/2006:14.0.0.0</date.5_time.6>
            <map.1_projection.4>Mercator</map.1_projection.4>
      </Image.2>
      <Image.2>
            <url.1>http://localhost/temperatureMapF/other/sfc_con_temp-
4_mod.gif</url.1>
            <width.1>640</width.1>
            <height.1>425</height.1>
            <begin.7_latitude.2>48.07</begin.7_latitude.2>
            <begin.7_longitude.1>-133.34</begin.7_longitude.1>
            <end.8_latitude.2>21.35</end.8_latitude.2>
            <end.8_longitude.1>-72.32</end.8_longitude.1>
            <date.5_time.6>03/04/2006:13.0.0.0</date.5_time.6>
            <map.1_projection.4>Mercator</map.1_projection.4>
      </Image.2>
      <Image.2>
            <url.1>http://localhost/temperatureMapF/other/sfc_con_temp-
5_mod.gif</url.1>
            <width.1>640</width.1>
            <height.1>425</height.1>
            <begin.7_latitude.2>48.07</begin.7_latitude.2>
            <begin.7_longitude.1>-133.34</begin.7_longitude.1>
            <end.8_latitude.2>21.35</end.8_latitude.2>
            <end.8_longitude.1>-72.32</end.8_longitude.1>
            <date.5_time.6>03/04/2006:12.0.0.0</date.5_time.6>
            <map.1_projection.4>Mercator</map.1_projection.4>
      </Image.2>
      <Image.2>
            <url.1>http://localhost/temperatureMapF/other/sfc_con_temp-
6_mod.gif</url.1>
            <width.1>640</width.1>
            <height.1>425</height.1>
            <begin.7_latitude.2>48.07</begin.7_latitude.2>
            <begin.7_longitude.1>-133.34</begin.7_longitude.1>
            <end.8_latitude.2>21.35</end.8_latitude.2>
            <end.8_longitude.1>-72.32</end.8_longitude.1>
            <date.5_time.6>03/04/2006:11.0.0.0</date.5_time.6>
            <map.1_projection.4>Mercator</map.1_projection.4>
      </Image.2>
      <Image.2>
            <url.1>http://localhost/temperatureMapF/other/sfc_con_temp-
7_mod.gif</url.1>
            <width.1>640</width.1>
            <height.1>425</height.1>
            <begin.7_latitude.2>48.07</begin.7_latitude.2>
            <begin.7_longitude.1>-133.34</begin.7_longitude.1>
            <end.8_latitude.2>21.35</end.8_latitude.2>
            <end.8_longitude.1>-72.32</end.8_longitude.1>
            <date.5_time.6>03/04/2006:10.0.0.0</date.5_time.6>
            <map.1_projection.4>Mercator</map.1_projection.4>
      </Image.2>
      <Image.2>
            <url.1>http://localhost/temperatureMapF/other/sfc_con_temp-
8_mod.gipcf2 </url.1>
            <width.1>640</width.1>
            <height.1>425</height.1>
            <begin.7_latitude.2>48.07</begin.7_latitude.2>
            <begin.7_longitude.1>-133.34</begin.7_longitude.1>
            <end.8_latitude.2>21.35</end.8_latitude.2>
            <end.8_longitude.1>-72.32</end.8_longitude.1>
            <date.5_time.6>03/04/2006:9.0.0.0</date.5_time.6>
            <map.1_projection.4>Mercator</map.1_projection.4>
      </Image.2>
      <Image.2>
            <url.1>http://localhost/temperatureMapF/other/sfc_con_temp-
9_mod.gif</url.1>
            <width.1>640</width.1>
            <height.1>425</height.1>
            <begin.7_latitude.2>48.07</begin.7_latitude.2>
            <begin.7_longitude.1>-133.34</begin.7_longitude.1>
            <end.8_latitude.2>21.35</end.8_latitude.2>
            <end.8_longitude.1>-72.32</end.8_longitude.1>
            <date.5_time.6>03/04/2006:8.0.0.0</date.5_time.6>
            <map.1_projection.4>Mercator</map.1_projection.4>
      </Image.2>
      <Image.2>
            <url.1>http://localhost/temperatureMapF/other/sfc_con_temp-
10_mod.gpu102 ?</url.1>
            <width.1>640</width.1>
            <height.1>425</height.1>
            <begin.7_latitude.2>48.07</begin.7_latitude.2>
            <begin.7_longitude.1>-133.34</begin.7_longitude.1>
            <end.8_latitude.2>21.35</end.8_latitude.2>
            <end.8_longitude.1>-72.32</end.8_longitude.1>
            <date.5_time.6>03/04/2006:7.0.0.0</date.5_time.6>
            <map.1_projection.4>Mercator</map.1_projection.4>
      </Image.2>
      <Image.2>
            <url.1>http://localhost/temperatureMapF/other/sfc_con_temp-
11_mod.gif</url.1>
            <width.1>640</width.1>
            <height.1>425</height.1>
            <begin.7_latitude.2>48.07</begin.7_latitude.2>
            <begin.7_longitude.1>-133.34</begin.7_longitude.1>
            <end.8_latitude.2>21.35</end.8_latitude.2>
            <end.8_longitude.1>-72.32</end.8_longitude.1>
            <date.5_time.6>03/04/2006:6.0.0.0</date.5_time.6>
            <map.1_projection.4>Mercator</map.1_projection.4>
      </Image.2>
</spatial.1_data.1_descriptor.1>
```