Extending the Hierarchical Systems Knowledge Representation Framework: Interfacing With Geographic Information Systems

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
NOV 13 2008
LIBRARIES

by

Reesa Brooke Phillips

S.B., E.E. & C.S. M.I.T., 2007

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2008

© 2008 Massachusetts Institute of Technology. All rights reserved.

Author .....
Department of Electrical Engineering and Computer Science
May 23, 2008

Certified by.....
Daniel E. Hastings
Professor of Engineering Systems and Aeronautics and Astronautics
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Thesis

ARCHIVES

Extending the Hierarchical Systems Knowledge Representation Framework: Interfacing With Geographic Information Systems

by

Reesa Brooke Phillips

Submitted to the Department of Electrical Engineering and Computer Science
on May 23, 2008, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

This thesis presents the design and implementation of a geographic information systems framework in which engineering systems can be created and analyzed. This framework extends the hierarchical systems knowledge representation framework to allow geospatial information to be attributed to model objects and viewed within geographic information system tools. The addition of geospatial information allows analysts to use spatial analysis to better learn about engineering systems.

Thesis Supervisor: Daniel E. Hastings
Title: Professor of Engineering Systems and Aeronautics and Astronautics

Acknowledgments

First, I would like to thank Professor Hastings for his role as thesis advisor as well as for his advice and continuous support of this project. I would also like to thank Jason Bartolomei for his thesis work from which this project was formed and for being an endless reservoir of ideas and advice. On the software development side, I would like to thank Igor Sylvester for his original design and implementation on which this project was built and Shannon Iyo for his hard work and contributions. Thanks are also due to Jennifer Wilds, David Broniatowski, and Nirav Shah for all of their invaluable contributions. Last, but definitely not least, I would like to thank God through whom all things are possible and my family for their continued support of me and of my work.

Contents

List of Figures

Chapter 1

Introduction

The real world is large and complicated. From the phenomena of nature to human behavior in different environments, complexity abounds. Oftentimes this complexity makes it difficult to understand, navigate, and manipulate the real world. The systems approach to modeling and analyzing the real world demystifies it by addressing some of the complexities. Real world systems exist in time and space and therefore should be temporally and geospatially analyzed for better understanding. For example, to simplify its complexity, a real estate company could be modeled as a system of several people such as buyers, sellers, and real estate agents working together to buy and sell property. To better serve its customers, the real estate company geospatially analyzes property in terms of proximity to grocery stores, schools, commercial zones, crime, etc. This thesis describes an extension of a methodology for performing such geospatial analysis.

Chapter two of the thesis explains the Engineering System Matrix (ESM) methodology used to model and to analyze real world systems.

Chapter three describes the design and implementation of the Frog application, a software tool used to model and to analyze ESMs.

Chapter four leaps into the heart of the work executed for this thesis: the design and implementation of the geospatial information systems (GIS) framework that provides geospatial

capabilities to the Frog application. It also describes the graphical user interface that was created for interacting with the GIS framework.

Chapter five discusses the contributions made by this thesis, lists known issues, and describes future directions of research.

1.1 Background

The work presented in this thesis is built upon previous work accomplished at the Massachusetts Institute of Technology. Jason Bartolomei laid the foundation for this project within his doctoral research [1] by revealing the Engineering Systems Matrix (ESM) as a means to model and to analyze engineering systems. Through his work on the framework for representing hierarchical systems knowledge [8], Igor Sylvester designed and implemented a software analysis tool (Frog) to provide an environment in which Bartolomei's ESM could be created. Shannon Iyo extended and improved upon Sylvester's implementation of Frog in his work by providing a constraint framework for use in analyzing an ESM [4]. This thesis project extends and improves upon both Sylvester and Iyo's implementation of the Frog analysis tool for complex engineering systems.

Collecting, organizing, managing, and interpreting data can be quite daunting especially for engineering systems analysts who frequently acquire copious amounts of data during field work. The Frog systems analysis tool was designed specifically to simplify these tasks. Within the application, collecting, organizing, managing, and interpreting data corresponds to entering, editing, storing, and analyzing data, respectively. Frog allows users to quickly and efficiently enter and edit data in an organized manner and, in turn, stores the data. It also allows users to

define numerical constraints between components in the system model which enables some numerical analysis. However, numerical analysis is only one of many different forms of analysis.

Among the forms of analyses that are desirable to perform on an engineering system is geospatial analysis. An engineering system has parts that exist in time and space so it is important to understand how the spatial properties of these parts change over time and affect the system itself. The goal of the work executed and presented here is to provide geospatial analysis capabilities to the Frog application. Geospatial analysis is achieved within the application by interfacing with geographic information systems (GIS) tools to create an environment in which user defined geospatial data can be attributed to system model components. This environment merges two, namely Frog and Google Earth, in the interest of fostering interesting and new geospatial analyses.

Chapter 2

Engineering Systems and the ESM

2.1 Engineering Systems

Complex systems are everywhere. They exist within many disciplines of study from ecology and economics to politics and finance. Within these disciplines, several useful models have been created to represent and analyze the structure of such complex systems in order to better understand them. Complex systems also exist across disciplines. However, until recently complex systems that cross disciplines have remained unstudied. In 1999, MIT established the Engineering Systems Division (ESD) in order to pursue the study of complex technological systems and products considered in their broader environmental, financial, legal, organizational, and political context. ESD answers the call to better understand complex systems involving interactions across domains by modeling such systems as engineering systems.

An engineering system is a representation of a real-world system. It is comprised of social, technical, functional, process, and other components which interact in different ways. The field of engineering systems involves understanding these different interactions. In this sense, society itself, and most of its sub-structures, are complex adaptive and evolving engineering systems. For example, a jazz club venue could be represented as an engineering system. The social components of the system would be musicians performing, customers enjoying the music and socializing, and employees working at the venue. The technical components include musical instruments, audio systems, computer systems, cash registers, and any other physical parts of the

system. The functional components model the higher level objectives and goals of the system and its parts. In this example, such goals include providing entertainment in the form of jazz music, maintaining a clean environment, and generating profit. Process components model the tasks and activities carried out by the system and its parts in order to accomplish its objectives and goals such as the musicians playing jazz music to provide entertainment and employees cleaning up after customers to maintain a clean environment.

While engineering systems can be readily identified, the difficulty lies in developing a modeling framework for engineering systems in the general sense. Several modeling frameworks exist for the express purpose of describing and providing a means to analyze engineering systems. However, such frameworks fall short of accomplishing their goals. The Design Structure Matrix (DSM) [6], Department of Defense Architecture Framework (DoDAF) [2, 5], and Sussman's Complex Large Integrated Open System (CLIOS) [3, 7] all fail to capture change within the system over time or to model the dynamics across and interactions between all system domains [1]. In addition, neither the DoDAF nor the CLIOS are conducive for quantitative analysis of the system [1]. Because of the complexities of and the requirements to understand engineering systems, existing modeling frameworks are limited in their ability to handle these systems.

2.2 Engineering Systems Matrix

In Bartolomei's PhD thesis, he presented the Engineering Systems Matrix (ESM) as a new modeling framework to address the shortcomings of existing engineering systems models. The ESM models an engineering system as a directed graph of nodes and edges. Nodes represent components of the system; nodes are organized into different classes just as system components

are categorized by type (social, technical, functional, and process). This is important for analysts because it provides structure to their data by grouping components together in a meaningful way.

Directed edges represent the interactions and relationships between these system components. Each edge, also known as a relation, has a label associated with it to describe the type of interaction or relationship it represents. For example, in the engineering system model of the jazz club venue, relations could model the employer-employee relationship that exists between the club owner and a musician regularly booked at the venue (Person A employs Person B), or the musician's responsibilities at the venue (Person B performs Task A).

System components, interactions, and relationships within any engineering system have numerous parameters or characteristics; these are modeled as attributes in the ESM. An attribute is simply an annotation describing a property of that component, interaction, or relationship. Each attribute has a descriptive label and quantitative or qualitative value(s). A musician's musical instruments, amount of musical training, and salary can be modeled as attributes. The relation modeling his responsibility at the venue might have an attribute of duration and frequency.

Using attributes and other elements, the ESM also models one of the most important properties of any engineering system: time. Each quantitative or qualitative value associated with an attribute is designated a time period and represented by attribute elements. Attributes can have multiple attribute elements representing its different values during different time periods. All systems can be expected to change over time so it follows that time dependent dynamics should

play a role in the analysis of a system. This aspect of the ESM allows such temporally sensitive analysis to occur. For example, the musician's salary attribute can be expected to have multiple corresponding values over time due to raises.

Figure 2-1 shows the class diagram for the ESM.

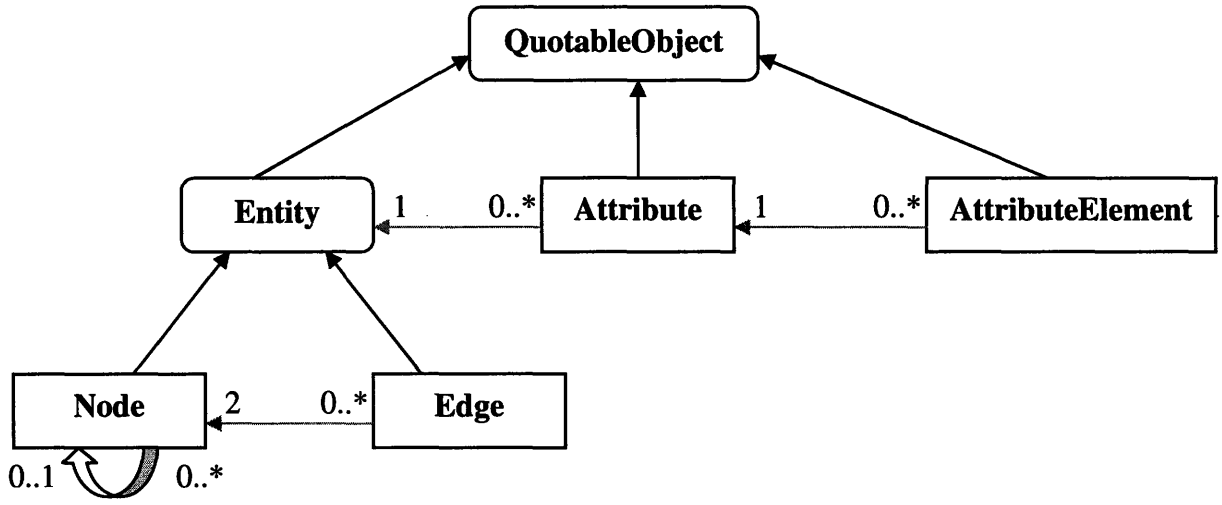


Figure 2-1. Object Model Diagram for Engineering Systems Matrix Model.

Chapter 3

The Frog Analysis Tool

The primary tasks of an engineering analyst are collecting, organizing, and analyzing data. However, engineering systems is a relatively new field of research so there are few if any analysis tools specifically designed for engineering analysts to carry out these tasks. To address this, Bartolomei and Sylvester collaborated to create a software tool to help assist analysts. This tool, known as Frog, is based on the ESM model described in the preceding section and enables engineering analysts to carry out all of their primary tasks. The Frog program is the application on which the geographic information systems framework presented in this thesis is implemented.

3.1 System Design

3.1.1 Data Representation

Frog's ESM data model is designed with a hierarchical structure as described in Sylvester's Masters of Engineering thesis. ESM nodes are represented as nodes in an acyclic graph, also known as a tree. The advantage of employing this hierarchical structure is that it allows nodes to inherit from ancestral nodes. This means that all attributes of an ancestral node are inherited by its children nodes. Such inheritance is desirable to indicate an "is-a" relationship between nodes. For example, a saxophonist is a musician and therefore could be modeled in a hierarchical manner with a node representing a musician as the parent node of that representing a saxophonist. The design choice allows nodes to be grouped together and common attributes to be automatically inherited rather than forcing the user to recreate them.

All nodes within an ESM share one attribute in common and that attribute represents existence. Any node represented in an ESM must exist during some time period. Therefore, the hierarchical structure of each ESM has a single root node with only one attribute to represent existence. This root node is not a tangible entity within the engineering system and has no other purpose than to serve as a starting point for the hierarchy. However, due to the hierarchical structure of the application, the root node serves as the ancestral node from which all nodes inherit existence.

Figure 3-1 shows the hierarchical structure of the application.

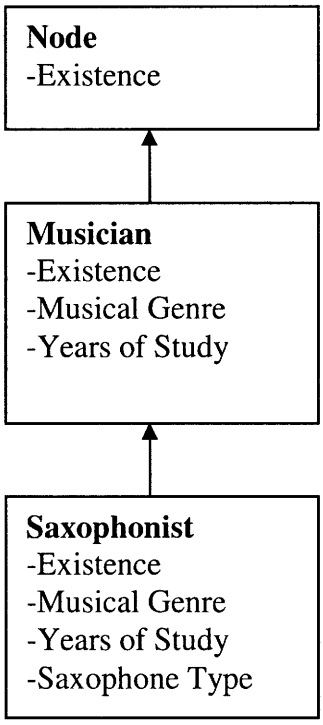


Figure 3-1. An Example of the Hierarchical Structure of the Application. The root node, Node, has an Existence attribute which it passes down to its child node, Musician. Musician has two additional attributes, Musical Genre and Years of Study. These two attributes and the inherited Existence attribute are passed down to its child node, Saxophonist.

3.1.2 Naming Conventions

Naming conventions for referring to nodes, edges, and attributes are important in using the Frog application. Sylvester defined them in his thesis work and they are described again here. The naming convention for nodes resembles that of a file system. Similarly to how the full pathname for a file specifies the path from the root drive, the pathname of a node specifies the path from the root node (which is named "node"):

[node pathname] = [path from root].[node name]

ie. node.child.grandchild

An edge is referred to by the name of the source node appended with '>', the relation name, another '>', and the target node:

[relation pathname] = [source node]>[relation name]>[target node]

ie. node.child1>relation>node.child2

An attribute is referred to by the pathname of the node or relation it describes appended with ':' and the name of the attribute:

[attribute pathname] = [node or relation pathname]:attribute

ie. node.child:attribute or node.child1>relation>node.child2:attribute

Refer to Figure 3-2 to see an example of naming conventions.

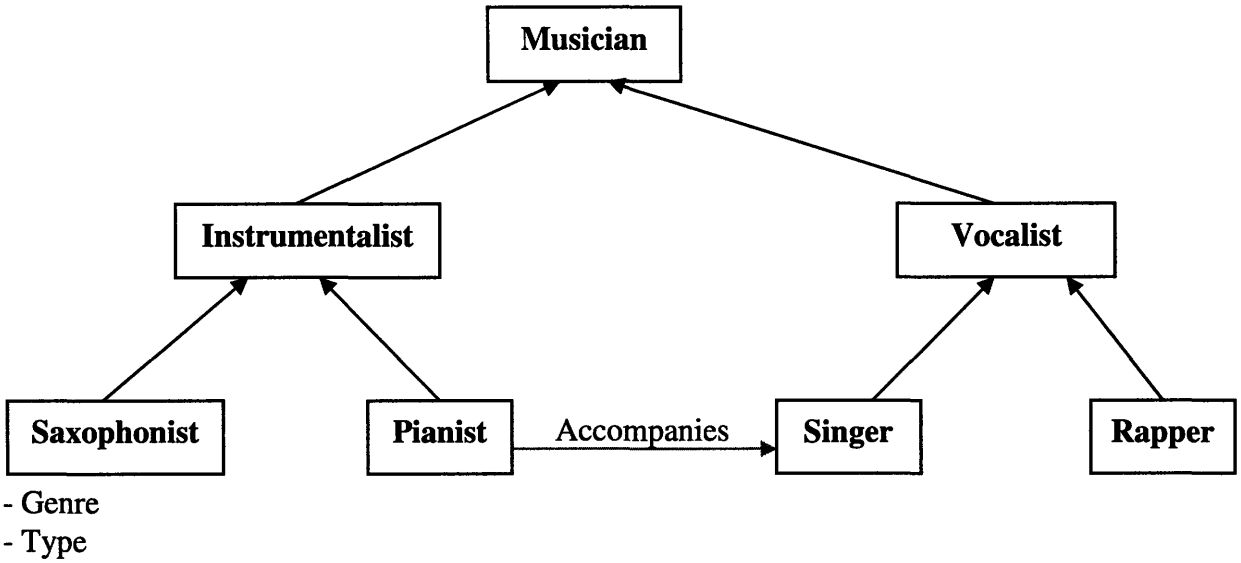


Figure 3-2. Example of Naming Convention in the Application. There are 7 nodes: Musician, Instrumentalist, Vocalist, Saxophonist, Pianist, Singer, and Rapper. There is an Accompanies relation between Pianist and Singer. Saxophonist has two attributes, Genre and Type. The attributes of the Saxophonist are referred to as *Musician.Instrumentalist.Saxophonist: Genre* and *Musician.Instrumentalist.Saxophonist: Type*. The Accompanies relation between Pianist and Singer is referred to as *Musician.Instrumentalist.Pianist>Accompanies>Musican.Vocalist.Singer*.

3.2 System Implementation

The architecture of the Frog application is a server-client design. With the server-client design, several clients can connect to and be updated by a single server which stores and persists the knowledge database. The server uses the Extensible Markup – Remote Procedure Call (XML-RPC) protocol to serve data to its one or more remote client applications. Using one server and separating it from the client allows the data to be stored centrally and accessed remotely by multiple users. The client application is written in Java SE 1.6 using the Swing GUI to provide platform independence and easy deliverability over the Internet through Java Web Start. The server application was originally written in the Franz implementation of Common Lisp to provide flexibility during the prototyping stage of development. However, Iyo converted it to

Java SE 1.6 through Oracle's Berkeley DB Java Edition to resolve the impedance mismatch occurring at the interface between the client and server applications.

3.2.1 Server Architecture

The server architecture that Iyo implemented and presented in his thesis consists of three major parts: the Spidr class, the database wrapper for Berkeley DB, and the XML-RPC server. For each data model object of the ESM, there is a Java class for converting between Java data and Berkeley DB data.

The Spidr class handles all ESM operations and is implemented in `frog.server.Spidr.java`. It contains all model objects and provides all methods to read from or write to the model objects. The Berkeley DB wrapper is implemented in `frog.server.SpidrEnvironment.java` and `frog.server.SpidrDatabase.java`. These classes are responsible for setting up, reading, and writing to the Berkeley DB databases. The XML-RPC server is designed using the Apache XML-RPC library and implemented in `frog.server.RPCServer.java`. The RPCServer class handles remote client requests by forwarding them to Spidr.

3.2.2 Filters

The graphical user interface implemented by Sylvester for the Frog application contains an interactive matrix view of the ESM with nodes represented in the diagonal cells and relations represented in off diagonal cells. The off diagonal cells display the number of relations that exist between two nodes and allow the user to create more. While the matrix view is useful to see the number of relations that exist and their distribution within the ESM, it provides no information

about the types of relations that exist or their associated values during certain time periods. In order to make the matrix view more informative, it was decided that the matrix view needed filter capabilities.

In order to filter the matrix view by relation (and attribute), a class representing a relation filter was created. Each relation filter has the following filtering parameters: name, attribute name, date, source node category, and target node category. The relation filter is implemented in **`frog.model.filter.RelationFilter.java`**. It contains the following methods:

- **`apply(Entity entity)`**: Returns **`true`** if the given **`entity`** is an edge with the same name as that of the relation filter.

- **`isWithinMatrix(Entity entity)`**: Returns **`true`** if the given **`entity`** is an edge with the same source node category and target node category as those of the relation filter.

- **`getValue(Edge edge)`**: Calls the **`isWithinMatrix`** method and if it returns **`false`**, returns **`FILTER_OUT`**[1]. Calls the **`apply`** method and if it returns **`false`**, returns **`NO_VALUE`**[2]. Otherwise, returns the value of the attribute specified by the attribute name of the relation filter for the date specified by the date of the relation filter. Returns **`NO_VALUE`** if the attribute does not exist or have a value for the given date.

[1] **`FILTER_OUT`** is a **`String`** value indicating that the given edge is not within the sub-matrix of the current filter.
[2] **`NO_VALUE`** is a **`String`** value indicating that the given edge is within the sub-matrix but does not have the same name as that of the filter or does not have a value for the given attribute and date.

The filter manager, which is implemented in `frog.gui.filterManager.CreateFilterDialog.java`, is the graphical user interface that creates relation filters. For a selected sub-matrix, the filter manager creates a list of all existing relation names and displays them in a radio button group allowing the user to select the relation by which to filter (as seen in Figure 3-3 below).

Figure 3-3. Create Filter Dialog I.

Once the user selects the desired relation, the manager dialog expands to display a list of all existing attribute names for that relation (as seen in Figure 3-4 below).

Figure 3-4. Create Filter Dialog II.

Finally, when the user selects the desired attribute, the manager dialog expands to display a textbox in which the user can enter a date by which to filter (as seen in Figure 3-5).

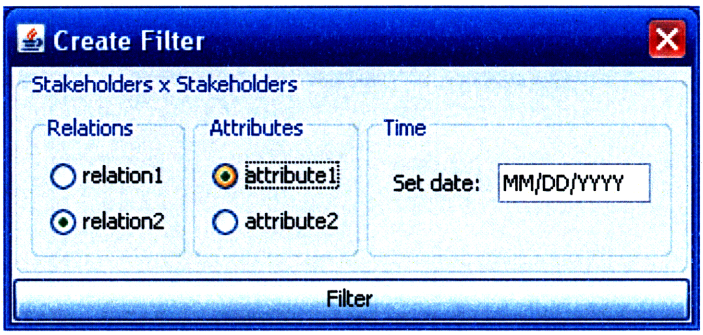


Figure 3-5. Create Filter Dialog III.

The filter is then created with these parameters and the filtered matrix view and filter key panel are updated (as seen in Figures 3-6 and 3-7, respectively).

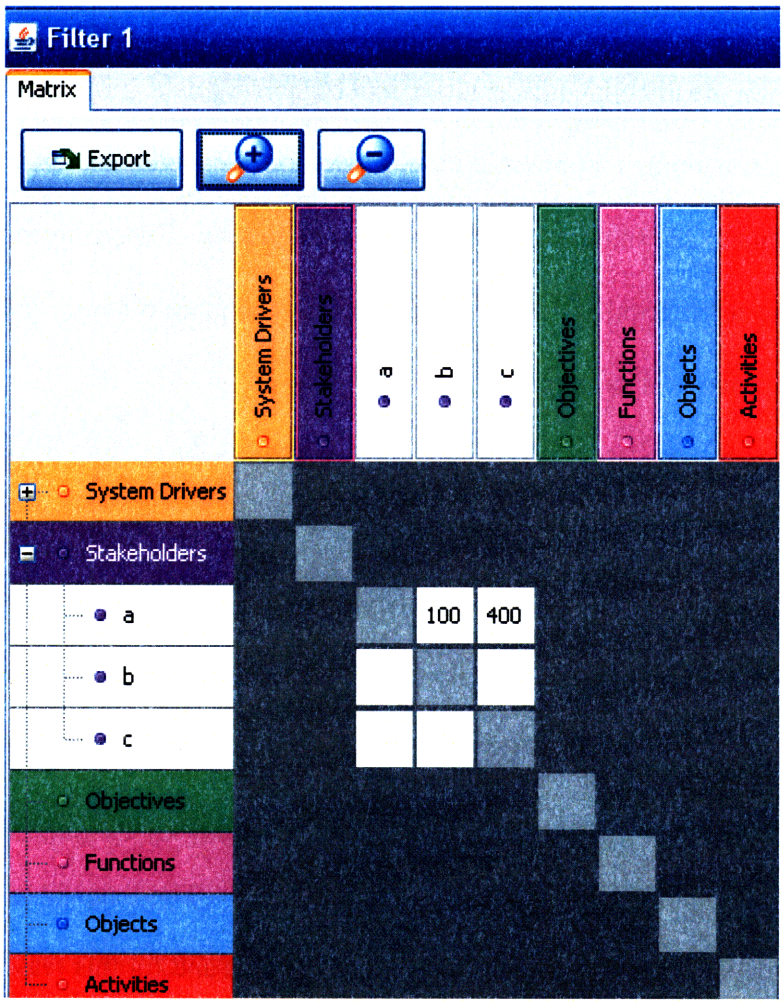


Figure 3-6. Filtered Matrix View.

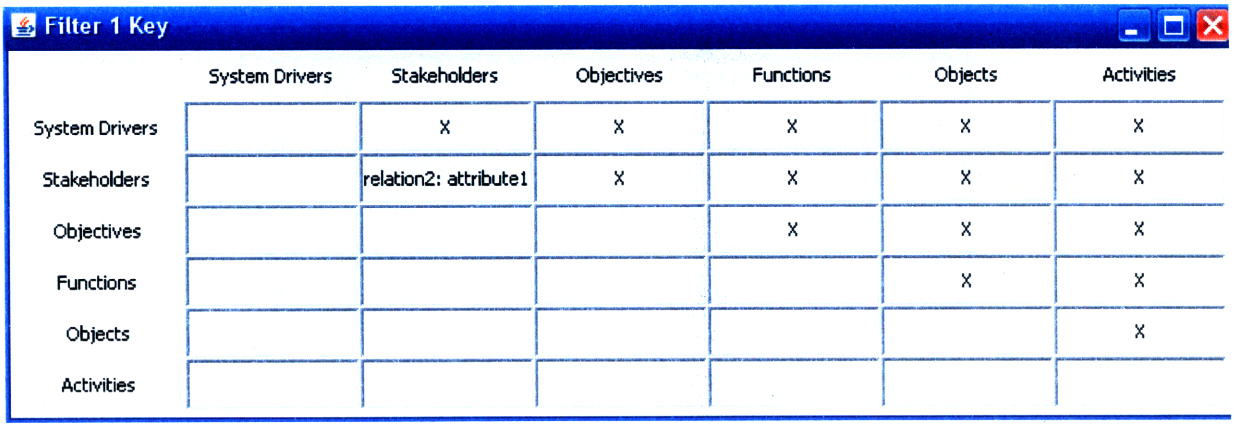


Figure 3-7. Filter Key Panel.

3.2.3 Exporting and Importing

The Frog application greatly expedites entering and organizing data and provides some useful methods of analysis, making it a powerful tool. Most tools, however, gain power by providing means to export data to and import data from other applications. Exporting and importing capabilities became a top priority since this thesis work is about interfacing with other applications to improve analysis capabilities.

There are several useful data analysis tools that already exist so rather than recreating them within the application, Frog can export data to be imported by these analysis tools. From the matrix view of the application, one can export matrix data to a comma separated values (CSV) file in order to import data into Microsoft Excel and analyze data within that environment. However, the matrix excludes so much information that is contained within the ESM model such as attributes of nodes and relations and details about relations. The Extensible Markup Language (XML) is a universally accepted standard for structuring data and for this reason is used to structure the ESM data for exporting. Exporting to XML is implemented within the

`frog.model.Spidr.java`. It contains a method that converts all of the data objects within an ESM into XML elements and writes them to an XML file.

To mirror the exporting function, the importing function accepts XML files. Importing XML files is implemented within `frog.GUI.XMLHandler.XMLParser.java`. To prevent errors from being introduced into the server, each XML file is validated against an XML schema (see Appendix C to view XML schema used). If the XML file is verified to be valid, the XML elements are parsed and used to create ESM objects. Importing capabilities are useful not only to provide a means for importing data from other applications, but also to provide flexibility in entering data. Users have the option of whether to use the Frog application GUI for entering data or to organize their data in an XML file that can then be imported into the application. For a user with programming knowledge, organizing data in an XML file can be expedited with the use of a script, making data entry effortless.

Importing and exporting functionality is important when transporting data between applications, but it is also important in several other ways. Backup and recovery of data, detection of data corruption, and server conversions are all assisted by import and export utilities. Without import and export functionality, all of the existing data would have needed to be reentered when the server application was converted from the Franz implementation of Common Lisp to Oracle's Berkeley DB Java Edition. Instead, the data was simply exported from the old server and imported into the new server through the client application, reducing a task that could easily have taken hours into a matter of minutes.

Chapter 4

Geographic Information System Integration

Engineering analysts can use the Frog analysis tool to enter, edit, and manage data collections. They are also able to perform some numerical analysis using the constraint framework provided. However, it was previously lacking in other forms of analysis such as geospatial analysis. The geographic information system integration is the solution to the lack of geospatial analysis.

4.1 Design

There are two primary goals behind interfacing with a geographic information system. The first goal is to extend the Frog application to allow geospatial information to be attributed to its objects easily. The second goal is to supply a means for that information to be used in a meaningful way.

The first goal of allowing users to add geospatial information to system components and interactions with ease, actually involves two essential design questions. The first is how to represent geospatial information of system objects; the second is the ease with which an analyst can add the geospatial information to system objects. Since the ESM already provides a data model in which attributes model the properties of system components and interactions, the first design decision hinged upon whether or not attributes can adequately represent the geospatial information of system objects. It was determined that they can. Since geospatial information is represented as attributes, users can add geospatial information in the same way they would add any other attribute. However, an environment in which one can view the geospatial information

would provide more flexibility and efficiency. This decision presented the most opportunity to be creative in terms of graphical user interface design, addressing the second design question.

The second goal of using the geospatial information in a meaningful way involved two key decisions. The first was whether to build a tool to analyze the geospatial information or to use an existing GIS tool. The decision was to use an existing GIS tool for the following reasons:

- Any tool created with current Java GIS toolkits to analyze the geospatial information would only provide basic functionality. The dynamic data such as traffic and weather patterns would be lost.

- Several GIS tools provide a means via the Internet through which their users can share their own geographic overlays which could be extremely useful to analysts.

- There are so many open source tools available that are powerful in their own right. Time and resources would be conserved for another task by taking advantage of such tools rather than starting from scratch.

The decision to use an existing GIS tool presented another question – which GIS tool should be used? Due to its popularity and extensive features, Google Earth was chosen. Google Earth has become a very powerful tool over the past few years creating a considerable community of developers. Towards the end of 2006, Google Earth released a beta version of its Application Programming Interface (API) implemented in Component Object Model (COM) [10] (see

Appendix D to view Google Earth API). Also, its Keyhole Markup Language (KML) file format is now an international standard according to the Open Geospatial Consortium (OGC), the organization responsible for setting standards for geospatial and location based services [11].

4.2 Implementation

The geographic information system integration was implemented in Java using Jawin [9], which is a Java/Win32 interoperability library, as an extension of the Frog analysis application. The implementation required changes to the client application. Model and user interface classes were modified in the client and a new GIS graphical user interface was added.

4.2.1 Data Representation

As stated in the design section, attributes are used to represent geospatial information. The design choice is convenient, promotes reuse of code, and allows the graphical user interface to remain internally consistent. Geospatial properties can be added, modified, and removed the same way as other properties of an ESM within the Frog application.

4.2.2 Naming Convention

The attribute representing geospatial information is called "Coordinate". For convenience and to prevent errors in data entry, the coordinate attribute is added to the root node forcing all nodes and relations to inherit geospatial properties. Although this decision breaks the real-world metaphor since all system components do not have physical locations, error prevention

outweighs strictly adhering to the metaphor in this case. The naming convention to refer to the coordinate attribute of a node or relation is as follows:

[Coordinate pathname] = [node or relation pathname]:Coordinate

ie. node.child:Coordinate or node.child1>relation>node.child2:Coordinate

4.2.3 Google Earth Model

As stated before, Google Earth has provided an Application Programming Interface to allow software developers to control the application programmatically. Unfortunately, this API is implemented in Component Object Model (COM). Interfacing between Java and COM is a difficult task because Java is designed to be platform independent while COM is a Microsoft Windows language. The developers of Java, Sun Microsystems, do not provide any resources to facilitate interoperability between Java and COM. Fortunately, there are third party libraries such as the Jacob project and the Java/Win32 interoperability project (Jawin) that handle interfacing between Java and COM.

The model to interface with Google Earth is implemented in `frog.model.GoogleEarthModel.java` using Jawin. The GoogleEarthModel class contains the entry point to the Google Earth application and all methods to interact with it. The following methods are included within the class:

- `getMainHwnd()`: Returns the window handle for the main window of the Google Earth application.

- `getRenderHwnd()`: Returns the window handle for the rendering window, in which all the drawing takes place, of the Google Earth application.

- `getLatitude()`: Returns the latitude coordinate for the center point of the current camera position.

- `getLongitude()`: Returns the longitude coordinate for the center point of the current camera position.

- `setCamera(double latitude, double longitude)`: Sets the camera position to the given latitude and longitude coordinates.

- `openKMLFile(String filename)`: Opens the KML file with the given filename in Google Earth.

- `loadKMLData(String data)`: Loads the given KML data in Google Earth.

4.3 GIS Graphical User Interface

4.3.1 Frog Environment

There were several factors to consider when designing the graphical user interface for adding geospatial coordinates to system components and interactions in the Frog environment. One

important factor was remaining internally consistent. Since geospatial information is modeled as an attribute of all system components and relations, users of the Frog application should be able to modify and remove them in the same way they can remove the attribute representing existence. The Frog application allows a user to input one value for each attribute, but there are two values associated with a geospatial coordinate: longitude and latitude. Therefore, acknowledging that each coordinate has two values, the `frog.gui.googleEarth.CreateCoordinateDialog.java` was created to allow the user to input both values when adding to the coordinate attribute:

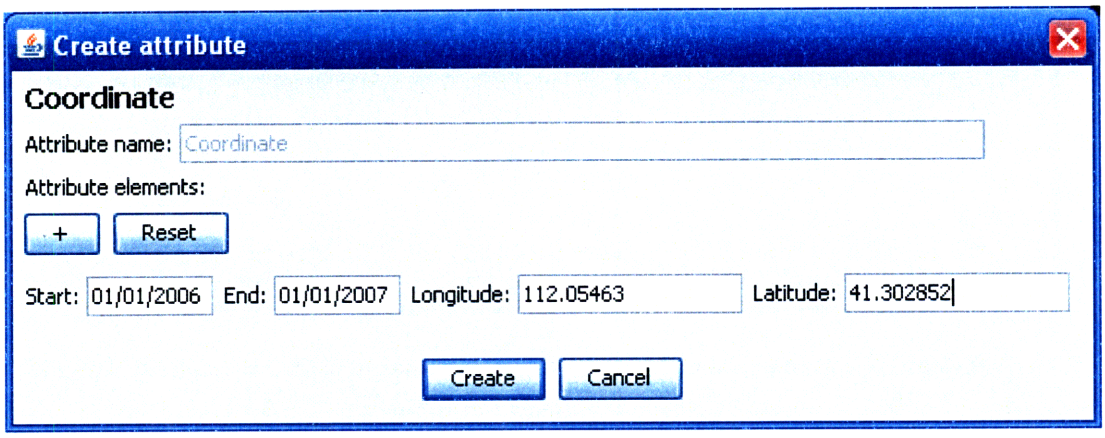


Figure 4-1. Create Coordinate Dialog I.

Although the user enters two values for the coordinate attribute, the two values are represented as one delimited by a comma to remain internally consistent with other attributes (as shown in Figure 4-2).

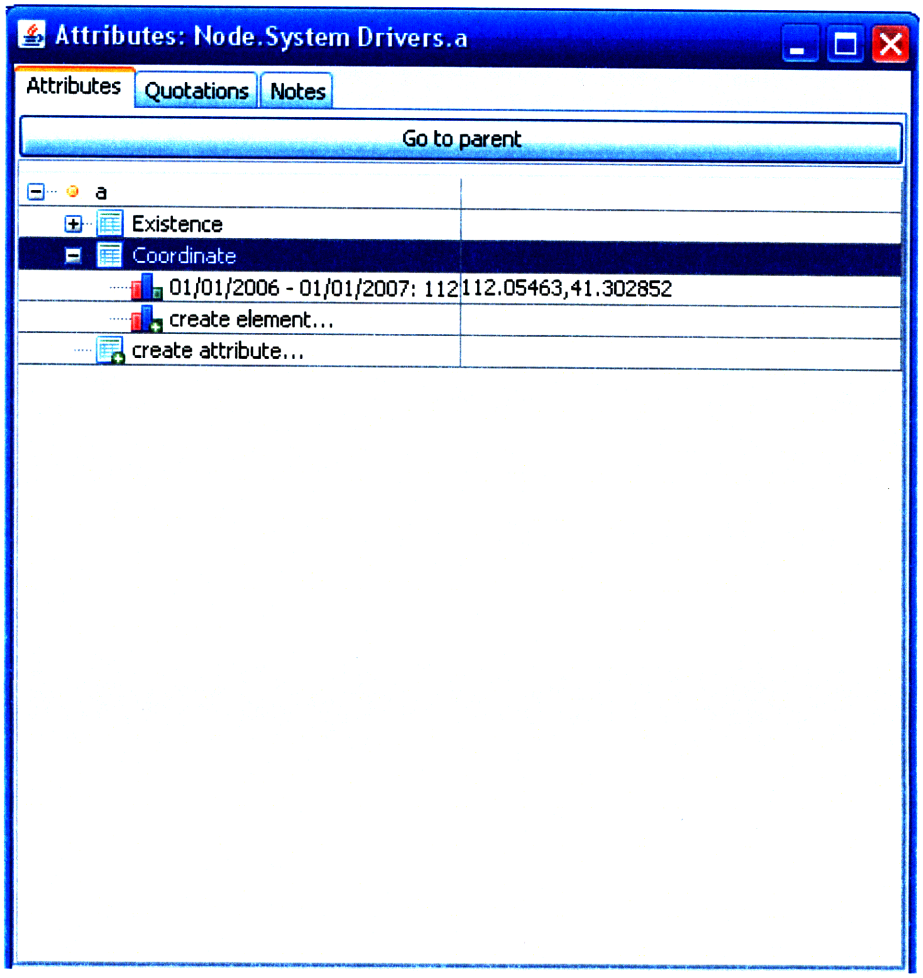


Figure 4-2. Attribute Panel. Coordinate value is represented as one value in the form: longitude, latitude.

4.3.2 Frog/Google Earth Environment

Allowing a user to enter geospatial information in the way detailed in the preceding section is useful, simple, and consistent, but it lacks visual feedback about the actual geospatial location. There is no feedback by which users can verify that the latitude and longitude coordinates they entered correspond with the location they had in mind. Providing visual feedback requires an environment that merges the Frog and Google Earth environments. The environment has three essential parts: the Google Earth view, the Google Earth controller, and the Frog controller.

The actual embedding of a visual of Google Earth is complicated by the different implementing languages, Java and COM. It is further complicated by the fact that there is currently no way provided to fully embed the Google Earth application in another application. Fortunately, the only part of Google Earth needed for the visual is the rendering window in which all the drawing takes place. A workaround within **`frog.gui.GoogleEarthFrame.java`** hides the main window of the Google Earth application and sets the parent of the rendering window to the GoogleEarthFrame window. The rendering window allows the same controls that exist when using Google Earth as a standalone application (ie. left double-click to zoom in, right double-click to zoom out, etc.).

The Google Earth controller is responsible for interacting with the Google Earth view and is implemented in **`frog.gui.googleEarth.GoogleEarthControlPanel.java`**. The Google Earth controller allows a user to enter coordinates to update the view and to get the current coordinates from the view:

Figure 4-3. Google Earth Control Panel.

The Frog controller is responsible for interacting with the Frog application and is implemented in **`frog.gui.googleEarth.CreateNodePanel.java`** and **`frog.gui.googleEarth.CreateEdgePanel.java`**. Using the ESM naming conventions for nodes and edges, the user can add nodes and edges to the ESM or modify existing nodes and edges using the interface in Figure 4-4.

Figure 4-4. Create Node and Edge Panels I.

To reduce the cognitive load on the user, the text field acts as a search box similar to that seen in a search engine, displaying a list of suggestions as the user types.

Figure 4-5. Create Node and Edge Panels II. Possible completions are displayed as the user types.

After entering the node or edge to be added or modified, the CreateCoordinateDialog is displayed with the current coordinates as seen in Figure 4-6 below.

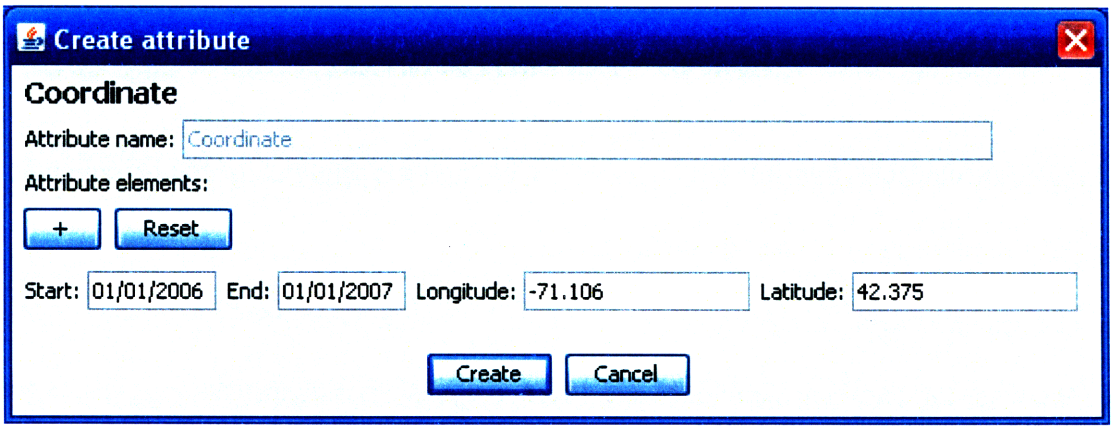


Figure 4-6. Create Coordinate Dialog II. Displays current Google Earth coordinates.

Once the user selects the create button, the application creates the attribute and provides visual feedback by creating a Google Earth placemark at the entered coordinates for the given time interval:

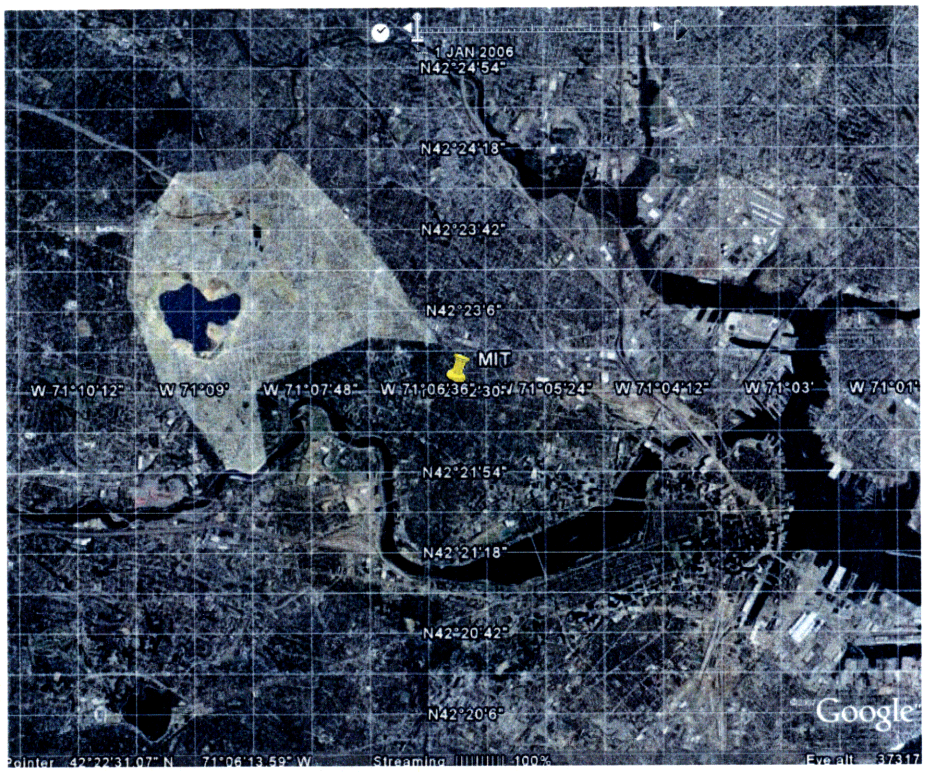


Figure 4-7. Google Earth View. Visual feedback is displayed after creating a coordinate attribute.

The Google Earth view, Google Earth controller, and Frog controller combine to produce the following:

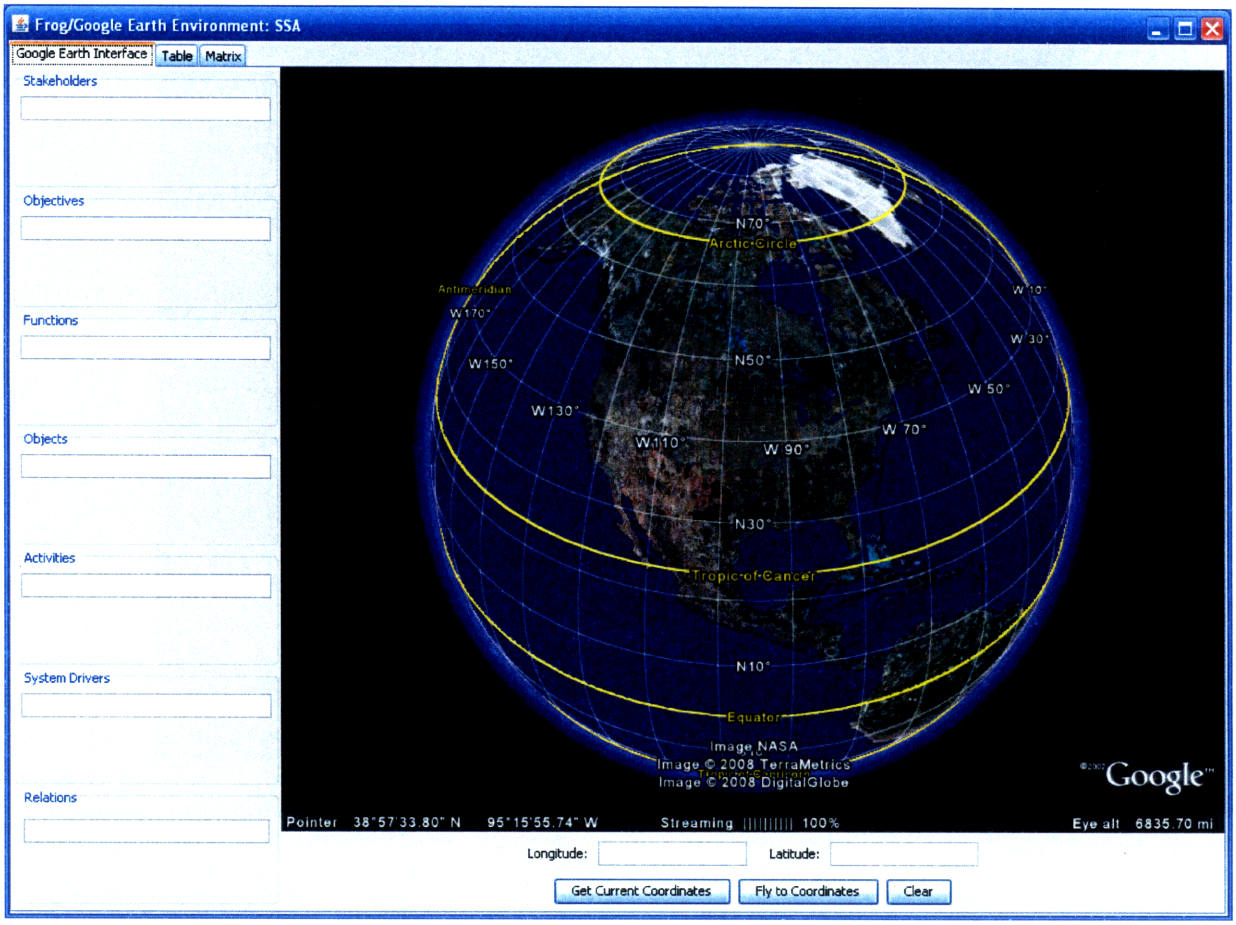


Figure 4-8. Frog/Google Earth Environment.

4.3.3 Exporting Geospatial Information

The preceding sections describe the process of adding geospatial data to an engineering system in the Frog application. Adding geospatial data to an ESM, however, would not be complete without the ability to export that data to a GIS tool for analysis. In order to export data to Google Earth, the coordinate attributes containing the geospatial information must be converted

to KML elements. This conversion is implemented in **`frog.model.Spidr.java`** in the following methods:

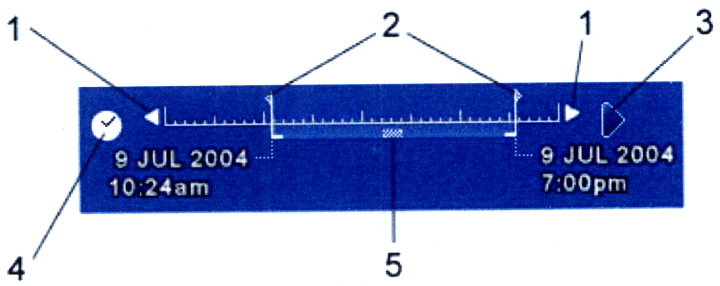
- **`spidrToKML()`**: Converts all objects with geospatial coordinates to KML elements.

- **`spidrToKML(Date date)`**: Converts all objects with geospatial coordinates for the given date to KML elements.

In KML, each element is represented by a placemark object. The following is the KML placemark for a system object named Object A that resided at longitude coordinate 77.24957 and latitude coordinate 38.54278 during the time interval from May 20, 1990 to March 14, 1995:

```
<Placemark>
      <name>a</name>
      <TimeSpan>
            <begin>1990-05-20</begin>
            <end>1995-03-14</end>
      </TimeSpan>
      <Point>
            <coordinates> 77.24957,38.54278,0</coordinates>
      </Point>
</Placemark>
```

When using **`spidrToKML()`**, a placemark is created for each system object for each time interval. This allows the user to view geospatial changes within the system over time. For example, if Object A moves from the coordinates given above to a new position on March 15, 1995 and then moves again on July 4, 2000, three placemarks are created – one for each time interval. In Google Earth, there is a time slider, as seen in Figure 4-9, that allows the user to change the time or time interval displayed.



1. Click these arrows to move the time range earlier or later.
2. Drag these range markers to the right or left to re-define the time range of data displayed.
3. Click this to play an animation of the sequence.
4. Click this to change options for the time slider.
5. Drag this to move the time range earlier or later.

Figure 4-9. Google Earth Time Slider [12].

Using this time slider, the user can view how Object A moves over time, as shown in Figures 4-10, 4-11, 4-12, and 4-13.

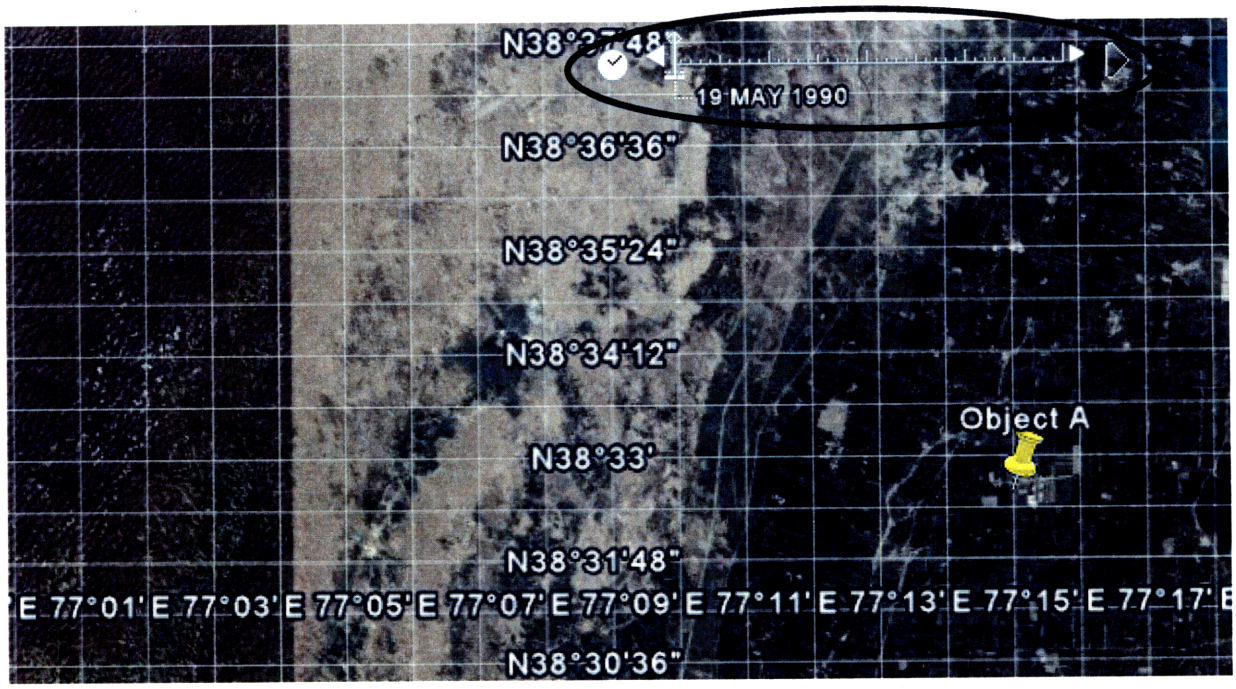


Figure 4-10. Object A on May 19, 1990. Time slider is circled.

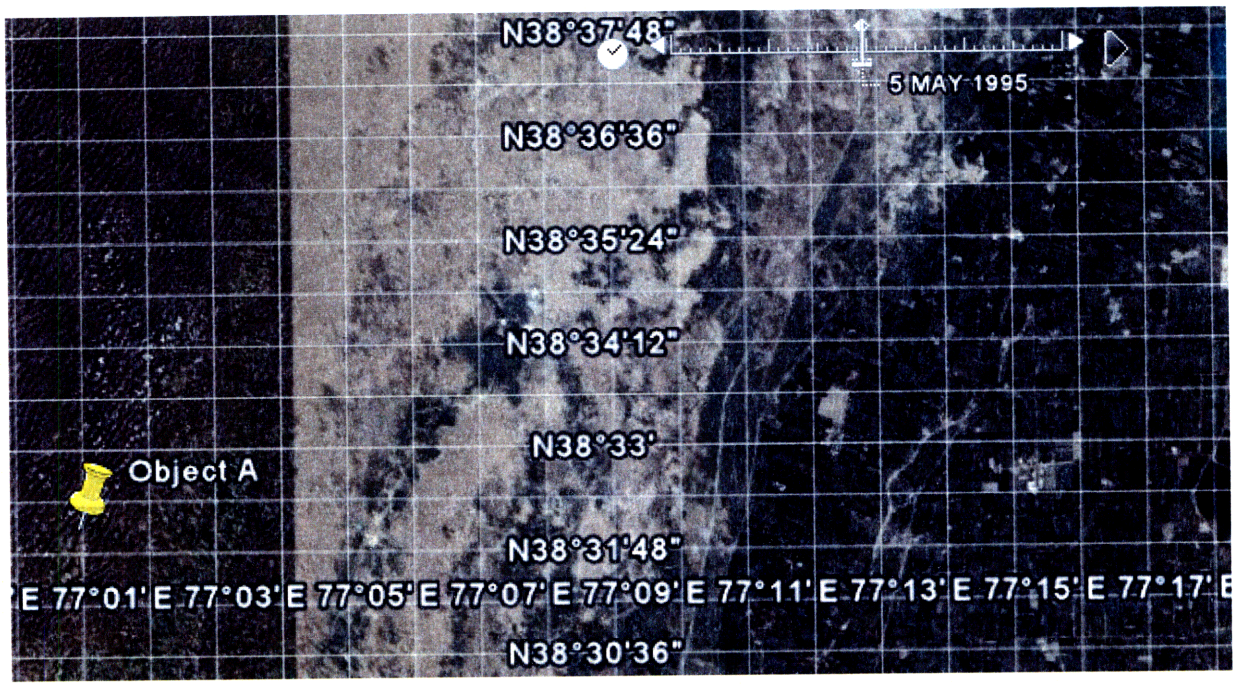


Figure 4-11. Object A on May 5, 1995.

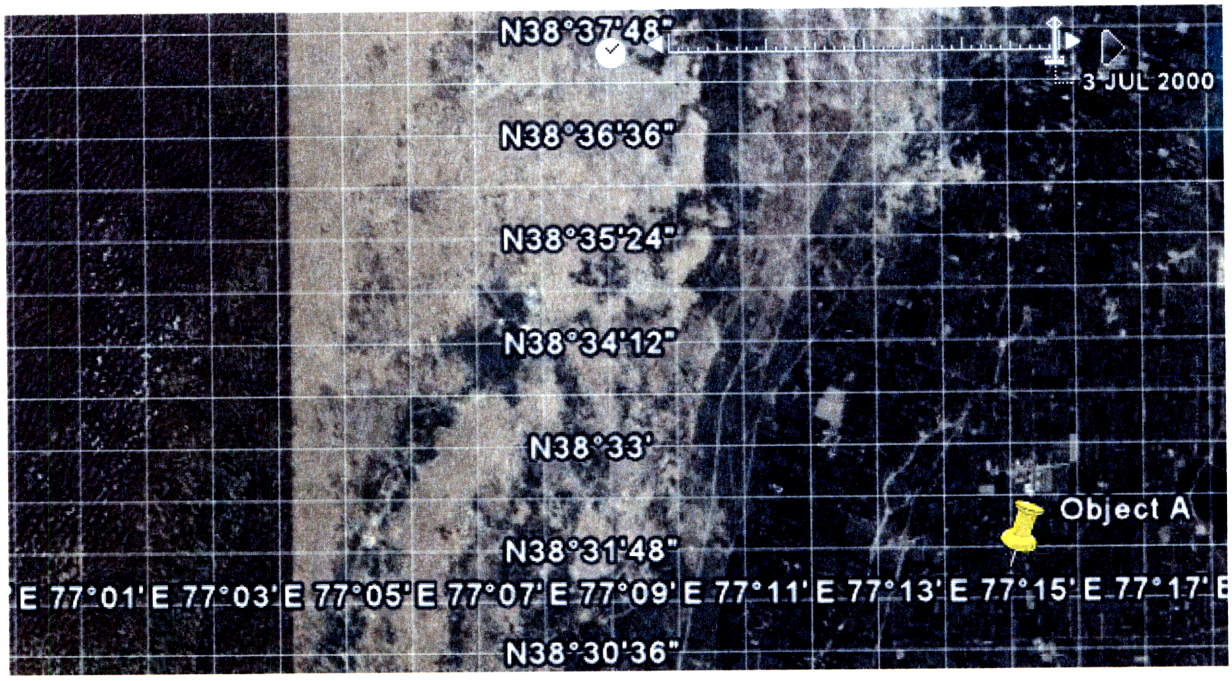


Figure 4-12. Object A on July 3, 2000.

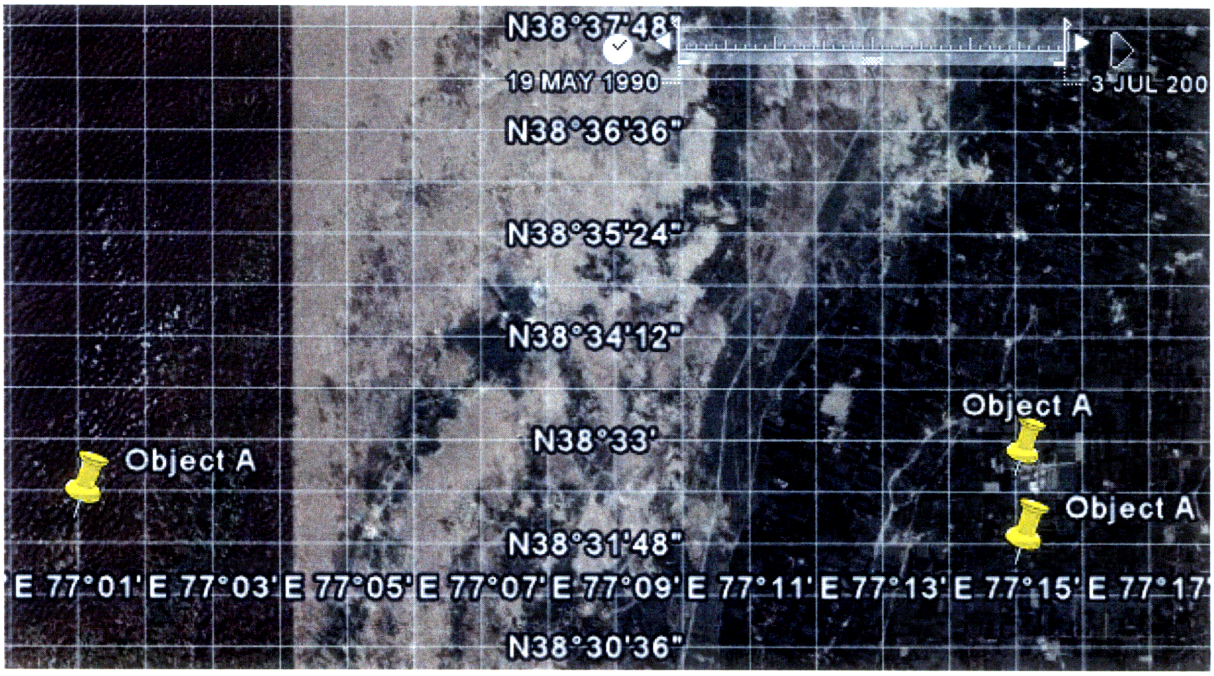


Figure 4-13. Object A between May 19, 1990 and July 3, 2000.

When using `spidrToKML(Data date)`, a placemark is created for each system object for that specific date. This allows the user to view the system in one snapshot of time as seen in Figure 4-14 below.

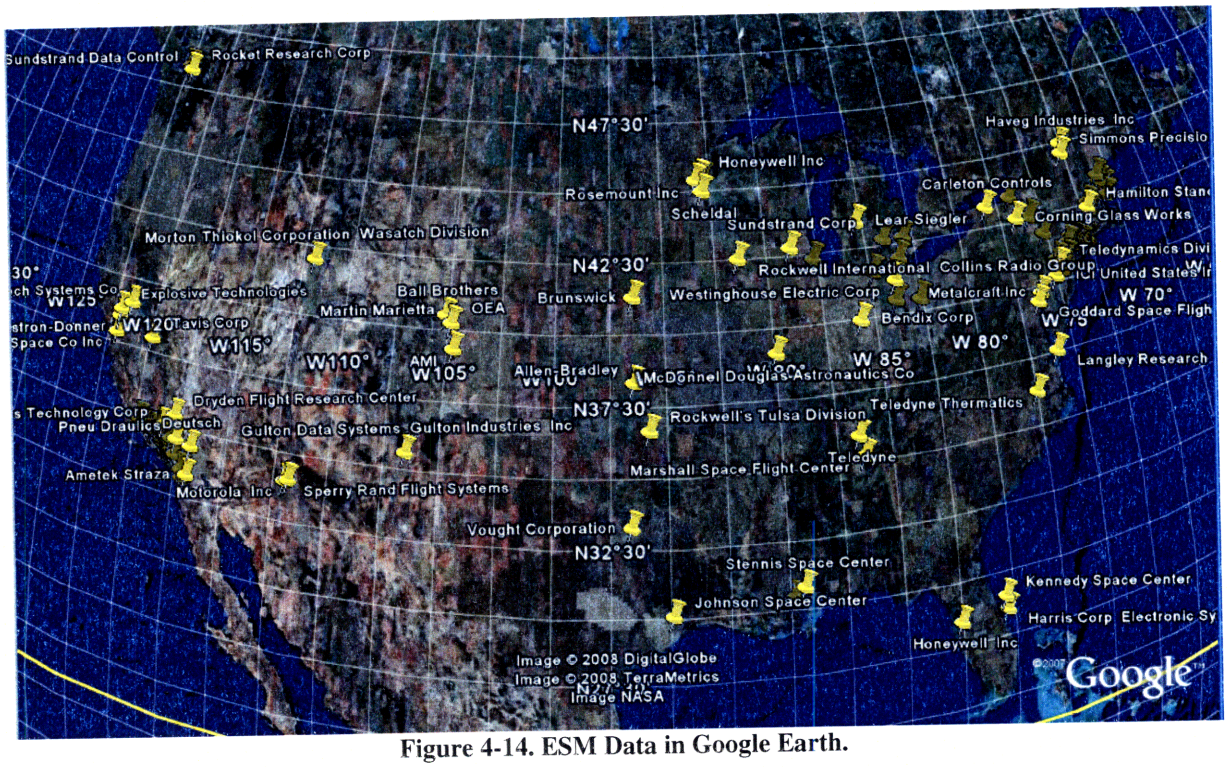


Figure 4-14. ESM Data in Google Earth.

Chapter 5

Conclusion

5.1 Contributions

The work presented here extends the Frog analysis tool with the addition of filters, XML importing and exporting, and geospatial analysis capabilities. The addition of filtering capabilities adds a layer of analysis to the ESM matrix view of the application. Exporting and importing XML files allows flexibility in ESM creation, easy data backup and recovery, and information transfer between Frog and other applications. The geographic information systems integration designed in this thesis creates an environment in which an ESM can be created with geospatial information. It also provides the means to export an ESM for geospatial analysis to powerful GIS tools. In conclusion, this new GIS framework adds a layer of geospatial capability and provides users with greater options for analysis.

5.2 Known Issues

The following is a list of known issues in the implementation of the geographic information system environment that should be addressed in future work:

- The Google Earth API is written in COM which forces the application to rely on the Microsoft Windows platform. Perhaps Google Earth will release APIs for the other platforms it supports, but until then, the GIS environment can only run on Windows

machines. The reliance on Windows takes away from the platform independence achieved from implementing the application in Java.

- Google Earth can only be embedded within an application using a workaround and even then it is not fully embedded. This issue prevents the GIS environment from harnessing the full power of Google Earth and its features.

5.3 Future Work

There are several logical improvements and extensions that can be made to the geospatial information systems framework presented within this thesis.

One possible improvement is to allow users to search for locations without having to know the longitude and latitude coordinates within the Frog/Google Earth Environment. Allowing the user to search would enable more efficiency when finding the coordinates of a location. It would also help geographically challenged users that have no idea where a certain location is. The search could resemble that used by Google Earth as shown in Figure 5-1.

Figure 5-1. Google Earth Search Feature.

A possible extension to the GIS framework is to use another GIS tool for visual feedback when entering coordinates. Google Earth currently has one API and it is written in COM which makes it dependent on the Microsoft Windows platform. Platform independence in the Frog application could be achieved again by using a different GIS tool that has a platform independent API or an API for each widely used platform (Linux, Mac OS X, Microsoft Windows, and Solaris). Using a GIS tool that can be fully embedded would also allow analysis to occur in one environment instead of needing to export data.

Another extension is to provide the ability to use the geospatial information to form geospatial knowledge. Geospatial knowledge can be derived by identifying patterns, relationships, and rules from geospatial information. Perhaps this can be accomplished using the constraint framework which provides mathematical analysis. This might also involve using other GIS tools.

Appendix A

Selected Model Source Code

A.1 RelationFilter.java

```
public class RelationFilter extends DefaultFilter {
    public final static String NO_VALUE = "@NO_VALUE";
    public final static String FILTER_OUT = "@FILTER_OUT";

    private String name = null;
    private int sourceNodeCategory = -1;
    private int targetNodeCategory = -1;
    private String attributeName = "existance";
    private Date date = Date.ANY_DATE;

    public RelationFilter() {
        super();
    }

    public RelationFilter(Edge exampleEdge, String attributeName, Date date) {
        super();
        try {
            name = exampleEdge.getShortName().toLowerCase();
            sourceNodeCategory = exampleEdge.getOutNode().getCategory();
            targetNodeCategory = exampleEdge.getInNode().getCategory();
        } catch (ObjectDeletedException e) {
            e.printStackTrace();
        } catch (ClientConnectionException e) {
            e.printStackTrace();
        }
        this.attributeName = attributeName;
        this.date = date;
    }

    public RelationFilter(String relationName, int sourceCategory, int targetCategory,
            String attributeName, Date date) {
        super();
        name = relationName.toLowerCase();
        sourceNodeCategory = sourceCategory;
        targetNodeCategory = targetCategory;
        this.attributeName = attributeName;
        this.date = date;
    }
```

```
//Returns true if the given entity is an edge with the same relation name as this filter
@Override
public boolean apply(Entity entity) {
    if (entity instanceof Edge) {
        try {
            String edgeName = ((Edge) entity).getShortName().toLowerCase();
            return edgeName.equals(name);
        } catch (ObjectDeletedException e) {
            e.printStackTrace();
        } catch (ClientConnectionException e) {
            e.printStackTrace();
        }
    }
    return false;
}

//Returns true if the given entity is within the submatrix of this filter
public boolean isWithinMatrix(Entity entity){
    if (entity instanceof Edge) {
        try {
            Edge edge = (Edge) entity;
            int sourceCat = edge.getOutNode().getCategory();
            int targetCat = edge.getInNode().getCategory();
            return ((sourceCat == sourceNodeCategory)
                && (targetCat == targetNodeCategory));
        } catch (ObjectDeletedException e) {
            e.printStackTrace();
        } catch (ClientConnectionException e) {
            e.printStackTrace();
        }
    }
    return false;
}

//Returns the value of the given edge
public String getValue(Edge edge) {
    //Returns the filter out value if the given edge is not within the submatrix of this filter
    if (!isWithinMatrix(edge)) { return FILTER_OUT; }

    //Returns no value if the given edge does not apply to this filter
    if (!apply(edge)) { return NO_VALUE; }

    //Returns the value of the corresponding attribute element if it exists
    Attribute attribute = null;
    try {
        for (Attribute a : edge.getAttributes()) {
            if (a.getShortName().equals(attributeName)) {
                attribute = a;
                break;
            }
        }
```

```
            if (attribute == null) { return NO_VALUE; }
            AttributeElement e = attribute.getElement(date);
            if (e != null && !e.isFalse()) {
                return e.getValue();
            } else { return NO_VALUE; }
        } catch (ObjectDeletedException e1) {
            e1.printStackTrace();
        } catch (ClientConnectionException e1) {
            e1.printStackTrace();
        }
        return NO_VALUE;
    }

    public int getSourceNodeCategory() {
        return sourceNodeCategory;
    }

    public int getTargetNodeCategory() {
        return targetNodeCategory;
    }

    public String getAttributeName() {
        return attributeName;
    }

    public String getRelationName() {
        return name;
    }
}
```

A.2 Spidr.java

```
public class Spidr extends StateObject {
    private static final long serialVersionUID = -1409508119309339921L;
    private Map<Integer, DatabaseObject> cache = new HashMap<Integer, DatabaseObject>();
    private BulletinBoard bulletinBoard = new SpidrBulletinBoard();

    public Spidr(Client client, Integer id) {
        super(client, id);
    }

    public Spidr(Client client, String name, List<String> rootNodes)
            throws ClientConnectionException {
        super(client);
        MakeSpidrCommand cmd = new MakeSpidrCommand(client, name, rootNodes);
        cmd.run();
        id = cmd.getId();
    }
```

```
public Spidr(Client client, String name) throws ClientConnectionException {
    super(client);
    MakeSpidrCommand cmd = new MakeSpidrCommand(client, name);
    cmd.run();
    id = cmd.getId();
}

public DatabaseObject parse(String s) throws ClientConnectionException,
        ObjectDeletedException {
    return parse(s, true);
}

public DatabaseObject parse(String s, boolean createIfNotFound)
        throws ClientConnectionException, ObjectDeletedException {
    CreateCommand cmd = new CreateCommand(this, s, createIfNotFound);
    cmd.run();
    return cmd.getSObject();
}

@SuppressWarnings("unchecked")
public List<DatabaseObject> query(String query)
        throws ClientConnectionException {
    Map ret = (Map) client.invoke("spidr.query", new Object[ ] { id, query });
    List<DatabaseObject> result = new ArrayList<DatabaseObject>();
    Object[ ] nodeIDs = (Object[ ]) ret.get("nodes");
    if (nodeIDs != null) {
        for (int i = 0; i < nodeIDs.length; i++) {
            result.add((Node) get((Integer) nodeIDs[i], Node.class));
        }
    }
    Object[ ] edgeIDs = (Object[ ]) ret.get("edges");
    if (edgeIDs != null) {
        for (int i = 0; i < edgeIDs.length; i++) {
            result.add((Edge) get((Integer) edgeIDs[i], Edge.class));
        }
    }
    Object[ ] attIDs = (Object[ ]) ret.get("attributes");
    if (attIDs != null) {
        for (Object attID : attIDs) {
            result.add((Attribute) get((Integer) attID, Attribute.class));
        }
    }
    return result;
}

public Attribute makeAttribute(String name, Entity entity)
        throws ObjectDeletedException, ClientConnectionException {
    MakeAttributeCommand cmd = new MakeAttributeCommand(this, name, entity);
    cmd.run();
    return cmd.getAttribute();
}
```

```
public AttributeElement makeAttributeElement(Attribute attribute,
        TimeInterval timeInterval, String value)
        throws ObjectDeletedException, ClientConnectionException {
    MakeAttributeElementCommand cmd = new MakeAttributeElementCommand
                                                (this, attribute, timeInterval, value);
    cmd.run();
    return cmd.getAttributeElement();
}

public Constraint makeConstraint(String equation, Attribute dependentAttribute)
        throws ObjectDeletedException, ClientConnectionException {

    MakeConstraintCommand cmd = new MakeConstraintCommand(this, equation,
                                                dependentAttribute);
    cmd.run();
    return cmd.getConstraint();
}

public Edge makeEdge(String name, Node in, Node out) throws ObjectDeletedException,
        ClientConnectionException {
    MakeEdgeCommand cmd = new MakeEdgeCommand(this, name, in, out);
    cmd.run();
    return cmd.getEdge();
}

public Node makeNode(String name, Node parent)
        throws ObjectDeletedException, ClientConnectionException {
    MakeNodeCommand cmd = new MakeNodeCommand(this, name, parent);
    cmd.run();
    return cmd.getNode();
}

public Quotation makeDocumentQuote(QuotableObject qObject, Document document,
        int start, int end) throws ObjectDeletedException, ClientConnectionException {
    MakeQuotationCommand cmd = new MakeQuotationCommand(this, qObject,
                                                document, start, end);
    cmd.run();
    return cmd.getDocumentQuote();
}

public Folder makeFolder(String name, Folder parent) throws ObjectDeletedException,
        ClientConnectionException {
    MakeFolderCommand cmd = new MakeFolderCommand(this, name, parent);
    cmd.run();
    return cmd.getFolder();
}

public Document makeDocument(String name, byte[ ] content, Folder parent)
        throws ClientConnectionException, ObjectDeletedException {
    MakeDocumentCommand cmd = new MakeDocumentCommand(this, name, content,
                                                parent);
```

```
    cmd.run();
    return cmd.getDocument();
}

public void flush(DatabaseObject sObject) {
    cache.remove(sObject);
}

@SuppressWarnings("unchecked")
public DatabaseObject reGet(Integer id, Class cl) {
    try {
        DatabaseObject newObject = (DatabaseObject) cl.getConstructor(
        new Class[ ] { Spidr.class, Integer.class }).newInstance(new Object[ ] { this, id });
        cache.put(id, newObject);

        // Subscribe it on the bulletin board
        getBulletinBoard().subscribe(newObject, id);
        return newObject;
    } catch (Exception e) {
        e.printStackTrace();
        throw new RuntimeException();
    }
}

@SuppressWarnings("unchecked")
public DatabaseObject get(Integer id, Class cl) {
    DatabaseObject cachedObject = cache.get(id);
    if (cachedObject != null) {
        return cachedObject;
    } else {
        return reGet(id, cl);
    }
}

public String getName() throws ClientConnectionException, ObjectDeletedException {
    load();
    return (String) data.get("name");
}

public Edge getRootEdge() throws ObjectDeletedException, ClientConnectionException {
    load();
    return (Edge) get((Integer) data.get("rootEdge"), Edge.class);
}

public Folder getRootFolder() throws ObjectDeletedException, ClientConnectionException {
    load();
    return (Folder) get((Integer) data.get("rootFolder"), Folder.class);
}

public Node getRootNode() throws ClientConnectionException {
    return (Node) get((Integer) data.get("rootNode"), Node.class);
```

```
}

public List<Constraint> getConstraints() throws ObjectDeletedException,
        ClientConnectionException {
    load();
    List<Constraint> constraints = new ArrayList<Constraint>();

    Object[ ] constraintIds = (Object[ ]) data.get("constraints");
    if (constraintIds != null) {
        for (Object id : constraintIds) {
            constraints.add((Constraint) get((Integer) id, Constraint.class));
        }
    }

    return constraints;
}

public void remove() throws ClientConnectionException, ObjectDeletedException {
    RemoveSpidrCommand cmd = new RemoveSpidrCommand(this);
    cmd.run();
}

@Override
protected boolean reload2() throws ClientConnectionException {
    super.reload2();
    return true;
}

@Override
public String toString() {
    try {
        return getName();
    } catch (ClientConnectionException e) {
        e.printStackTrace();
        return id.toString();
    } catch (ObjectDeletedException e) {
        return "DELETED";
    }
}

//Returns String of this Spidr in XML form
public String spidrToString() {
    String result = "<?xml version=\"1.0\" encoding=\"ISO-8859-1\"?>\n";
    result += "<frog>\n";
    try {
        result += "<spidr name=\"" + formatString(getName()) + "\" rootNodes=\"";

        //Adds top nodes to String
        List<Node> nodes = getRootNode().getChildren();
        List<Node> temp = new ArrayList<Node>();
        List<Edge> edges = new ArrayList<Edge>();
```

```
for (int i = 0; i < nodes.size(); i++) {
    Node node = nodes.get(i);
    result += formatString(node.getShortName());
    if (i != nodes.size() - 1){ result += ","; }
        temp.addAll(node.getChildren());
        edges.addAll(node.getInEdges());
}
result += "\" />\n";
while (!temp.isEmpty()) {
    Node node = temp.remove(0);

    //Adds each node to String
    result += "<node name=\"" + formatString(node.getShortName())
            + "\" parent=\"" + formatString(node.getParent().getLongName()) + "\">\n";
    List<Attribute> attributes = node.getAttributes();
    if (attributes.isEmpty()){ continue; }

    //Adds the node's attributes and attribute elements to String
    result += "<attributes>\n";
    for (Attribute attr : attributes) {
        result += "<attribute name=\"" + formatString(attr.getShortName()) + "\" />\n";
        List<AttributeElement> elements = attr.getElements();
        for (AttributeElement element : elements) {
            TimeInterval time = element.getTimeInterval();
            result+= "<element name=\"" + formatString(element.getShortName())
                    + "\" start=\"" + time.start.getDateInMs()
                    + "\" end=\"" + time.end.getDateInMs()
                    + "\" value=\"" + formatString(element.getValue()) + "\" />\n";
        }
    }
    result += "</attributes>\n";
    result += "</node>\n";
    if (!node.getChildren().isEmpty()) {
        temp.addAll(node.getChildren());
    }
    edges.addAll(node.getInEdges());
}
for (Edge edge : edges) {

    //Adds each edge to the String
    result += "<edge name=\"" + formatString(edge.getShortName())
            + "\" in=\""      + formatString(edge.getInNode().getLongName())
            + "\" out=\"" + formatString(edge.getOutNode().getLongName()) + "\">\n";
    List<Attribute> attributes = edge.getAttributes();
    if (attributes.isEmpty()){ continue; }

    //Adds the edge's attributes and attribute elements to String
    result += "<attributes>\n";
    for (Attribute attr : attributes) {
        result += "<attribute name=\"" + formatString(attr.getShortName()) + "\" />\n";
        List<AttributeElement> elements = attr.getElements();
```

```
                for (AttributeElement element : elements) {
                    TimeInterval time = element.getTimeInterval();
                    result+= "<element name=\"" + formatString(element.getShortName())
                            + "\" start=\"" + time.start.getDateInMs()
                            + "\" end=\"" + time.end.getDateInMs()
                            + "\" value=\"" + formatString(element.getValue()) + "\" />\n";
                }
            }
            result += "</attributes>\n";
            result += "</edge>\n";
        }
    } catch (ClientConnectionException e) {
        e.printStackTrace();
    } catch (ObjectDeletedException e) {
        e.printStackTrace();
    }
    return result + "</frog>";
}

//Returns String of this Spidr in KML form
public String spidrToKML() {
    String result = "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n"
                    + "<kml xmlns=\"http://earth.google.com/kml/2.2\">\n"
                    + "<Document>\n<open>1</open>\n";
    String firstLatitude = "";
    String firstLongitude = "";
    List<Node> nodes;
    boolean anyPoints = false;
    try {
        nodes = getRootNode().getChildren();
        List<Node> temp = new ArrayList<Node>();
        for (Node node : nodes) {
            temp.addAll(node.getChildren());
        }
        while (!temp.isEmpty()) {
            Node node = temp.remove(0);
            List<Attribute> attributes = node.getAttributes();
            if (attributes.isEmpty()){ continue; }
            for (Attribute attr : attributes) {
                String name = attr.getShortName();
                if (!name.equals("Coordinate")){ continue; }
                List<AttributeElement> elements = attr.getElements();
                if (elements.isEmpty()){ break; }
                for (AttributeElement ae : elements) {

                    //Adds placemark to String for each attribute element of each Coordinate
                    //attribute
                    result += "<Placemark>\n<name>"
                            + node.getShortName() + "</name>\n";
                    anyPoints = true;
                    String[ ] value = ae.getValue().split(",");
```

```
                    if (value.length != 2){ continue; }
                    result += "<Point>\n<coordinates>" + value[0] + ","
                            + value[1] + ",0</coordinates>\n</Point>\n";
                    Date startTime = ae.getTimeInterval().getStart();
                    Date endTime = ae.getTimeInterval().getEnd();
                    String start = startTime.getYear() + "-0" + startTime.getMonth() + "-0"
                                + startTime.getDay();
                    String end = endTime.getYear() + "-0" + endTime.getMonth() + "-0"
                                + endTime.getDay();
                    result += "<TimeSpan><begin>" + start + "</begin><end>"+ end
                            + "</end></TimeSpan>\n";
                    result += "</Placemark>\n";
                    if (firstLatitude.equals("")) {
                        firstLatitude = value[1];
                        firstLongitude = value[0];
                    }
                }
                break;
            }
            if (!node.getChildren().isEmpty()) {
                temp.addAll(node.getChildren());
            }
        }

        //Adds LookAt element to String to set the camera position to the first
        //(longitude, latitude) coordinate in the KML file
        result += "<LookAt>\n<longitude>" + firstLongitude + "</longitude>\n"
                +"<latitude>" + firstLatitude + "</latitude>\n" + "<altitude>300</altitude>\n"
                + "<altitudeMode>clampToGround</altitudeMode>\n" + "</LookAt>\n";
        result += "</Document>\n</kml>";
    } catch (ClientConnectionException e) {
        e.printStackTrace();
    } catch (ObjectDeletedException e) {
        e.printStackTrace();
    }
    return (anyPoints) ? result : "";
}

//Returns String of this Spidr for the given date in KML form
public String spidrToKML(Date d) {
    String result = "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n"
                    + "<kml xmlns=\"http://earth.google.com/kml/2.2\">\n"
                    + "<Document>\n<open>1</open>\n";
    String firstLatitude = "";
    String firstLongitude = "";
    List<Node> nodes;
    boolean anyPoints = false;
    try {
        nodes = getRootNode().getChildren();
        List<Node> temp = new ArrayList<Node>();
        for (Node node : nodes) {
```

```
            temp.addAll(node.getChildren());
        }
        while (!temp.isEmpty()) {
            Node node = temp.remove(0);
            List<Attribute> attributes = node.getAttributes();
            if (attributes.isEmpty()){ continue; }
            for (Attribute attr : attributes) {
                String name = attr.getShortName();
                if (!name.equals("Coordinate")){ continue; }
                AttributeElement ae = attr.getElement(d);
                if (ae == null){ break; }

                //Adds placemark element to String for each attribute element of each
                //Coordinate attribute with the given date
                result += "<Placemark>\n<name>" + node.getShortName()
                        + "</name>\n";
                anyPoints = true;
                String[ ] value = ae.getValue().split(",");
                if (value.length != 2){ break; }
                result += "<Point>\n<coordinates>" + value[0] + "," + value[1]
                        + ",0</coordinates>\n</Point>\n";
                result += "</Placemark>\n";
                if (firstLatitude.equals("")) {
                    firstLatitude = value[1];
                    firstLongitude = value[0];
                }
                break;
            }
            if (!node.getChildren().isEmpty()) {
                temp.addAll(node.getChildren());
            }
        }

        //Adds LookAt element to String to set the camera position to the first
        //(longitude, latitude) coordinate in the KML file
        result += "<LookAt>\n<longitude>" + firstLongitude + "</longitude>\n"
                +"<latitude>" + firstLatitude + "</latitude>\n" + "<altitude>300</altitude>\n"
                + "<altitudeMode>clampToGround</altitudeMode>\n" + "</LookAt>\n";
        result += "</Document>\n</kml>";
    } catch (ClientConnectionException e) {
        e.printStackTrace();
    } catch (ObjectDeletedException e) {
        e.printStackTrace();
    }
    return (anyPoints) ? result : "";
}

private String formatString(String in) {
    if (in == null){ return ""; }
    return in.replace("&", "&amp;");
}
```

```
    @Override
    public Icon getIcon() {
        return null;
    }

    public BulletinBoard getBulletinBoard() {
        return bulletinBoard;
    }
}
```

A.3 XMLParser.java

```
public class XMLParser extends DefaultHandler {
    /** Constants used for JAXP 1.2 */
    public static final String JAXP_SCHEMA_LANGUAGE =
        "http://java.sun.com/xml/jaxp/properties/schemaLanguage";
    public static final String W3C_XML_SCHEMA = "http://www.w3.org/2001/XMLSchema";
    public static final String JAXP_SCHEMA_SOURCE =
        "http://java.sun.com/xml/jaxp/properties/schemaSource";
    private Spidr spidr;
    private Hashtable<String, QuotableObject> quotableObjects =
        new Hashtable<String, QuotableObject>();
    private Client client = null;
    private Entity currentEntity = null;
    private QuotableObject currentQObject = null;
    private Attribute att = null;

    public XMLParser(Client client){
        super();
        this.client = client;
    }

    @Override
    public void startElement(String namespaceURI, String localName, String qName,
        Attributes atts) throws SAXException {

        String name;

        //Parses spidr from XML element
        if (localName=="spidr"){
            name = atts.getValue("name");
            String[ ] st = atts.getValue("rootNodes").split(",");
            List<String> rootNodes = new ArrayList<String>();
            for (String s : st){
                rootNodes.add(s);
            }
            try {
                boolean found = false;
```

```
Spidr[ ] spidrs = client.getSpidrs();

//Searches to see if spidr already exists
for (int i=0; i<spidrs.length; i++){
    try {
        if (spidrs[i].getName()!=null && spidrs[i].getName().equals(name)){
            spidr = spidrs[i];
            found = true;
            break;
        }
    } catch (ObjectDeletedException e) {
        e.printStackTrace();
    }
}

//Creates spidr if it does not exist
if (!found){
    spidr = client.makeSpidr(name, rootNodes);
}

//Loads spidr and all existing nodes, edges, and attributes
try {
    spidr.load();
    List<Node> nodes = spidr.getRootNode().getChildren();
    List<Edge> edges = new ArrayList<Edge>();
    for(Node node : nodes){
        quotableObjects.put(node.getLongName(), node);
        for(Attribute a : node.getAttributes()){
            quotableObjects.put(a.getName(), a);
        }
    }
    while(!nodes.isEmpty()){
        Node node = nodes.remove(0);
        edges.addAll(node.getInEdges());
        if (!node.getChildren().isEmpty()) {
            nodes.addAll(node.getChildren());
            for(Node n : node.getChildren()){
                quotableObjects.put(n.getLongName(), n);
                for(Attribute a : n.getAttributes()){
                    quotableObjects.put(a.getName(), a);
                }
            }
        }
        for(Edge e : edges){
            quotableObjects.put(e.getShortName()
                + e.getInNode().getLongName()
                + e.getOutNode().getLongName(), e);
            for(Attribute a : e.getAttributes()){
                quotableObjects.put(a.getName(), a);
            }
        }
```

```
                edges.clear();
            }
        } catch (ObjectDeletedException e) {
            e.printStackTrace();
        }
    } catch (ClientConnectionException e) {
        e.printStackTrace();
    }
}

//Parses node from XML element
else if (localName=="node"){
    name = atts.getValue("name");
    String parentString = atts.getValue("parent");

    //Checks if node already exists and creates it if it does not
    if (!quotableObjects.containsKey(parentString+"."+name)){
        Node parent = null;
        if (parentString!=null){
            parent = (Node) quotableObjects.get(parentString);
        }
        else {
            System.err.println("Node "+name+"'s parent does not exist: Check XML file.");
        }
        try {
            currentEntity = spidr.makeNode(name, parent);
            currentQObject = currentEntity;
            quotableObjects.put(((Node)currentEntity).getLongName(), currentEntity);
            for(Attribute a : currentEntity.getAttributes()){
                quotableObjects.put(a.getName(), a);
            }
            att = null;
        } catch (ObjectDeletedException e) {
            e.printStackTrace();
        } catch (ClientConnectionException e) {
            e.printStackTrace();
        }
    }
    else {
        currentEntity = (Node) quotableObjects.get(parentString+"."+name);
        currentQObject = currentEntity;
        System.out.println("Node: "+parentString+"."+name+" already exists.");
    }
}

//Parses edge from XML element
else if (localName=="edge"){
    name = atts.getValue("name");
    String inString = atts.getValue("in"), outString = atts.getValue("out");
```

```
    //Checks if edge already exists and creates it if it does not
    if(!quotableObjects.containsKey(name+inString+outString)){
        Node in = null, out = null;
        if (inString!=null && outString!=null){
            in = (Node) quotableObjects.get(inString);
            out = (Node) quotableObjects.get(outString);
        }
        else if (inString==null){
            System.err.println("Edge "+name
                    +"'s corresponding in node does not exist: Check XML file.");
        }
        else if (outString==null){
            System.err.println("Edge "+name
                    +"'s corresponding out node does not exist: Check XML file.");
        }
        try {
            currentEntity = spidr.makeEdge(name, in, out);
            currentQObject = currentEntity;
            quotableObjects.put(name+in.getLongName()+out.getLongName(),
                                    currentEntity);
            for(Attribute a : currentEntity.getAttributes()){
                quotableObjects.put(a.getName(), a);
            }
            att = null;
        } catch (ObjectDeletedException e) {
            e.printStackTrace();
        } catch (ClientConnectionException e) {
            e.printStackTrace();
        }
    }
    else{
        currentEntity = (Edge) quotableObjects.get(name+inString+outString);
        currentQObject = currentEntity;
        System.out.println("Edge: "+name+inString+outString+" already exists.");
    }
}

//Parses attribute from XML element
else if (localName=="attribute"){
    name = atts.getValue("name");
    String entityString = atts.getValue("entity");
    Entity entity = null;
    if (entityString!=null){
        entity = (Entity) quotableObjects.get(entityString);
    }
    else {
        if (currentEntity!=null){
            entity = currentEntity;
        }
        else {
            System.err.println("Attribute "+name
```

```
                        +"'s corresponding entity does not exist: Check XML file.");
            }
        }

        //Checks if attribute's parent entity exists and creates attribute if it does
        if (entity!=null){
            try {
                if(quotableObjects.containsKey(entity.getName()+":"+name)){
                    att = (Attribute) quotableObjects.get(entity.getName()+":"+name);
                }
                else{
                    att = spidr.makeAttribute(name, entity);
                    if (att!=null){
                        quotableObjects.put(entity.getName()+":"+name, att);
                    }
                }
                currentQObject = att;
            } catch (ObjectDeletedException e) {
                e.printStackTrace();
            } catch (ClientConnectionException e) {
                e.printStackTrace();
            }
        }
    }

    //Parses attribute element from XML element
    else if (localName=="element"){
        name = atts.getValue("name");
        String attributeString = atts.getValue("attribute");
        Attribute attribute = null;
        if (attributeString!=null){
            attribute = (Attribute) quotableObjects.get(attributeString);
        }
        else{
            if (att!=null){
                attribute = att;
            }
            else {
                System.err.println("Attribute element "+name
                        +"'s corresponding attribute does not exist: Check XML file.");
            }
        }

        //Checks if attribute exists and creates attribute element if it does
        if(attribute!=null){
            long start = Long.parseLong(atts.getValue("start"));
            long end = Long.parseLong(atts.getValue("end"));
            String value = atts.getValue("value");
            try {
                currentQObject = spidr.makeAttributeElement(attribute,
                                        new TimeInterval(start, end), value);
```

```
                quotableObjects.put(attribute.getName()+name, currentQObject);
            } catch (ObjectDeletedException e) {
                e.printStackTrace();
            } catch (ClientConnectionException e) {
                e.printStackTrace();
            }
        }
    }
}

public static void parse(String[ ] args, Client client) throws ParserConfigurationException,
                                                                SAXException{
    String filename = null;
    boolean dtdValidate = false;
    boolean xsdValidate = false;
    String schemaSource = null;

    // Parse arguments
    for (int i = 0; i < args.length; i++) {
        if (args[i].equals("-dtd")) {
            dtdValidate = true;
        } else if (args[i].equals("-xsd")) {
            xsdValidate = true;
        } else if (args[i].equals("-xsdss")) {
            if (i == args.length - 1) {
                usage();
            }
            xsdValidate = true;
            schemaSource = args[++i];
        } else if (args[i].equals("-usage")) {
            usage();
        } else if (args[i].equals("-help")) {
            usage();
        } else {
            filename = args[i];

            // Must be last arg
            if (i != args.length - 1) {
                usage();
            }
        }
    }
    if (filename == null) {
        usage();
    }

    // Create a JAXP SAXParserFactory and configure it
    SAXParserFactory spf = SAXParserFactory.newInstance();
```

```
    // Set namespaceAware to true to get a parser that corresponds to
    // the default SAX2 namespace feature setting.
    spf.setNamespaceAware(true);

    // Validation part 1: set whether validation is on
    spf.setValidating(dtdValidate || xsdValidate);

    // Create a JAXP SAXParser
    SAXParser saxParser = spf.newSAXParser();

    // Validation part 2a: set the schema language if necessary
    if (xsdValidate) {
        try {
            saxParser.setProperty(JAXP_SCHEMA_LANGUAGE, W3C_XML_SCHEMA);
        } catch (SAXNotRecognizedException x) {
            // This can happen if the parser does not support JAXP 1.2
            System.err.println("Error: JAXP SAXParser property not recognized: "
                    + JAXP_SCHEMA_LANGUAGE);
            System.err.println("Check to see if parser conforms to JAXP 1.2 spec.");
            System.exit(1);
        }
    }

    // Validation part 2b: Set the schema source, if any.
    if (schemaSource != null) {
        saxParser.setProperty(JAXP_SCHEMA_SOURCE,
            XMLParser.class.getResourceAsStream(schemaSource));
    }

    // Get the encapsulated SAX XMLReader
    XMLReader xmlReader = saxParser.getXMLReader();

    // Set the ContentHandler of the XMLReader
    xmlReader.setContentHandler(new XMLParser(client));

    // Set an ErrorHandler before parsing
    xmlReader.setErrorHandler(new MyErrorHandler(System.err));

    // Tell the XMLReader to parse the XML document
    try {
        xmlReader.parse(new File(filename).toURI().toURL().toString());
    } catch (MalformedURLException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

/** Prints error message to console **/
private static void usage() {
    System.err.println("Usage: SAXLocalNameCount [-options] <file.xml>");
```

```
        System.err.println("    -dtd = DTD validation");
        System.err.println("    -xsd | -xsdss <file.xsd> = W3C XML Schema validation using"
                           + "xsi: hints");
        System.err.println("       in instance document or schema source <file.xsd>");
        System.err.println("    -xsdss <file> = W3C XML Schema validation using schema"
                           + "source <file>");
        System.err.println("    -usage or -help = this message");
        System.exit(1);
    }

    //Error handler to report errors and warnings
    private static class MyErrorHandler implements ErrorHandler {
        /** Error handler output goes here */
        private PrintStream out;

        MyErrorHandler(PrintStream out) {
            this.out = out;
        }

        //Returns a string describing parse exception details
        private String getParseExceptionInfo(SAXParseException spe) {
            String systemId = spe.getSystemId();
            if (systemId == null) {
                systemId = "null";
            }
            String info = "URI=" + systemId + " Line=" + spe.getLineNumber() + ": "
                                + spe.getMessage();
            return info;
        }

        // The following methods are standard SAX ErrorHandler methods.
        // Refer to SAX documentation for more info.
        public void warning(SAXParseException spe) throws SAXException {
            out.println("Warning: " + getParseExceptionInfo(spe));
        }

        public void error(SAXParseException spe) throws SAXException {
            String message = "Error: " + getParseExceptionInfo(spe);
            throw new SAXException(message);
        }

        public void fatalError(SAXParseException spe) throws SAXException {
            String message = "Fatal Error: " + getParseExceptionInfo(spe);
            throw new SAXException(message);
        }
    }
}
```

A.4 GoogleEarthModel.java

```
public class GoogleEarthModel {

    private DispatchPtr applicationGE;

    public GoogleEarthModel(){
        try {
            //Launches Google Earth
            applicationGE = new DispatchPtr("GoogleEarth.ApplicationGE");
        } catch (COMException e) {
            e.printStackTrace();
        }
    }

    //Returns the window handle of the main window of Google Earth
    public int getMainHwnd() {
        int mainHwnd = 0;
        try {
            mainHwnd = (Integer) applicationGE.invoke("GetMainHwnd");
        } catch (COMException e) {
            e.printStackTrace();
        }
        return mainHwnd;
    }

    //Returns the window handle of the rendering window of Google Earth
    public int getRenderHwnd() {
        int renderHwnd = 0;
        try {
            renderHwnd = (Integer) applicationGE.invoke("GetRenderHwnd");
        } catch (COMException e) {
            e.printStackTrace();
        }
        return renderHwnd;
    }

    //Returns the current latitude coordinate displayed in Google Earth
    public String getLatitude() throws COMException{
        DispatchPtr point = (DispatchPtr) applicationGE.invoke
                                        ("GetPointOnTerrainFromScreenCoords",0,0);
        return point.get("Latitude").toString();
    }

    //Returns the current longitude coordinate displayed in Google Earth
    public String getLongitude() throws COMException{
        DispatchPtr point = (DispatchPtr) applicationGE.invoke
                                        ("GetPointOnTerrainFromScreenCoords",0,0);
        return point.get("Longitude").toString();
    }
```

```
    //Sets the camera in Google Earth to display the given (longitude, latitude) coordinate
    public void setCamera(double latitude, double longitude) throws COMException{
        Object[ ] args = {latitude, longitude, 200, 1, 200000, 0, 0, 1};
        applicationGE.invokeN("SetCameraParams", args);
    }

    //Opens the KML file with the given filename in Google Earth
    public Object openKMLFile(String filename) throws COMException {
        return applicationGE.invoke("OpenKmlFile",filename,0);
    }

    //Loads the given KML data in Google Earth
    public Object loadKMLData(String data) throws COMException{
        return applicationGE.invoke("LoadKmlData",data);
    }

}
```

Appendix B

Selected Graphical User Interface Source Code

B.1 CreateFilterDialog.java

```
public class CreateFilterDialog extends CreateDialog {

    private static final long serialVersionUID = -8677653442326727408L;
    private Node source, target;
    private List<Node> sourceChildren = new ArrayList<Node>();
    private List<Node> targetChildren = new ArrayList<Node>();
    private List<String> names = new ArrayList<String>();
    private ButtonGroup relationGroup = new ButtonGroup();
    private ButtonGroup attributeGroup = new ButtonGroup();
    private List<JRadioButton> filters = new ArrayList<JRadioButton>();
    private List<Edge> edges = new ArrayList<Edge>();
    private JButton filterButton = new JButton("Filter");
    private JPanel topPanel = new JPanel();
    private JPanel attributePane;
    private JPanel datePane;
    private JTextField dateField = new JTextField(Date.ANY_DATE.toString());
    private String attribute = "";
    private String relation = "";
    private String title = "";
    private FrogTreeModel model;
    private ESMViewPanel esm;

    public CreateFilterDialog(Point location, Node first, Node second, FrogTreeModel model,
                                                                        ESMViewPanel esm){
        super();

        this.model = model;
        this.esm = esm;
        source = first.getParent();
        target = second.getParent();
        setDefaultCloseOperation(JDialog.DISPOSE_ON_CLOSE);
        List<Node> temp = new ArrayList<Node>();

        try {
            while (!source.getParent().isRootNode()){
                source = source.getParent();
            }

            while (!target.getParent().isRootNode()){
```

```
            target = target.getParent();
        }

        sourceChildren.addAll(source.getChildren());
        temp.addAll(sourceChildren);

        while(!temp.isEmpty()){
            Node n = temp.remove(0);
            if (!n.getChildren().isEmpty()){
                temp.addAll(n.getChildren());
                sourceChildren.addAll(n.getChildren());
            }
        }

        if (source.equals(target)){
            targetChildren.addAll(sourceChildren);
        }
        else{
            targetChildren.addAll(target.getChildren());
            temp.addAll(targetChildren);

            while(!temp.isEmpty()){
                Node n = temp.remove(0);
                if (!n.getChildren().isEmpty()){
                    temp.addAll(n.getChildren());
                    targetChildren.addAll(n.getChildren());
                }
            }
        }

        title = source.getShortName()+" x "+target.getShortName();
        TitledBorder titledBorder = new TitledBorder(title);
        topPanel.setBorder(titledBorder);
        setMinimumSize(new Dimension(titledBorder.getMinimumSize(topPanel).width+5,
                                     titledBorder.getMinimumSize(topPanel).height));

    } catch (ClientConnectionException e) {
        e.printStackTrace();
    } catch (ObjectDeletedException e) {
        e.printStackTrace();
    }

    topPanel.setLayout(new BorderLayout());

    ActionListener ac = new ActionListener(){
        public void actionPerformed(ActionEvent e) {
            JRadioButton btn = (JRadioButton) e.getSource();
            relation = btn.getText();
            List<Attribute> attributes = new ArrayList<Attribute>();
            for (Edge edge : edges){
                try {
```

```
            if (edge.getShortName().toLowerCase().equals(relation)){
                attributes = edge.getAttributes();
                break;
            }
        } catch (ObjectDeletedException e1) {
            e1.printStackTrace();
        } catch (ClientConnectionException e1) {
            e1.printStackTrace();
        }

    }

    if (!attributes.isEmpty()){
        attributePane = new JPanel();
        attributePane.setBorder(new TitledBorder("Attributes"));
        attributePane.setLayout(new GridLayout(attributes.size(), 1));
        for (Attribute a : attributes){
            JRadioButton attr = new JRadioButton(a.toString());
            attr.addActionListener(new ActionListener(){
                public void actionPerformed(ActionEvent e) {
                    attribute = ((JRadioButton)e.getSource()).getText();
                    datePane = new JPanel(new FlowLayout(FlowLayout.LEFT));
                    datePane.setBorder(new TitledBorder("Time"));
                    datePane.add(new JLabel(" Set date: "));
                    Dimension oldSize = dateField.getPreferredSize();
                    Dimension newSize = new Dimension(oldSize.width + 10,
                                                      oldSize.height);
                    dateField.setPreferredSize(newSize);
                    datePane.add(dateField);
                    topPanel.add(datePane, BorderLayout.EAST);
                    CreateFilterDialog.this.refresh();
                }

            });
            attributeGroup.add(attr);
            attributePane.add(attr);
        }

        topPanel.add(attributePane, BorderLayout.CENTER);
        CreateFilterDialog.this.refresh();
    }
    else{
        if (attributePane!=null){
            topPanel.remove(attributePane);
        }

        if (datePane!=null){
            topPanel.remove(datePane);
        }

        CreateFilterDialog.this.refresh();
```

```
        }
    }

};

for (Node node : sourceChildren){
    try {
        for (Edge edge : node.getOutEdges()){
            if (!edges.contains(edge) && targetChildren.contains(edge.getInNode())){
                edges.add(edge);
                String name = edge.getShortName().toLowerCase();
                if(!names.contains(name)){
                    names.add(name);
                    JRadioButton filter = new JRadioButton(name);
                    filter.addActionListener(ac);
                    relationGroup.add(filter);
                    filters.add(filter);
                }
            }
        }
    } catch (ClientConnectionException e) {
        e.printStackTrace();
    } catch (ObjectDeletedException e) {
        e.printStackTrace();
    }
}

filterButton.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e) {
        RelationFilter filter = new RelationFilter(relation, source.getCategory(),
                target.getCategory(), attribute, Date.parseDate(dateField.getText()));
        if (relation=="" || attribute==""){
            JOptionPane.showMessageDialog(CreateFilterDialog.this,
                                                    "No attribute which to apply filter.");
        }
        else {
            CreateFilterDialog.this.model.getFilter().add(filter);
            CreateFilterDialog.this.model.reload();
            CreateFilterDialog.this.esm.clearESMCache();
            CreateFilterDialog.this.esm.updateColumns();
            CreateFilterDialog.this.esm.flagToRepaint();
            CreateFilterDialog.this.esm.refreshMatrix();
            CreateFilterDialog.this.dispose();
        }
    }
});

if (!filters.isEmpty()){
    JPanel btnpne = new JPanel();
    btnpne.setLayout(new GridLayout(filters.size(), 1));
    for (JRadioButton btn : filters){
```

```
                btnpne.add(btn);
            }
            btnpne.setBorder(new TitledBorder("Relations"));

            setTitle("Create Filter");
            setLayout(new BorderLayout());
            setLocation(location);
            topPanel.add(btnpne, BorderLayout.WEST);
            add(topPanel, BorderLayout.CENTER);

            JPanel bottomPane = new JPanel(new BorderLayout());
            bottomPane.add(filterButton, BorderLayout.CENTER);

            add(bottomPane, BorderLayout.SOUTH);
            pack();
            setVisible(true);
        }
        else{
            JOptionPane.showMessageDialog(this, "No possible relations to filter.", title,
                                                    JOptionPane.INFORMATION_MESSAGE);
            dispose();
        }

    }

    private void refresh(){
        pack();
        repaint();
        setVisible(true);
    }

}
```

B.2 CreateCoordinateDialog.java

```
public class CreateCoordinateDialog extends CreateDialog {

    private static final long serialVersionUID = 603191828643098722L;
    private Attribute coordinate;
    private List<AttributeInputPanel> attributeInputPanels;
    private JTextField nameField;
    private JPanel inputPanel;
    private JButton createButton;
    private String latitude="", longitude="";
    private List<CoordinateListener> listeners = new ArrayList<CoordinateListener>();

    public CreateCoordinateDialog(Attribute coordinate, Component parentComponent,
                                  String latitude, String longitude) {
```

```
super(parentComponent);
this.latitude = latitude;
this.longitude = longitude;
this.coordinate = coordinate;
attributeInputPanels = new LinkedList<AttributeInputPanel>();
setTitle("Create attribute");
setLayout(new BorderLayout());

JPanel topPanel = new JPanel();
topPanel.setLayout(new BoxLayout(topPanel, BoxLayout.Y_AXIS));
Font font = topPanel.getFont().deriveFont(Font.BOLD, 14);

JPanel quotablePanel = new JPanel();
quotablePanel.setLayout(new BoxLayout(quotablePanel, BoxLayout.X_AXIS));

try {
    JLabel quotableLabel = new JLabel(coordinate.getShortName());
    quotableLabel.setFont(font);
    quotablePanel.add(quotableLabel);
    quotablePanel.add(Box.createHorizontalGlue());
    topPanel.add(quotablePanel);
} catch (ObjectDeletedException e1) {
    e1.printStackTrace();
} catch (ClientConnectionException e1) {
    e1.printStackTrace();
}

JPanel namePanel = new JPanel();
namePanel.setLayout(new BoxLayout(namePanel, BoxLayout.X_AXIS));
nameField = new JTextField("Coordinate");
namePanel.setBorder(new EmptyBorder(5, 0, 0, 0));
namePanel.add(new JLabel("Attribute name: "));
nameField.setEnabled(false);
namePanel.add(nameField);
namePanel.add(Box.createHorizontalStrut(50));

topPanel.add(namePanel);
topPanel.setBorder(new EmptyBorder(5, 7, 5, 5));
add(topPanel, BorderLayout.PAGE_START);

inputPanel = new JPanel();
inputPanel.setBorder(new EmptyBorder(0, 5, 0, 5));
inputPanel.setLayout(new BoxLayout(inputPanel, BoxLayout.Y_AXIS));
add(inputPanel, BorderLayout.CENTER);

JPanel buttonPanel = new JPanel();
buttonPanel.setLayout(new BoxLayout(buttonPanel, BoxLayout.Y_AXIS));

createButton = new JButton("Create");
createButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent evt) {
```

```
            doCreate();
        }
    });

    final JButton cancelButton = new JButton("Cancel");
    cancelButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent evt) {
            doCancel();
        }
    });
    cancelButton.setMnemonic(KeyEvent.VK_ESCAPE);

    JPanel bottomButtonPanel = new JPanel();
    bottomButtonPanel.setLayout(new FlowLayout());
    bottomButtonPanel.add(createButton);
    bottomButtonPanel.add(cancelButton);
    bottomButtonPanel.setBorder(new EmptyBorder(5, 5, 5, 5));

    buttonPanel.add(bottomButtonPanel);
    add(buttonPanel, BorderLayout.PAGE_END);

    // Ensure the text field always gets the first focus.
    addComponentListener(new ComponentAdapter() {
        @Override
        public void componentShown(ComponentEvent ce) {
            doReset();
        }
    });

    // Handle window closing correctly.
    setDefaultCloseOperation(DO_NOTHING_ON_CLOSE);
    addWindowListener(new WindowAdapter() {
        @Override
        public void windowClosing(WindowEvent we) {
            doCancel();
        }
    });

    // Set the create button as the default
    getRootPane().setDefaultButton(createButton);

    // This is so the dialog will close with the ESC key
    KeyStroke escape = KeyStroke.getKeyStroke(KeyEvent.VK_ESCAPE, 0, false);
    Action escapeAction = new AbstractAction() {
        private static final long serialVersionUID = 1L;

        public void actionPerformed(ActionEvent e) {
            doCancel();
        }
    };
    getRootPane().getInputMap(JComponent.WHEN_IN_FOCUSED_WINDOW).put(
```

```
                                escape, "ESCAPE");
        getRootPane().getActionMap().put("ESCAPE", escapeAction);
    }

    //Create the appropriate attribute and add it to the database
    private void doCreate() {
        // Open the database
        try {
            // Check for improper dates
            for (AttributeInputPanel inputPanel : attributeInputPanels) {
                Date start = inputPanel.getStartDate();
                Date end = inputPanel.getEndDate();
                if ((start == null) || (end == null)) {
                    continue;
                } else if (start.compareTo(end) > 0) {
                    JOptionPane.showMessageDialog(this, "The start date ("+ start
                            + ") you entered was after the end date ("+ end + ").", "Date error",
                                                    JOptionPane.ERROR_MESSAGE);
                    return;
                }
            }

            for (AttributeInputPanel inputPanel : attributeInputPanels) {
                Date start = inputPanel.getStartDate();
                Date end = inputPanel.getEndDate();
                if ((start == null) || (end == null)) { continue; }
                    TimeInterval ti = new TimeInterval(start.getDateInMs(), end.getDateInMs());
                    double lat = Double.parseDouble(inputPanel.getLatitude());
                    double lon = Double.parseDouble(inputPanel.getLongitude());
                    if (lat<-90||lat>90){
                        JOptionPane.showMessageDialog(this,
                                                "Latitude coordinate must be between -90 and 90");
                    }
                    else if(lon<-180||lon>180){
                        JOptionPane.showMessageDialog(this,
                                        "Longitude coordinate must be between -180 and 180");
                    }
                    else{
                        String value = lon+","+lat;
                        AttributeElement ae =
                            coordinate.getSpidr().makeAttributeElement(coordinate, ti, value);
                    if(ae!=null){
                        for(CoordinateListener cl : listeners){
                            cl.fireCreated(coordinate, ae);
                        }
                    }
                }
            }
        } catch(NumberFormatException e){
            JOptionPane.showMessageDialog(this,
                                "Latitude and longitude coordinates must be numbers.");
```

```
        } catch (ClientConnectionException e) {
            e.printStackTrace();
        } catch (ObjectDeletedException e) {
            e.printStackTrace();
        }
        fireCreated();
        dispose();
    }

    //Called when the cancel button is pressed
    private void doCancel() {
        dispose();
    }

    /** Adds a new blank attribute input area */
    private void doAddAttributeInputPanel() {
        AttributeInputPanel aip;
        aip = new AttributeInputPanel();
        attributeInputPanels.add(aip);
        inputPanel.add(aip);
        doResize();
        aip.focus();
    }

    /** Resets the input panel */
    private void doReset() {
        attributeInputPanels.clear();
        inputPanel.removeAll();
        AttributeInputPanel aip;
        aip = new AttributeInputPanel();
        JPanel topButtonPanel = new JPanel();
        topButtonPanel.setLayout(new BoxLayout(topButtonPanel, BoxLayout.X_AXIS));
        JButton resetButton = new JButton("Reset");
        resetButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                doReset();
            }
        });
        JButton plusButton = new JButton("+");
        plusButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                doAddAttributeInputPanel();
            }
        });

        topButtonPanel.add(plusButton);
        topButtonPanel.add(Box.createHorizontalStrut(5));
        topButtonPanel.add(resetButton);
        topButtonPanel.add(Box.createHorizontalGlue());
        topButtonPanel.setBorder(new EmptyBorder(5, 0, 5, 5));
        topButtonPanel.setMaximumSize(new Dimension(topButtonPanel
```

```
                .getMaximumSize().width, topButtonPanel.getPreferredSize().height));

        JPanel labelPanel = new JPanel();
        labelPanel.setLayout(new BoxLayout(labelPanel, BoxLayout.X_AXIS));
        labelPanel.setBorder(new EmptyBorder(0, 2, 0, 0));
        labelPanel.add(new JLabel("Attribute elements:"));
        labelPanel.add(Box.createHorizontalGlue());

        inputPanel.add(labelPanel);
        inputPanel.add(topButtonPanel);
        attributeInputPanels.add(aip);
        inputPanel.add(aip);
        doResize();
        nameField.requestFocus();
        getRootPane().setDefaultButton(createButton);
    }

    // Resize the dialog box
    private void doResize() {
        int preferredHeight = 45;
        int preferredWidth = 0;

        for (Component c : getComponents()) {
            preferredWidth = Math.max(preferredWidth, c.getPreferredSize().width);
            preferredHeight += c.getPreferredSize().height;
        }

        setSize(preferredWidth, preferredHeight);
        validate();
        repaint();
    }

    public void addListener(CoordinateListener cl){
        listeners.add(cl);
    }

    private class AttributeInputPanel extends JPanel {

        private static final long serialVersionUID = 1L;
        private JTextField dateInput1, dateInput2;
        private JTextField valueInput1, valueInput2;

        public AttributeInputPanel() {
            this("01/01/2006", "01/01/2007");
        }

        public AttributeInputPanel(String startDate, String endDate) {
            setLayout(new BoxLayout(this, BoxLayout.X_AXIS));
            setBorder(new EmptyBorder(5, 2, 0, 5));

            add(new JLabel("Start: "));
```

```
dateInput1 = new JTextField(startDate);
dateInput1.setPreferredSize(new Dimension(50,
                                          dateInput1.getPreferredSize().height));
dateInput1.addFocusListener(new FocusListener() {
    public void focusGained(FocusEvent arg0) {
        dateInput1.selectAll();
    }

    public void focusLost(FocusEvent arg0) { }
});
add(dateInput1);

add(new JLabel("  End: "));
dateInput2 = new JTextField(endDate);
dateInput2.setPreferredSize(new Dimension(50,
                                          dateInput2.getPreferredSize().height));
dateInput2.addFocusListener(new FocusListener() {
    public void focusGained(FocusEvent arg0) {
        dateInput2.selectAll();
    }

    public void focusLost(FocusEvent arg0) { }
});
add(dateInput2);

add(new JLabel("  Longitude: "));
if(longitude.isEmpty())
    valueInput2 = new JTextField("0");
else{
    valueInput2 = new JTextField(longitude);
}
valueInput2.setPreferredSize(new Dimension(100,
                                          valueInput2.getPreferredSize().height));
add(valueInput2);
valueInput2.addFocusListener(new FocusListener() {
    public void focusGained(FocusEvent arg0) {
        valueInput2.selectAll();
    }

    public void focusLost(FocusEvent arg0) { }
});

add(new JLabel("  Latitude: "));
if(latitude.isEmpty()){
    valueInput1 = new JTextField("0");
}
else{
    valueInput1 = new JTextField(latitude);
}
valueInput1.setPreferredSize(new Dimension(100,
                                          valueInput1.getPreferredSize().height));
```

```
            add(valueInput1);
            valueInput1.addFocusListener(new FocusListener() {
                public void focusGained(FocusEvent arg0) {
                    valueInput1.selectAll();
                }
                public void focusLost(FocusEvent arg0) { }
            });
        }

        @Override
        public Dimension getMaximumSize() {
            return new Dimension(super.getMaximumSize().width,
                                                    super.getPreferredSize().height);
        }

        @Override
        public Dimension getMinimumSize() {
            return new Dimension(super.getMinimumSize().width,
                                                    super.getPreferredSize().height);
        }

        @Override
        public Dimension getPreferredSize() {
            int fudge = 70;
            return new Dimension(super.getPreferredSize().width + fudge,
                                                    super.getPreferredSize().height);
        }

        public Date getStartDate() {
            return Date.parseDate(dateInput1.getText());
        }

        public Date getEndDate() {
            return Date.parseDate(dateInput2.getText());
        }

        public String getLatitude() {
            return valueInput1.getText();
        }

        public String getLongitude(){
            return valueInput2.getText();
        }

        public void focus() {
            dateInput1.requestFocus();
        }
    }
}
```

B.3 GoogleEarthFrame.java

```
public class GoogleEarthFrame extends JFrame {
    private static final long serialVersionUID = 9058020377334818391L;
    private JPanel gePanel = new JPanel();
    private JPanel esmPanel = new JPanel();
    private GoogleEarthControlPanel controlPanel;
    private GoogleEarthModel geModel;

    private int appWidth = 800, appHeight = 640;
    private int geRenderHwnd;

    private Spidr spidr;
    private List<Node> rootNodes = new ArrayList<Node>();

    public GoogleEarthFrame(Spidr s){
        spidr = s;
        File file = createFile();

        try {
            rootNodes = spidr.getRootNode().getChildren();
        } catch (ClientConnectionException e) {
            e.printStackTrace();
        } catch (ObjectDeletedException e) {
            e.printStackTrace();
        }

        try {
            setTitle("Frog/Google Earth Environment: "+spidr.getName());
        } catch (ClientConnectionException e1) {
            e1.printStackTrace();
        } catch (ObjectDeletedException e1) {
            e1.printStackTrace();
        }
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

        geModel = new GoogleEarthModel();
        if(file!=null){
            try {
                geModel.openKMLFile(file.getPath());
            } catch (COMException e1) {
                e1.printStackTrace();
            }
        }

        setName("Google Earth");
        JTabbedPane tabbedPane = new JTabbedPane();
        controlPanel = new GoogleEarthControlPanel(this, geModel);
        gePanel.setPreferredSize(new Dimension(appWidth, appHeight));
        JPanel mainPanel = new JPanel();
```

```
esmPanel.setLayout(new GridLayout(rootNodes.size()+1, 1));

CreateEdgePanel cep = new CreateEdgePanel(spidr);
cep.addListener(controlPanel);

for(Node n : rootNodes){
    try {
        CreateNodePanel cnp = new CreateNodePanel(spidr, n);
        cnp.addListener(controlPanel);
        cep.addAll(cnp.getInEdges());
        esmPanel.add(cnp);
    } catch (ObjectDeletedException e1) {
        e1.printStackTrace();
    } catch (ClientConnectionException e1) {
        e1.printStackTrace();
    }
}

cep.sortNames();
cep.setUp();
esmPanel.add(cep);

mainPanel.setLayout(new BorderLayout());
mainPanel.add(controlPanel, BorderLayout.SOUTH);
mainPanel.add(gePanel, BorderLayout.CENTER);

JPanel gePane = new JPanel();
gePane.setLayout(new BorderLayout());
gePane.add(mainPanel, BorderLayout.CENTER);
gePane.add(esmPanel, BorderLayout.WEST);

TableViewPanel tvPanel = new TableViewPanel(spidr);
tvPanel.setPreferredSize(new Dimension(appWidth, appHeight));
ESMViewPanel esmPanel = new ESMViewPanel(spidr, false);
tabbedPane.addTab("Google Earth Interface", gePane);
tabbedPane.addTab("Table", tvPanel);
tabbedPane.addTab("Matrix", esmPanel);
getContentPane().add(tabbedPane);

setVisible(true);
pack();
attachRenderHwnd();

gePane.addComponentListener(new ComponentListener(){
    public void componentHidden(ComponentEvent e) {
        gePanel.setPreferredSize(new Dimension(1,1));
        resizeGERenderHwnd();
        repaint();
    }
```

```
            public void componentShown(ComponentEvent e) {
                setGEPanelSize();
                controlPanel.grabFocus();
            }

            public void componentMoved(ComponentEvent e) {}
            public void componentResized(ComponentEvent e) {}

        });

        addComponentListener(new ComponentListener(){
            public void componentResized(ComponentEvent e) {
                if(gePanel.isShowing())
                setGEPanelSize();
            }

            public void componentHidden(ComponentEvent e) {}
            public void componentMoved(ComponentEvent e) {}
            public void componentShown(ComponentEvent e) {}

        });
    }

    @Override
    public void dispose() {
        super.dispose();
        quitGE();
    }

    private void setGEPanelSize(){
        int width = gePanel.getSize().width;
        int height = gePanel.getSize().height;
        if(width>1 && height>1){
            gePanel.setPreferredSize(new Dimension(width, height));
        }
        else if(width>1){
            gePanel.setPreferredSize(new Dimension(width,1));
        }
        else if(height>1){
            gePanel.setPreferredSize(new Dimension(1,height));
        }
        else{
            gePanel.setPreferredSize(new Dimension(1,1));
        }
        resizeGERenderHwnd();
        repaint();
    }

    //Returns window handle of this frame
    private int getGUIHwnd() {
        int hwnd = 0;
```

```
        try {
            FuncPtr findWindow = new FuncPtr("USER32.DLL", "FindWindowW");
            hwnd = findWindow.invoke_I(null, this.getTitle(), ReturnFlags.CHECK_FALSE);
        } catch (COMException e) {
            e.printStackTrace();
        }


        return hwnd;
    }

    //Attaches the render window of Google Earth to this frame
    private void attachRenderHwnd() {
        int geMainHwnd = geModel.getMainHwnd();
        geRenderHwnd = geModel.getRenderHwnd();
        try {
            User32.ShowWindow(geMainHwnd, 0); // hide Google Earth main window
        } catch (COMException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }

        resizeGERenderHwnd();

        // attach Google Earth render window to the GUI
        try {
            FuncPtr setParent = new FuncPtr("USER32.DLL", "SetParent");
            setParent.invoke_I(geRenderHwnd, getGUIHwnd(), ReturnFlags.CHECK_FALSE);
        } catch (COMException e) {
            e.printStackTrace();
        }
    }

    //Resize the render window of Google Earth
    private void resizeGERenderHwnd() {
        try {
            FuncPtr moveWindow = new FuncPtr("USER32.DLL", "MoveWindow");
            // create a NakedByteStream for the serialization of Java variables
            NakedByteStream nbs = new NakedByteStream();
            // wrap it in a LittleEndianOutputStream
            LittleEndianOutputStream leos = new LittleEndianOutputStream(nbs);
            // and then write the Java arguments
            leos.writeInt(geRenderHwnd); // Handle to the window
            // Specifies the new position of the left side of the window
            leos.writeInt(esmPanel.getPreferredSize().width+2);
            // Specifies the new position of the top of the window
            leos.writeInt(22);
            //Specifies the new width of the window
            leos.writeInt((int) gePanel.getPreferredSize().width);
            //Specifies the new height of the window
```

```
            leos.writeInt((int) gePanel.getPreferredSize().height);
            //Specifies whether the window is to be repainted
            leos.writeBoolean(false);
            moveWindow.invoke("IIIII:I:", 24, nbs, null, ReturnFlags.CHECK_FALSE);
        } catch (IOException e) {
            e.printStackTrace();
        } catch (COMException e) {
            e.printStackTrace();
        }
    }

    //Quits Google Earth
    private void quitGE() {
        try {
            FuncPtr endTask = new FuncPtr("USER32.DLL", "EndTask");
            //create a NakedByteStream for the serialization of Java variables
            NakedByteStream nbs = new NakedByteStream();
            // wrap it in a LittleEndianOutputStream
            LittleEndianOutputStream leos = new LittleEndianOutputStream(nbs);
            //and then write the Java arguments
            leos.writeInt(geModel.getMainHwnd()); //Handle to the window to be closed.
            leos.writeInt(0); //Ignored. Must be FALSE.
            leos.writeInt(0); //A TRUE for this parameter will force the destruction of the window if
            //an initial attempt fails to gently close the window using WM_CLOSE. With a FALSE
            //for this parameter, only the close with WM_CLOSE is attempted.
            endTask.invoke("III:I:", 12, nbs, null, ReturnFlags.CHECK_FALSE);
        } catch (COMException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    //Creates temporary KML file from current Spidr if it has geospatial coordinates
    private File createFile(){
        FileOutputStream fos;
        PrintStream ps;
        File file = new File("C://temp.kml");
        try {
            file.createNewFile();
            file.deleteOnExit();
            String spidrString = spidr.spidrToKML();
            if (!spidrString.equals("")){
                file.createNewFile();
                fos = new FileOutputStream(file);
                ps = new PrintStream(fos);
                ps.println(spidrString);
                ps.close();
                return file;
            }
        } catch (IOException e) {
```

```
            e.printStackTrace();
        }
        return null;
    }

}
```

B.4 GoogleEarthControlPanel.java

```
public class GoogleEarthControlPanel extends JPanel implements AutocompletePanelListener,
                                                                CoordinateListener {

    private static final long serialVersionUID = -8019760635427483430L;
    private JTextField latitude = new JTextField(15);
    private JTextField longitude = new JTextField(15);
    private JButton getCoordinates = new JButton("Get Current Coordinates");
    private JButton clear = new JButton("Clear");
    private JButton flyToCoordinates = new JButton("Fly to Coordinates");
    private GoogleEarthFrame geFrame;
    private GoogleEarthModel geModel;

    public GoogleEarthControlPanel(GoogleEarthFrame frame, GoogleEarthModel ge){
        geFrame = frame;
        geModel = ge;

        JPanel fieldPanel = new JPanel();
        JPanel buttonPanel = new JPanel();

        fieldPanel.add(new JLabel("Longitude: "));
        fieldPanel.add(longitude);
        longitude.grabFocus();
        JLabel strut = new JLabel();
        strut.setPreferredSize(new Dimension(10,1));
        fieldPanel.add(strut);
        fieldPanel.add(new JLabel("Latitude: "));
        fieldPanel.add(latitude);
        buttonPanel.add(getCoordinates);
        buttonPanel.add(flyToCoordinates);
        buttonPanel.add(clear);

        setLayout(new BorderLayout());
        add(fieldPanel, BorderLayout.NORTH);
        add(buttonPanel, BorderLayout.SOUTH);

        getCoordinates.addActionListener(new ActionListener(){

            public void actionPerformed(ActionEvent arg0) {
                try {
```

```
                latitude.setText(geModel.getLatitude());
                longitude.setText(geModel.getLongitude());
            } catch (COMException e) {
                e.printStackTrace();
            }

        }

    });

    flyToCoordinates.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent arg0) {
            if(latitude.getText().isEmpty()||longitude.getText().isEmpty()){
                JOptionPane.showMessageDialog(geFrame,
                                                "Enter latitude and longitude coordinates");
            }
            else{
                try{
                    double lat = Double.parseDouble(latitude.getText());
                    double lon = Double.parseDouble(longitude.getText());
                    if (lat<-90||lat>90){
                        JOptionPane.showMessageDialog(geFrame,
                                        "Latitude coordinate must be between -90 and 90");
                    }
                    else if(lon<-180||lon>180){
                        JOptionPane.showMessageDialog(geFrame,
                                "Longitude coordinate must be between -180 and 180");
                    }
                    else{
                        geModel.setCamera(lat, lon);
                    }
                } catch (COMException e) {
                    e.printStackTrace();
                }catch(NumberFormatException e){
                    JOptionPane.showMessageDialog(geFrame,
                                "Latitude and longitude coordinates must be numbers");
                }
            }
            GoogleEarthControlPanel.this.longitude.grabFocus();
        }
    });

    clear.addActionListener(new ActionListener(){
        public void actionPerformed(ActionEvent e) {
            GoogleEarthControlPanel.this.latitude.setText("");
            GoogleEarthControlPanel.this.longitude.setText("");
            GoogleEarthControlPanel.this.longitude.grabFocus();
        }
    });

}
```

```
public void entityCreated(AutocompletePanel source, Spidr s, Entity n) {
    String latText = latitude.getText();
    String longText = longitude.getText();
    if(latText.isEmpty()||longText.isEmpty()){
        try {
            latText = geModel.getLatitude();
            longText = geModel.getLongitude();
        } catch (COMException e) {
            e.printStackTrace();
        }
    }
    try {
        Attribute coordinate = s.makeAttribute("Coordinate", n);
        if(coordinate==null){
            coordinate = (Attribute) s.parse(n.getName()+":Coordinate");
        }
        CreateCoordinateDialog ccd = new CreateCoordinateDialog(coordinate, this, latText,
                                                                    longText);
        ccd.setVisible(true);
        ccd.addListener(this);

    } catch (ClientConnectionException e) {
        e.printStackTrace();
    } catch (ObjectDeletedException e) {
        e.printStackTrace();
    }
}

public void fireCreated(Attribute a, AttributeElement ae) {
    try {
        Entity e = a.getEntity();
        String name = e.getShortName();
        String value = ae.getValue();
        String[ ] coordinate = value.split(",");
        Date startTime = ae.getTimeInterval().getStart();
        Date endTime = ae.getTimeInterval().getEnd();
        String start = startTime.getYear()+"-0"+startTime.getMonth()+"-0"
                                                            +startTime.getDay();
        String end = endTime.getYear()+"-0"+endTime.getMonth()+"-0"+endTime.getDay();
        System.out.println(start+","+end);
        String data = "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n"
                    + "<kml xmlns=\"http://earth.google.com/kml/2.2\">\n"
                    + <Placemark>\n<name>" + name+"</name>\n";
        data+="<Point>\n<coordinates>"+coordinate[0]+","+coordinate[1]
              +",0</coordinates>\n</Point>\n";
        data+="<TimeSpan><begin>"+start+"</begin><end>"+end+"</end></TimeSpan>\n";
        data+="</Placemark>\n</kml>";
        geModel.loadKMLData(data);
        geModel.setCamera(Double.parseDouble(coordinate[1]),
                                                Double.parseDouble(coordinate[0]));
    } catch (ObjectDeletedException e) {
```

```
            e.printStackTrace();
        } catch (ClientConnectionException e) {
            e.printStackTrace();
        } catch (COMException e) {
            e.printStackTrace();
        }

    }

}
```

B.5 AutocompletePanel.java

```
public abstract class AutocompletePanel extends JPanel {

    private static final long serialVersionUID = -7341972119482784979L;
    protected Spidr spidr;
    protected WordList names = new WordList();
    protected JTextField name = new JTextField(25);
    private JList suggestions = new JList();
    private JScrollPane listScroller;
    private boolean suggestionsVisible = false;
    private boolean hoverSuggestions = false;
    private List<AutocompletePanelListener> listeners = new
                                                ArrayList<AutocompletePanelListener>();
    public AutocompletePanel(Spidr s, String str){
        spidr = s;
        setBorder(new TitledBorder(str));
        double[ ][ ] size = {{TableLayout.FILL}, {20,TableLayout.FILL}};
        setLayout(new TableLayout(size));
        add(name, "0,0");

        name.addFocusListener(new FocusListener() {
            public void focusGained(FocusEvent arg0) {}
            public void focusLost(FocusEvent arg0) {
                if(!hoverSuggestions)
                    removeSuggestions();
                }
            });

        suggestions.addMouseListener(new MouseListener(){
            public void mousePressed(MouseEvent e) { }
            public void mouseReleased(MouseEvent e) { }
            public void mouseClicked(MouseEvent e) {
                String val = (String) suggestions.getModel().getElementAt(
                                                suggestions.locationToIndex(e.getPoint()));
                if(val!=null){
```

```
                selectSuggestion(val);
            }
        }

        public void mouseEntered(MouseEvent e) {
            suggestions.setSelectedIndex(suggestions.locationToIndex(e.getPoint()));
            hoverSuggestions = true;
        }

        public void mouseExited(MouseEvent e) {
            suggestions.setSelectedValue(null, false);
            hoverSuggestions = false;
        }
    });

    suggestions.addMouseMotionListener(new MouseMotionListener(){
        public void mouseDragged(MouseEvent e) {}
        public void mouseMoved(MouseEvent e) {
            suggestions.setSelectedIndex(suggestions.locationToIndex(e.getPoint()));
            hoverSuggestions = true;
        }
    });

    name.addKeyListener(new KeyAdapter() {
        @Override
        public void keyReleased(KeyEvent ke) {
            if(ke.getKeyCode() == KeyEvent.VK_DOWN){
                keyDown();
            }
            else if(ke.getKeyCode()==KeyEvent.VK_UP){
                keyUp();
            }
            else if (ke.getKeyCode() == KeyEvent.VK_ENTER && suggestionsVisible
                                                    && suggestions.getSelectedValue()!=null){
                String text = (String) suggestions.getSelectedValue();
                selectSuggestion(text);
            }
            else if (ke.getKeyCode() == KeyEvent.VK_ENTER
                                                            && !name.getText().isEmpty()) {
                addOrModifyEntity();
            }
            else if (name.getText().isEmpty() && suggestionsVisible){
                removeSuggestions();
            }
            else if(!name.getText().isEmpty()){
                find(name.getText());
            }
        }
    });

    setPreferredSize(new Dimension(name.getPreferredSize().width+20, 100));
```

```
}

/**
* Adds all the items in a given list to the list of suggestions
* @param list items to be added
*/
public void addAll(List<String> list){
    names.addAll(list);
}

// Sorts the names in the autocomplete list box of suggestions
public void sortNames(){
    names.sort();
}

// Sets up the autocomplete list box of suggestions
public void setUp(){
    suggestions.setListData(names.getWords().toArray());
    listScroller = new JScrollPane(suggestions);
}

public void addListener(AutocompletePanelListener c){
    listeners.add(c);
}

public boolean removeListener(AutocompletePanelListener c){
    return listeners.remove(c);
}

/**
* Notifies all listeners that an entity has been created
* @param e created entity
*/
protected void entityCreated(Entity e){
    for(AutocompletePanelListener cnpl : listeners){
        cnpl.entityCreated(this, spidr, e);
    }
}

/**
* Returns edge name formatted with long name of nodes
* @param s edge name
* @return edge name formatted with full name
*/
protected String formatEdge(String s){
    String result = s;
    if (s.indexOf("Node.")==0){
        result = s.replaceFirst("Node.", "");
    }
    if (result.indexOf(">Node.")!=-1){
        result = result.replace("Node.", "");
```

```
        }
        return result;
    }

    // Adds entity to the spidr if it does not already exist
    protected abstract void addOrModifyEntity();

    private void addSuggestions(){
        add(listScroller, "0,1");
        repaint();
        revalidate();
        suggestionsVisible = true;
    }

    private void removeSuggestions(){
        remove(listScroller);
        repaint();
        revalidate();
        suggestionsVisible = false;
    }

    private void selectSuggestion(String val){
        name.setText(val);
        removeSuggestions();
        name.grabFocus();
        hoverSuggestions = false;
    }

    private void keyDown(){
        if(suggestionsVisible){
            if (suggestions.getSelectedIndex()==-1 ||
                        suggestions.getSelectedIndex()==suggestions.getModel().getSize()-1){
                suggestions.setSelectedIndex(0);
            }
            else{
                suggestions.setSelectedIndex(suggestions.getSelectedIndex()+1);
            }
        }
    }

    private void keyUp(){
        if(suggestionsVisible){
            if (suggestions.getSelectedIndex()==-1 || suggestions.getSelectedIndex()==0){
                suggestions.setSelectedIndex(suggestions.getModel().getSize()-1);
            }
            else{
                suggestions.setSelectedIndex(suggestions.getSelectedIndex()-1);
            }
        }
    }
```

```
    //Finds possible completions for the given String
    private void find(String s){
        Object[ ] wordsArray = names.find(s).toArray();
        suggestions.setListData(wordsArray);
        if(wordsArray.length==0 && suggestionsVisible){
            removeSuggestions();
        }
        else if(wordsArray.length==1
                        && ((String)wordsArray[0]).toLowerCase().equals(s.toLowerCase())){
            removeSuggestions();
        }
        else if (!suggestionsVisible && wordsArray.length!=0){
            addSuggestions();
        }
    }
}
```

B.6 CreateNodePanel.java

```
public class CreateNodePanel extends AutocompletePanel {
    private static final long serialVersionUID = 52474552906396771 41L;
    private List<String> inEdges = new ArrayList<String>();
    private Node parent;
    private String parentName;

    public CreateNodePanel(Spidr s, Node p) throws ObjectDeletedException,
                                                    ClientConnectionException{
        super(s,p.getShortName());
        name.setToolTipText("Add or modify node");
        parent = p;

        try {
            parentName = p.getShortName();
            List<Node> children = new ArrayList<Node>();
            List<Edge> edges = new ArrayList<Edge>();
            children.addAll(parent.getChildren());
            List<String> temp = new ArrayList<String>();
            while(!children.isEmpty()){
                Node n = children.remove(0);
                String name = n.getName();
                temp.add(removeLeadingNode(name));
                if(!n.getChildren().isEmpty()){
                    children.addAll(n.getChildren());
                }
                if(!n.getInEdges().isEmpty()){
                    edges.addAll(n.getInEdges());
                }
            }
```

```
            while(!edges.isEmpty()){
                Edge e = edges.remove(0);
                String name = e.getName();
                inEdges.add(formatEdge(name));
            }

            names.load(temp);
            names.sort();
            setUp();
        } catch (ClientConnectionException e1) {
            e1.printStackTrace();
        } catch (ObjectDeletedException e1) {
            e1.printStackTrace();
        }
    }

    public List<String> getInEdges(){
        return inEdges;
    }

    protected void addOrModifyEntity(){
        try {
            String nodeName = name.getText();
            name.setText("");
            if(nodeName.indexOf(":")!=-1 || nodeName.indexOf("<")!=-1
                                                    ||nodeName.indexOf(">")!=-1){
            }
            else{
                Node newNode = (Node) spidr.parse(parentName+"."+nodeName);
                if(newNode==null){
                    newNode = spidr.makeNode(nodeName, parent);
                }
                if(newNode!=null){
                    names.add(nodeName);
                    entityCreated(newNode);
                }
            }
        } catch (ObjectDeletedException e) {
            e.printStackTrace();
        } catch (ClientConnectionException e) {
            e.printStackTrace();
        }
    }

    private String removeLeadingNode(String s) {
        String fullParentName = "Node."+parentName;
        if (s.indexOf(fullParentName) != 0) { return s; }
        return s.substring(fullParentName.length()+1);
    }
}
```

Appendix C

Frog XML Schema

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:complexType name="spidr">
    <xs:attribute name="name" type="xs:string" use="required" />
    <xs:attribute name="rootNodes" type="xs:string"/>
  </xs:complexType>

  <xs:complexType name="root">
    <xs:attribute name="name" type="xs:string" use="required" />
  </xs:complexType>

  <xs:complexType name="node">
    <xs:sequence minOccurs="0">
      <xs:element name="quotes" minOccurs="0">
        <xs:complexType>
          <xs:group ref="connectedQuotes" minOccurs="0" maxOccurs="unbounded" />
        </xs:complexType>
      </xs:element>
      <xs:element name="attributes" minOccurs="0">
        <xs:complexType>
          <xs:group ref="connectedAttributes" minOccurs="0" maxOccurs="unbounded" />
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="name" type="xs:string" use="required" />
    <xs:attribute name="parent" type="xs:string" use="required" />
  </xs:complexType>

  <xs:complexType name="edge">
    <xs:sequence minOccurs="0">
      <xs:element name="quotes" minOccurs="0">
        <xs:complexType>
          <xs:group ref="connectedQuotes" minOccurs="0" maxOccurs="unbounded" />
        </xs:complexType>
      </xs:element>
      <xs:element name="attributes" minOccurs="0">
        <xs:complexType>
          <xs:group ref="connectedAttributes" minOccurs="0" maxOccurs="unbounded" />
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="name" type="xs:string" use="required" />
    <xs:attribute name="in" type="xs:string" use="required" />
```

```
    <xs:attribute name="out" type="xs:string" use="required" />
</xs:complexType>

<xs:complexType name="attribute">
    <xs:sequence minOccurs="0">
        <xs:element name="quotes">
            <xs:complexType>
                <xs:group ref="connectedQuotes" minOccurs="0" maxOccurs="unbounded" />
            </xs:complexType>
        </xs:element>
    </xs:sequence>
    <xs:attribute name="name" type="xs:string" use="required" />
    <xs:attribute name="entity" type="xs:string" use="required" />
</xs:complexType>

<xs:complexType name="element">
    <xs:attribute name="name" type="xs:string" use="required" />
    <xs:attribute name="attribute" type="xs:string" use="required" />
    <xs:attribute name="start" type="xs:long" use="required" />
    <xs:attribute name="end" type="xs:long" use="required" />
    <xs:attribute name="value" type="xs:string" use="required" />
</xs:complexType>

<xs:complexType name="connectedAttribute">
    <xs:sequence minOccurs="0">
        <xs:element name="quotes">
            <xs:complexType>
                <xs:group ref="connectedQuotes" minOccurs="0" maxOccurs="unbounded" />
            </xs:complexType>
        </xs:element>
    </xs:sequence>
    <xs:attribute name="name" type="xs:string" use="required"/>
    <xs:attribute name="entity" type="xs:string"/>
</xs:complexType>

<xs:complexType name="connectedElement">
    <xs:sequence minOccurs="0">
        <xs:element name="quotes">
            <xs:complexType>
                <xs:group ref="connectedQuotes" minOccurs="0" maxOccurs="unbounded" />
            </xs:complexType>
        </xs:element>
    </xs:sequence>
    <xs:attribute name="name" type="xs:string" use="required"/>
    <xs:attribute name="attribute" type="xs:string"/>
    <xs:attribute name="start" type="xs:long" use="required" />
    <xs:attribute name="end" type="xs:long" use="required" />
    <xs:attribute name="value" type="xs:string" use="required" />
</xs:complexType>

<xs:complexType name="connectedQuote">
```

```
    <xs:attribute name="object" type="xs:string"/>
    <xs:attribute name="document" type="xs:string" use="required"/>
    <xs:attribute name="start" type="xs:int" use="required"/>
    <xs:attribute name="end" type="xs:int" use="required"/>
</xs:complexType>

<xs:complexType name="quote">
    <xs:attribute name="object" type="xs:string" use="required"/>
    <xs:attribute name="document" type="xs:string" use="required"/>
    <xs:attribute name="start" type="xs:int" use="required"/>
    <xs:attribute name="end" type="xs:int" use="required"/>
</xs:complexType>

<xs:complexType name="folder">
    <xs:attribute name="name" type="xs:string" use="required"/>
    <xs:attribute name="parent" type="xs:string" use="required"/>
</xs:complexType>

<xs:complexType name="document">
    <xs:sequence>
        <xs:element name="content" type="byteArray"/>
    </xs:sequence>
    <xs:attribute name="name" type="xs:string" use="required"/>
    <xs:attribute name="parent" type="xs:string" use="required"/>
</xs:complexType>

<xs:complexType name="byteArray">
    <xs:sequence>
        <xs:element name="item" maxOccurs="unbounded">
            <xs:complexType>
                <xs:attribute name="info" type="xs:byte" use="required"/>
            </xs:complexType>
        </xs:element>
    </xs:sequence>
</xs:complexType>
<xs:element name="frog">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="spidr" type="spidr" minOccurs="0" maxOccurs="unbounded" />
            <xs:element name="node" type="node" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="edge" type="edge" minOccurs="0" maxOccurs="unbounded" />
            <xs:element name="attribute" type="attribute" minOccurs="0"
                maxOccurs="unbounded" />
            <xs:element name="element" type="element" minOccurs="0"
                maxOccurs="unbounded"/>
            <xs:element name="quote" type="quote" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>

<xs:group name="connectedAttributes">
```

```
      <xs:choice>
        <xs:element name="attribute" type="connectedAttribute" maxOccurs="unbounded"/>
        <xs:element name="element" type="connectedElement" minOccurs="0"
              maxOccurs="unbounded"/>
      </xs:choice>
    </xs:group>

    <xs:group name="connectedQuotes">
      <xs:sequence>
        <xs:element name="quote" type="connectedQuote" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:group>
</xs:schema>
```

Appendix D

Google Earth COM API

```
import "oaidl.idl";
import "ocidl.idl";

cpp_quote("#include <winerror.h>")
// Google Earth COM API specific error codes.
cpp_quote("#define E_INVALID_OR_DELETED_FEATURE
        MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF, 0x8001)")
cpp_quote("#define E_APPLICATION_UNINITIALIZED
        MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF, 0x8002)")
cpp_quote("#define S_TELEPORTED
        MAKE_HRESULT(SEVERITY_SUCCESS, FACILITY_ITF, 0x8003)")
cpp_quote("#define E_FEATURE_HAS_NO_VIEW
        MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF, 0x8004)")
cpp_quote("#define E_USAGE_LIMIT_EXCEEDED
        MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF, 0x8005)")
cpp_quote("#define E_INFINITE_NUMBER
        MAKE_HRESULT(SEVERITY_ERROR, FACILITY_ITF, 0x8006)")

    interface ITimeGE;
    interface ITimeIntervalGE;
    interface ICameraInfoGE;
    interface ITourControllerGE;
    interface ISearchControllerGE;
    interface IAnimationControllerGE;
    interface IViewExtentsGE;
    interface IFeatureGE;
    interface IFeatureCollectionGE;
    interface IPointOnTerrainGE;
    interface IApplicationGE;

    typedef [ uuid(ECF071D1-75F4-4F61-A5D9-D96EEAF5EC1D), v1_enum ] enum {
        RelativeToGroundAltitudeGE = 1, AbsoluteAltitudeGE = 2 } AltitudeModeGE;

    [
        object,
        uuid(E39391AE-51C0-4FBD-9042-F9C5B6094445),
        dual,
        helpstring("ITimeGE Interface"),
        pointer_default(unique)
    ]
    interface ITimeGE : IDispatch {
        typedef [ v1_enum ] enum { TimeNegativeInfinityGE = -1, TimeFiniteGE = 0,
            TimePositiveInfinityGE = 1 } TimeTypeGE;
        [propget, id(1), helpstring("property Type")]
```

```
    HRESULT Type([out, retval] TimeTypeGE *pType);
    [propput, id(1), helpstring("property Type")] HRESULT Type([in] TimeTypeGE type);
    [propget, id(2), helpstring("property Year")] HRESULT Year([out, retval] int *pYear);
    [propput, id(2), helpstring("property Year")] HRESULT Year([in] int year);
    [propget, id(3), helpstring("property Month")] HRESULT Month([out, retval] int *pMonth);
    [propput, id(3), helpstring("property Month")] HRESULT Month([in] int month);
    [propget, id(4), helpstring("property Day")] HRESULT Day([out, retval] int *pDay);
    [propput, id(4), helpstring("property Day")] HRESULT Day([in] int day);
    [propget, id(5), helpstring("property Hour")] HRESULT Hour([out, retval] int *pHour);
    [propput, id(5), helpstring("property Hour")] HRESULT Hour([in] int hour);
    [propget, id(6), helpstring("property Minute")]
    HRESULT Minute([out, retval] int *pMinute);
    [propput, id(6), helpstring("property Minute")] HRESULT Minute([in] int minute);
    [propget, id(7), helpstring("property Second")]
    HRESULT Second([out, retval] int *pSecond);
    [propput, id(7), helpstring("property Second")] HRESULT Second([in] int second);
    [propget, id(8), helpstring("property TimeZone")]
    HRESULT TimeZone([out, retval] double *pTimeZone);
    [propput, id(8), helpstring("property TimeZone")]
    HRESULT TimeZone([in] double timeZone);
    [id(9), helpstring("method Clone")] HRESULT Clone([out, retval] ITimeGE **pTime);
    [id(10), helpstring("method ConvertToZone")]
    HRESULT ConvertToZone( [in] double timeZone, [out, retval] ITimeGE **pTime);
};
[
    object,
    uuid(D794FE36-10B1-4E7E-959D-9638794D2A1B),
    dual,
    helpstring("ITimeIntervalGE Interface"),
    pointer_default(unique)
]
interface ITimeIntervalGE : IDispatch {
    [propget, id(1), helpstring("property BeginTime")]
    HRESULT BeginTime([out, retval] ITimeGE **pTime);
    [propget, id(2), helpstring("property EndTime")]
    HRESULT EndTime([out, retval] ITimeGE **pTime);
};
[
    object,
    uuid(08D46BCD-AF56-4175-999E-6DDC3771C64E),
    dual,
    helpstring("ICameraInfoGE Interface"),
    pointer_default(unique)
]
interface ICameraInfoGE : IDispatch {
    [propget, id(1), helpstring("property FocusPointLatitude")]
    HRESULT FocusPointLatitude([out, retval] double *pLat);
    [propput, id(1), helpstring("property FocusPointLatitude")]
    HRESULT FocusPointLatitude([in] double lat);
    [propget, id(2), helpstring("property FocusPointLongitude")]
    HRESULT FocusPointLongitude([out, retval] double *pLon);
```

```
    [propput, id(2), helpstring("property FocusPointLongitude")]
    HRESULT FocusPointLongitude([in] double lon);
    [propget, id(3), helpstring("property FocusPointAltitude")]
    HRESULT FocusPointAltitude([out, retval] double *pAlt);
    [propput, id(3), helpstring("property FocusPointAltitude")]
    HRESULT FocusPointAltitude([in] double alt);
    [propget, id(4), helpstring("property AltitudeModeGE")]
    HRESULT FocusPointAltitudeMode([out, retval] AltitudeModeGE *pAltMode);
    [propput, id(4), helpstring("property AltitudeModeGE")]
    HRESULT FocusPointAltitudeMode([in] AltitudeModeGE altMode);
    [propget, id(5), helpstring("property Range")]
    HRESULT Range([out, retval] double *pRange);
    [propput, id(5), helpstring("property Range")] HRESULT Range([in] double range);
    [propget, id(6), helpstring("property Tilt")] HRESULT Tilt([out, retval] double *pTilt);
    [propput, id(6), helpstring("property Tilt")] HRESULT Tilt([in] double tilt);
    [propget, id(7), helpstring("property Azimuth")]
    HRESULT Azimuth([out, retval] double *pAzimuth);
    [propput, id(7), helpstring("property Azimuth")] HRESULT Azimuth([in] double azimuth);
};

[
    object,
    uuid(D08577E0-365E-4216-B1A4-19353EAC1602),
    dual,
    helpstring("ITourControllerGE Interface"),
    pointer_default(unique)
]
interface ITourControllerGE : IDispatch {
    [id(1), helpstring("method PlayOrPause")] HRESULT PlayOrPause();
    [id(2), helpstring("method Stop")] HRESULT Stop();
    [propget, id(3), helpstring("property Speed")]
    HRESULT Speed([out, retval] double *pSpeed);
    [propput, id(3), helpstring("property Speed")] HRESULT Speed([in] double speed);
    [propget, id(4), helpstring("property PauseDelay")]
    HRESULT PauseDelay([out, retval] double *pPauseDelay);
    [propput, id(4), helpstring("property PauseDelay")]
    HRESULT PauseDelay([in] double pauseDelay);
    [propget, id(5), helpstring("property Cycles")]
    HRESULT Cycles([out, retval] int *pCycles);
    [propput, id(5), helpstring("property Cycles")] HRESULT Cycles([in] int cycles);
};

[
    object,
    uuid(524E5B0F-D593-45A6-9F87-1BAE7D338373),
    dual,
    helpstring("ISearchControllerGE Interface"),
    pointer_default(unique)
]
interface ISearchControllerGE : IDispatch {
    [id(1), helpstring("method Search")] HRESULT Search([in] BSTR searchString);
```

```
    [id(2), helpstring("method IsSearchInProgress")]
    HRESULT IsSearchInProgress([out, retval] BOOL *inProgress);
    [id(3), helpstring("method GetResults")]
    HRESULT GetResults([out, retval] IFeatureCollectionGE **pResults);
    [id(4), helpstring("method ClearResults")] HRESULT ClearResults();
};

[
    object,
    uuid(BE5E5F15-8EC4-4dCC-B48D-9957D2DE4D05),
    dual,
    helpstring("IAnimationControllerGE Interface"),
    pointer_default(unique)
]
interface IAnimationControllerGE : IDispatch {
    [id(1), helpstring("method Play")] HRESULT Play();
    [id(2), helpstring("method Pause")] HRESULT Pause();
    [propget, id(3), helpstring("property SliderTimeInterval")]
    HRESULT SliderTimeInterval([out, retval] ITimeIntervalGE **pInterval);
    [propget, id(4), helpstring("property CurrentTimeInterval")]
    HRESULT CurrentTimeInterval([out, retval] ITimeIntervalGE **pInterval);
    [propput, id(4), helpstring("property CurrentTimeInterval")]
    HRESULT CurrentTimeInterval([in] ITimeIntervalGE *interval);
};

[
    object,
    uuid(865AB2C1-38C5-492B-8B71-AC73F5A7A43D),
    dual,
    helpstring("IViewExtentsGE Interface"),
    pointer_default(unique)
]
interface IViewExtentsGE : IDispatch {
    [propget, id(1), helpstring("property North")] HRESULT North([out, retval] double *pVal);
    [propget, id(2), helpstring("property South")] HRESULT South([out, retval] double *pVal);
    [propget, id(3), helpstring("property East")] HRESULT East([out, retval] double *pVal);
    [propget, id(4), helpstring("property West")] HRESULT West([out, retval] double *pVal);
};

[
    object,
    uuid(92547B06-0007-4820-B76A-C84E402CA709),
    dual,
    helpstring("IFeatureGE Interface"),
    pointer_default(unique)
]
interface IFeatureGE : IDispatch {
    [propget, id(1), helpstring("property Name")]
    HRESULT Name([out, retval] BSTR *name);
    [propget, id(2), helpstring("property Visibility")]
    HRESULT Visibility([out, retval] BOOL *pVal);
```

```
    [propput, id(2), helpstring("property Visibility")] HRESULT Visibility([in] BOOL newVal);
    [propget, id(3), helpstring("property HasView")]
    HRESULT HasView([out, retval] BOOL *pVal);
    [propget, id(4), helpstring("property Highlighted")]
    HRESULT Highlighted([out, retval] BOOL *pVal);
    [id(5), helpstring("method Highlight")] HRESULT Highlight();
    [id(6), helpstring("method GetParent")]
    HRESULT GetParent([out, retval] IFeatureGE **pParent);
    [id(7), helpstring("method GetChildren")]
    HRESULT GetChildren([out, retval] IFeatureCollectionGE **pChildren);
    [propget, id(8), helpstring("property TimeInterval")]
    HRESULT TimeInterval([out, retval] ITimeIntervalGE **pInterval);
};

[
    object,
    uuid(851D25E7-785F-4DB7-95F9-A0EF7E836C44),
    dual,
    helpstring("IFeatureCollectionGE Interface"),
    pointer_default(unique)
]
interface IFeatureCollectionGE : IDispatch {
    [propget, id(DISPID_NEWENUM), helpstring("property _NewEnum")]
    HRESULT _NewEnum([out, retval] IUnknown** ppUnk);
    [propget, id(DISPID_VALUE), helpstring("property Item")]
    HRESULT Item([in] long index, [out, retval] IFeatureGE **pFeature);
    [propget, id(1), helpstring("property Count")] HRESULT Count([out, retval] long *count);
};

[
    object,
    uuid(F4F7B301-7C59-4851-BA97-C51F110B590F),
    dual,
    helpstring("IPointOnTerrainGE Interface"),
    pointer_default(unique)
]
interface IPointOnTerrainGE  : IDispatch {
    [propget, id(1), helpstring("property Latitude")]
    HRESULT Latitude([out, retval] double *pLat);
    [propget, id(2), helpstring("property Longitude")]
    HRESULT Longitude([out, retval] double *pLon);
    [propget, id(3), helpstring("property Altitude")]
    HRESULT Altitude([out, retval] double *pAlt);
    [propget, id(4), helpstring("ProjectedOntoGlobe")]
    HRESULT ProjectedOntoGlobe([out, retval] BOOL *pProjected);
    [propget, id(5), helpstring("ZeroElevationExaggeration")]
    HRESULT ZeroElevationExaggeration([out, retval] BOOL *pZeroExag);
};

[
    object,
```

```
    uuid(2830837B-D4E8-48C6-B6EE-04633372ABE4),
    dual,
    helpstring("IApplicationGE Interface"),
    pointer_default(unique)
]
interface IApplicationGE : IDispatch {
    [id(1), helpstring("method GetCamera")]
    HRESULT GetCamera([in] BOOL considerTerrain,
    [out, retval] ICameraInfoGE **pCamera);
    [id(2), helpstring("method SetCamera")]
    HRESULT SetCamera([in] ICameraInfoGE *camera, [in] double speed);
    [id(3), helpstring("method SetCameraParams")]
    HRESULT SetCameraParams([in] double lat, [in] double lon, [in] double alt,
    [in] AltitudeModeGE altMode, [in] double range, [in] double tilt, [in] double azimuth,
    [in] double speed);
    [propget, id(4), helpstring("property StreamingProgressPercentage")]
    HRESULT StreamingProgressPercentage([out, retval] long *pVal);
    [id(5), helpstring("method SaveScreenShot")]
    HRESULT SaveScreenShot([in] BSTR fileName, [in] long quality);
    [id(6), helpstring("method OpenKmlFile")]
    HRESULT OpenKmlFile([in] BSTR fileName, [in] BOOL suppressMessages);
    [id(7), helpstring("method LoadKmlData")]
    HRESULT LoadKmlData([in] BSTR *kmlData);
    [propget, id(8), helpstring("property AutoPilotSpeed")]
    HRESULT AutoPilotSpeed([out, retval] double *pVal);
    [propput, id(8), helpstring("property AutoPilotSpeed")]
    HRESULT AutoPilotSpeed([in] double newVal);
    [propget, id(9), helpstring("property ViewExtents")]
    HRESULT ViewExtents([out, retval] IViewExtentsGE **pVal);
    [id(10), helpstring("method GetFeatureByName")]
    HRESULT GetFeatureByName([in] BSTR name, [out, retval] IFeatureGE **pFeature);
    [id(11), helpstring("method GetFeatureByHref")]
    HRESULT GetFeatureByHref([in] BSTR href, [out, retval] IFeatureGE **pFeature);
    [id(12), helpstring("method SetFeatureView")]
    HRESULT SetFeatureView([in] IFeatureGE *feature, [in] double speed);
    [id(13), helpstring("method GetPointOnTerrainFromScreenCoords")]
    HRESULT GetPointOnTerrainFromScreenCoords([in] double screen_x,
            [in] double screen_y, [out, retval] IPointOnTerrainGE **pPoint);
    typedef [ v1_enum ] enum { EnterpriseClientGE = 0, ProGE = 1, PlusGE = 2,
            FreeGE = 5, UnknownGE = 0xFF } AppTypeGE;
    [propget, id(14), helpstring("property VersionMajor")]
    HRESULT VersionMajor([out, retval] int *major);
    [propget, id(15), helpstring("property VersionMinor")]
    HRESULT VersionMinor([out, retval] int *minor);
    [propget, id(16), helpstring("property VersionBuild")]
    HRESULT VersionBuild([out, retval] int *build);
    [propget, id(17), helpstring("property VersionAppTye")]
    HRESULT VersionAppType([out, retval] AppTypeGE *appType);
    [id(18), helpstring("method IsInitialized")]
    HRESULT IsInitialized([out, retval] BOOL *isInitialized);
    [id(19), helpstring("method IsOnline")] HRESULT IsOnline([out, retval] BOOL *isOnline);
```

```
        [id(20), helpstring("method Login")] HRESULT Login();
        [id(21), helpstring("method Logout")] HRESULT Logout();
        [id(22), helpstring("method ShowDescriptionBalloon")]
        HRESULT ShowDescriptionBalloon([in] IFeatureGE *feature);
        [id(23), helpstring("method HideDescriptionBalloons")]
        HRESULT HideDescriptionBalloons();
        [id(24), helpstring("method GetHighlightedFeature")]
        HRESULT GetHighlightedFeature([out, retval] IFeatureGE **pFeature);
        [id(25), helpstring("method GetMyPlaces")]
        HRESULT GetMyPlaces([out, retval] IFeatureGE **pMyPlaces);
        [id(26), helpstring("method GetTemporaryPlaces")]
        HRESULT GetTemporaryPlaces([out, retval] IFeatureGE **pTemporaryPlaces);
        [id(27), helpstring("method GetLayersDatabases")]
        HRESULT GetLayersDatabases([out, retval] IFeatureCollectionGE **pDatabases);
        [propget, id(28), helpstring("property ElevationExaggeration")]
        HRESULT ElevationExaggeration([out, retval] double *pExaggeration);
        [propput, id(28), helpstring("property ElevationExaggeration")]
        HRESULT ElevationExaggeration([in] double exaggeration);
        [id(29), helpstring("method GetMainHwnd")]
        HRESULT GetMainHwnd([out, retval] OLE_HANDLE *hwnd);
        [propget, id(30), helpstring("property TourController")]
        HRESULT TourController([out, retval] ITourControllerGE **pTourController);
        [propget, id(31), helpstring("property SearchController")]
        HRESULT SearchController([out, retval] ISearchControllerGE **pSearchController);
        [propget, id(32), helpstring("property AnimationController")]
        HRESULT AnimationController([out, retval] IAnimationControllerGE
                **pAnimationController);
        [id(33), helpstring("method GetRenderHwnd")]
        HRESULT GetRenderHwnd([out, retval] OLE_HANDLE *hwnd);
    };

[
    uuid(3476FAB2-687F-4EA6-9AC2-88D72DC7D7FC),
    version(1.0),
    helpstring("Earth 1.0 Type Library")
]
library EARTHLib
{
    importlib("stdole32.tlb");
    importlib("stdole2.tlb");

    [
        uuid(8097D7E9-DB9E-4AEF-9B28-61D82A1DF784),
        helpstring("ApplicationGE Class")
    ]
    coclass ApplicationGE {
        [default] interface IApplicationGE;
    };
    [
        uuid(1AEDB68D-18A7-4CA9-B41B-3CE7E59FAB24),
        helpstring("TimeGE Class")
```

```
]
coclass TimeGE {
    [default] interface ITimeGE;
};
[
    uuid(42DF0D46-7D49-4AE5-8EF6-9CA6E41EFEC1),
    helpstring("TimeIntervalGE Class")
]
coclass TimeIntervalGE {
    [default] interface ITimeIntervalGE;
};
[
    uuid(645EEE5A-BD51-4C05-A6AF-6F2CF8950AAB),
    helpstring("CameraInfoGE Class")
]
coclass CameraInfoGE {
    [default] interface ICameraInfoGE;
};
[
    uuid(D93BF052-FC68-4DB6-A4F8-A4DC9BEEB1C0),
    helpstring("ViewExtentsGE Class")
]
coclass ViewExtentsGE {
    [default] interface IViewExtentsGE;
};
[
    uuid(77C4C807-E257-43AD-BB3F-7CA88760BD29),
    helpstring("TourControllerGE Class")
]
coclass TourControllerGE {
    [default] interface ITourControllerGE;
};
[
    uuid(A4F65992-5738-475B-9C16-CF102BCDE153),
    helpstring("SearchControllerGE Class")
]
coclass SearchControllerGE {
    [default] interface ISearchControllerGE;
};
[
    uuid(1A239250-B650-4B63-B4CF-7FCC4DC07DC6),
    helpstring("AnimationControllerGE Class")
]
coclass AnimationControllerGE {
    [default] interface IAnimationControllerGE;
};
[
    uuid(CBD4FB70-F00B-4963-B249-4B056E6A981A),
    helpstring("FeatureGE Class")
]
coclass FeatureGE {
```

```
        [default] interface IFeatureGE;
    };
    [
        uuid(9059C329-4661-49B2-9984-8753C45DB7B9),
        helpstring("FeatureCollectionGE class")
    ]
    coclass FeatureCollectionGE {
        [default] interface IFeatureCollectionGE;
    };
    [
        uuid(1796A329-04C1-4C07-B28E-E4A807935C06),
        helpstring("PointOnTerrainGE class")
    ]
    coclass PointOnTerrainGE {
        [default] interface IPointOnTerrainGE;
    };
};
```

Bibliography

[1] Bartolomei, Jason. *Qualitative Knowledge Construction for Engineering Systems: Extending the Design Structure Matrix Methodology in Scope and Procedure.* PhD thesis, Massachusetts Institute of Technology, 2007.

[2] Cooper, C. A., M. L. Ewoldt, et al. (2005). A Systems Architectural Model for Man-Packable/Operable Intelligence, Surveillance, and Reconnaissance Mini/Micro Aerial Vehicles. *Department of Aeronautical Engineering*. Wright-Patterson Air Force Base OH, AFIT.

[3] Dodder, R., J. McConnell, et al. (2005). The Concept for Using the "CLIOS Process": Integrating the Study of Physical and Policy Systems Using Mexico City as an Example. *MIT Engineering Systems Working Papers*. Cambridge MA, MIT.

[4] Shannon Iyo. A Constraint Framework for Analysis of Engineering Systems. Master's thesis, Massachusetts Institute of Technology, 2008.

[5] Richards, M. G., N. B. Shah, et al. (2007). *Architecture Frameworks in System Design: Motivation, Theory, and Implementation.* 2007 Annual Symposium for International Council on System Engineering (INCOSE), San Diego CA.

[6] Steward, D. V. (1981). "The Design Structure System: A Method for Managing the Design of Complex Systems." *IEEE Transactions on Engineering Management* **28**: 71-74.

[7] Sussman (2000). Toward Engineering Systems as a Discipline. Cambridge, Massachusetts Institute of Technology: 6.

[8] Sylvester, Igor. A Hierarchical Systems Knowledge Representation Framework. Master's thesis, Massachusetts Institute of Technology, 2007.

[9] Jawin – A Java/Win32 Interoperability Project. 23 March 2005. SourceForge.net. 22 April 2008 < http://jawinproject.sourceforge.net/>.

[10] Taylor, Frank. "Google Announces a Programmable Interface to Google Earth." Gearthblog.com. 01 Oct. 2006. Google Earth Blog. 22 April 2008 <http://www.gearthblog.com/blog/archives/2006/10/google_announces_a_p.html>.

[11] Taylor, Frank. "Google Earth's KML Now International Standard." Gearthblog.com. 14 April 2008. Google Earth Blog. 22 April 2008 <http://www.gearthblog.com/blog/archives/2008/04/google_earths_kml_now_international.html>.

[12] "Using GPS Devices with Google Earth."earth.google.com. 15 April 2008. Google Earth User Guide. 25 April 2008 < http://earth.google.com/userguide/v4/ug_gps.html>.