Complex Relationships for the Semantic Web

Sanjeev Thacker, Amit Sheth, and , Shuchi Patel
Large Scale Distributed Information Systems (LSDIS) Lab
Department of Computer Science, University of Georgia
Athens, GA 30602 USA

http://lsdis.cs.uga.edu
Email: {sanjeev, amit, shuchi}@cs.uga.edu

1 Introduction

Relationships are fundamental to supporting semantics [Wie97, She96], and hence to the Semantic Web [LHL, FM01]. Till date, focus has been on simple relationships such as is-a and is-part-of, as in DAML/OIL [Ont]. In this work, we adapt our earlier work on MREF [SS98] to develop a framework for supporting complex relationships. A framework to manage complex relationships as discussed here becomes the basis for knowledge discovery from the information interlinked by the Semantic Web.

Our work primarily builds upon earlier research in integrating information systems that has also been applied to exploiting web-accessible distributed across heterogeneous information sources. Primary focus of information integration systems has been to model these diverse data sources and integrate the data by resolving the heterogeneity involved to provide global views of domains (one point access) for querying. We shift the focus from modeling of the information sources for purpose of querying to extracting useful knowledge from these information sources. This, we believe, can be achieved by modeling the complex relationships among the domains to study and explore the interaction that exists between them. In addition to information source and relationship modeling, operations are also modeled as part of the knowledge to exploit the semantics involved in performing complex information requests across multiple domains. The system's framework provides a support for knowledge discovery. Knowledge representation and support for relationships, which are fundamental to the concept of Semantic Web are described in this chapter.

Consider the capability provided by current research prototypes to support integration of information from diverse sources of data over a domain to provide the user with a unified structured (homogeneous) view of that domain for querying. On an integrated view of "earthquakes" one can ask queries of the nature "find information of all the quakes that occurred in California since 1990". However, there is still a limitation on the type of queries that can be answered using such integrated domain views. Assuming views on earthquakes and nuclear tests did exist, can one answer the question "Do nuclear tests cause earthquakes?" How can one study such relationships between the two domains based on the data available on diverse web accessible sources? Let us consider a known relationship between air pollution and vegetation. Assuming the necessary views did exist can a question like "How does air-pollution affect vegetation" be answered using only the integrated views? These queries are beyond the realm of the existing systems. There is a

compelling need to be able to model the semantic correlation of data across the domains and then be able to pose complex information requests.

Support for semantic correlation involving complex relationships has been is demonstrated in the InfoQuilt[1], which provides a framework and a platform to answer complex information requests of the above nature. The novel features of InfoQuilt are:

- A mechanism to express and understand the complex semantic inter-domain relationships
- A powerful interface which can use the semantic knowledge about domains and their relationships to construct complex information requests known as Information Scapes (IScapes). Most information integration systems (e.g., SIMS [AKS96]) focus on efficiently retrieving data on a single domain only. IScapes, on the other hand, are requests that could span across one or more domains and involve inter-domain relationships too. These information requests have a much higher degree of expressiveness compared to keyword based search techniques provided by web-based search tools and database queries that focus only on the syntax and structure of the data.
- Ability to analyze the data available from multitude of heterogeneous autonomous sources to explore and study potential relationships using IScapes. Although InfoQuilt provides the tools necessary for data analysis, human involvement is a part of the process. This forms the basis of the knowledge discovery supported by InfoQuilt.
- A framework to support complex operations that can be used for post-processing of data retrieved from the sources (*e.g.* statistical analysis) and as complex operators needed to define inter-domain relationships.

Major issues addressed in building the InfoQuilt system are as follows:

- A large portion of the data on the web is either unstructured or semi-structured. There are syntactic, structural and semantic heterogeneities across different sources providing information about the same domain [KS97, She99]. In addition to these inconsistencies there is overlapping of information
- The knowledge of domains and their sources need to be modeled so that the system is able to automatically identify the sources of data that could possibly provide useful information with respect to the information request while carefully pruning the others, maximizing the ability to obtain faster and more comprehensive results from the same available sources
- Representation of complex relationships spanning across multiple domains involves modeling the semantics involved in their interaction
- Posing complex information requests containing embedded semantic information requires the system to able to understand the semantics of the request to retrieve relevant results corresponding to a request
- Exploring new relationships is of special interest to us which might require an ability to post-process the data obtained from the sources in several ways

---

[1] One of the incarnations of the InfoQuilt system, as applied to the geographic information as part of the NSF Digital Library initiative is the ADEPT-UGA system [http://alexandria.uscb.edu/]

This chapter focuses on our work on semantic correlation and complex relationships, in the hope of supporting "deeper semantics" in the Semantic web. The chapter is organized as follows. Section 2 focuses on our approach to modeling the domains, inter-domain relationships, information sources available to the system, and functions that form the system's knowledge base. Section 3 addresses the use of IScapes in our system to develop and deploy complex information requests that are answered using the system knowledge and set of resources available to the system. The knowledge discovery, which is the ability of InfoQuilt to study domains, mine and analyze the data available from the sources and explore relationships, is described in Section 4. Section 5 discusses the use of visual toolkits that are provided to construct knowledge base and construct and execute IScapes. We compare our approach to other state of the art information integration systems in section 6 and present our conclusions and planned future improvements in section 7.

2 Knowledge Modeling

The representation of information sources in the system and creation of an "integrated structured view" of the domain that the users can use are two of the critical issues that need to be addressed in any information integration system. The two main approaches used for information integration are [Dus97]:

- **Source-Centric:** The integrated view of the domain is modeled first. Both user queries and source views are specified in terms of that view. For each information request, the integration system needs to plan how to answer it using the source views.
- **Query-Centric:** User queries are in terms of views synthesized at a mediator that are defined on source views. After a view expansion at the mediator, the query is translated to a logical plan that is composed of source views.

The term 'view' here implies some form of representation of the domain and not a view in a database. InfoQuilt adopts the source-centric approach. The source-centric approach has the following advantages over the query-centric approach. First, the integrated views of domains are created independent of their source views. Second, the source views are independent of each other and are described in terms of their domain model. Hence, adding, removing or modifying a source can be done dynamically without affecting any other source views. In the query-centric approach, adding and removing sources additionally requires redefining the domain view and the sources need to be mapped with each other making schema integration difficult. Lastly the source-centric approach is more suitable for sources other than database sources.

This work has been built upon work done in [Ber98, Lak00]. InfoQuilt system knowledge comprises of knowledge about the domains, resources accessible to it, inter-domain relationships and complex operations like functions and simulations. We use ontologies to describe the domains to the system. An ontology is meant to capture useful semantics of the domain such as the domain characteristics, the set of terms and concepts of interest involved and their meanings. Section 2.1 describes how a domain is represented using an ontology. We will use the terms ontology and domain interchangeably in the rest of the chapter unless explicitly specified. There can be multiple resources accessible to the system and an ontology can have several resources that provide relevant data on it.

Section 2.2 describes how we define resources in terms of the ontologies. Section 2.3 describes how complex relationships between two or more domains are modeled. Finally, section 2.4 describes how we support operations such as simulations and functions in the system. InfoQuilt uses XML as the knowledge representation language. We provide a visual tool called *Knowledge Builder*, that lets a user easily create and update the knowledge base of the system. Section 2.5 describes the Knowledge Builder and its functionalities in detail.

2.1 Ontology (Domain Modeling)

An ontology should be able to provide a good understanding of the domain it represents. This includes related terms/concepts, their definitions and/or meanings, their relationships with each other, and characteristics of the domain. It also helps in resolving differences between the models of the domain used by the available sources. This is done by mapping the data available from all resources for that domain from the local model used by the resource to the model specified by the ontology, which the user can then use. The user can create a classification of real-world entities and represent them as ontologies in the system. This classification is based on the human perception of the world broken down into several domains and their sub-domains and so on (real world objects as perceived from a modeling standpoint). The representation of an ontology comprises of related terms (attributes), its place in the domain hierarchy, domain rules, and functional dependencies. The ontology of a sub-domain is said to inherit from the ontology of its parent domain. A child ontology inherits all attributes, domain rules and functional dependencies of its parent ontology.

Example 1:

Consider the domain of disasters. They could be sub-categorized as natural disasters and man-made disasters. Natural disasters could be further sub-categorized into earthquakes, volcanoes, tsunamis, etc. Similarly, man-made disasters can have sub-categories like nuclear tests. An example hierarchy is shown in Figure 1. An ontology's place in the domain hierarchy is represented by specifying its parent ontology, if any. However, it is not necessary for an ontology to be a part of some hierarchy.

[to
ar

4
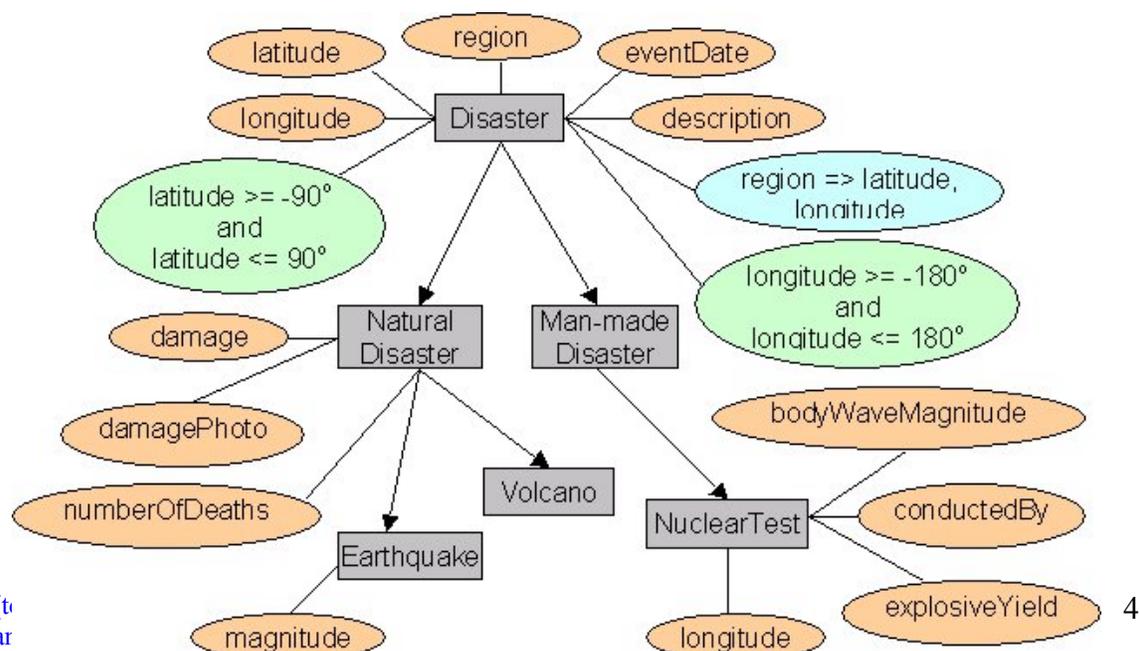
Figure 1. Example domain hierarchy

2.1.1 Attributes

The terms and concepts of the domain are represented as attributes of the ontology. As mentioned earlier, ontology inherits all attributes of its parent ontology. The meaning of each attribute and its syntax (type and format) is standardized so that it has a precise interpretation and use.

Example 2:

An earthquake can have attributes like the date it occurred, region where it occurred, latitude and longitude for its epicenter, a description, number of people that died, magnitude and an image showing some damage or fault line as seen in Figure 1. We will also use the following notation to represent ontology.

```
Earthquake ( eventDate, description, region, magnitude,
             latitude, longitude, numberOfDeaths,
             damagePhoto );
```

2.1.2 Domain rules

The characteristics and semantics of a domain are described by domain rules and functional dependencies. Domain rules describe a condition that always holds true for all entities in the domain. These can be used for query validation and optimization of query plans.

Example 3:

A simple fact that the latitude of an earthquake should lie between -90° and 90° can be described by the following rule on the ontology `Earthquake`:

```
latitude >= -90 and latitude <= 90
```

The ontology can then be represented as follows:

```
Earthquake ( eventDate, description, region, magnitude,
             latitude, longitude, numberOfDeaths,
             damagePhoto,
             latitude >= -90 and latitude <= 90 );
```

2.1.3 Functional Dependencies (FD)

A functional dependency specifies that two entities having the same values for all attributes appearing on the left-hand side (LHS) of the FD will have the same values for the corresponding variables appearing on the right-hand side (RHS). The domain rules and FDs help in understanding the characteristics of the domains. Web sources constitute a large portion of the set of resources available to the system. The integrated domain model is a comprehensive model and it is very likely to come across resources that do not provide certain attributes of the domain model. The information about the FDs of a domain is used by the system to retrieve information (attributes) that is missing from a

resource by using another resource (an *associate resource*) thereby deducing extra information and retrieving more comprehensive results with the same available resources.

Example 4:
A functional dependency on the ontology `Earthquake` could be that the `region` implies the `latitude` and `longitude` of the place. This can be represented as:

```
region -> latitude longitude
```

The ontology can therefore be represented as follows:

```
Earthquake ( eventDate, description, region, magnitude,
             latitude, longitude, numberOfDeaths,
             damagePhoto,
             latitude >= -90 and latitude <= 90,
             region -> latitude longitude );
```

Assume that the system has access to two sources that provide information about earthquakes - `SignificantEarthquakesDB` that provides the `eventDate`, `description`, `region`, `magnitude`, `latitude`, `longitude`, `numberOfDeaths`, and `damagePhoto` for significant earthquakes (say, with a magnitude > 5) all over the world and `EarthquakesAfter1990` that provides `eventDate`, `region`, `magnitude`, `numberOfDeaths`, and `damagePhoto` for earthquakes that occurred after January 1, 1990. Suppose the user has the following information request:

"Find all earthquakes with epicenter in a 5000 mile radius area around the location at latitude 60.790 North and longitude 97.570 East."

Assume that the calculation of the distance between two locations, given their latitudes and longitudes, is possible. Also, the ontology for earthquake does have attributes `latitude` and `longitude` that specify these. So the system should be able to answer the query. However, the resource `EarthquakesAfter1990` does not supply values for these attributes. Hence, we would need to return only the information that was available from the resource `SignificantEarthquakesDB`. The results that the user sees will have all significant earthquakes (with magnitude > 5) that occurred in that region. But to be able to return *all* earthquakes, the system can use the knowledge about the domain that `region -> latitude longitude`. The system can check if it can deduce the latitudes and longitudes of epicenters of any earthquakes in `EarthquakesAfter1990` by checking the regions of earthquakes available from `SignificantEarthquakesDB`. For all the earthquakes for which the latitude and longitude of the epicenter could be deduced, the system can now check if they fall within the 5000 mile radius area. Using the resource `EarthquakesAfter1990` to answer this query could not have been possible without the use of the FD.

2.2 Resources
Information sources in InfoQuilt are described in terms of their corresponding ontologies. This description is meant to supply details about the resource, which can be used for

efficient planning. It consists of a set of attributes, binding patterns (BP), data characteristic (DC) rules, and local completeness (LC) rules. Sections 2.2.1 to 2.2.4 describe each of them and the rationale for including them in the knowledge base. The goal is: [LRO96].

- To be able to add, modify and delete resources for an ontology dynamically without affecting the systems knowledge
- To be able to specify the sources in a manner such that one can declaratively query them
- To be able to identify the exact usefulness of resources in the context of a particular IScape and prune the others

2.2.1 Resource Attributes

Since the way the resource perceives its data can be different than our perception represented by the ontology of the domain, the data needs to be mapped after it is retrieved. This allows for interoperability between sources with heterogeneous data [Gun00]. Additionally, since the ontology is ideally a comprehensive perception of the domain, it is common to come across resources that supply only a part of the information. In other words, a resource may not provide values for all the attributes of the ontology. The resource's description therefore lists the attributes that it can provide.

Example 5:

Consider the resources `SignificantEarthquakesDB` and `EarthquakesAfter1990`. The example mentions a list of attributes that the resources provide values for. The resources would therefore be represented as follows:

```
SignificantEarthquakesDB ( eventDate, description, region,
                           magnitude, latitude, longitude,
                           numberOfDeaths, damagePhoto );

EarthquakesAfter1990 ( eventDate, region, magnitude,
                       numberOfDeaths, damagePhoto );
```

We will use the notation above to represent resources.

2.2.2 Binding Pattern (BP)

The sources accessible to the system that are not databases do not have the ability to execute queries. Most of the web sources fall in this category. However, some web sources have a limited query capability. This is supported by allowing users to search, based on some attributes. These web sources require that values for some attributes be provided to retrieve results. The source can be queried only by specifying values for those attributes. Additionally, some sources may provide finite number of optional ways to query (different sets of attributes). For example, a user can query a site providing information about movies by actors, director, title, etc. Such query limitations and characteristics of resources are important to consider while planning a query. Some of the common motivations for having such limitations are [Dus97]:

- **Privacy/Security**

For example, a company might not want one to be able to get a listing of all its employees but allows a search on an employee by name.

- **Performance**
  The data could be indexed and would then be more efficient to query the source on the indexed attributes for efficient retrieval.
- **Efficiency**
  The data might be enormous and hence it might not be efficient to query all of it. The values supplied for bound attributes are used to narrow down the search and limit the data retrieved.

They are represented in the knowledge base as a set of binding patterns (BP). A BP is a set of attributes that the system must be able to supply values for in order to query the resource. If the resource has several BPs, the system can select the most appropriate one. The values for the BP can be obtained from the query constraint or provided by some other resource.

Example 6:
Consider the `Flight` ontology that represents a flight from one city to another within United States. Most web sites for air travel reservation have forms that require the user to specify source, destination, dates of travel, etc. One or more combinations of values could be required as input for the web site to obtain any useful information. One of the possible combinations of BP could be:

```
[fromCity, fromState, toCity, toState, departureDate]
```

Suppose we use the AirTran Airway web site (www.airtran.com) as a source. The resource would then be represented as follows:

```
AirTranAirways ( airlineCompany, flightNumber, fromCity,
                 fromState, toCity, toState, departureDate,
                 fare, departureTime, arrivalTime,
                 [fromCity, fromState, toCity, toState,
                 departureDate] );
```

2.2.3 Data Characteristic (DC) Rules
Data characteristic rules are similar to domain rules discussed in section 2.1.2 but they apply only to the specific resource. Knowledge about the data characteristics of a resource can be useful for the system to predict whether a resource will provide any useful results for the particular information request. This can be used to prune the resource if it is possible to infer from the DC rules that the resource will not provide any useful results for the Iscape. It can also be used to optimize the query plan by eliminating a constraint if it can be deduced that the constraint will always be true for all the data retrieved from that resource. The DC rules are also used to select an appropriate associate resource for a resource with BP or missing attributes.

Example 7:
Consider the `AirTranAirways` resource introduced in the previous example that provides data for the ontology Flight. We know that it only provides information about all flights

operated by AirTran Airways. This characteristic of the resource can be represented as follows:

```
AirTranAirways ( airlineCompany, flightNumber, fromCity,
                 fromState, toCity, toState, departureDate,
                 fare, departureTime, arrivalTime,
                 [dc] airlineCompany = "AirTran Airways",
                 [fromCity, fromState, toCity, toState,
                 departureDate] );
```

We use [dc] and [lc] to distinguish data characteristic (DC) rules from local completeness (LC) rules. LC rules are described in section 2.2.4.

Consider the following query using the ontology `Flight`.

"*Find all the flights operated by Delta Airlines from Boston, MA to Los Angeles, CA on February 19, 2001.*"

Clearly, `AirTranAirways` will not provide any information about a Delta flight. The system can use this knowledge to deduce that the resource `AirTranAirways` should not be used to answer this query.

Now suppose the user modifies the query to look for flights operated by AirTran Airways. This time the system knows that it can use `AirTranAirways`. Additionally, it can also eliminate the check for "AirTran Airways" since all flights available from AirTran Airways are known to be operated by AirTran Airways.

2.2.4 Local Completeness (LC) Rules
A local completeness (LC) rule on a resource has the same format as a DC rule or a domain rule. But it has a different interpretation. A characteristic of web sources is overlapping and incomplete information. Hence, using just one source to answer an information request in many cases does not guarantee retrieval of all the possible information. The general approach to this solution has been to use all the sources (for that ontology) and compute a union of the results retrieved from all of them to provide maximum possible information to every request. However, it is also possible to find sources that do provide complete information about some subset of the domain. LC rules are used to model this. They specify that the resource is complete for a subset(s) of information on a particular ontology. In other words, the information contained in the resource is *all* the information for the subset (specified by the rule) of the domain. Hence, the system cannot retrieve any additional information (for that subset) by querying other sources.

Example 8:
Consider the AirTranAirways resource used earlier again. We know that information about *all* flights operated by AirTran Airways will be available from it. It is thus locally complete for all flights with airlineCompany = "AirTran Airways".

```
AirTranAirways ( airlineCompany, flightNumber, fromCity,
```

```
                      fromState, toCity, toState, departureDate,
                      fare, departureTime, arrivalTime,
                      [dc] airlineCompany = "AirTran Airways",
                      [lc] airlineCompany = "AirTran Airways",
                      [fromCity, fromState, toCity, toState,
                       departureDate] );
```

Now, any information request that needs only the subset of flights that are operated by AirTran Airways can use only `AirTranAirways` to retrieve data about all such flights. This would be faster than querying all sources available for `Flight`.

2.3 Inter-Ontological Relationships

The real-world entities that were classified into domains and sub-domains can be related to each other in various ways. These relationships can be very simple like the "is a" relationship implied by inheritance. For example, a nuclear test "is a" man-made disaster and so on. They can also be very complex, for example, an earthquake can "cause" a tsunami. A relationship may involve more than two domains (ontologies). Such complex inter-ontological relationships represent real-world concepts and characteristics. A novel feature of InfoQuilt is that it provides an infrastructure to model and learn about such complex relationships between different ontologies and to use them to specify information requests. Modeling such inter-ontological relationships requires understanding the semantics involved in their interaction and cannot be expressed using only simple relational and logical operators. The relationships already known can be directly modeled (defined) using the Knowledge Builder toolkit. Additionally, we may want to explore hypothetical relationships and formalize them, if they can be established by using the available information sources. Established relationships can be used to study the interaction between the involved ontologies. Further two relationships "A affects B" and "B affects C" could lead to transitive finding "A affects C".

Example 9:
Consider the relationship between a nuclear test and an earthquake. We use the `Earthquake` ontology from the example 4 here. Suppose we model nuclear test as follows:

```
NuclearTest ( testSite, explosiveYield, bodyWaveMagnitude, testType,
              eventDate, conductedBy, latitude, longitude,

                bodyWaveMagnitude > 0,
                bodyWaveMagnitude < 10,
                testSite -> latitude longitude );
```

We can say that some nuclear test could have "caused" an earthquake if we see that the earthquake occurred "some time after" the nuclear test was conducted and "in nearby region". Notice the use of operators "some time after" and "in nearby region". These are specialized user defined operators that are not a part of the set of relational operators (<, >, <=, etc.). These are possible due to InfoQuilt's ability to support user defined functions, as described in section 2.4. For now, assume that there are two functions called `dateDifference` and `distance` available to the system. The function `dateDifference` takes two dates as arguments and returns number of days from date1 to date2. The

10

function `distance` takes the latitudes and longitudes of two places and calculates the distance between them. Given that we can use these functions, we can represent the relationship as follows:

```
NuclearTest Causes Earthquake:

dateDifference(NuclearTest.eventDate, Earthquake.eventDate) < 30
                            AND
distance(NuclearTest.latitude, NuclearTest.longitude,
        Earthquake,latitude, Earthquake.longitude) < 10000
```

The values 30 and 10000 here are arbitrary. In fact, a user can try different values to analyze the data. This is a part of the knowledge discovery support that the system's framework provides, as described further in section 4.

2.4 Operations

A distinguishing feature of InfoQuilt is its framework to support use of operations. As seen in the previous example, we used the functions `dateDifference` and `distance` as operators to describe a complex inter-ontological relationship between `NuclearTest` and `Earthquake`. The user can also use them to specify constraints in their information requests.

Example 10:

Consider the information request:

"Find all earthquakes with epicenter in a 5000 mile radius area of the location at latitude 60.790 North and longitude 97.570 East"

The system needs to know how it can calculate the distance between two points, given their latitudes and longitudes, in order to check which earthquakes' epicenters fall in the range specified.

Operations are also used to resolve syntactic heterogeneities between sources by providing a fuzzy matching mechanism to map the two sources. Consider the use of functional dependency `region -> latitude longitude` from example 4 to solve the missing attribute (latitude and longitude) problem of the source `EarthquakesAfter1990`. It is highly unlikely that the sources `SignificantEarthquakesDB` and `EarthquakesAfter1990` will have exact same values for the attribute region. The value available from one source could be "Nevada, USA" and that from another source could be "NV, USA". The two are semantically equal but syntactically unequal. [KS96]

Another important advantage of using operations is that the system can support complex post-processing of data. An interesting form of post-processing is the use of simulation programs. These independent programs can be integrated with the system. For instance, researchers in the field of Geographic Information Systems (GIS) use simulation programs to forecast characteristics like urban growth in a region based on a model.

InfoQuilt supports the use of such simulations like any other operation. They provide valuable additional information that is not often available from the resources directly.

Example 11:
Clarke's Urban Growth Model [Cla] is a model to forecast the urban growth in a region based on information about the areas in the region where it is known that growth cannot occur due to some reasons, roads, slopes and vegetation in the region. It requires that this information be specified as a set of maps and generates a series of maps showing the progressive urban growth using a specified time step. This simulation can be run on data that can be retrieved from some resource (or multiple resources) that has the maps needed assuming that they are in the format that the program expects.

To be able to dynamically and easily add new operations, update and delete existing ones, InfoQuilt maintains what is known as a *Function Store*. The Function Store contains information about all the functions known to the system. We will use the terms *function* and *operation* interchangeably in the rest of the chapter. The description of a function contains its name, a description of its functionality, a list of arguments that it takes as inputs along with their types and descriptions of what they are, the type of return value, and information about how the system can use it. The user provides an implementation for the function. Once the implementation is provided and it has been added to the Function Store, the system can make use of it as described earlier.

3. Information Scapes
The goal of the InfoQuilt system is to develop a framework for knowledge discovery support through use of the available resources of data. Users can form complex information requests on data available from multiple heterogeneous distributed resources. Note that we use the term "information request" instead of "query". A query generally explicitly specifies the exact sources (tables in a RDBMS) that need to be used and how the data from these sources should be integrated (join conditions in a RDBMS). Additionally, it does not "understand" what the user is asking. An information request, on the other hand, can understand what the user is enquiring about by embedding semantic information within the request[2]. In InfoQuilt, we refer to them as Information Scapes or *IScapes*. This work has been built upon the work done in [Pal00, SS98].

 Example 12:
Again consider the following information request.

"Find all earthquakes with epicenter in a 5000 mile radius area of the location at latitude 60.790 North and longitude 97.570 East and find all tsunamis that they might have caused."

In addition to the obvious constraints, the system needs to understand what the user means by saying "*find all tsunamis that might have been caused due to the earthquakes*".

---

[2] Use of ontologies, context and relationships are critical in defining information requests and in supporting semantics – see for example DS-6 proceedings, esp. [Wie97] and [She96]

The relationship that *an earthquake caused a tsunami* is a complex inter-ontological relationship.

Any system that needs to answer such information requests would need a comprehensive knowledge of the terms involved and how they are related. An IScape is specified in terms of the components of the knowledge base of the system such as ontologies, inter-ontological relationships and operations, which the system understands. This helps the system in understanding the semantics of the request. Additionally, it abstracts the user from having to know the actual sources that will be used by the system to answer it and how the data retrieved from these sources will be integrated.

An IScape is defined as a set of information semantically related in a complex manner. We use XML to represent an IScape. It specifies the ontologies involved, inter-ontological relationships, preset constraint, runtime configurable constraint, how the results should be grouped, any aggregations that need to be computed or constraints that need to be applied to the grouped data and finally the information that needs to be returned in the result to the user. The ontologies in the IScape identify the domains that are involved in the IScape and the inter-ontological relationships specify how they are related to each other. The preset constraint and the runtime configurable constraint are filters used to describe the subset of data that the user is interested in. For example, a user may be interested in earthquakes that occurred in only a certain region and had a magnitude greater than 5. The difference between the preset constraint and the runtime constraint is that the runtime constraint can be set at the time of executing the IScape. In the IScape template certain constraints are pre-set, while some others can be configured at execution. The runtime constraint forms an important mechanism to support the knowledge discovery, as described in section 4. Constraints are essentially conjunctive boolean expressions. The results of the IScape can be grouped based on attributes and/or specified functions. If the results are to be grouped, the user can also specify any aggregations that the user wants to be returned as a part of the result or specify additional constraint on the groups formed (similar to the HAVING clause in the SELECT statement in SQL). The aggregates supported by the system are sum (SUM), average (AVG), count (COUNT), minimum (MIN) and maximum (MAX). Finally, the user specifies a list of all the information that is to be returned as a part of the result. We refer to this as the *projection list*. It could include certain information modeled by the ontologies directly (attributes), aggregates, values of functions evaluated on the data, and results of some simulation programs.

A graphical toolkit known as the *IScape Builder* is available for the user to construct and execute these IScapes and analyze the results. It provides a platform for ease of development and deployment of IScapes using the knowledge base without having to understand the underlying formats used by the system.

4. Knowledge Discovery

InfoQuilt provides a framework that allows users to access data available from a multitude of diverse autonomous distributed resources and provide tools that help them to analyze the data to gain a better understanding of the domains and the inter-domain

relationships as well as help users to explore the possibilities of new relationships. This section discusses the conceptual framework using a series of examples, while the next section discusses some of the software components of the InfoQuilt framework.

The inter-ontological relationships defined in the knowledge base of the system describe the interaction that exists between domains as they contain embedded semantic information. Existing relationships provide a scope for discovering new aspects of relationships through transitive learning as discussed in example 13.

Example 13:
Consider the ontologies `Earthquake, Tsunami` and `Environment`. Assume that the relationships *Earthquake affects Environment*, *Earthquake causes Tsunami* and *Tsunami affects Environment* are defined and known to the system. We can see that since Earthquake causes a Tsunami and Tsunami affects the environment, effectively this is another way in which an earthquake affects the environment (by causing a tsunami). If this aspect of the relationship between an earthquake and environment was not considered earlier, it can be studied further.

Another valuable source of knowledge discovery is studying existing IScapes that make use of the ontologies, their resources and relationships to retrieve information that is of interest to the users. The results obtained from IScapes can be analyzed further by post processing of the result data. For example, the Clarke UGM model forecasts the future patterns of urban growth using information about urban areas, roads, slopes, vegetation in those areas and information about areas where no urban growth can occur.

For the users that are well-versed with the domain, the InfoQuilt framework supports exploring new relationships. The data available from various sources can be analyzed by using charts, statistical analysis techniques, etc. to study and explore trends or aspects of the domain. Such analysis can be used to hypothesize new relationships between domains and to see if the data invalidates the hypothesis or supports it sufficiently as demonstrated by example 14.

Example 14:[3]
Several researchers in the past have expressed their concern over nuclear tests as one of the causes of earthquakes and suggested that there could be a direct connection between the two. The underground nuclear tests cause shock waves, which travel as ripples along the crust of the earth and weaken it, thereby making it more susceptible to earthquakes. Although this issue has been addressed before, it still remains a hypothesis that is not conclusively and scientifically proven. Suppose we want to explore this hypothetical relationship.

---

[3] The information presented in this example is based on the findings of Gary T. Whiteford published in the paper "Earthquakes and Nuclear Testing: Dangerous Patterns and Trends" presented at the 2nd Annual Conference on the United Nations and World Peace in Seattle, Washington on April 14, 1989. It has been recognized as the most exhaustive study yet of the correlation between nuclear testing and earthquakes. [Whi89]

14

Consider the `NuclearTest` and `Earthquake` ontologies again. We assume that the system has access to sufficient resources for both the ontologies such that they together provide sufficient information for the analysis. These resources include non-traditional, non-database sources like web-based sources.

If the hypothesis is true, then we should be able to see an increase in the number of earthquakes that have occurred after the nuclear testing started. We proceed as follows. First, we check to see when nuclear testing began.

IScape 1:
"*Find when was the earliest recorded nuclear test conducted.*"

We find that nuclear testing began in 1950. Next we check the trend of the number of earthquakes that have occurred since the nuclear testing started. It is believed that some earthquakes below the intensity of 5.8 on the Richter scale would have passed unrecorded in the earlier part of the century when measuring devices were less sensitive and less ubiquitous. But for bigger earthquakes, the records are detailed and complete [Whi89]. We therefore check the number of earthquakes with a magnitude 5.8 or higher occurring every year in this century.

IScape 2:
"*Find the total number of earthquakes with a magnitude 5.8 or higher on the Richter scale per year starting from year 1900.*"

We can then plot a chart to analyze the trend in the number of earthquakes occurring every year. It reveals that there seems to be a sudden increase in the number of earthquakes since 1950. We modify the query to try to approximately quantify this rise.

IScape 3:
"*Find the average number of earthquakes with a magnitude 5.8 or higher on the Richter scale per year for the period 1900-1949 and for the period 1950-present.*"

We see that in the period 1900-1949, the average rate of earthquakes was 68 per year and that for 1950-present[4] was 127 per year, that is it almost doubled [Whi89].

Next, we try to analyze the same data grouping the earthquakes by their magnitudes.

IScape 4:
"*For each group of earthquakes with magnitudes in the ranges 5.8-6, 6-7, 7-8, 8-9, and magnitudes higher than 9 on the Richter scale per year starting from year 1900, find the average number of earthquakes.*"

---

[4] The period of 1950-1989 implies the period 1950-1989, since the data presented here was published by Gary T. Whiteford in 1989.

The results show that the average number of earthquakes with magnitude greater than 7 on the Richter scale have remained practically constant over the century (about 19) [Whi89]. We can therefore deduce that the earthquakes caused by nuclear tests usually are of magnitudes less than 7 on the Richter scale. We can then try to explore the data at a finer level of granularity by trying to look for specific instances of earthquakes that occurred within a certain period of time after a nuclear test was conducted in a near by region.

IScape 5:
"*Find nuclear tests conducted after January 1, 1950 and find any earthquakes that occurred not later than a certain number of days after the test and such that its epicenter was located no farther than a certain distance from the test site.*"

Note the use of "*not later than a certain number of days*" and "*no farther than a certain distance*". The IScape does not specify the value for the time period and the distance. These are defined as runtime configurable parameters, which the user can use to form a constraint while executing the IScape. The user can hence supply different values for them and execute the IScape repeatedly to analyze the data for different values without constructing it repeatedly from scratch. This is where runtime configurable constraint is useful. Also, note the use of functions as user-defined operators to calculate the difference in date (`dateDifference`) and the distance between the epicenter of the earthquake and the nuclear test site (`distance`). Some of the interesting results that can be found by exploring earthquakes occurring that occurred no later than 30 days after the test and with their epicenter no farther than 5000 miles from the test site are listed below.

- China conducted a nuclear test on October 6, 1983 at Lop Nor test site. USSR conducted two tests, one on the same day and another on October 26, 1983, both at Easter Kazakh or Semipalitinsk test site. There was an earthquake of magnitude 6 on the Richter scale in Erzurum, Turkey on October 30, 1983, which killed about 1300 people. The epicenter of the earthquake was about 2000 miles away from the test site in China and about 3500 miles away from the test site in USSR. The second USSR test was just 4 days before the quake.
- USSR conducted a test on September 15, 1978 at Easter Kazakh or Semipalitinsk test site. There was an earthquake in Tabas, Iran on September 16, 1978. The epicenter was about 2300 miles away from the test site.
- More recently, India conducted a nuclear test at its Pokaran test site in Rajasthan on May 11, 1998. Pakistan conducted two nuclear tests, one on May 28, 1998 at Chagai test site and another on May 30, 1998. There were two earthquakes that occurred soon after these tests. One was in Egypt and Israel on May 28, 1998 with its epicenter about 4500 miles away from both test sites and another in Afghanistan, Tajikistan region on May 30, 1998, with a magnitude of 6.9 and its epicenter about 750 miles away from the Pokaran test site and 710 miles from Chagai test site.

5 Visual Interfaces

This section describes the graphical tools we provide to aid in creation of the knowledge base, creation and execution of Iscapes and for monitoring the execution of Iscapes. These components of the InfoQuilt system are most relevant to the knowledge discovery which is the focus of this chapter, the remaining components - primarily the distributed agent based run-time system is described in [PS01].

5.1 Knowledge Builder

The Knowledge Builder (KB) is the graphical toolkit that helps the user in creating the knowledge base of the system. It allows the user to declaratively specify ontologies, relationships, resources and functions (including simulations). Use of KB provides the following advantages:

- The user does not need to know the format of the XML specification used by the system to represent ontologies, relationships and functions. The KB provides an easy to follow graphical interface that the users can use and thereby provides an abstraction from the details of the formats used internally by the system. See figure 2.
- The KB provides tools that help the user relate the information in the knowledge base in a better way. For example, the user can look at the entire knowledge base as a graphical tree that lists all the ontologies defined in the system, their details including rules and FDs defined on them, relationships involving the ontology with their details and resources available for the ontologies with their details like the attributes they supply, the data characteristic rules, the local completeness rules and the binding patterns that they need.
- The KB structures the knowledge in a manner that allows for easy access to knowledge and in the form that can help one understand and learn about a domain.
- The KB also helps the user while trying to modify the knowledge base. For example, a user may want to remove an attribute from an ontology. The KB will not allow this if the attribute is being used in a rule on the ontology, a FD on the ontology, set of attributes provided by some resource, a data characteristic rule on a resource, a local completeness rule on a resource or a binding pattern on a resource. The user is required to manually take care of these before removing the attribute. If the knowledge base is huge, it may be a tedious and error-prone task to go through the specifications of all these to find where the attribute is being used. Even worse, the user would have to go through the XML specifications in the absence of a tool like KB to correctly make the modification. Using the graphical tree display provided by the KB, the user finds it easier locate such uses of the attribute. See Figure 3.
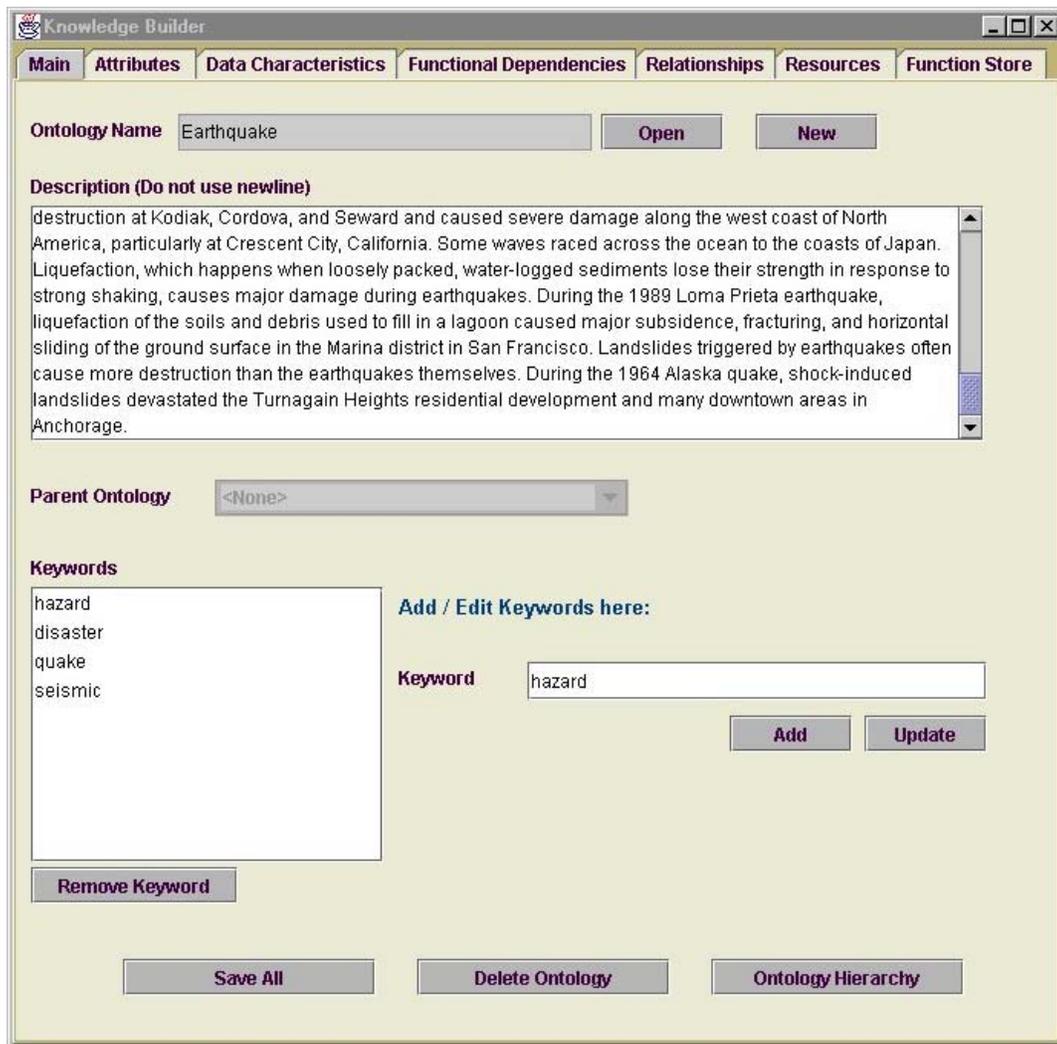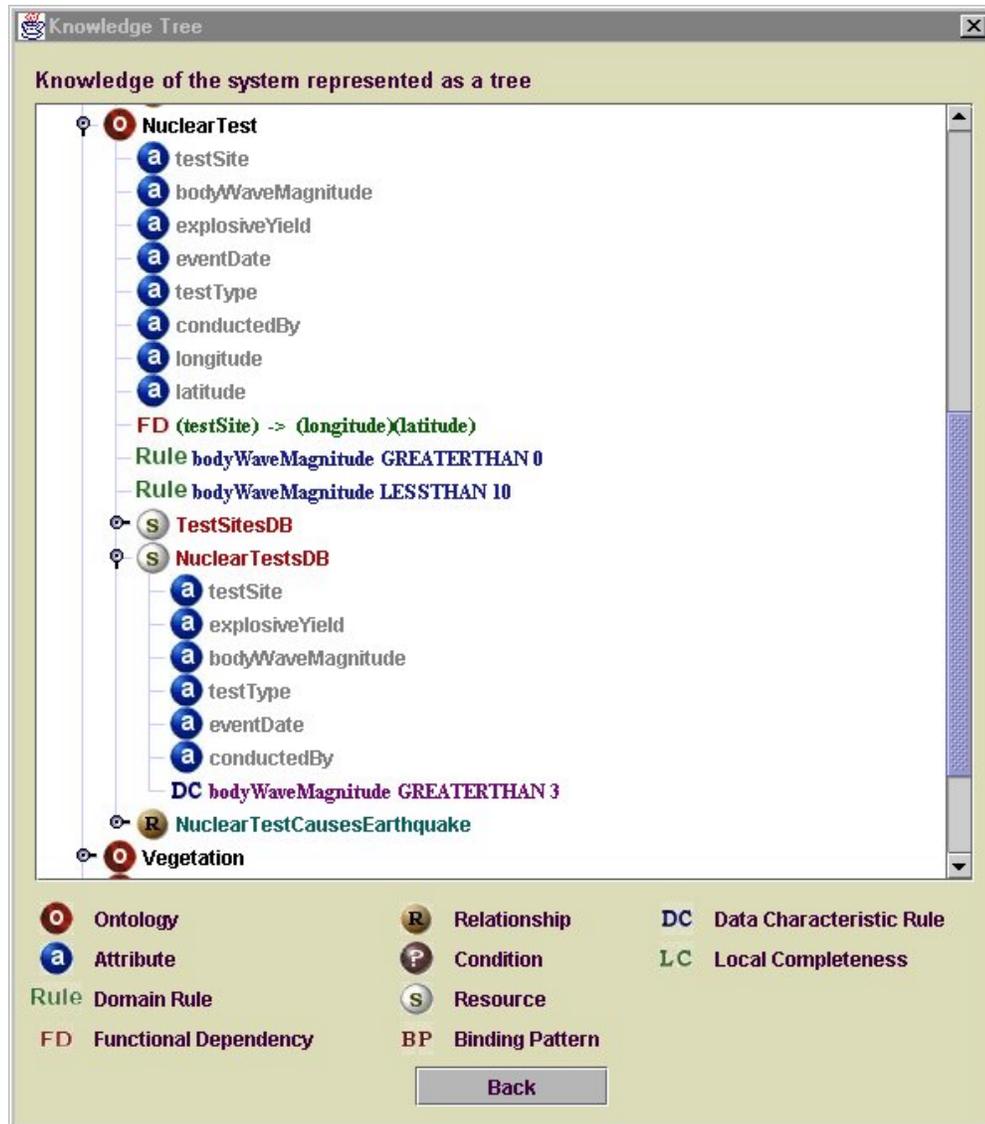
Figure 2: Knowledge Builder

Figure 3: Ontology hierarchy

5.2 IScape Builder

The IScape Builder (IB) is a stand-alone Java application that provides a graphical interface to create and execute IScapes. Use of this tool has the following advantages:

- It provides a simple and intuitive interface that allows the user to create and execute IScapes in a step-by-step manner. See Figure 4.
- The user does not need to be aware of the format of the XML specification used internally by the system to represent IScapes.
- It is aware of the knowledge base of the system. The user therefore does not need to look it up to create new IScapes. For example, names of ontologies, relationships and functions appear in combo boxes wherever required. The user can easily make his selection from there.

- It implements basic validity checks to make sure that the IScape being created is valid. For example, if it involves use of a function call, the tool makes sure that the user has specified where the values for all the arguments to the function should be supplied from. It also performs a type match.
- It can provide various tools to help users better analyze the results of the IScapes. For example, the current version of the IB provides a charting tool, which allows the user to create charts to analyze the results. This helps in providing a learning environment to the users.
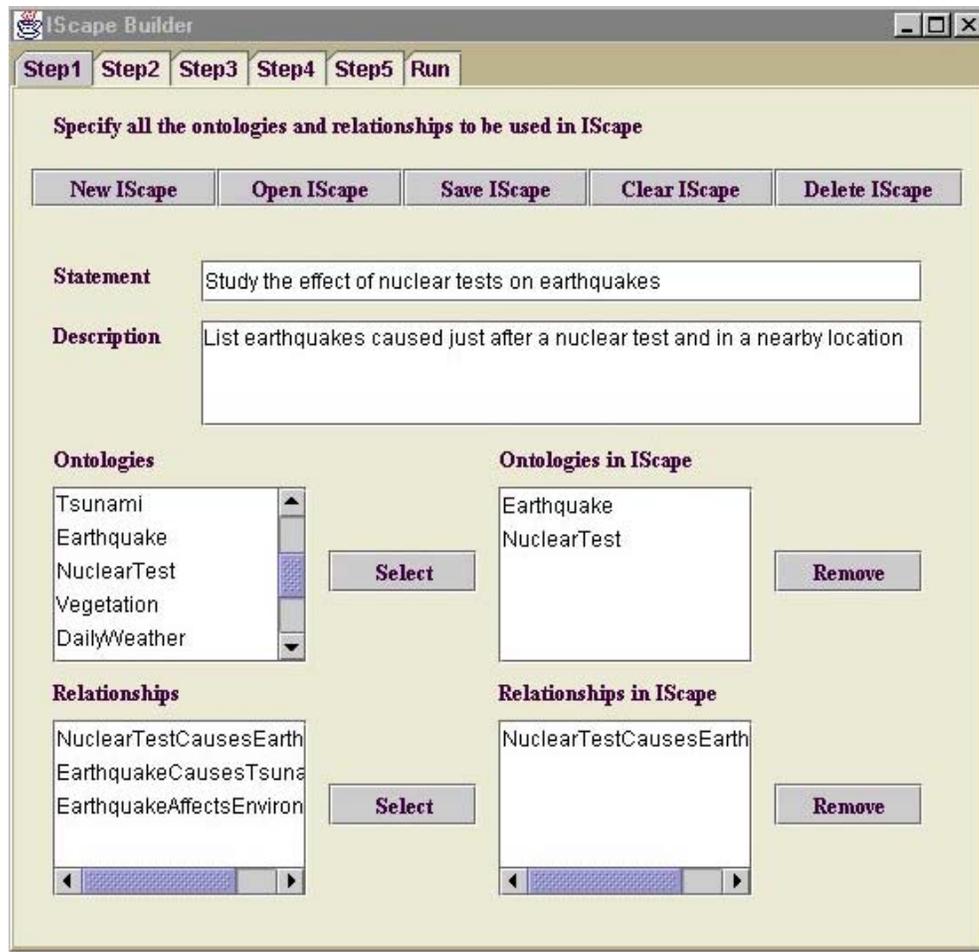

Figure 4: Iscape Builder

5.3 Web-accessible interface to execute IScapes

We also provide a web-accessible interface that provides a learning environment and allows execution of already existing IScapes.

- The interface is web-accessible. It allows execution of existing IScapes by setting the runtime constraint. However, it does not allow users to create new IScapes. The result of executing the `NuclearTest` causes `Earthquake` is demonstrated in Figure 5.

- It describes the entire knowledge base of the system that helps the users in understanding the domains that are modeled by the ontologies, how the ontologies are related to each other in complex ways (inter-ontological relationships like `affects`, `causes`, etc) and the functions and simulations available in the system.
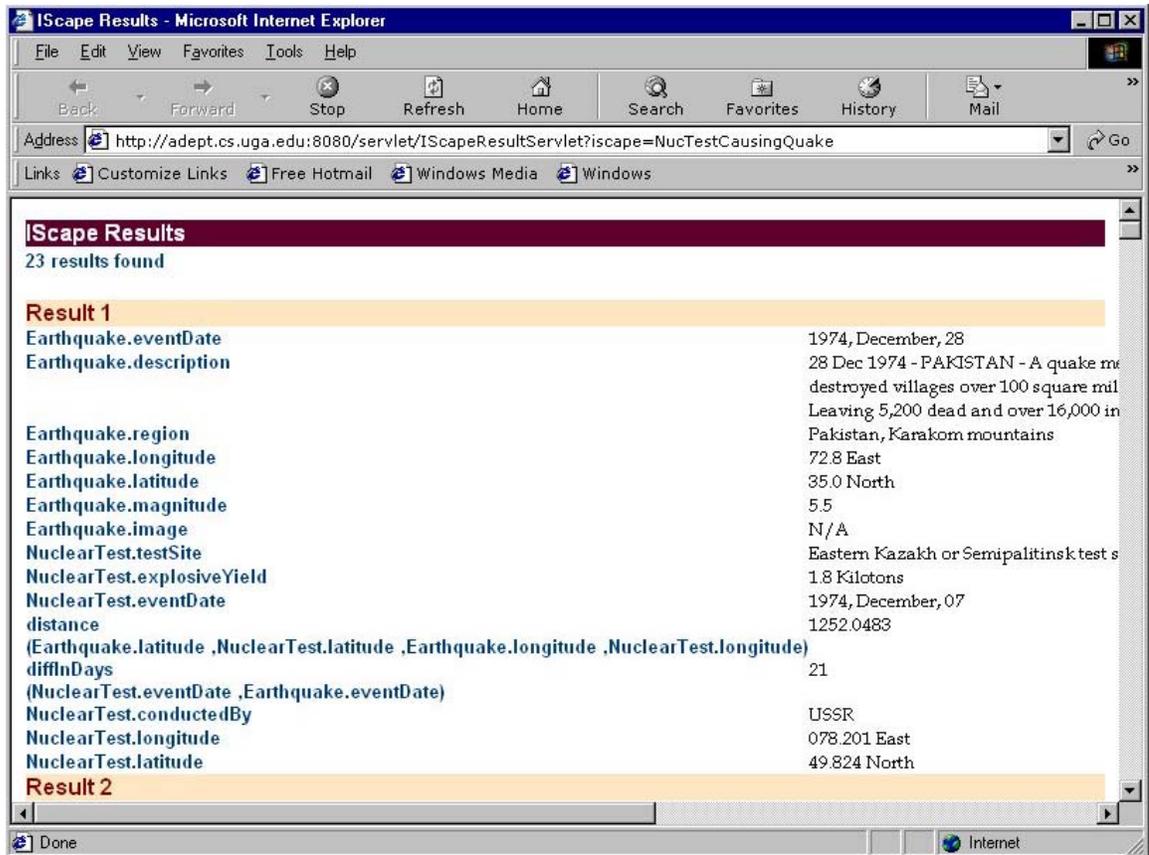

Figure 5: Iscape Results using the Web interface

5.4 IScape Processing Monitor

The IScape Processing Monitor (IPM) is a graphical interface used to monitor the execution of IScapes. The user can choose to run the IScape with or without the processing monitor. The web-accessible interface however does not support the monitor.

The various agents in the runtime system generate log entries that are sent to the monitor and displayed as color-coded table. The log entries can be very detailed. They include a time stamp, the name of the agent sending the log, a brief message, a more descriptive message and associated data, if any. The associated data could be, for example, the execution plan generated by the planner or the results at various stages in the IScape execution.

The IPM is very useful in the following ways:
- It helps in monitoring the execution of the IScape as it proceeds and locate any failures easily.

- The availability of time-stamps with all the log entries allows us to evaluate which phases of the IScape processing are taking the most time. This helps us evaluate our system better and identify areas that need improvement.
- The detailed log messages generated by various agents describe in considerable detail exactly what is going on during processing. The IPM is therefore extremely useful as a high-level debugging tool.

IScape Processing Monitor

IScape Find all nuclear tests conducted after January 1, 1985 with seismic body wave magnitude > 5 and find all earthquakes that could have been caused due to these tests

Processing Log :

Message Id	Time Stamp	Message From	Brief Message
0	12:02:44.171	User Agent	Started processing
1	12:02:44.453	Broker Agent	Started processing
2	12:02:48.406	Planning Agent	Applied Domain Rules to IScape's constraint
3	12:02:48.500	Planning Agent	Checking IScape's constraint and relationshi...
4	12:02:48.593	Planning Agent	Selecting sources for Earthquake
5	12:02:48.687	Planning Agent	Selecting sources for NuclearTest
6	12:02:48.812	Planning Agent	Plan created by the planner. Returning it to Br...
7	12:02:49.125	Broker Agent	Received plan from planner
8	12:02:50.015	CorrelationAgent	Executing IScape
9	12:02:57.906	TestSitesDB Resource Agent	Queried TestSitesDB
10	12:02:58.093	SignificantEarthquakesDB Resource Agent	Queried SignificantEarthquakesDB
11	12:02:58.171	Correlation Agent	Computed Union
12	12:02:58.156	NuclearTestsDB Resource Agent	Queried NuclearTestsDB
13	12:02:58.515	Correlation Agent	Evaluated Join
14	12:02:58.687	CorrelationAgent	SelectNodeProcessor done processing
15	12:02:58.734	Correlation Agent	Computed Union
16	12:02:59.109	Correlation Agent	Evaluated functions on a relation.
17	12:02:59.187	CorrelationAgent	SelectNodeProcessor done processing
18	12:02:59.781	Correlation Agent	Evaluated Relationship on a set of relations
19	12:02:59.828	Correlation Agent	Evaluated projection on a relation
20	12:02:59.937	Broker Agent	Received IScape results from Correlation Ag...
21	12:03:00.078	User Agent	Returning final result to client

Detailed Log Message   Show Associated Data

Figure 6: Iscape Processing Monitor

6 Related Work

SIMS [AKS96], TSIMMIS [CMH+94], Information Manifold [LRO96], OBSERVER [MIKS00] and InfoSleuth [BBB97] are some of the efforts to integrate information from multiple heterogeneous sources for querying. Most other systems focus on retrieving and integrating "data" from multiple sources and not on the "learning, exploring and understanding" aspects. The following are the features of InfoQuilt that are not supported by any other systems:

- A framework to study, analyze and learn about domains and inter-domain relationships.
- Functions and simulations to post-process the data retrieved from the resources thereby adding value to the data
- Complex relationships and constraints representing the semantic correlation of information across multiple domains that cannot be expressed using relational and logical operators can be modeled
- Powerful semantic query interface (IScapes)

Additionally, in comparing the general approach of information integration to the other systems we identify the following noteworthy differences.

The mediator in SIMS [AKS96] system is specialized to a single application domain, which is source dependent and query dependent. An application domain in SIMS models a single hierarchy of classes. It does not support inter-ontological relationships. SIMS uses Loom [Gre90], a member of the KL-ONE family knowledge representation systems to model domains as well as describe information sources. They do not consider use of local completeness information about sources and support only one binding pattern per web resource.

OBSERVER [MIKS00] uses ontologies to describe information sources and inter-ontology relationships like synonyms, hyponyms and hypernyms across terms in different ontologies to be able to translate a query specified using some ontology that the user selected into another query that uses related ontologies describing relevant information. This approach of using relationships to achieve interoperability between the sources is limited to basic relationships.

TSIMMIS [CMH+94, GPQ+95] uses a mediator-based architecture [Wie93] with a query-centric approach. It uses Mediator Specification Language (MSL) to define mediators. The specification encodes how the system should use the available resources. The mediators are then generated automatically from these specifications. Since the MSL definitions need to be created manually, adding or removing information sources is not easy as it requires updating them after determining how the sources should be used to answer the queries and then recompiling them. It has a set of pre-defined query templates that it knows to answer. User queries are then answered by relating them to the existing query templates. The types of queries that can be answered using this approach are limited compared to our system.

Information Manifold (IM) [LRO96] uses an approach similar to ours in that the user creates a *world view*, a collection of virtual relations and classes. The world view however does not capture semantics of the domains as InfoQuilt can using domain rules and FDs. Information sources are described to the system as a query over the relations in the world view. Locally complete resources cannot be modeled precisely. IM uses capability records to capture query capability limitations of sources. These records specify, among others, a set of input parameters and minimum and maximum number of inputs allowed. The system then arbitrarily selects a subset of the set of input parameters with at least the minimum number of allowed parameters in the set. The subset selected is arbitrary. Therefore, the capability records cannot precisely specify the binding patterns.

Semantic network is a related concept in terms of modeling knowledge and relationships [MBJK90]. Unlike information integration systems, the main purpose of semantic network is to create a logical, orderly and aesthetically consistent relationship of pages (page is an elemental unit of semantic network). The relationships modeled are hierarchical or similarity based are can be used to understand semantics involved within

and across domains, but are limited in terms of their ability to model the complex semantic relationships like the "cause" or "affects" relationships.

7 Conclusion And Future Work

This chapter discussed our approach for knowledge discovery on the evolving Semantic Web. It involves modeling knowledge in terms of complex relationships among Web-accessible semantically related but otherwise heterogeneous information. Iscape, a representation used for this comprises of ontologies, inter-ontological relationships, information sources and complex operations including functions and simulations. The chapter described the use of IScapes to construct and deploy complex information requests. Of particular interest was the use of inter-ontological relationships and functions to answer those requests. Simulations can also be integrated with the system to perform post-query analysis on the result. Iscape also form the basis for knowledge discovery, through the ability to study and explore new (complex) relationships.

The IScapes, inter-ontological relationships, support for functions and knowledge discovery are the distinguishing features of InfoQuilt. The system can easily adapt to new sources of information and is very scalable. Further, the system has been implemented and it makes use of query planning and optimization with a multi-threaded execution to exploit the parallelism in the plans [PS01].

Some of the improvements to the system planned are as follows. The system will be enhanced to allow the use of recursive query plans, which would further expand its query capability. It needs to make use of inductive learning to infer rules and use them appropriately, which can speed up query processing [HK94]. In using the current framework to support simulations, further investigation might be necessary to support simulations with more complex interactions.

8 References

[AHK96] Y. Arens, C. Hsu and C. A. Knoblock. Query processing in the SIMS information mediator. In Austin Tate, editor, Advanced Planning Technology. The AAAI Press, Menlo Park, CA, 1996.

[AKS96] Y. Arens, C. A. Knoblock, and W. Shen. Query reformulation for dynamic information integration. Journal of Intelligent Information Systems, Vol. 6, pp. 99-130, 1996.

[BBB97] R. J. Bayardo Jr., W. Bohrer, R. Brice, et al. InfoSleuth: Agent-Based Semantic Integration of Information in Open and Dynamic Environments. In SIGMOD-97, pp. 195-206, Tucson, AZ, USA, May 1997.

[B98] C. Bertram. InfoQuilt: Semantic Correlation of Heterogeneous Distributed Assets. Masters Thesis, Computer Science Department, University of Georgia, 1998.

[Cla] Clarke's Urban Growth Model, Project Gigalopolis, Department of Geography, University of California, Santa Barbara.
http://www.ncgia.ucsb.edu/projects/gig/ncgia.html

[CMH+94] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom. The TSIMMIS Project: Integration of Heterogeneous Information Sources. Proceedings of 10th Anniversary Meeting of the Information Processing Society of Japan, pp. 7-18, Tokyo, Japan, 1994.

[Dus97] O. M. Duschka. Query Planning and Optimization in Information Integration. Ph. D. Thesis, Computer Science Department, Stanford University, 1997.

[FM01] D. Fensel and M. Musen, Eds., The Semantic Web: A Brain for Humankind, Special issue of IEEE Intelligent Systems on Semantic Web Technology, March-april 2001.

[GPQ+95] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, and J. Widom. The TSIMMIS Approach to Mediation: Data Models and Languages. In Proceedings of NGITS (Next Generation Information Technologies and Systems), 1995.

[Gun00] M. Guntamadugu. MÉTIS: Automating Metabase Creation from Multiple Heterogeneous Sources. Masters Thesis, Computer Science Department, University of Georgia, 2000

[HK94] Chun-Nan Hsu and Craig A. Knoblock. Rule Induction for Semantic Query Optimization. Proceedings on the 11th International Conference on Machine Learning, San Mateo, CA, 1994

[KS96] V. Kashyap, A. Sheth. Schematic and Semantic Similarities between Database Objects: A Context-based Approach - in the VLDB Journal 5 (4).

[KS97] V. Kashyap, A. Sheth. Semantic Heterogeneity in Global Information Systems: The Role of Metadata, Context and Ontologies. M. Papzoglou, and G. Schlageter, (Eds.), Academic Press,1997

[KS00] V. Kashyap. A. Sheth. Information Brokering Across Heterogeneous Digital Data – A Metadata-based Approach. Kluwer Academic Publishers, 2000.

[Lak00] Laksminarayanan S. Achieving Semantic Inter-operability in Digital Libraries by use of Inter-Ontological Relations. Masters Thesis, Computer Science Department, University of Georgia, 2000.

[LHL] Time Berners-Lee, James Hendler and Ora Lassila. The Semantic Web published as an article in Scientific American.
http://www.sciam.com/2001/0501issue/0501berners-lee.html

[LRO96] A. Y. Levy, A. Rajaraman, J. J. Ordille. Querying heterogeneous information sources using source descriptions. Proceedings of the 22nd International Conference on Very Large Databases VLDB-96, Bombay, India, September 1996.

[MBJK90] J. Mylopoulus, A. Borgida, M. Jarke, M. Koubarakis. Telos: Representing knowledge about information systems. ACM transaction on information systems, 1990.

[MIKS00] E. Mena, A. Illarramendi, V. Kashyap, A. P. Sheth. OBSERVER: An approach for query processing in global information systems based on interoperation across pre-existing ontologies. International Journal on Distributed and Parallel Databases, Vol. 8, No. 2, pp. 223-271, April 2000.

[Ont] Ontology-based information exchange for knowledge management and electronic commerce. http://www.ontoweb.org.

[Pal00] N. Palsena. A collaborative Approach to learning Using Information Landscapes. Masters Thesis, Computer Science Department, University of Georgia, 2000.

[PS01] S. Patel and A. Sheth. Planning And Optimizing Semantic Information Requests On Heterogeneous Information Sources Using Semantically Modeled Domain And Resource Characteristics, 2001. LSDIS Technical Report, University of Georgia, March 2001

[She96] A. Sheth, "Data Semantics: What, Where, and How?", in Data Semantics (IFIP Transactions), R. Meersman and L. Mark, Eds., Chapman and Hall, London, 1996, pp. 601-610.

[She99] A. Sheth. Changing focus on Interoperability: From System, Syntax, Structure to Semantics. In M. Goodchild, M. Egenhofer, R. Fegeas, and C. Kottman, editors, Interoperating Geographic Information Systems, Kluwer Academic Publishers, 1999.

[SS98] K. Shah and A. Sheth. Logical Information Modeling of Web-accessible Heterogeneous Digital Assets: Proceedings of the Forum on Research and Technology Advances in Digital Libraries (ADL), Santa Barbara, CA, April, 1998.

[Wie93] G. Wiederhold. Mediators in the architecture of future information systems. IEEE Computer, 25(3), pp. 38-49.

[Wie97] G. Wiederhold. Value-added mediation in Large-Scale Information Systems. Database Application Semantics, Chapman and Hall, 1997.

[Whi89] G. T. Whiteford. Earthquakes and Nuclear Testing: Dangerous Patterns and Trends. In proceedings of the 2nd Annual Conference on the United Nations and World Peace, Seattle, Washington, April 1989.