Complex Relationship and Knowledge Discovery Support in the InfoQuilt System

Amit Sheth, Sanjeev Thacker and Shuchi Patel
Large Scale Distributed Information Systems Lab
Computer Science Department, University of Georgia
http://lsdis.cs.uga.edu, amit@cs.uga.edu

Abstract
Support for semantic content is becoming more common in Web-accessible information systems. We see this support emerging with the use of ontologies and machine-readable, annotated documents. The practice of domain modeling coupled with the extraction of domain-specific, contextually relevant metadata also supports the use of semantics.. These advancements enable knowledge discovery approaches that define complex relationships between data that is autonomously collected and managed. The InfoQuilt[1] system supports one such knowledge discovery approach. This paper presents (parts of) the InfoQuilt system with the focus on its use for modeling and utilizing complex semantic inter-domain relationships to enable human-assisted knowledge discovery over Web-accessible heterogeneous data. This includes the specification and execution of Information Scale (IScapes), a semantically rich information request and correlation mechanism.

1. Introduction

1.1 Motivation

A large amount of information is broadly accessible as a result of affordable distributed computing infrastructures, especially the Internet and the World Wide Web. The second generation of the World Wide Web called the *Semantic Web* will go beyond providing content for humans to view, and will enable computer programs or agents to "understand" better the information content available in web sources [LHL01] and other open sources. To take maximum advantage of this increased "awareness" of computers, we need to support meaningful information requests that capture user's information need more precisely than is possible with present-day techniques and enable human-assisted knowledge discovery.

A great deal of research into enabling technologies for the Semantic Web and semantic interoperability in information systems has focused on domain knowledge representation through the use of ontologies. Current state-of-the-art ontological representational schemes represent knowledge as a hierarchical taxonomy of concepts and relationships such as is-a/role-of, instance-of/member-of and part-of. Systems based on these schemes and associated "crisp logic" based reasoning or inference mechanisms [Dec01] support queries of limited complexity [DHM+01]. Additional research in query languages and

[1] One of the incarnations of the InfoQuilt system, as applied to the geographic information as part of the NSF Digital Library II initiative is the ADEPT-UGA system [Ade]. This research was funded in part by National Science Foundation grant IIS-9817432.

query processing is rapidly continuing. Relationships between concepts are the primary building block of any method of supporting semantics [She96,Wie97]. The vast majority of research to date has focused on processing hierarchical relationships that are mathematically well defined (e.g. subsumption to identify placement of a concept in a concept hierarchy, transitivity computation to support parts hierarchy). The focus of the InfoQuilt system is to extend support for semantics by supporting computations involving lateral, user-defined relationships. Furthermore, InfoQuilt aims to support human-assisted knowledge discovery by allowing users to pose questions that involve complex and hypothetical relationships amongst concepts both within and across domains. In doing so, users gain a better understanding of their domains of study and the interactions between them. Relationships across domains (e.g. causal relationships) may not necessarily be hierarchical in nature and may involve complex information requests involving user-defined functions and fuzzy or approximate match of objects. These types of relationships therefore require a richer environment in terms of expressiveness and computation. For example, a user may want to know "Does Nuclear Testing *cause* Earthquakes?" Answering such a question requires correlation of data from sources belonging to the domain Natural-Disasters**.**Earthquake with data from sources belonging to the Nuclear-Weapons**.**Nuclear-Testing domain. Such a correlation is only possible if, among other things, the user's notion of "cause" is clearly understood and exploited. This involves the use of ontologies of the involved domains for shared understanding of the terms and their relationships. Furthermore, the user should be allowed to express their meaning (or definition) of the causal relationship. In our example, the meaning of "cause" could be based on the proximity in time and distance between the two events, nuclear tests and earthquakes. This meaning should be exploited when correlating data from the different sources. Subsequent investigation of the relationship by refining and posing other questions based on the results presented, may lead the user to a better understanding of the nature of the interaction between the two events. This process is what we refer to as Human-Assisted KNowledge Discovery (HAND), or Hypothesis Driven Knowledge Discovery. In this paper, we present the InfoQuilt system and describe its approach to the challenge of enabling such a knowledge discovery on the Semantic Web.

1.2 Background

Domain modeling, especially using ontologies, is a crucial step in enabling the Semantic Web [BHL01, FHLW02], as well as other advanced capabilities such as HAND. Ontologies become the basis of metadata and knowledge sharing as they represent ontological commitment or the agreement among the domain experts and users about the terms or concepts and relationships between them. Relationships between entities (terms or concepts) are the basis of capturing, representing, and supporting semantics. Present day ontology languages, such as those based on RDF/RDFS, F-Logic, Conceptual Graphs, SHOE, Description Logic (including DAML+OIL) provide formal representations of relationships and the ability to have relatively efficient inferencing capabilities [SemWeb].

However, the complex, user-defined, inter-ontological relationships that HAND requires are beyond the expressive and computational capability of languages like the ones listed

above. One example of a deficiency shared by most of these languages is the absence of a mechanism that can model complex operators. Consider the relationship of an Earthquake causing a Tsunami. Temporal and spatial proximity between the two events are two aspects of this relationship [EFDC02]. The use of simple equality to compute them would be incorrect or insufficient because these proximities involve some approximate matching, which is context-specific. The tsunami could occur a few hours after the earthquake. Calculating the temporal proximity in this case requires fuzzy or approximate comparison. InfoQuilt supports such operators by allowing the use of user-defined functions as operators. Such functions can additionally be used for fuzzy or approximate matching of data values from two sources. It is very likely to come across two values (for the same term) that mean the same but are not "equal" syntactically. For example, "Nevada Test Site, Nevada, US" and "Nevada Test Site (NTS), NV, USA" both refer to the same nuclear test site. However, the two values are syntactically not equal. Another use of functions is for complex post-processing of data. Simulation is a particularly interesting example. For example, simulation programs are widely used in the field of Geographical Information Systems to forecast patterns of urban land-cover, land use, deforestation, water consumption, pollution, etc. If these programs are available as user-defined functions, they can be run using data available from multiple autonomous data sources.

Apart from the Internet and the web, a large number of other types of open sources (e.g. databases, premium on-line services, limited addition publications and "gray literature") can also be accessed over a network. One of the most basic requirements for HAND is the availability of large amounts of data. In practice, this is distributed over a large number of heterogeneous and autonomous sources. Hence, the ability to handle wide varieties of data sources (which may contain static or dynamic data) is crucial.

The use of metadata is often makes searching data more efficient. However, in order to discover "knowledge", it is crucial for the system to be able to "understand" the data. The system particularly needs to extract domain-specific or contextually relevant metadata from all the data sources. This type of metadata is extremely useful to integrate information from multiple sources in a more meaningful way using user-defined relationships. Recently, significant commercial success has been achieved in automatically extracting domain specific (or contextually relevant) metadata and using it to provide semantic search over relevant attributes (as opposed to keyword based searches provided by traditional search engines) [SAB01, SBA+02]. Consider the information request *"Find all movies that were directed by Clint Eastwood"*. Because the movie objects have associated metadata for actors as well as directors, semantic search can avoid returning movies in which Clint Eastwood was an actor but not a director. Support for semantics through the use of domain-specific attributes provides early examples of realizing the promise of the Semantic Web.

In addition to modeling the semantics of the domains, knowledge of the characteristics of the domains of interest gives us the ability to optimize the processing of information requests. . The InfoQuilt system accomplishes this by utilizing information about the characteristics and limitations of the available data sources when selecting the sources

that are relevant to a particular information request. Such information includes use of domain characteristics, functional dependencies, characteristics of the data available from each resource, local completeness of the resources, binding patterns on the resources, etc.

This paper discusses a form of knowledge discovery, HAND, supported in the InfoQuilt system. The following are the core capabilities that make HAND possible:

- Domain modeling that uses ontologies to describe terms and concepts relevant to the domain, domain rules and functional dependencies to model domain characteristics, and support of complex user-defined inter-ontological relationships
- Rich and powerful querying mechanism known as *IScape* that utilizes the domain ontologies, inter-ontological relationships, and user defined functions to accurately describe a user's information need
- User defined functions as complex operators, especially for fuzzy/approximate matching, complex post-processing (esp. simulations) and result analysis (e.g., chart creator)
- Ability to handle heterogeneous, static or dynamic content
- Information Extraction: Semi-Automatically or automatically create domain-specific or contextually relevant metadata
- Information request processing utilizing domain and resource characteristics (domain rules, functional dependencies, resource rules, local completeness, binding patterns)

The concepts and algorithms described in the paper are implemented and functional in the current version of the system. The remainder of the paper is organized as follows. Section 2 shows how the knowledgebase of the system is modeled. This includes domain ontologies, inter-ontological relationships, and user-defined functions. Section 3 describes how *IScapes*, complex information requests such as the one discussed earlier, can be specified to the InfoQuilt system. Section 4 describes how InfoQuilt supports Human Assisted Knowledge Discovery (HAND) and decision-making by exploring hypothetical relationships with an extensive example. Section 5 describes the runtime architecture of InfoQuilt. Section 6 discusses how information from the diverse heterogeneous source can be extracted and integrated. It also describes MÉTIS, a toolkit for automating the creation of an integrated metabase using several heterogeneous and diverse sources. Section 7 explains how data sources are modeled in the system. Section 8 describes how an IScape specified using high-level domain models and complex relationships are translated into practical execution plans that specify exactly how the request is to be answered. The processing of an IScape is multi-threaded and parallel and can be monitored as it progresses. Section 9 describes IScape execution and monitoring. Section 10 describes the various tools provided by InfoQuilt to help the administrator to create and manage knowledgebase and to help the users to create and deploy IScapes. We compare our work with other related efforts in section 11. Finally, section 12 presents our conclusions and plans for future enhancements.

2. Knowledge Modeling

The InfoQuilt system maintains information about the domains of interest, complex real-world relationships between them, and various operations that can be performed on the data. This forms the Knowledgebase of the system. In a typical scenario in using the InfoQuilt system, an administrator would first create the knowledgebase of the system. A user can then use this knowledgebase to define IScapes and execute them. This section describes how the knowledgebase of the system is modeled.

2.1 Domain Modeling

InfoQuilt uses ontologies to model the domains of interest. Ontologies capture useful semantics about the domains that they model such as the terms and concepts of interest, their meanings, relationships between them, and the characteristics of the domain. The terms and concepts of the domain are represented as attributes of the classes in the ontology. Classes in ontologies are modeled in a hierarchy that allows modeling simple "is-a" relationships. Additionally, the administrator can also model useful characteristics of the domain as domain rules and functional dependencies. Ontologies provide a structured, homogeneous view over all the available data sources. It is used to standardize the meaning, description, and the representation of the attributes across the sources. When all the resources are mapped to this integrated view, resolving source differences and schema integration become easier.

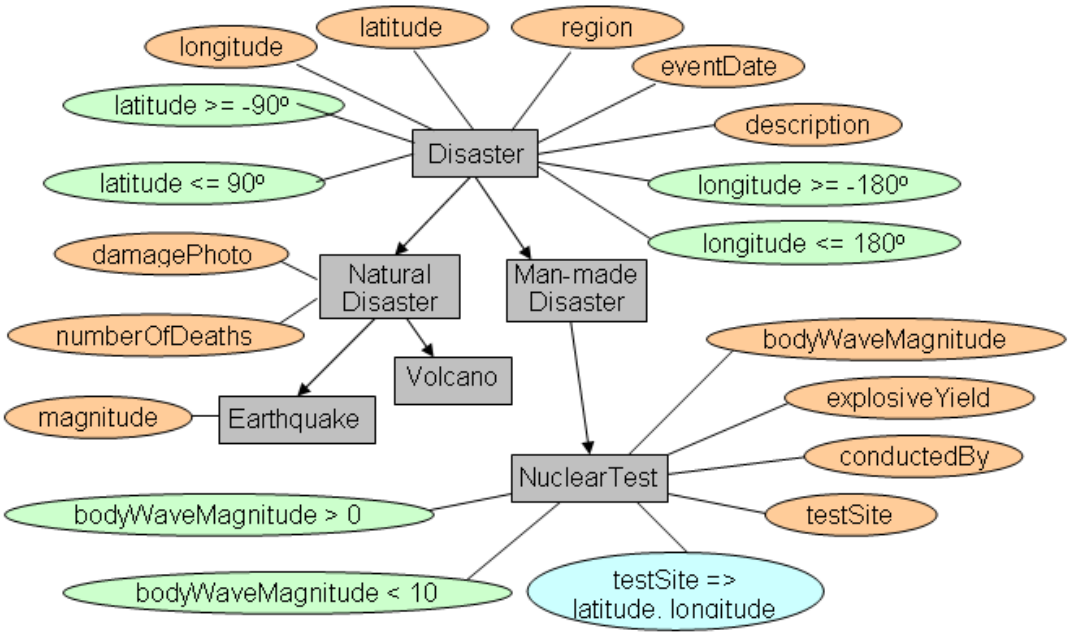


Figure 1: Disaster Ontology (partial)

An example ontology is shown in **Figure 1**. A class inherits all the attributes, domain rules and functional dependencies defined on its parent class. 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). 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.

2.1.1 Attributes

The terms and concepts of the domain are represented as attributes of the classes of the ontology. The meaning of each attribute is standardized so that it has a precise interpretation and use. For example, 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. We will also use the following notation to represent a class in an ontology.

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

Here, the attributes latitude, longitude, region, eventDate, and description are inherited from the class `Disaster` and the attributes damagePhoto and numberOfDeaths are inherited from `NaturalDisaster`.

2.1.2 Domain Rules

Domain rules and functional dependencies describe the characteristics and semantics of a domain. Domain rules describe a condition that always holds true for the domain. These can be used for query validation and optimization of query plans. For example, a simple fact that the latitude of a location on earth should lie between -90^o and 90^o can be described by the following rules on the ontology `Disaster`:

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

Similarly, we can also model the fact that the longitude of a location should be in the range -180^o to $+180^o$ using two domain rules. The domain rules defined for a class are inherited by the child classes. The ontology `Earthquake` therefore inherits these domain rules `Disaster`. We represent it using the following notation:

```
Earthquake (latitude, longitude, region, eventDate, description,
             damagePhoto, numberOfDeaths, magnitude,
             latitude >= -90, latitude <= 90,
             longitude >= -180, longitude <= 180 );
```

2.1.3 Functional Dependencies (FD)

A functional dependency (FD), like in databases, 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). It is very likely to come across resources that do not provide values for certain attributes. The information about the FDs of a domain is used to retrieve information (attribute values) 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, as described in section 8. For example, a functional

dependency on the class `NuclearTest` is that the `testSite` of a nuclear test implies the `latitude` and `longitude` of the place. This can be represented as:

```
testSite -> latitude longitude
```

The ontology `NuclearTest` can therefore be represented as follows:

```
NuclearTest ( latitude, longitude, eventDate, description,
              testSite, conductedBy, explosiveYield, bodyWaveMagnitude,
              latitude >= -90, latitude <= 90,
              longitude >= -180, longitude <= 180,
              bodyWaveMagnitude > 0, bodyWaveMagnitude < 10,
              testSite -> latitude longitude );
```

2.2 Inter-Ontological Relationships

Entities in the real world are related to each other in various ways [SK93, KS96, KS98]. These relationships can be very simple like "is-a", "is-part-of", "is-similar-to", "is-associated-with" which are hierarchical or similarity based and help to relate the entities in a very basic manner. For example, a nuclear test "is-a" man-made disaster. Most Information Integration systems do not support any types of inter-ontological relationships, although there is significant research interest in addressing mismatch between ontologies [SM01, K01]. Support for such simple relationships allows us to relate information in a simple way. OBSERVER is an example of a system that supports inter-ontological relationships to deal with vocabulary heterogeneity between multiple ontologies, using synonyms, hyponyms, and hypernyms [MIKS00]. However, several real-world relationships between entities are much more complex and it is not possible to express them using simple relational and logical operators. For example, consider the relationships "earthquake causes tsunami", "air-pollution affects vegetation", and "nuclear test causes earthquake". These are some examples of the complex relationships that typically span across multiple classes or ontologies. To express such relationships one needs to be able to model the semantics involved in the interaction between the domains. A novel feature of InfoQuilt is that it provides an infrastructure to model such complex relationships and use them to specify IScapes. This forms the basis of Human Assisted Knowledge Discovery (HAND) as discussed further in section 4.

Consider the relationship between a nuclear test and an earthquake. 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*. The constraints "some time after" and "in nearby region" need specialized operators. Use of such operators is necessary to model relationships because evaluations of relationships need evaluations of constraints and computation of values. Several such evaluations and computations cannot be expressed as expressions using only relational and logical operators. In our example, the sub-constraint "in nearby region" is spatially approximate. Both Earthquake and NuclearTest ontologies represent their exact location in terms of latitude and longitude. Using the = operator to check if the two occurred in the same region does not make sense in the context of these domains and relationship because the two are geographical phenomena that span large areas. The region matching thus has a spatial scope. InfoQuilt supports user defined functions (see section 2.3), which can be

used as specialized operators to model relationships. 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 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 paradigm that the system supports, as described further in section 4.

2.3 Operations

A distinguishing feature of InfoQuilt is its framework to support user-defined operations. As seen in section 2.2, we used the user-defined 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 additional constraints in their IScapes. For example, consider the IScape:

"*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. The distance function can again be used here.

These user-defined functions are also helpful for supporting a context-specific fuzzy or approximate matching of attribute values. For example, assume that we have two data sources for the domain of earthquakes. It is quite possible that two values of an attribute testSite retrieved from the two sources may be syntactically unequal but refer to the same location. Considering the example given earlier, the value available from one source could be "Nevada Test Site, Nevada, USA" and that from another source could be "Nevada Site, NV, USA". The two are semantically equal but syntactically unequal [KS96]. Fuzzy or approximate matching functions can be useful in comparing the two values.

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. 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. For example, Clarke's Urban Growth Model [Cla] is a model to forecast the

urban growth in a region in several ways. For example it can use information about the areas in the region where it is known that growth cannot occur for some reason (roads, slopes, vegetation, etc.). 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 provides these maps. Another way to forecast urban growth in a region using the Clarke UGM model is to use a set of maps of the urban area patterns from past years. Figure 2 shows how the urban growth in Washington D.C. area is forecasted for the year 2025 using this method.

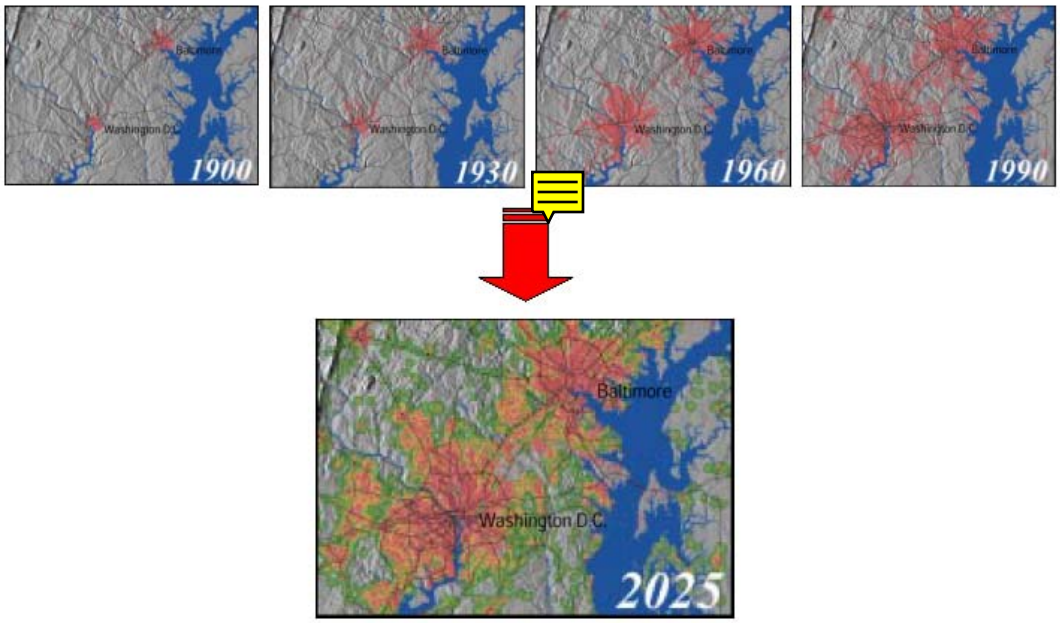


Figure 2: Clarke Urban Growth Model (UGM)

To be able to dynamically and easily add new operations as well as update and delete existing ones, InfoQuilt maintains what is known as a *Function Store*. The Function Store is a component of the knowledgebase of the system. It contains information about all the functions known to the system. We will use the terms *function* and *operation* interchangeably in the rest of the paper. 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. Additional tools discussed in Sections 6.2 and 10 make it easier to incorporate such functions as part of complex relations that can be defined by IScapes, including mapping for input and output data types.

3. Information Scapes (IScapes)

We have used the phrase "information request" or IScapes when referring to the type of question InfoQuilt aims to answer over the term "query" for several reasons. 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 IScape, on the other hand, can understand what the user is inquiring about by embedding semantic

information within the request[2]. It is also able to identify the exact usefulness of the sources available to it for a given request. InfoQuilt is unique in its ability to model such IScapes. This work has been built upon the work done in [Pal00], and Metadata REFerence link (MREF), discussed in[SS98, SK96]. An IScape is specified using concepts belonging to one or more ontologies, and may involve complex relationships including user-defined operations over concepts chosen from one or more ontologies. It is the responsibility of the InfoQuilt system to generate queries over resources, in a way that is somewhat similar to generation of queries over component databases in a federated database, or wrapped resources in a mediator system. Two main differences are the need to support ontology to resource mapping, as well the need to support the computation involving complex relationships, which require mappings involving metadata and ontologies, as well as user-defined functions. This is explained next.

Example:
Consider the following IScape.

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

An IScape allows users to query and analyze the data available from diverse autonomous sources. It allows specification semantic correlation between data from multiple sources, and supports discovery by allowing human-guided discovery of relationships between data by changing parameters defining a complex relationship. Thus we define IScape as "*a computing paradigm that allows users to query and analyze the data available from a diverse autonomous sources, gain better understanding of the domains and their interactions as well as discover and study relationships*."

Any system that needs to answer such IScapes would require a comprehensive knowledge of the terms involved and how they are related. An IScape is specified in terms of the components of the knowledgebase of the system such as ontologies, inter-ontological relationships and operations. This helps the system in understanding the semantics of the request. By stating an IScape in this fashion, it prevents the user from having to know the actual sources that will be used by the system and how the data retrieved from these sources will be integrated. This integration includes how the results should be grouped, any aggregations that need to be computed, constraints that need to be applied to the grouped data, and projection upon the resulting set of information. In particular, an IScape specifies the following:

- the ontologies involved,

[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].

- inter-ontological relationships,
- preset constraints,
- runtime configurable constraints,
- 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
- information that needs to be returned in the result to the user.

Let's briefly reviewed these components.

The ontologies in the IScape identify the domains that are involved in the IScape and the inter-ontological relationships specify the semantic interaction between the ontologies. The preset constraint and the runtime configurable constraint are filters used to describe the subset of data that the user is interested in, similar to the WHERE clause in an SQL query. For example, a user may be interested in earthquakes that occurred in only a certain region and had a magnitude greater than 5. As its name implies, a runtime constrain can be set at the time of executing the IScape, whereas present constraint are defined when the IScape is first constructed. The results of the IScape can be grouped based on attributes and/or values computed by functions. If the results are to be grouped, the user can also specify any aggregations 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 attribute values, computed aggregates, values of functions evaluated on the data, and results of simulation programs.

XML provides syntactic basis for an IScape (see [Lak00] for further details on IScape syntax) . A person uses a graphical toolkit known as the *IScape Builder* to construct and execute IScapes and analyze their results. That it, the IScape Builder is a graphical tool that supports ease of development and deployment of IScapes using the knowledgebase without having to understand the underlying formats used by the system. We describe the IScape Builder in detail in section 10.2, and provide XML representation of an IScape that is generated by this tool in Appendix A.

4. Human Assisted Knowledge Discovery (HAND)

Existing relationships in the knowledgebase provide a scope for discovering new aspects of relationships through transitive learning. For example, 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. The current system does not automatically apply transitive learning, the ISCAPE designer can explicitly specify relationships to effect traversal of transitive relationships.

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 (e.g., the example of the Clarke UGM. The data available from various sources can be queried by constructing IScapes and the results 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 validate any hypothetical relationships between domains and to see if the data validates or invalidates the hypothesis. For example, 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.

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. However, note that the user need not be aware of these data sources since the system abstracts him from them. To construct IScapes, the user works only with the components in the knowledgebase. 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:
"W*hen 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 per year with a magnitude 5.8 or higher on the Richter scale 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[3] was 127 per year indicating that it has 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 starting from year 1900, find the number of earthquakes.*"

The results show that the 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. 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.
- The 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.

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

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. InfoQuilt Runtime Architecture

InfoQuilt uses a multi-agent information brokering architecture at runtime. The other type of architecture used by some information integration systems is mediator-based [Wie92]. [KS00] discuss the information brokering architecture at great length and compare it with mediator-based architecture. Information brokering architecture uses a set of (possibly distributed) software agents that specialize in their specific tasks and a special Broker Agent that acts as a coordinator between them. The information brokering architecture itself used in InfoQuilt is not new (e.g. see [BBB97]). However, to be able to perform semantic information brokering, agents need to be more "intelligent" especially in the areas of sharing, exchanging and interoperating across different knowledge collections [SKL99]. The capabilities of the agents in InfoQuilt to support multiple domains, complex inter-domain relationships, operations and IScapes differentiate the system from previous agent-based systems. The current architecture is built upon work done in [Sin00, Ber98, Par98]. Figure 3 shows the runtime architecture of InfoQuilt system.

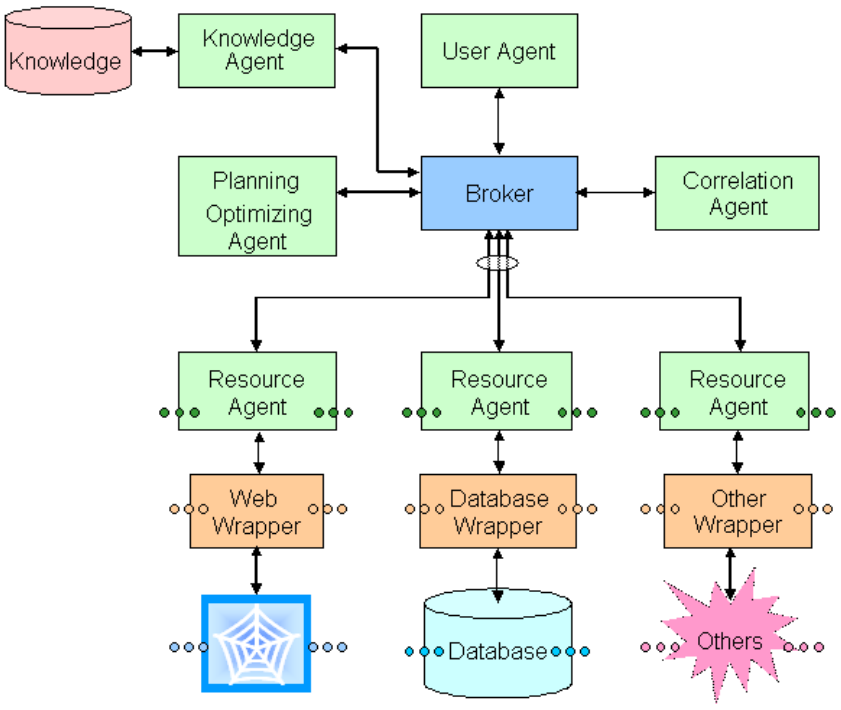


Figure 3: InfoQuilt Runtime Architecture

InfoQuilt creates a knowledgebase of information about the domains of interest to the user, inter-domain relationships, user-defined operations and available information

sources. The database "knowledge" in the figure represents this knowledgebase. The Knowledge Agent acts as an interface for any other agent in the system that needs to access it. As seen from the figure, InfoQuilt supports a variety of information sources such as databases, web-based sources, etc. There is one Resource Agent per resource in the system. The Resource Agent provides a standard interface to the rest of the system to query the corresponding resource. It may in turn use a wrapper and/or extractor that accesses the resource, retrieves data, and maps it to the domain model in the knowledgebase. We discuss this in more detail in section 6. The User Agent acts as the programming interface to the system to execute IScapes. The Broker Agent is responsible for brokering requests from various agents in the system and co-ordination of IScape processing. The Planning Agent is responsible for the creation of high-quality practical execution plans to execute the IScapes. The Correlation Agent executes the execution plan to retrieve data from the sources, correlate them and do any post-processing if required by the IScape.

The processing of an IScape follows the following steps:

1. The User Agent sends the IScape to the Broker Agent for processing.
2. The Broker Agent sends the IScape to the Planning Agent.
3. The Planning Agent creates an execution plan for the IScape after enquiring about the specifications of various domains, relationships, etc from the Knowledge Agent through the Broker Agent. (All interactions between the agents occur through the Broker). If the planning agent deduces that the IScape is semantically incorrect (see section 8), it informs the Broker Agent about it and aborts plan generation.
4. The Planning Agent returns the plan to the Broker Agent. If the plan generation was aborted, the Broker Agent responds back to the User Agent directly (skipping steps 5-7).
5. The Broker Agent then sends the plan to the Correlation Agent for processing.
6. The Correlation Agent starts executing the plan. The execution is completely multi-threaded. This allows the system to exploit the parallelism in the plan. We describe the execution in more detail in section 9.1.
7. The Correlation Agent returns the final result to the Broker Agent.
8. Finally, the Broker Agent forwards the result to the User Agent.

The InfoQuilt architecture supports easy dynamic addition and removal of resources from the system declaratively with an intuitive graphical toolkit. The Planning Agent automatically considers all newly added sources, disregards the eliminated sources and considers any modifications to the specifications of existing sources automatically when creating execution plans for IScapes. Similarly, domains, relationships, etc. can be dynamically added, removed and modified. The architecture is therefore dynamic.

6. Information Extraction and Integration

Our discussions till now assumed that data was somehow made available to the system from the available sources. This section describes how it is extracted from different sources. There are two ways to retrieve and maintain the extracted information – extract it as needed (i.e. when processing an IScape) and extract it offline and maintain it in a

local database. The first option is needed when the content available from the source is dynamic, which is it is updated frequently. However, if the content available is static, the second option is obviously preferable. Additionally, for static content, we can also integrate information from different (static) sources available for a given domain offline. Section 6.2 describes how this can be done semi-automatically.

6.1 Information Extraction

Information Extraction (IE) can be defined as the gathering of relevant data from a collection of documents [Mus99]. We use IE techniques to extract metadata from web sources relevant to the domains of interest. Two important metrics used to assess the performance of an IE system are *recall* and *precision*. Recall refers to how much of the information that should have been extracted was correctly extracted whereas precision refers to the reliability of the information extracted [Leh]. These measures are independent of each other. Statistics show that experienced analysts exhibit better precision (82%) and recall (79%) in manual information extraction over automated systems (57% precision and 53% recall). However, automating the extraction is still preferred for large quantities of information as various human factors come into play [Leh].

IE techniques can be used to extract information from both free text as well as semi-structured data. AutoSlog [Ril93], Crystal [SFAL95], and Hasten [Kru95] are examples of IE systems used to extract information from free text. We use IE techniques to extract information from web sources that are semi-structured. IE techniques for free text are out of scope here. [Gun00] gives an overview of types of IE techniques used for information extraction from semi-structured data [CM97, Sod99, HGC+97, SA99].

We use an extraction toolkit from Taalee/Voquette [SAB01, SBA+02] to extract information from individual web sources by semi-automatically creating wrappers for each source. Another example of a toolkit is XRAP [LPH00]. However, the information extracted can be structured, named, represented and stored differently at different sources. Therefore, these heterogeneity must be resolved and the information must be converted into a canonical form before it can be integrated.

6.2 Integrated Metabase Creation and MÉTIS

We refer to data about a real-word entity available from a source as an object. Since potentially several resources may provide data for a domain, it is common that several different objects retrieved from different sources actually represent the same real-world entity. Information integration systems usually provide a uniform means of representing the information from multiple sources, but do not integrate such objects into a single object. Such integrated objects have the advantage that they can be stored in a metabase. The system may then directly query this metabase instead of the sources from which these objects were integrated. Additionally, the integration of the objects allows scope for enhancing the information using multiple objects available for an entity. InfoQuilt provides MÉTIS (Metabase Creation Toolkit Integrating Multiple Heterogeneous *S*ources), a toolkit for automating the creation of such a metabase from multiple heterogeneous sources [Gun00]. A point to note is that object integration cannot be done

in real-time (while processing an IScape) since it is slow. It is therefore done offline. Also, it works best for sources with relatively static data. For sources that frequently update their data, updating the metabase frequently can be very expensive. A commercial system based on the research in the LSDIS lab that deals with dynamic or frequently updated source (e.g., CNN.com) is reported in [SBA+02]. However, this capability is not part of the current InfoQuilt implementation.

The MÉTIS toolkit [Gun00] is used to create a single repository of information for a domain by integrating information retrieved from multiple heterogeneous sources. These sources can be web sources, databases, etc. The information coming from each source is first converted into a canonical representation. Next, if an object representing the same entity is found in the metabase, the new object is merged with it. Otherwise, it is inserted into the metabase.

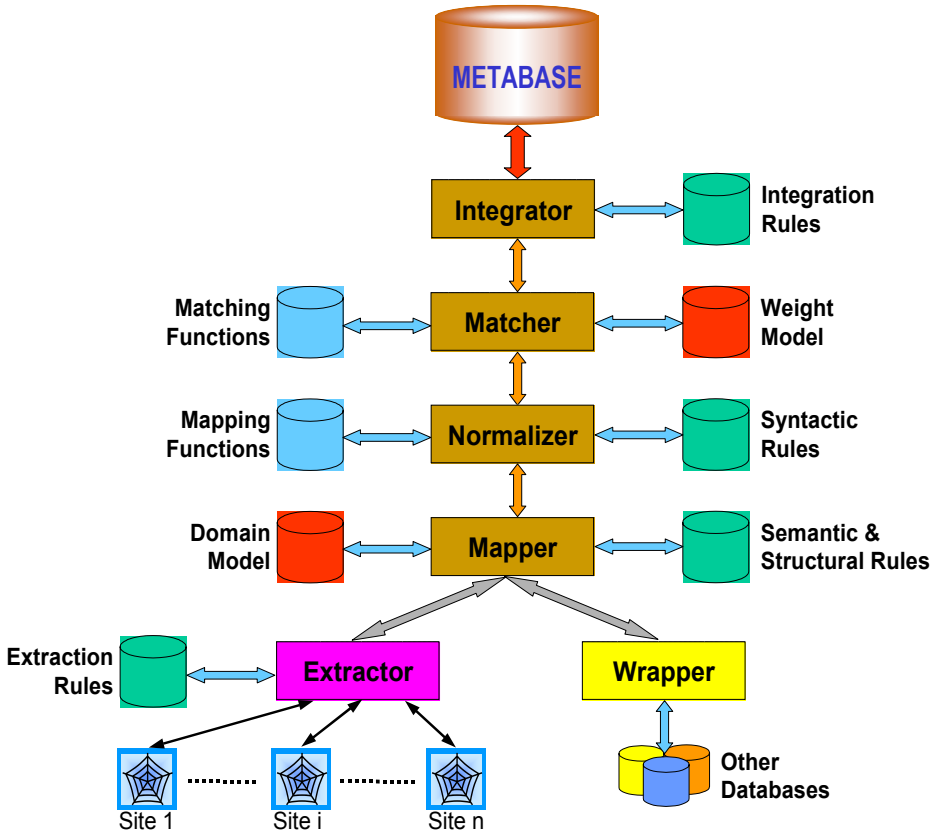


Figure 4: MÉTIS Architecture

Figure 4 shows the MÉTIS architecture. Data from web resources can be extracted using the most appropriate extraction tool, while a wrapper is written for each database to convert the data represented in the internal schema of the database into attribute-value pairs. These attributes may differ from the system's domain model in terms of structure as well as naming since the individual resources have their individual representations and models of the domains. The *Mapper* then uses a set of structural and semantic mapping rules to map the information in these objects to the system's domain model. Next, the *Normalizer* uses a set of syntactic rules to resolve syntactic heterogeneity between the

objects retrieved from different sources. For example, one source may represent a street address as "214 N. Peachtree Ave." and another may represent the same as value as "214 North Peachtree Avenue". The Normalizer resolves this type of differences. This may involve use of several mapping functions because resolving such differences is a very context and domain-specific task. The objects can now be compared. The *Matcher* uses a weight model to compare objects. Finally, if an object does not have a matching object in the metabase already, it is directly inserted into the metabase. If, however, a matching object is found, a set of integration rules is used by the *Integrator* to integrate the two objects and the metabase is updated.

7. Resource Modeling

In addition to creating a Resource Agent for each resource to be added to the system, a declarative description (model) of the resource is also created. Information sources in InfoQuilt are described in terms of the corresponding ontologies. This approach has the advantage that adding and removing sources to the system dynamically is very easy since it does not affect the ontologies or other resources. The resource models used in InfoQuilt capture the characteristics of the data provided by the resources and its query capabilities. The resource model consists of a list of attributes, binding patterns (BP), data characteristic (DC) rules, and local completeness (LC) rules.

7.1 Resource Attributes

Resource attributes are attributes for which the resource can provide values. Since the way the resource models its data can be different from 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] and is done by the Resource Agents. 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. The resource's description therefore lists the attributes for which it can provide values . For example, consider the resources `SignificantEarthquakesDB` and `EarthquakesAfter1990` for the class `Earthquake`.

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

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

We will use the notation above to represent resources.

7.2 Binding Patterns (BP)

Some web sources have a limited query capability. This is supported by allowing users to search based on some attribute(s) or combinations of attribute(s). These web sources require that values for some attributes be provided to retrieve any information from the resource. The source can be queried only by specifying values for those 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 processing an IScape. These are represented in the resource models 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(s).

For example, 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]
```

7.3 Data Characteristic (DC) Rules

Data characteristic (DC) rules are similar to domain rules. However, they apply only to the specific resource. Consider the Air Tran Airways web site as a resource for the `Flight` ontology. Flight models a direct flight from one city to another within US. We know that the AirTran Airways web site provides information about only the flights operated by AirTran Airways. This characteristic of the resource can be represented by a DC rule:

```
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 7.4.

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 a particular 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. 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 IScape.

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 "flights operated by AirTran Airways" since all flights available from `AirTranAirways` site are known to be operated by AirTran Airways.

7.4 Local Completeness (LC) Rules

A local completeness (LC) rule on a resource has the same format as a DC 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 IScape 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 domain) 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 of information on a particular domain. 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. For example, consider the AirTranAirways resource used earlier. 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`.

8. Planning and Optimization

As stated previously, an IScape is specified in terms of the components of the knowledgebase of the system such as ontologies, relationships, operations, etc. The system needs to then translate it into an execution plan that specifies exactly which resources should be used to answer the IScape, how are they to be queried, and how is the information retrieved from these resources to be integrated. As mentioned earlier, the Planning Agent is responsible for this. This section describes the process of creation of these execution plans, given an IScape. A detailed discussion appears in [PS01]. Consider the following example IScape.

Example:

We use the classes `NuclearTest` and `Earthquake` for this IScape. An inter-ontological relationship "`NuclearTest Causes Earthquake`" exists between them. The following are their specifications in the knowledgebase.

```
NuclearTest ( latitude, longitude, eventDate, description,
              testSite, conductedBy, explosiveYield, bodyWaveMagnitude,
              latitude >= -90, latitude <= 90,
              longitude >= -180, longitude <= 180,
              bodyWaveMagnitude > 0, bodyWaveMagnitude < 10,
              testSite -> latitude longitude );

Earthquake ( latitude, longitude, region, eventDate, description,
             damagePhoto, numberOfDeaths, magnitude,
             latitude >= -90, latitude <= 90,
             longitude >= -180, longitude <= 180 );

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

Following are the specifications of the information sources available to the system for these two ontologies.

```
NuclearTestsDB( testSite, explosiveYield, bodyWaveMagnitude,
                testType, eventDate, conductedBy,
                [dc]bodyWaveMagnitude > 3,
                [dc] eventDate > "January 1, 1985" );

NuclearTestSites( testSite, latitude, longitude );

SignificantEarthquakesDB( eventDate, description, region,
                          magnitude, latitude, longitude,
                          numberOfDeaths, damagePhoto,
                          [dc] eventDate > "January 1, 1970" );
```

NuclearTestsDB is a database of nuclear tests with a seismic body wave magnitude > 3 on the Richter scale and conducted after January 1, 1985. This is a local metabase created by extracting information about nuclear tests from several web sources such as a catalog compiled by Oklahoma Geological Survey Observatory [Okl], Trinity Atomic Web Site [Tri], and some others. The resource NuclearTestSites contains exact locations of nuclear test sites as their latitudes and longitudes. SignificantEarthquakesDB is also a local database created by extracting information from several web sources such as the USGS web site [USGS]. It has information only on earthquakes that occurred after January 1, 1970. The IScape to be processed is the following.

"*Find all nuclear tests conducted by India or Pakistan after January 1, 1995 with a seismic body wave magnitude > 4.5 and find all earthquakes that could have been caused due to these tests.*"

The Planning Agent uses the following rules to select the resources that are relevant to answer the IScape for each class in the IScape.

- **Locally Complete Sources**

First it makes use of the local completeness rules of the resources. If there exists a resource that is locally complete for some subset A of the domain of the ontology such that the part of the IScape's result that belongs to that domain is a subset of A, using only this resource for the domain is sufficient. Other resources need not be used (unless the selected resource has some attributes missing or has a binding pattern) since the local completeness condition implies that using any additional sources will not provide any extra information. A semantic optimization that is performed while using this rule is that the DC rules on the locally complete resource should not falsify the IScape's constraint. If it they do, then it implies that the IScape is semantically incorrect. No resources in this example are locally complete. So, we cannot apply this rule.

- **Non Locally Complete Sources**
  If a locally complete source could not be found, then the planner cannot be sure that all possible answers to the IScape can be found using the available sources. However, we would want to retrieve as much information as possible from them. It therefore considers *all* the resources that it can use. Again, it can make use of DC rules to prune sources that will not return any useful results.

  For the ontology `NuclearTest`, there are two resources. It selects both and applies the condition to them. No resources are eliminated as their DC rules do not falsify the IScape's constraint. Similarly, `SignificantEarthquakesDB`, the only available resource for Earthquake ontology, is selected. The DC rule on it does not falsify the IScape's constraint. However, consider a slight modification to the example. If the resource NuclearTestsDB provides data for tests conducted *before* January 1, 1985, then its DC rule would falsify the IScape's constraint. This is because the IScape is querying for tests *after* 1995. In this situation, the planner eliminates NuclearTestsDB from the list of sources that are to be used to answer the IScape since it will not provide any data useful for the IScape.

- **Binding Patterns**
  For the resource(s) selected, the planner needs to ensure that their query capability limitations (binding patterns) are respected. If a resource has binding pattern(s) associated with it, the plan needs to specify how the values for the binding pattern attributes will be supplied. There are three possible ways to do this.
    - The values can be supplied from the IScape constraint directly.
    - The values can also be supplied from attributes in other classes.
    - If either of the above two cases is not possible, then we can use another resource of the ontology as an associate resource with this one, to supply values for its binding pattern attributes. We will refer to the resource with the binding pattern as the primary resource.

  First, the planner tries to use the first method since it is the simplest and the fastest. If, however, the IScape constraint is unable to provide specific values for all attributes, it can make use of a sub-condition such as "`A.a = B.b`" in the constraint where `A` and `B` are classes and `a` and `b` are their attributes. If a resource

for ontology A is bound on the attribute a, the values retrieved for attribute b can be provided as binding pattern values to query it. If these two techniques cannot be applied, the planner can make use of an associate resource. The associate resource is another resource of the same ontology. Figure 5 shows how an associate resource can be used to retrieve values for binding pattern attributes. Suppose A, B, C, D, E and F are attributes of a class. The primary resource has a binding pattern on attributes A and B. A Binding Pattern Supplier (BP Supplier) node is used in the plan to retrieve arbitrary values for these attributes from the associate resource. These values are then used to query the primary resource. This technique allows us to use the primary resource to answer the IScape, which would not have been possible otherwise.

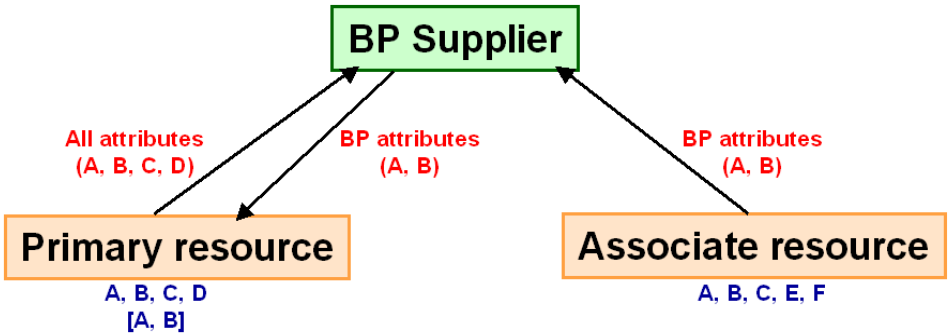


Figure 5: Use of associated resource to supply binding pattern values

For our example, none of the resources selected have binding patterns. Hence, this rule does not apply.

- **Associate resource to supply values for missing attributes**
  The resources selected should be able to provide all the attributes that the IScape needs. However, it is very common to come across resources that do not. If a resource has one or more missing attributes, the planner can use the functional dependencies defined for the ontology, involving all the missing attributes to see if it can couple it with some associate resource to retrieve values for those attributes for as many entities as possible. This is done by equating the values of attributes appearing on the LHS of the FD to join the data retrieved from the main resource with that from the associated resource. Without the knowledge about the FD, it would not be possible to form a plan that can thus deduce more information about entities using multiple resources in conjunction.

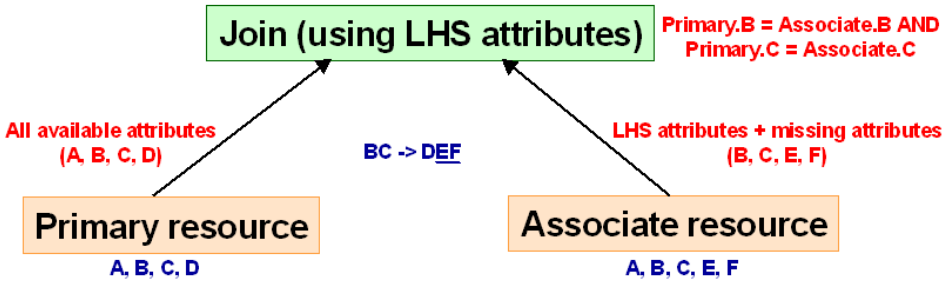


Figure 6: Use of Functional Dependency to retrieve values for missing attributes

Figure 6 illustrates this. The primary resource has two missing attributes – E and F. We use the functional dependency BC -> DEF defined on the class that the primary resource provides information for.

The attributes are equated using a default equality computing function defined for it in the function store or an exact match if no function was defined. The use of functions is necessary as it is highly unlikely that all sources will have exact same values for the attributes (syntactic heterogeneity) [Gun00]. For example, a nuclear test site available from one source could be "Nevada Test Site, Nevada, USA" and that from another source could be "Nevada Test Site, NV, USA". The two are semantically equal but syntactically unequal [SK93]. In Figure 6, it is assumed that no default equality function is available for attributes B and C.
If a FD or an associate resource could not be found, then the resource with the missing attributes cannot be used.

The attributes that the IScape uses (including the attributes that we project on and those needed to evaluate relationships and constraints) are:

```
( NuclearTest.testSite, NuclearTest.explosiveYield,
  NuclearTest.bodyWaveMagnitude, NuclearTest.testType,
  NuclearTest.eventDate, NuclearTest.conductedBy,
  NuclearTest.latitude, NuclearTest.longitude,
  Earthquake.eventDate, Earthquake.description,
  Earthquake.region, Earthquake.magnitude,
  Earthquake.latitude, Earthquake.longitude,
  Earthquake.numberOfDeaths, Earthquake.damagePhoto )
```

`NuclearTestsDB` has two missing attributes – `latitude` and `longitude`. The planner uses the FD "`testSite -> latitude longitude`" to retrieve their values using `NuclearTestSites` as an associate resource. The function `testSiteEquals` from the function store is used to equate the values of `testSite` from the two resources. Similarly, for `NuclearTestSites`, all attributes except `testSite`, `latitude`, and `longitude` are missing. However, there is no FD that can be used to retrieve values for these missing attributes using an associate resource. So, we eliminate it as a primary source.

Notice that the execution plan may need to use a source more than once. For example, a source could be used as a primary resource as well as an associate resource. However, it need not be accessed twice if it does not need any binding patterns. The Planning Agent identifies such resources and optimizes the plan by creating a single Resource Access node that retrieves all the information (attributes) needed. Every node in the plan that needs information from this resource then points to this Resource Access node.

Figure 7 shows the execution plan created for the IScape. The sub-conditions that the nuclear test should be of a magnitude > 4.5 and that the test should have been conducted by either India or Pakistan are moved down to the resource as shown. This is a standard optimization technique used in all databases as well

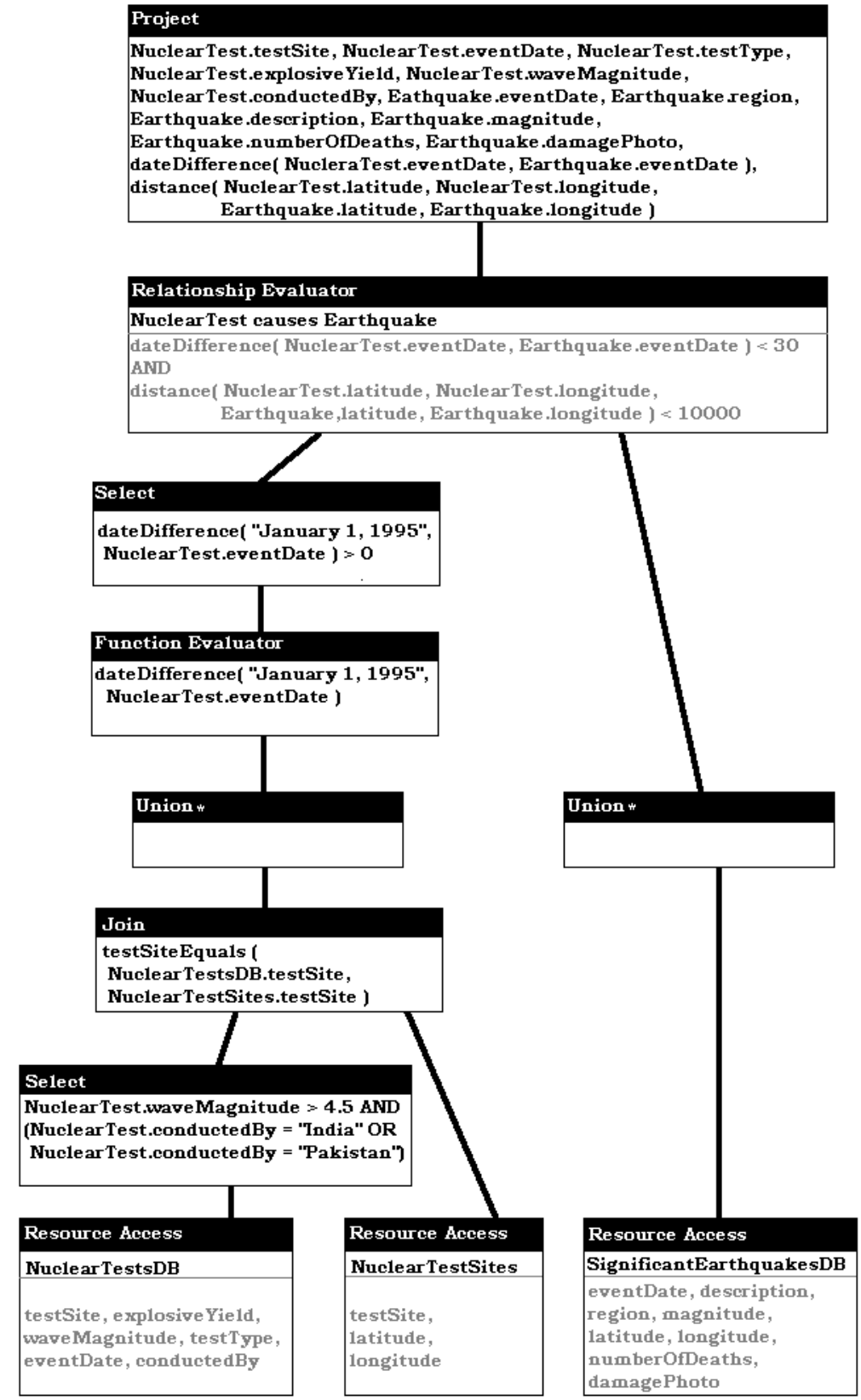


Figure 7: Execution Plan for IScape

Next, the planner determines the steps needed to integrate the data retrieved from these resources. As seen in **Figure 7**, for this example, it creates an intermediate union for each ontology. This intermediate union is to integrate the data from all resources used in the plan such that they do not need any binding pattern values supplied from attributes of other ontologies. In this example, all resources fall in that category. This union node is eliminated only if no such resource exists in the plan and is needed to avoid deadlocks that may potentially occur when resources retrieve values for their BP attributes from other classes. For details, see [PS01].

Next, the remaining sub-conditions involving attributes of a single ontology are evaluated. Here, the check if nuclear test was conducted after 1995 is such a sub-condition. Next, the data retrieved from the sources for all ontologies is related with each other using the specifications of the inter-ontological relationships. This may involve evaluation of functions. Next, all the remaining sub-conditions in the IScape's constraint should be evaluated. These sub-conditions are those that involve attributes of more than one ontology. They can now be evaluated since the information sets for all ontologies are now integrated. There are no such conditions remaining in this example. Next, all groupings and aggregations are computed. Again, this example does not need any. Finally, the result set should be projected on the attributes, function values, and aggregations in the projection list of the IScape.

9. IScape Execution and Monitoring

This section discusses the processing of IScapes and how it can be monitored.

9.1 IScape Processing

The User Agent is the entry point for the IScape in the system. The User Agent passes the IScape to the Broker Agent. The Broker Agent starts the processing of the IScape by coordinating the other agents. It first asks the Planning Agent to create an execution plan. The Planning Agent interacts with the Knowledge Agent (through the Broker Agent) to access information about domains, relationships, operations involved in the IScape and the resources available in the system for those ontologies. It creates an execution plan as described in section 8 and sends it back to the Broker. The Broker then sends the plan to the Correlation Agent, which is responsible for executing it. The final results compiled after executing the IScape are finally returned to the Broker and then the User Agent.

The execution of the plan in the Correlation Agent is multi-threaded and parallel. The Correlation Agent starts by creating a node processor for each node in the execution plan. Each node processor is a thread. The IScape processing is started by starting the node processor for the root node in the plan. Each thread first starts all the node processor threads from which it expects some input and waits for them to finish. Once all the inputs are ready, the node processor starts its own processing. Thus, nodes are evaluated as and when their input data becomes available. The processing is thus done in parallel. For example, the first step is usually to query the resources. All resources that do not need any input from other resources (for binding patterns) are queried simultaneously.

9.2 IScape Processing Monitor (IPM)

The processing of the IScape can be monitored using a graphical toolkit called the IScape Processing Monitor (IPM). Every agent sends detailed logs to the IPM to inform its status. For example, the Planning Agent sends detailed log messages that indicate the results of various stages of planning algorithm (e.g. selection of sources for a class). The log messages include a time stamp, a brief message, a detailed message and possibly some associated data. For example, as soon as the Planning Agent generates the plan, it sends it to the monitor. Similarly, the Correlation Agent sends the intermediate results generated by each node processor to the monitor. The log messages are displayed as color-coded entries in a table. Figure 8 shows a screenshot of an IPM after execution of an IScape. The IPM provides the following benefits:

- It helps in monitoring the execution of the IScape as it proceeds and localize any failures easily.
- The detailed log messages generated by the various agents describe in considerable detail exactly what is going on during processing. IPM is therefore extremely useful as a high-level debugging tool. The availability of temporary results generated at all stages of execution of the plan in the Correlation Agent also helps in this.
- 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.

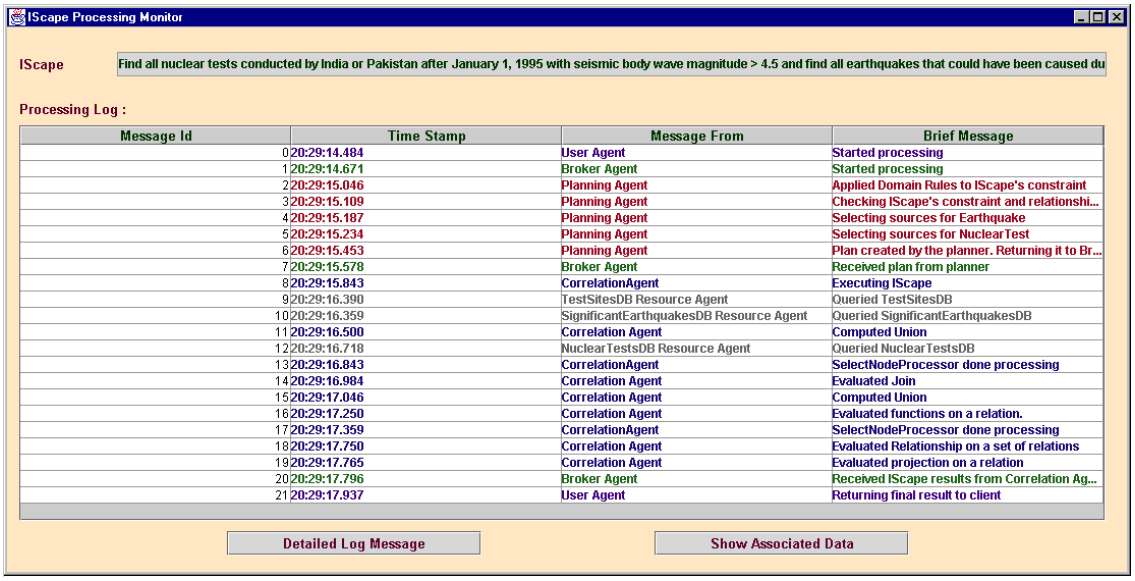


Figure 8: IScape Processing Monitor (IPM)

10. Graphical Tools

We now describe different graphical toolkits available to the user and interfaces to create and execute IScapes. These include the KnowledgeBuilder, which is used to create the knowledgebase of the system, the IScapeBuilder, a stand-alone client used to create and execute IScapes and a web-based interface that can be used to execute pre-defined IScapes. Additionally, InfoQuilt also provides an IScape Processing Monitor (IPM) as described earlier in section 9.2.

10.1 Knowledge Builder (KB)

The KnowledgeBuilder (KB) is an easy-to-use intuitive graphical tool to help an administrator create the specifications of domains (ontologies), inter-domain relationships, operations (functions and simulations) and the available information sources. These form the knowledgebase of the system as discussed in previous sections. Use of the KB has the following advantages:

- The administrator does not need to know the format of the XML specification used by the system internally to represent ontologies, relationships, functions, and resources. The KB abstracts the administrator from the technical details of the formats used internally by the system.
- It provides tools that help the user relate the information in the knowledgebase in a better way. For example, the user can look at the entire knowledgebase (except for function specifications) 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 characteristics 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. It can thus provide a comprehensive graphical view of the entire knowledgebase.

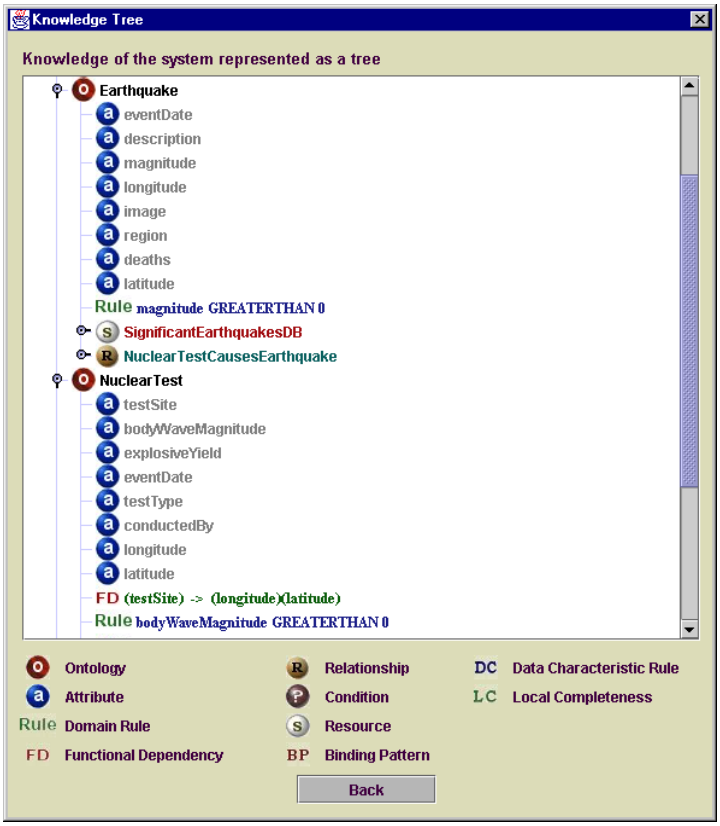


Figure 9: Knowledge Tree

- It helps the administrator in maintaining the knowledgebase and making modifications correctly. For example, suppose an attribute from an ontology needs to be removed. The KB will not allow this if the attribute appears in (a) a rule on the

ontology, (b) a FD on the ontology, (c) set of attributes provided by some resource, (d) a data characteristic rule on a resource, or (e) a local completeness rule on a resource or (f) a binding pattern on a resource. The user is required to individually take care of any such undesirable condition (the KB itself can be used to do this) before removing the attribute. If the knowledgebase 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 (Figure 9), the user can easily locate such uses of the attribute.

10.2 IScape Builder (IB)

The IScape Builder (IB) is a stand-alone Java application that provides a graphical interface to create and execute IScapes. IB provides the following benefits:

- It provides a simple and intuitive interface that allows the user to create and execute IScapes in a step-by-step manner.
- The user does not need to be aware of the XML representation used internally by the system to represent IScapes.
- It is integrated with the knowledgebase of the system. The user therefore does not need to look it up to create new IScapes. For example, the user can easily select from the names of ontologies, relationships, and functions that appear in drop down lists.
- It implements basic validity checks during IScape construction.
- 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. Tools like the IB can also additionally be designed to present data in a more intuitive format specific to particular domains. For example, if the only domains of interest are related to geography, many of them are likely to have a geo-reference indicating exact location on the earth. A more intuitive and appealing way of presenting this information could be via a map of the world with the exact location pointed out using a marker symbol. However, such tools are highly contextual and are not suited as a general technique.

Figure 10 shows the first in a series of five steps involved in the IScape creation and execution process. This step enables the user to select the ontologies that he/she wants the IScape to use. In addition this step also allows for the selection of the relationships that the user wants to exploit.

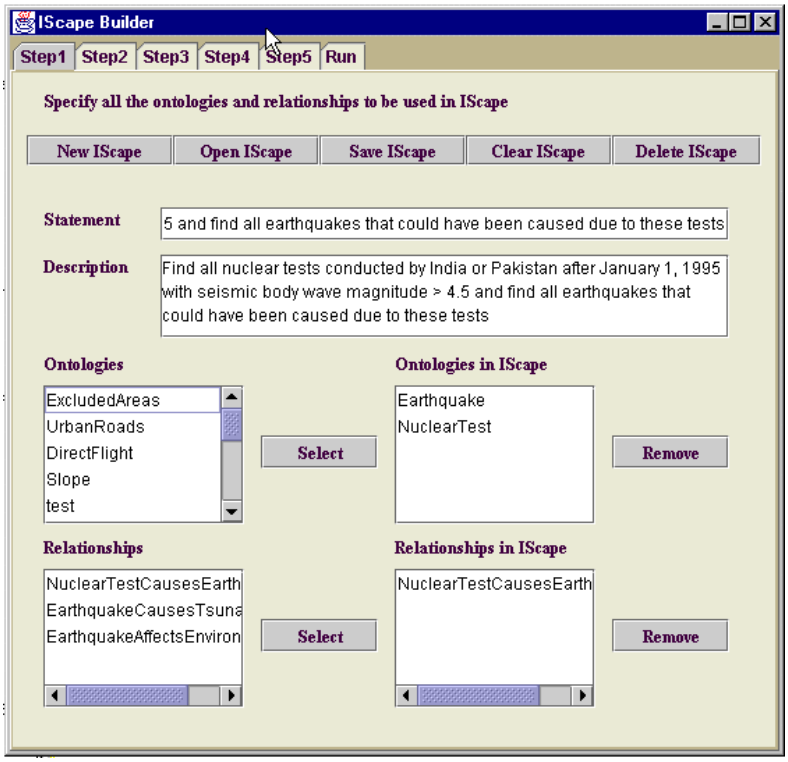


Figure 10: IScape Builder Step 1

The second step (Figure 11) enables the user to specify functions that operate on the attributes of the Ontologies used in the IScape. This step can also be used to specify aggregate operators on attributes needed to support the query using the IScape.

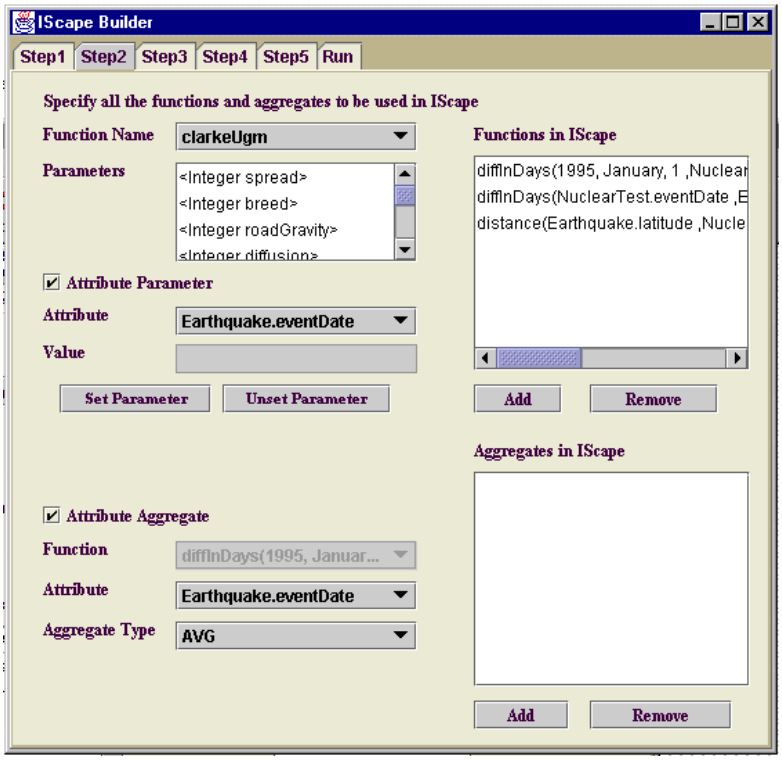


Figure 11: IScape Builder Step 2

Figure 12 shows the third step in the IScape creation process. This step enables the user to specify the conditions that make up the structure of the IScape. The basic conditions are conjoined to form the IScape.

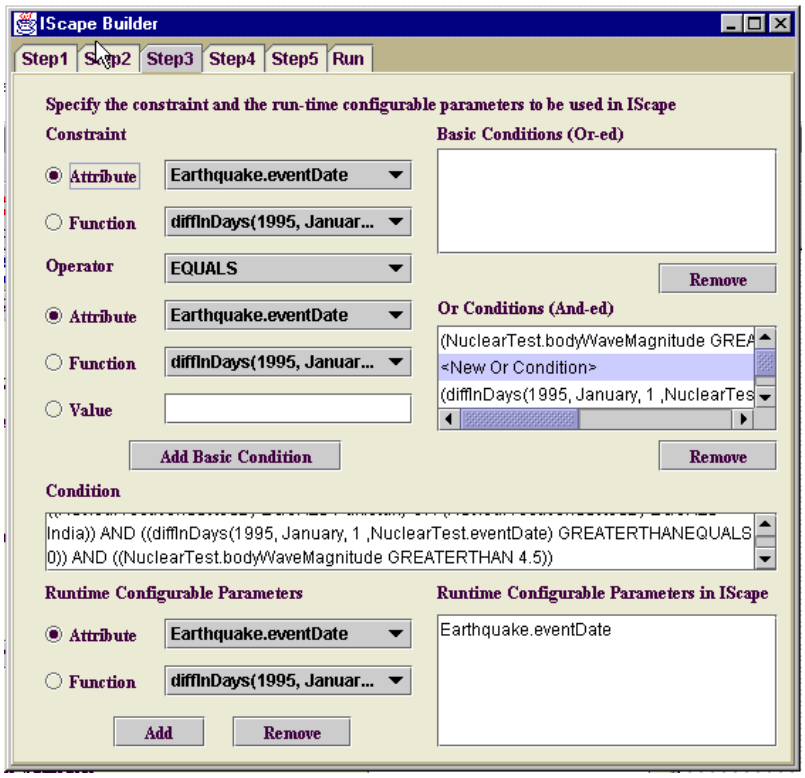


Figure12: IScape Builder Step 3

The fourth step shown in Figure 13 allows the user to specify additional constraints on the way the result of the IScape query should be grouped and what constraints apply to the members of the group.

The fifth step in the IScape creation process (Figure 14) enables the user to specify the runtime projection parameters. This is analogous to projecting an attribute of a relation in relational algebra. An example IScape specified in an XML file, which results from this five-step process, is shown in Appendix A.

After the creation process the user can run the IScape. The results are displayed as shown in Figure 15. It is possible to apply charting, filtering or visualization operations over the projection parameters.

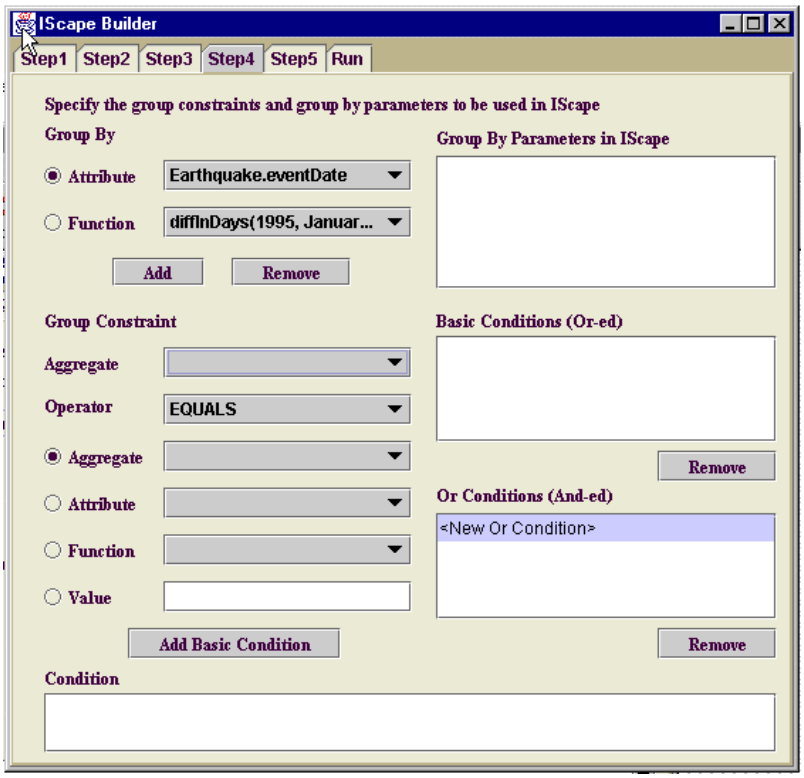


Figure 13: IScape Builder Step 4

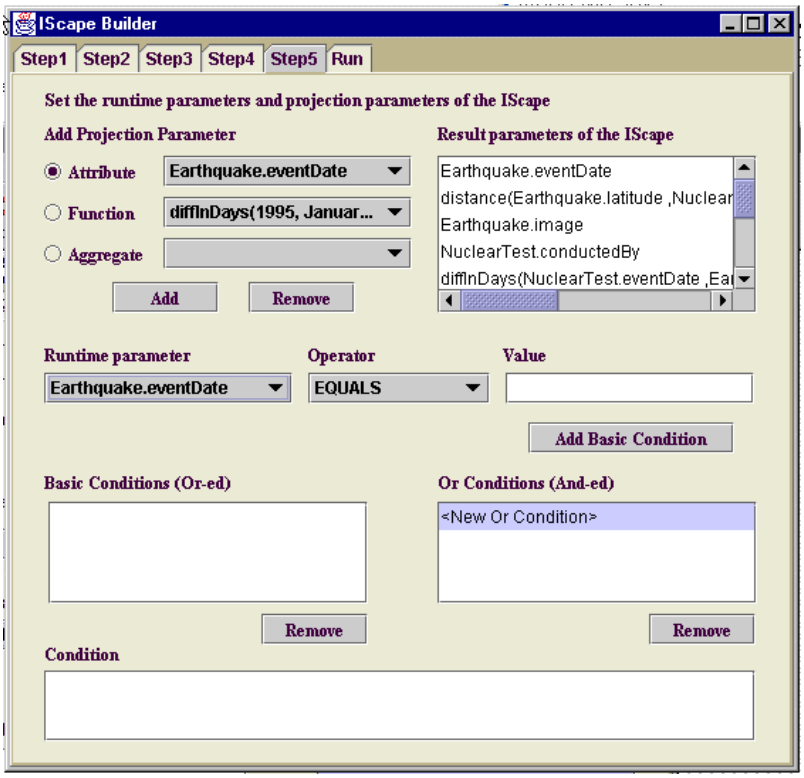
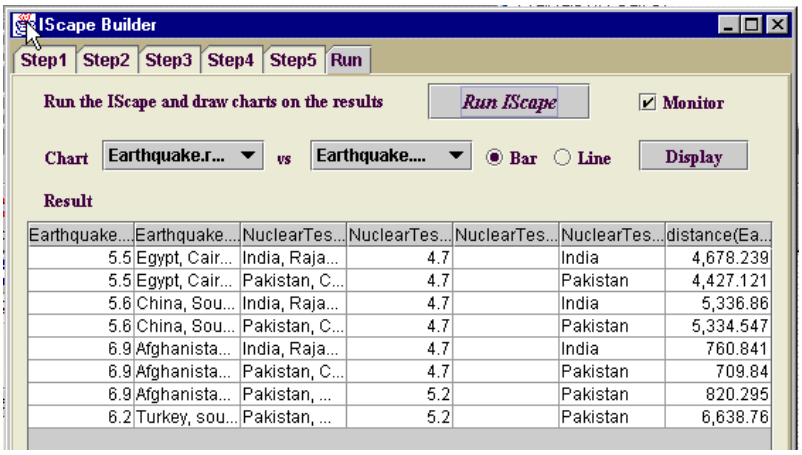


Figure 14: IScape Builder Step 5



Earthquake....	Earthquake....	NuclearTes...	NuclearTes...	NuclearTes...	NuclearTes...	distance(Ea...
5.5	Egypt, Cair...	India, Raja...	4.7		India	4,678.239
5.5	Egypt, Cair...	Pakistan, C...	4.7		Pakistan	4,427.121
5.6	China, Sou...	India, Raja...	4.7		India	5,336.86
5.6	China, Sou...	Pakistan, C...	4.7		Pakistan	5,334.547
6.9	Afghanista...	India, Raja...	4.7		India	760.841
6.9	Afghanista...	Pakistan, C...	4.7		Pakistan	709.84
6.9	Afghanista...	Pakistan, ...	5.2		Pakistan	820.295
6.2	Turkey, sou...	Pakistan, ...	5.2		Pakistan	6,638.76

Figure 15: IScape Result Window

10.3 Web-Accessible Interface to Execute IScapes

InfoQuilt also provide a web-accessible interface that provides a learning environment and allows execution of already defined IScapes (see Figure 16).

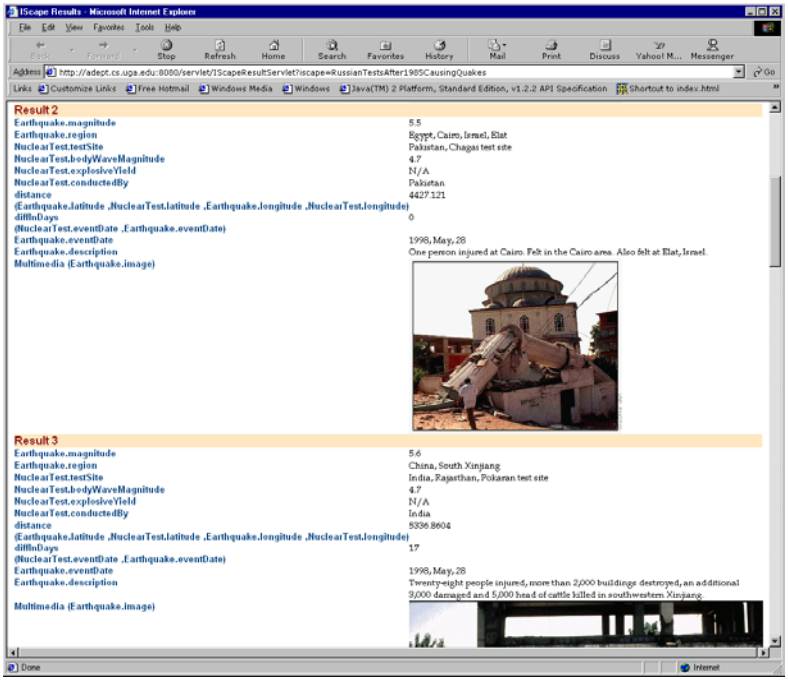


Figure 16: Sample Result Page using Web Interface to Execute IScapes

- The interface being web-accessible can be used from anywhere. However, it only allows execution of existing IScapes. It does not allow users to create new IScapes.
- It describes the entire knowledgebase 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.

11. Related Work

Most direct lineage of the InfoQuilt system to the past research is the areas of in information integration and semantic interoperability. Better known examples of heterogeneous data access and information integration systems include SIMS/ARIADNE [AK97, AKS96, KMA+01], TSIMMIS [CMH+94], Information Manifold [LRO96], GARLIC [HMN+99], InfoHarness [SS99] and HERMES [SAB+01,AE95]. An example of a system that investigated processing of queries involving multi-ontologies is OBSERVER [MIKS00]. The goal of InfoQuilt is to provide an environment where users can query, analyze, reason about inter-domain relationships and analyze data available from multiple sources (including web-based sources). However, most other systems focus only on retrieving and integrating data from multiple sources and not on the "exploring, understanding, and knowledge discovery" aspects. The following are the features of InfoQuilt that are not supported by any other systems:

- Ability to assist in discovery involving complex domain-specific relationships that may involve multiple ontologies
- Support for use of functions and simulations to post-process and thereby add value to data that is retrieved from the resources
- Support for complex relationships and constraints that cannot be expressed using relational and logical operators
- Powerful semantic query interface (IScapes)

The general approach of most of the systems is to model the domains and available information sources. They then use these models to translate a query specified by the user into an execution plan that specifies the relevant data sources that will be used and how information retrieved from them will be integrated. We compare our vision and approach with other systems with respect to the key features or focus of those systems below.

SIMS [AKS96], and its follow on ARIADNE [KMA+01] adopt the approach of creating a model of the domain using a knowledge representation system establishing a fixed vocabulary and accepting queries specified in the same language. However, its mediator is specialized to a single application domain [AHK96]. InfoQuilt supports multiple ontologies that may be completely independent of each other. An application domain in SIMS models a single hierarchy of classes. It also does not support inter-ontological relationships and functions. They do not consider use of local completeness information about sources and support only one binding pattern per web resource.

OBSERVER [MKSI96, 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 interesting. However, it is limited to basic relationships, primarily "is-a" as supported in description logic. InfoQuilt supports model complex relationships that may span multiple, independently developed ontologies.

TSIMMIS [CMH+94, GPQ+95] uses a mediator-based architecture [Wie92]. It uses a Mediator Specification Language (MSL) to define mediators, encoding how the system should use the resources. The mediators are then generated automatically from these specifications. Since the MSL definitions need to be created manually, adding or removing information sources 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 these templates. The query answering system (mediator) is thus query centric and can answer only a restricted set of queries. InfoQuilt has a dynamic planner that is not query-centric. It automatically considers newly added sources while planning IScapes.

Information Manifold [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. The user queries are also specified over relations in this world view. The sources can be specified to be either a subset of the domain or equal to a subset of the domain [LSK95]. So a resource that is locally complete for a part of information that it provides cannot be modeled appropriately. This does not capture local completeness information about the sources 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 minimum number of allowed parameters in the set. The subset selected is arbitrary. Therefore, the capability records cannot precisely specify the binding patterns. This approach would not work if the source needs very specific combinations of attributes as input.

Heterogeneous reasoning and mediator system (HERMES) [SAB+01,AE95] provides a framework to integrate information and additionally provides automation in various aspects of integration. Framework uses “Mediatory Programming Environment” (MPE) to extract and integrate information from various sources while resolving conflicts in a uniform way. A mediator is a logic program that uses predicates to execute external programs and contains clauses (subqueries) to express how the knowledge across the various domains(relational database, object-oriented database etc) is to be integrated. The framework is very extensible and can accommodate new knowledgebases. However, not much work has been done towards the development of the semantic foundation and it does not incorporate the use of web sources.

InfoHarness [SS99], a research as well as commercial system, uses metadata extraction methods, and provides integrated, rapid access to huge amounts of heterogeneous information, regardless of type, representation, location, and medium. InfoQuilt’s metadata extraction capability builds upon that of the InfoHarness system. GARLIC [HMN+99] focuses on provides ability to support queries specified an object description language against a integrated schema of wrapped heterogeneous data sources. Neither of

the system provides support for complex relationships or knowledge discovery of the type we have discussed earlier.

Recent Semantic Web based commercial web products such as the SCORE Technology from Voquette [SBA+02] and others listed at http://business.semanticweb.org increasingly recognize the importance of relationships between semantically related data from different sources. Some systems even support automatic recognition of domain-specific metadata based relationships, or semantically relate data by associating data to large (yet shallow) ontology and using that ontology to establish semantic relationships between data [App]. These systems do not yet support_complex relationships with changeable parameters and hence do not support knowledge discovery of the type InfoQuilt supports.

12. Conclusions

A basic information integration system focuses on accessing multiple heterogeneous and distributed sources and integrating the data available into a structured homogeneous data set. Adding support for simple relationships between entities that allow us to relate data in simple ways takes us to what is referred to as semantic web. Several commercial systems such as [SBA+02, App] have already achieved this. However, real word entities are related with each other in much more complex ways. We discussed the importance of support for such relationships, which takes us to the next step – Knowledge Discovery. We described the InfoQuilt system, which has the following distinguishing features:

- ability to model complex relationships involving multiple ontologies
- ability to use value-adding functions and simulations
- powerful query interface to describe complex information needs (IScapes)
- platform for specifying and performing human assisted knowledge discovery

This paper discussed our approach to modeling the knowledgebase of the system, which comprises of ontologies, inter-ontological relationships, information sources and complex operations including functions and simulations. It described the use of IScapes to construct and deploy IScapes. 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. This paper also described how all these features of InfoQuilt are put together to support human-assisted knowledge discovery. Furthermore, we described the runtime architecture of the system, gave an overview of the MÉTIS toolkit, which can be used to create an integrated homogeneous metabase from multiple heterogeneous sources, described how practical execution plans are created for the IScapes, how an IScape is processed in the system, and how this processing can be monitored to easily locate failures. We also described a number of tools with visual interfaces available for the user such as the Knowledge Builder, IScape Builder, the IScape Processing Monitor, etc. We described several examples throughout the paper that have been implemented and tested and explain how InfoQuilt can be used to better relate, integrate and use the large amount of information which is at our disposal to learn about domains of interest, their characteristics and their relationships with each other.

Some of the planned future enhancements are as follows:

- Inductive learning can be used to infer domain rules, FDs, and data characteristics in addition to those already specified by the administrator at the time of knowledgebase creation.
- The Planning Agent can create backup plans that the Correlation Agent can switch to on failure. The resources available to the system are completely autonomous. If any of them fails during IScape execution, the Correlation Agent can switch to a backup plan that could be a slight modification of the original so as to account for the failure(s).
- Simulations are currently supported using the framework used for functions. Simulation programs however are more complex and diverse in the kind of application (it could be an executable, a script, etc.), the method of accepting inputs, for example as files from local file systems, etc. The framework currently used may not suffice as it assumes that they are available as separate functions (through wrappers if needed). Several simulations however exist as programs written using languages supported by special software, for example, ArcInfo, and hence, maybe difficult to add to the system. Different types of simulation programs therefore need to be explored more thoroughly to provide a framework better suited to support a large class of them, if not all.

Acknowledgements

Reviewers' comments led us to provide more details than what we had provided in the first version of this paper. We thank for their diligent reviews. We thank David Avant, Kemafor Anyanwu, and Cartic Ramakrishnan for their assistance in improving the drafts. InfoQuilt has been a group project. Finally, we acknowledge past contributors, including Vipul Kashyap, Tarcisio Lima, Clemens Bertram, Krishnan Parasuraman, Kashitj Shah, Mukesh Guntamadugu, Sriram Lakshminarayan and Narayanan Palsena.

13. References[4]

[Ade] Alexandria Digital Earth Prototype. http://www.alexandria.ucsb.edu/

[AE95] S. Adali and R. Emery. A uniform framework for integrating knowledge in heterogeneous systems. Proceedings of the Eleventh IEEE International Conference of Data Engineering (March 1995).

[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.

[AK97] J. Ambite, and C. Knoblock. Planning by Rewriting: Efficiently generating high-quality plans. Proceedings of the 14th National Conference on Artificial Intelligence, Providence, RI, 1997.

[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.

[App] CIRCA Technology, Applied Semantics Inc., http://www.appliedsemantics.com

[4] All URLs are valid as of March 30, 2002.

[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.

[Ber98] 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 Gigalopolos, Department of Geography, University of California, Santa Barbara. http://www.ncgia.ucsb.edu/projects/gig/

[CM97] M. Califf and R. Mooney. Relational Learning of pattern-match rules for information extraction. Working papers of the ACL-97 Workshop in Natural Languages Learning 9-15. 1997.

[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.

[Dec01] S. Decker, Inference Engines for the Semantic Web. http://www.semanticweb.org/inference.html.

[DHM+01] G. Denker, J. R. Hobbs, D. Martin, S. Narayanan, R. Waldinger, Accessing Information and Services on the DAML-Enabled Web Proceedings of the Second International Workshop on the Semantic Web SemWeb'2001 Hongkong, China, May 1, 2001

[FEDC02] F. Fonseca, M. Egenhofer, C. Davis, and G. Camara. Semantic Granularity in Ontology-Driven Geographic Information Systems. AMAI Annals of Mathematics and Artificial Engineering, 2002 (to appear).

[FHLW02] D.Fensel, J.Hendler, H.Liebermann, and W.Wahlster, (Eds). Creating the Semantic Web. MIT Press, 2002.

[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

[HMN+99] L. Haas, R. Miller, B. Niswonger, M. Tork Roth, P. Schwarz, and E. Wimmers, "Transforming Heterogeneous Data with Database Middleware: Beyond Integration, Data Engineering Bulletin, 1999.

[HGC+97] J. Hammer, H. Garcia-Molina, J. Cho, R. Aranha, and A. Crespo. Extracting Semistructured Information from the Web. Proceedings of the Workshop on Management of Semistructured Data. Tucson, Arizona, May 1997.

[KMA+01] C. Knoblock, S. Minton, J. Ambite, N. Ashish, I. Muslea, A. Philpot, S. Tejada. The Ariadne Approach to Web-Based Information Integration. International Journal on Cooperative Information Systems (IJCIS). Special Issue on Intelligent Information Systems: Theory and Applications, 10(1/2), pp145-169, 2001.

[K01] M. Klein. Combining and relating ontologies: an analysis of problems and solutions In Workshop on Ontologies and Information Sharing, IJCAI'01, Seattle, USA, August 4-5, 2001

[Kru95] G. Krupka. Description of the sra system as used for muc 6. Proceedings of the Sixth Message Understanding Conference (MUC-6) 221 – 235. 1995.

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

[KS96] V. Kashyap, A. Sheth. Schematic and Semantic Similarities between Database Objects: A Context-based Approach - in the VLDB Journal 5 (4), 1996.

[KS98] V. Kashyap and A. Sheth. Semantic Heterogeneity in Global Information Systems: The Role of Metadata, Context and Ontologies, in M. Papazoglou and G. Schlageter (eds.), Cooperative Information Systems: Current Trends and Directions. Academic Press, 1998, pp.139-178.

[Lak00] S. Lakshminarayan, Semantic Interoperability in Digital Libraries using Inter-ontological Relationships, M.S. Thesis, Computer Science, Univ. of Georgia, August 2000.

[Leh] W. Lehnert. Information Extraction. Natural Language Processing Laboratory, University of Massachusetts. http://www-nlp.cs.umass.edu/

[LHL01] T. Berners-Lee, J. Hendler, O. Lassila. The Semantic Web. Scientific American, 284 (5), 2001, pp.28-37

[LPH00] L. Liu, C. Pu, and W. Han. XWRAP: An XML Enabled Wrapper Construction System for Web Information Sources. In Int'l Conference on Data Engineering, 2000, pp. 611—621.

[LRO96] A. Y. Levy, A. Rajaraman, and 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.

[LSK95] A.Y. Levy, D. Srivastava, and T. Kirk. Data Model and Query Evaluation in Global Information Systems. International Journal on Intelligent Information Systems, pp. 121-143, 1995.

[MKSI96 E. Mena, V. Kashyap, A. Sheth, A. Illarramendi, "OBSERVER: An Approach for Query Processing in Global Information Systems Based on Interoperation Across Pre-existing Ontologies", Proceedings of the 1st IFCIS Intl. Conference on Cooperative Information Systems (CoopIS'96), Brussels, Belgium, June 1996, pp. 14-25.

[MIKS00] E. Mena, A. Illarramendi, V. Kashyap, and 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.

[Mus99] I. Muslea. Extraction Patterns for Information Extraction Tasks: A Survey. The AAAI-99 Workshop on Machine Learning for Information Extraction, 1999. http://www.isi.edu/~muslea/PS/ml4ie-aaai99.ps

[Okl] Oklahoma Geological Survey Observatory. Catalog of Nuclear Tests compiled by James E. Lawson Jr. http://www.okgeosurvey1.gov/level2/nuke.cat.index.html

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

[Par98] K. Parasuraman. A Multi-Agent System For Information Brokering In InfoQuilt. Masters Thesis, Computer Science Department, University of Georgia, 1998.

[PS01] S. Patel and A. Sheth. Planning And Optimizing Semantic Information Requests On Heterogeneous Information Sources Using Semantically Modeled Domain And Resource Characteristics. LSDIS Technical Report, University of Georgia, March 2001. http://lsdis.cs.uga.edu/proj/iq/pubs/planning_optimization.doc
An abridged version of this report appears in *Proceedings of the 6th Intl Conf on Cooperative Information Systems (CoopIS),* Trento, Italy, September 5-7, 2001, pp. 135-149.

[Ril93] E. Riloff. Automatically constructing a dictionary for information extraction tasks. Proceedings of the 11th National Conference on Artificial Intelligence (AAAI-93) 811 – 816. 1993.

[SA99] A. Sahuguet and F. Azavant. W4F: a WysiWyg Web Wrapper Factory. 1999. http://db.cis.upenn.edu/DL/WWW8/index.html

[SAB+01] V.S.Subramanian, Sibel Adali, Anne Brink, Ross Emery, James J. Lu, Adil Rajput, Timothy J. Rogers, Robert Ross and Charles Ward. HERMES: Heterogeneous Reasoning and Mediator System. Submitted for publication. http://www.cs.umd.edu//projects/hermes/overview/paper/index.html

[SBA+02] Sheth, A., C. Bertram, D. Avant, B. Hammond, K. Kochut, Y. Warke. Semantic Content Management for Enterprises and the Web, IEEE Internet computing, July/August, 2002 (to appear). A version is available as a technical white paper from Voquette, Inc. http://www.voquette.com/demo.

[SemWeb] S. Decker (Ed.), The Semantic Web Community Portal. http://www.semanticweb.org

[SFAL95] S. Soderland, D. Fisher, J. Aseltine and W. Lehnert. Crystal: Inducing a conceptual dictionary. Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95) 1314 – 1319. 1995.

[She96] A. Sheth. Data Semantics: What, Where and How? Proceedings of the 6th IFIP Working Conference on Data Semantics (DS-6), R. Meersman and L. Mark (Eds.), Chapman and Hall, London, UK, 1996.

[Sin00] D. Singh. An Agents Based Architecture for Query Planning and Cost modeling of Web Sources. Masters Thesis. Computer Science Department, University of Georgia, 2000.

[SK93] A. Sheth and V. Kashyap. So Far (Schematically) yet So Near (Semantically). Proc. of the DS-5 Semantics of Interoperable Database Systems, Lorne, Australia; In IFIP Transactions A-25, North-Holland, 1993

[SK96] A. Sheth and V. Kashyap. Media-independent correlation of information: What? How? Proceedings of the First IEEE Conference Metadata Conference, April 1996.

[SKL99] A. Sheth, V. Kashyap, and T. Lima. Semantic Information Brokering: How Can a Multi-Agent Approach Help? Cooperative Information Agents III, Lecture Notes in Artificial Intelligence. M. Klusch, O. Shehory, G. Weiss (Eds.), Vol. 1652, Berlin et al: Springer-Verlag 292-311, July 1999.

[SM01] G. Stumme and A. Maedche, FCA-MERGE: Bottom-Up Merging of Ontologies. Proceedings of the IJCAI-01 Workshop on Ontologies and Information Sharing, Seattle, USA, August 4-5, 2001.

[Sod99] S. Soderland. Learning information extraction rules for semi-structured and free text. Machine Learning, 44(1-3):233--272, 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'98) Santa Barbara, CA. 1998, pp. 266-275,

[SS99] K. Shah and A. Sheth, "InfoHarness: Managing Distributed, Heterogeneous Information," IEEE Internet Computing, 3 (6), November/December 1999.

[SAB01] Sheth, A., D. Avant, C. Bertram. "System and Method for Creating Semantic Web and Its Applications in Browsing, Searching, Profiling, Personalization and Advertisement" US Patent # 6,311,194), October 30, 2001.

[Tri] Trinity Atomic Web Site. Gallery of US Nuclear Tests. http://nuketesting.enviroweb.org/hew/Usa/Tests/

[USGS] U.S. Geological Survey. http://www.usgs.gov

[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.

[Wie92] 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.

APPENDIX A
IScape FORMAT

The code snippet shown below shows actual output of the IScape builder tool for the example IScape as a result of the five step IScape creation process (with some added comments). This IScape was created to test and help validate the hypothesis that the Nuclear test can cause Earthquakes (i.e., data sources provide evidence to support the NuclearTestCausesEarthquate relationship).

```
<?xml version="1.0" ?>
<!DOCTYPE Ontology (View Source for full doctype...)>
- <IScape>
<! --  textual description of the query -- >
<Statement>Find all nuclear tests conducted by India or Pakistan after January 1,
    1995 with seismic body wave magnitude > 4.5 and find all earthquakes that
    could have been caused due to these tests</Statement>
<Description>Find all nuclear tests conducted by India or Pakistan after January 1,
    1995 with seismic body wave magnitude > 4.5 and find all earthquakes that
    could have been caused due to these tests</Description>
- <Ontologies>
<! -- Ontologies are selected when designing an ISCape (IScape Builder Step 1) -->
<Ontology>Earthquake</Ontology>
<Ontology>NuclearTest</Ontology>
  </Ontologies>

<! -- Relationships are selected during Step 1 of the IScape Building process. -->

<Relationship>NuclearTestCausesEarthquake</Relationship>

- <Constraint>
<! -- The constraints that qualify relationship are defined during Step 3 of the IScape
    Building process. These constraints are applied to the data sources. -->
- <And>
- <Or>
- <BasicCondition>
- <FunctionOperand>
<! -- Function is added by the user during Step 2 of the IScape Building process. This
    function was created by Knowledge Builder, and its definition is stored in a
    functionStore.java, When Knowledge Builder is closed, functions are compiled
    and stored in appropriate  .class files. -- >
<FunctionName>diffInDays</FunctionName>
```

<ParameterList>
<ValueParam>**1995, January, 1**</ValueParam>
<AttributeParam Ontology="**NuclearTest**">**eventDate**</AttributeParam>
</ParameterList>
</FunctionOperand>
<Operator>**GREATERTHANEQUALS**</Operator>
<ValueOperand>**0**</ValueOperand>
</BasicCondition>
</Or>
<Or>
<BasicCondition>
<AttributeOperand Ontology="**NuclearTest**">**conductedBy**</AttributeOperand>
<Operator>**EQUALS**</Operator>
<ValueOperand>**Pakistan**</ValueOperand>
</BasicCondition>
<BasicCondition>
<AttributeOperand Ontology="**NuclearTest**">**conductedBy**</AttributeOperand>
<Operator>**EQUALS**</Operator>
<ValueOperand>**India**</ValueOperand>
</BasicCondition>
</Or>
<Or>
<BasicCondition>
<AttributeOperand
Ontology="**NuclearTest**">**bodyWaveMagnitude**</AttributeOperand>
<Operator>**GREATERTHAN**</Operator>
<ValueOperand>**4.5**</ValueOperand>
</BasicCondition>
</Or>
</And>
</Constraint>
<RuntimeConfigurableConstraints>

*<AttributeOperand Ontology="**Earthquake**">**eventDate**</AttributeOperand>*
</RuntimeConfigurableConstraints>
<! -- This IScape does not use the GroupBy clause and the Group Constraints that can be added during Step 4 of the IScape Building process. -->

<! -- Projections specify the output parameters for the data set that supports the IScape. These are specified in Step 5 of the IScape Building process -- >
<Projections>
*<AttributeOperand Ontology="**Earthquake**">**magnitude**</AttributeOperand>*
*<AttributeOperand Ontology="**Earthquake**">**region**</AttributeOperand>*
*<AttributeOperand Ontology="**NuclearTest**">**testSite**</AttributeOperand>*
<AttributeOperand
*Ontology="**NuclearTest**">**bodyWaveMagnitude**</AttributeOperand>*
*<AttributeOperand Ontology="**NuclearTest**">**explosiveYield**</AttributeOperand>*
*<AttributeOperand Ontology="**NuclearTest**">**conductedBy**</AttributeOperand>*
<FunctionOperand>
*<FunctionName>**distance**</FunctionName>*
<ParameterList>
*<AttributeParam Ontology="**Earthquake**">**latitude**</AttributeParam>*

<AttributeParam Ontology="**NuclearTest**">**latitude**</AttributeParam>
<AttributeParam Ontology="**Earthquake**">**longitude**</AttributeParam>
<AttributeParam Ontology="**NuclearTest**">**longitude**</AttributeParam>
</ParameterList>
</FunctionOperand>
- <FunctionOperand>
<FunctionName>**diffInDays**</FunctionName>
- <ParameterList>
<AttributeParam Ontology="**NuclearTest**">**eventDate**</AttributeParam>
<AttributeParam Ontology="**Earthquake**">**eventDate**</AttributeParam>
</ParameterList>
</FunctionOperand>
<AttributeOperand Ontology="**Earthquake**">**eventDate**</AttributeOperand>
<AttributeOperand Ontology="**Earthquake**">**description**</AttributeOperand>
<AttributeOperand Ontology="**Earthquake**">**image**</AttributeOperand>
</Projections>
- <FunctionOperands>
- <FunctionOperand>
<FunctionName>**diffInDays**</FunctionName>
- <ParameterList>
<AttributeParam Ontology="**NuclearTest**">**eventDate**</AttributeParam>
<AttributeParam Ontology="**Earthquake**">**eventDate**</AttributeParam>
</ParameterList>
</FunctionOperand>
- <FunctionOperand>
<FunctionName>**diffInDays**</FunctionName>
- <ParameterList>
<ValueParam>**1995, January, 1**</ValueParam>
<AttributeParam Ontology="**NuclearTest**">**eventDate**</AttributeParam>
</ParameterList>
</FunctionOperand>
- <FunctionOperand>
<FunctionName>**distance**</FunctionName>
- <ParameterList>
<AttributeParam Ontology="**Earthquake**">**latitude**</AttributeParam>
<AttributeParam Ontology="**NuclearTest**">**latitude**</AttributeParam>
<AttributeParam Ontology="**Earthquake**">**longitude**</AttributeParam>
<AttributeParam Ontology="**NuclearTest**">**longitude**</AttributeParam>
</ParameterList>
</FunctionOperand>
</FunctionOperands>
<AggregationOperands />
</IScape>