Clemens Bertram
Semantic Correlation of Heterogeneous Distributed Assets in InfoQuilt
(Under the direction of Amit Sheth)

As the WWW makes a vast amount of information assets of various types accessible to the Internet user, it becomes increasingly important to retrieve assets that a) are semantically related and b) cover a broad range of asset types. Most current Web search engines like Yahoo or Excite that provide only keyword–search capabilities fall short of either requirement. Attribute– and content–search make use of metadata, modeled in domain–specific and domain–independent ontologies. They can substantially enhance the result to a given information request. In this thesis, we formalize and detail the Metadata Reference Link (MREF), a means to correlate information across ontologies, asset types, and resources on a resource–independent level, integrating all three search techniques for best results. The multi–agent system that processes the MREF and a detailed scenario involving multiple ontologies and real world data are also discussed.

Index words: MREF, InfoQuilt, semantic correlation, metadata, XML, RDF, Java, agent technology

Semantic Correlation of Heterogeneous Distributed Assets in InfoQuilt

by

Clemens Bertram

A Master's Thesis Submitted to the Graduate Faculty
of The University of Georgia in Partial Fulfillment
of the
Requirements for the Degree

MASTER OF SCIENCE

Athens, Georgia

1998

Semantic Correlation of Heterogeneous Distributed Assets in InfoQuilt

by

Clemens Bertram

Approved:

_____

Major Professor

_____

Date

Approved:

_____

Graduate Dean

_____

Date

Acknowledgements

I would like to thank Professor Vernon Meentemeyer and numerous other meteorologists for their valuable expert feedback during the development of the weather–related ontologies.

Dr. Covington deserves special thanks for providing and maintaining the UGA LaTeX style file.

The writing of this thesis would not have been possible without the active support of the Large Scale Distributed Information Systems (LSDIS) lab under Dr. Sheth, especially the members working on the InfoQuilt project, namely, Dr. Amit Sheth, Kshitij Shah, and Krishnan Parasuraman. Their help, suggestions, and guidance have been very helpful to me and I am grateful that I had the opportunity to work with them.

A Research Assistant with Dr. Sheth, I worked on the very interesting and challenging project *VideoAnywhere* with the private industry (see Chapter 5). Not only did it give me valuable insights into the industrial world, it also encouraged me to write down important findings in a publishable paper. I am thankful that the LSDIS lab provided me with the software, hardware, and office that made the many hours of work much more enjoyable.

Last but not least, I want to thank my committee and in particular my supervisor Dr. Sheth for spending many hours of reading through my thesis and giving me helpful suggestions how to improve it.

Contents

List of Figures

Chapter 1

Introduction

The Internet is the largest data resource available today. Unfortunately, a lot of assets are not easily accessible for various reasons, e.g., they might be hidden behind different types of query interfaces or are not available to the public. Most current search engines like Yahoo! or Excite search index only textual data, and even if an interesting asset is retrieved by a keyword–query, it is easily overlooked among the thousands of other results. Recently, a lot of research has been done towards offering Web search for heterogeneous assets, either using the familiar keyword–search [Sco, Mag] or, increasingly, content–based search [Vir, Pro].

If we think of the WWW today as a "physical Web", that is, a network of mostly hardcoded hyperlinks, then the next step is to move on to a dynamic, "logical Web" that makes browsing smarter and more efficient: The Internet user should not need to know the peculiarities of many different search engines in order to retrieve the information he or she is looking for. Rather, it should be possible to either follow a hyperlink or compose an information request that leads to a collection of heterogeneous assets that are semantically related and match the user's request as closely as possible. For that purpose, we need a general interface that "offers *location*, *model*, and *interface* transparency" [GP98], and an underlying information brokering architecture that

a) supports this transparency,

b) enables semantic correlation of information assets, and

c) is dynamic and extensible to account for the steady growth of the WWW.

InfoQuilt is such an architecture. It combines the basic ideas of mediators[1] with a multi–agent infrastructure similar to InfoSleuth [BBB+97, ISW].

Mediators and metasearchers are software modules that export an integrated view of heterogeneous sources. They provide the user with the capability of running a single query against a multitude of related information sources (databases, video servers, ...) and automate rather tedious tasks such as selecting resources or formulating the same query for multiple interfaces. Systems like TSIMMIS [CGMH+94, TSIW] are aimed at automating the generation of mediators.

Resource Agents in InfoQuilt are similar to mediators (see Chapter 2), but unlike in the mediator and InfoSleuth architecture, the resources that they maintain are often (but not always) metabases that manage only the *metadata* of assets that are located elsewhere. Other important differences to the otherwise similar InfoSleuth architecture include

- InfoQuilt does not restrict the interaction with the user to applets; rather, it allows the direct embedding of a "logical link" (see below) in any Web page.
- InfoSleuth uses only a few global ontologies and does not support information correlation. InfoQuilt correlates information across multiple ontologies that need not be known to every resource.
- In InfoQuilt, we integrate multiple search techniques (keyword, attribute, content–based) to get more precise results.

In this thesis, we discuss the *Metadata Reference Link* (MREF), which provides a general formalism for specifying information requests independent from resources, data models, or query interfaces. Thus, it serves as a generalized description of the

[1] see [GP98] for an in–depth overview

assets that the user wants to retrieve and achieves the above mentioned transparency. SIMS [ACHK, SIMW] and Ontobroker [DEFS98] accept queries that are similarly general[2], but unlike MREF, they use only one ontology, do not integrate different search techniques, and are not well embedded in the WWW architecture, which are some of the key strengths of MREF as we will show.

A major shortcoming of current search engines is that they (often) fail to produce semantically related results. The main problem is that the keywords have been indexed out of their respective contexts. We can overcome this problem by correlating *semantic information* about entities that are involved in the information request. For example, when looking for information on El Niño and its consequences on the fish population at the coast of Peru, it is possible to express the whole query in terms of the entities "El Niño" and "Fish", their attributes (e.g., "region"), and their relations (e.g., "affects"). *Metadata* [SK98] contain such semantic information about entities, and they can be accessed through ontologies that model entities and their relations among each other in a specific context (see Chapter 3 for details about the use of ontologies in InfoQuilt). In our example, we get the needed context by referring to the "Climate" and "Organism" ontologies. Relations such as "affects" provide links between both ontologies, thus enabling information correlation across multiple domains. Metadata support descriptions that are much more precise than any keyword query can ever be. MREFs make extensive use of metadata to correlate information across multiple ontologies and to describe resulting assets in terms of their attributes and relations. Chapter 4 details and formalizes this idea.

The concept of an MREF is not new. It was introduced to the public in [SS98]. However, at that time it was neither formalized nor did it correlate entities across ontologies. The main accomplishment of this thesis is to fully integrate the MREF

[2]SIMS uses Loom as the query language, Ontobroker has developed a proprietary format.

with ontologies, put it on a solid, formal basis, and make it consistent with the current draft of the Resource Description Format.

This thesis is organized as follows: Chapter 2 presents the InfoQuilt agent architecture and explains how an MREF is processed. Chapter 3 discusses design and implementation issues of ontologies in general and details the ontologies, which we have developed for our sample scenario "El Niño". The MREF itself is described and formalized in Chapter 4. An example MREF is discussed in detail. Chapter 5 gives some insights into a video metabase and how it is managed. Finally, Chapter 6 summarizes our work and outlines future directions.

Chapter 2

InfoQuilt Architecture

This chapter describes the agent infrastructure of InfoQuilt. It is designed to support the generation and processing of an information request, which is expressed by an MREF, and the display of the resulting information assets. The first section gives a broad overview of the general agent interaction during the MREF processing. The second section is concerned with the query pre– and post–processing that is coordinated by the User Agent. Finally, the third section tracks the actions and events that happen from the moment when the user issues an information request to the display of a Web page that contains the links to and metadata of the retrieved assets.

2.1 Agent Infrastructure

InfoQuilt uses a multi-agent system[1] to process the MREF. Six agent types are involved in this task: the User Agent, one or more Broker Agents, an Ontology Agent, a Query–Planning Agent, and possibly many Resource Agents (see Figure 2.1). Additional InfoQuilt components include Encapsulator Agents, which help keep the metabases up–to–date, and a User Profile Manager, which maintains both user–specific and generic profiles that are used to enrich queries and filter results.

InfoQuilt deals with a number of interoperability issues that arise on various levels when designing an information brokering architecture [She98] that is built

[1] See [HS98] for an overview of agent and multi-agent technology.

upon heterogeneous data and metadata resources. In particular, it enables system, syntactic, representational, structural, and semantic interoperability:

- All agents are implemented in Java, a *system*–independent language.
- The use of XML, the emerging standard language for the Internet, ensures *syntactic* interoperability by defining the syntax of a valid document, so that there is, for instance, no ambiguity concerning the representation of special characters.
- The agents communicate using the Knowledge Query and Manipulation Language (KQML) [FLM95]. KQML standardizes the way in which the information exchange between agents is *represented* by using so–called "performatives".
- MREF, which is built on top of RDF, imposes a certain *structure* on the description of resources (see Chapter 4 for details).
- Lastly, the OBSERVER subsystem of InfoQuilt[2] provides *semantic* interoperability by correlating information across different ontologies using synonyms, hypernyms, and hyponyms [MIKS98].

The design and implementation issues involved in the agent infrastructure are the subjects of the related Master's thesis [Par98].

2.1.1 User Agent

The User Agent is the main "communication partner" for the user. It resides on the InfoQuilt server and is responsible for

- retrieving the MREF (possibly generated on–the–fly),

[2]OBSERVER is not a part of this thesis.

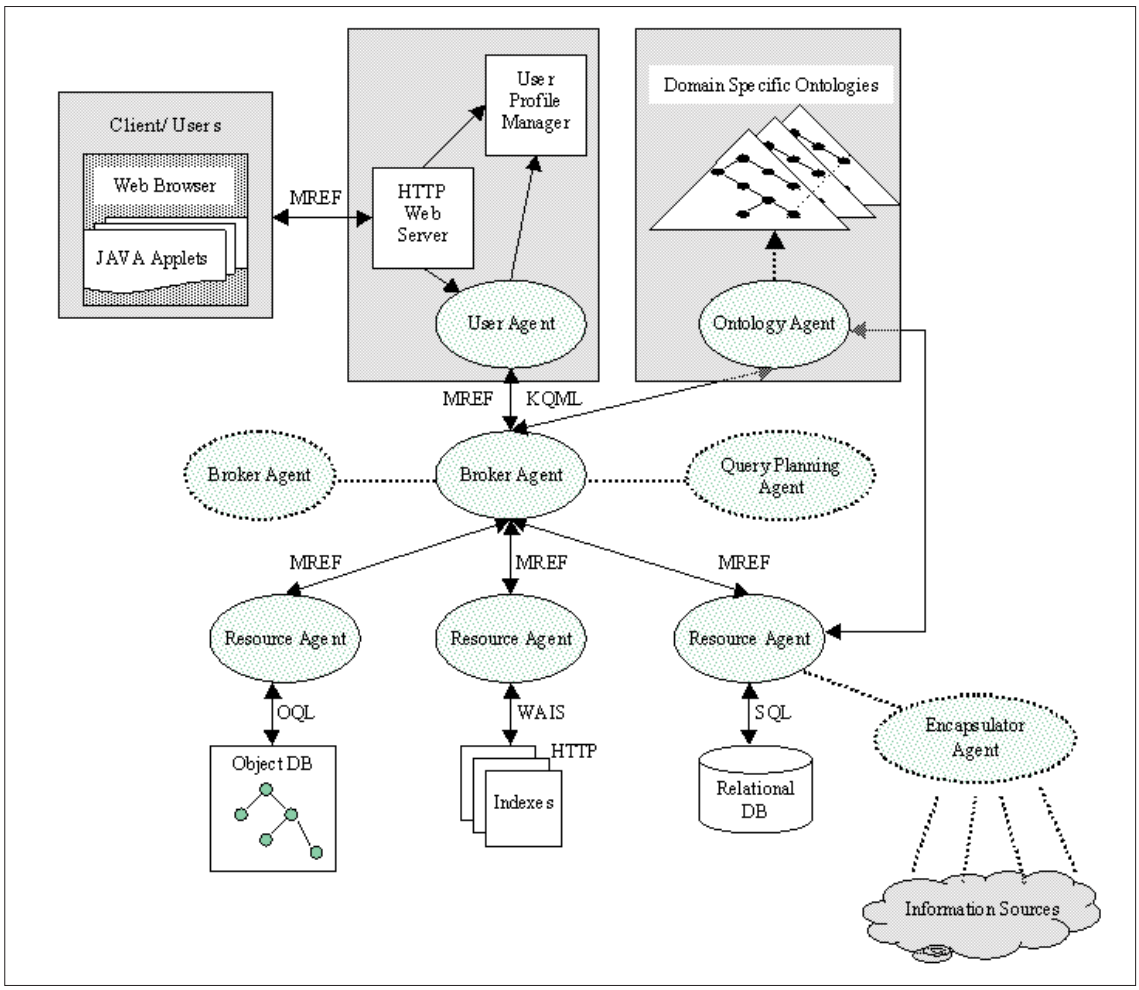

Figure 2.1: InfoQuilt Agent Infrastructure

- retrieving (and caching) a profile for the user,
- altering the MREF according to that user profile,
- forwarding the MREF to a Broker Agent,
- retrieving, parsing, and filtering the results that come back from the Broker Agent.

The actions and interactions of the User Agent (see Figure 2.2) are described in more detail in Section 2.2.

2.1.2 Broker Agent

A Broker Agent serves as a "facilitator" in InfoQuilt. It receives the MREF and forwards it to those Resource Agents that the Query–Planning Agent recommended. For that purpose, the Broker Agent maintains an agent directory, which contains the necessary information about all the known Resource Agents. Once it has gathered the results from the Resource Agents, it sends them back to the User Agent.

2.1.3 Query–Planning Agent

The Query–Planning Agent interacts with the Broker Agent. Its task is to find out which Resource Agents are able to contribute to a successful MREF processing based on their respective capabilities.

2.1.4 Ontology Agent

The Ontology Agent is consulted by the Broker Agent and the Resource Agents. Its main purpose is to correlate information between different ontologies. In many cases, the ontologies that were used to build an MREF are different from those ontologies that are known to the Resource Agents. In order to correctly process the MREF, terms have to be mapped so that as much of the original semantics is preserved as possible. Query processing across multiple domain ontologies is an ongoing research issue that is dealt with in the OBSERVER subsystem of the InfoQuilt project[3] [MIKS98].

2.1.5 Resource Agent

Resource Agents work as wrappers around information resources such as metabases. Metabases maintain information about available assets. This information can be

[3]This subsystem is not integrated in the current InfoQuilt system.

used to intelligently retrieve assets based on their metadata rather than through simple (and often vague) keyword–searches. It is the responsibility of the Resource Agents to update "their" respective metabases on a regular basis. They do this by interacting with Encapsulator Agents (see Section 2.1.6).

Having joined the multi–agent system by advertising their presence and capabilities to the Broker Agent(s), their main task is to accept an MREF from a Broker Agent and to return information assets from the metabase that match the MREF description (see Chapter 4) as closely as possible[4]. In order to achieve this, they have to translate the generic MREF into a local query that is executed against the metabase. If necessary, Resource Agents interact with the Ontology Agent to translate terms used in the MREF ontologies to terms of the local ontologies.

2.1.6 Encapsulator Agent

Encapsulator Agents provide Resource Agents with new information that is needed to keep a metabase up–to–date. For that purpose, they have a number of metadata extractors, resource crawlers, etc. at their disposal that crawl the Web extracting metadata about various types of information resources such as databases, Web sites, video servers, a.s.o.

Encapsulator Agents work autonomously and advertise new metadata to all subscribing Resource Agents, who update their respective metabases. This benefits the Internet users because more information is accessible to them.

[4] A "rating" value is assigned to each asset that indicates how good a particular result is deemed by the Resource Agent. The problem of how to compare such values might later be dealt with by a Correlation Agent (see Chapter 6).

2.2 Query Pre– and Post–Processing

Usually, MREFs are embedded in an HTML anchor[5]. The user simply follows the link, which bears enough information for the User Agent to retrieve the pre–constructed MREF. Alternatively, it is possible to query an MREF repository for an MREF or to construct an MREF from scratch using a graphical user interface[6]. Once the user has composed an MREF or decided on one, he or she sends it to the User Agent servlet.

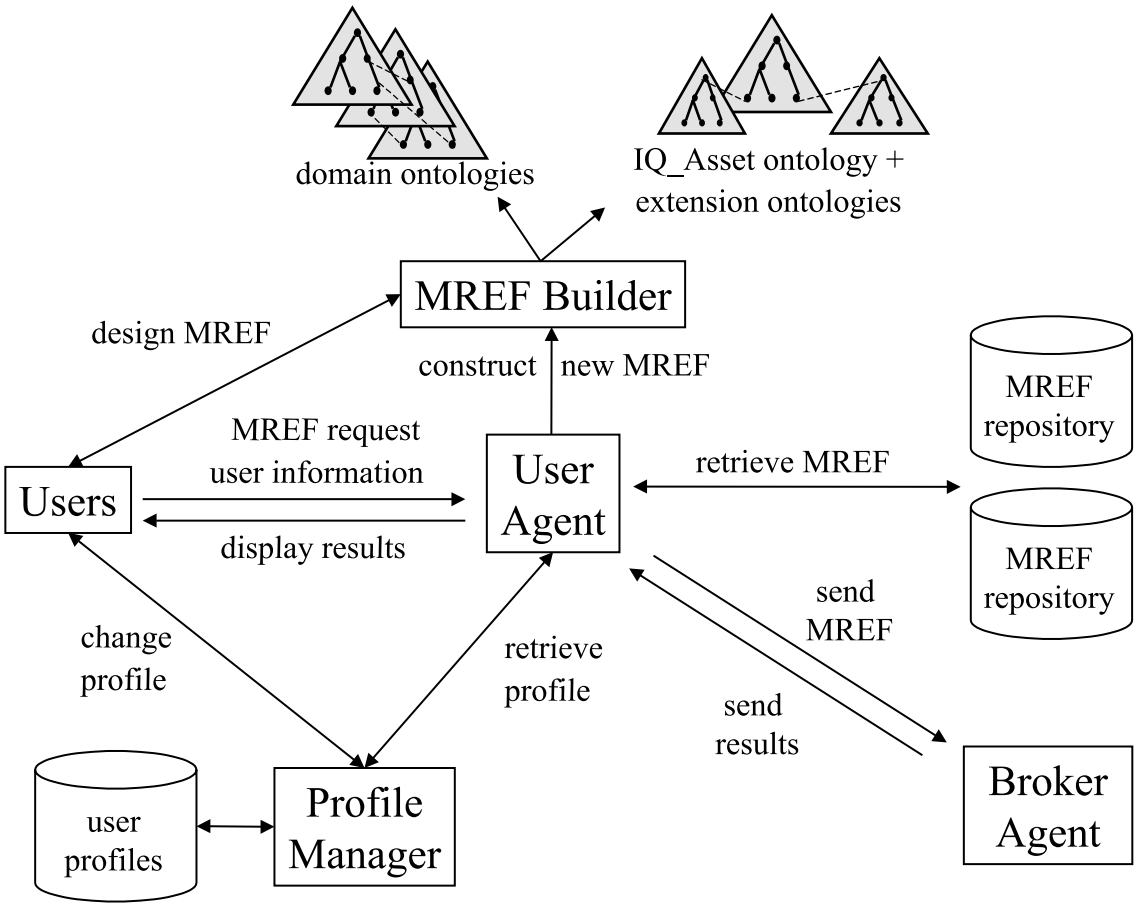

Figure 2.2: User Agent Interactions

[5] See Section 4.4 for the precise syntax.

[6] Although neither of these two options is currently implemented, they can be easily integrated into the whole system once they are developed.

User profiles can be used to customize the processing and display of an MREF. Profiles can be set for both single users or a group of users. Since not every user will want to set up and personalize his own profile, there is a general profile for casual users that is used whenever no personal profile can be found. If the user's browser is configured to allow the setting of cookies, then the Profile Manager can store some information (e.g., a login name) on the client's side. The User Agent can retrieve the user's profile by accessing the cookie and forwarding the information to the Profile Manager, who then returns the profile to the agent. For efficiency reasons, the User Agent stores the profile information with the current browser session, so that it does not have to be retrieved every time the user requests information.

The information stored in the user profile can be used both to pre–process the MREF before it is sent on to the Broker Agent, and to filter the set of results that is returned. For instance, if a user on a Sun Sparc is not able to play any type of video, it makes sense to set up the profile so that no video assets are included in the final result display. The profile can also specify whether the results should be displayed according to their rating or rather grouped by asset types[7].

2.3 A Complete Sample Run

Let us suppose the user is looking at a Web page with an embedded MREF. He requests to process it (e.g., by clicking on it), and the User Agent servlet is invoked with the location of the definition of this MREF. If this is the first time the user requests information, the User Agent tries to get the user profile by checking a cookie. It sends the cookie–information to the Profile Manager who will return the user's profile. If cookies are not allowed or the user does not have a profile yet, the User

[7] In the current implementation, we do not support the active use of profiles; however, the system provides the hooks for plugging in the needed modules as soon as they are implemented.

Agent asks the Profile Manager for the "casual users"–profile. The MREF is pre–processed according to the profile and then sent to the Broker Agent. The Broker Agent consults the Query–Planning Agent to find Resource Agents that are capable of contributing to the query processing and forwards the MREF to those agents. Each Resource Agent translates the MREF into a query string that is specific to its metabase. If it encounters a term it cannot understand, it interacts with the Ontology Agent hoping to get an appropriate translation into a term it will be able to process. Once the query is formed, the Resource Agent executes it against the metabase, and returns the results (if any) to the Broker Agent, who in turn forwards them directly to the User Agent. The User Agent assembles the results by filtering and ordering them according to the user profile. It then displays the assets in the browser as shown in Figure 2.3.

2.3.1 Result Display

The User Agent servlet displays a result set in a frameset with three frames, that contain a list of titles, an applet to display the metadata, and some general information including a legend, information on running Swing applets in browsers etc.

The frame on the left–hand side lists the asset titles along with their rating values and icons that indicate the respective asset types. All assets are formatted as HTML links. If the asset is not an HTML page, most browsers allow to save it to the local file system, some plug–ins even support file viewing of common file types such as pdf or Quicktime. By using the browser's native capabilities we overcome an applet's security restrictions, which would not generally grant the necessary permissions to perform those functions.

Moving the mouse over a link causes the display applet in the frame on the right–hand side to display all the metadata that are available about the corresponding asset. In order to avoid multiple servlet requests, the applet contacts the User Agent

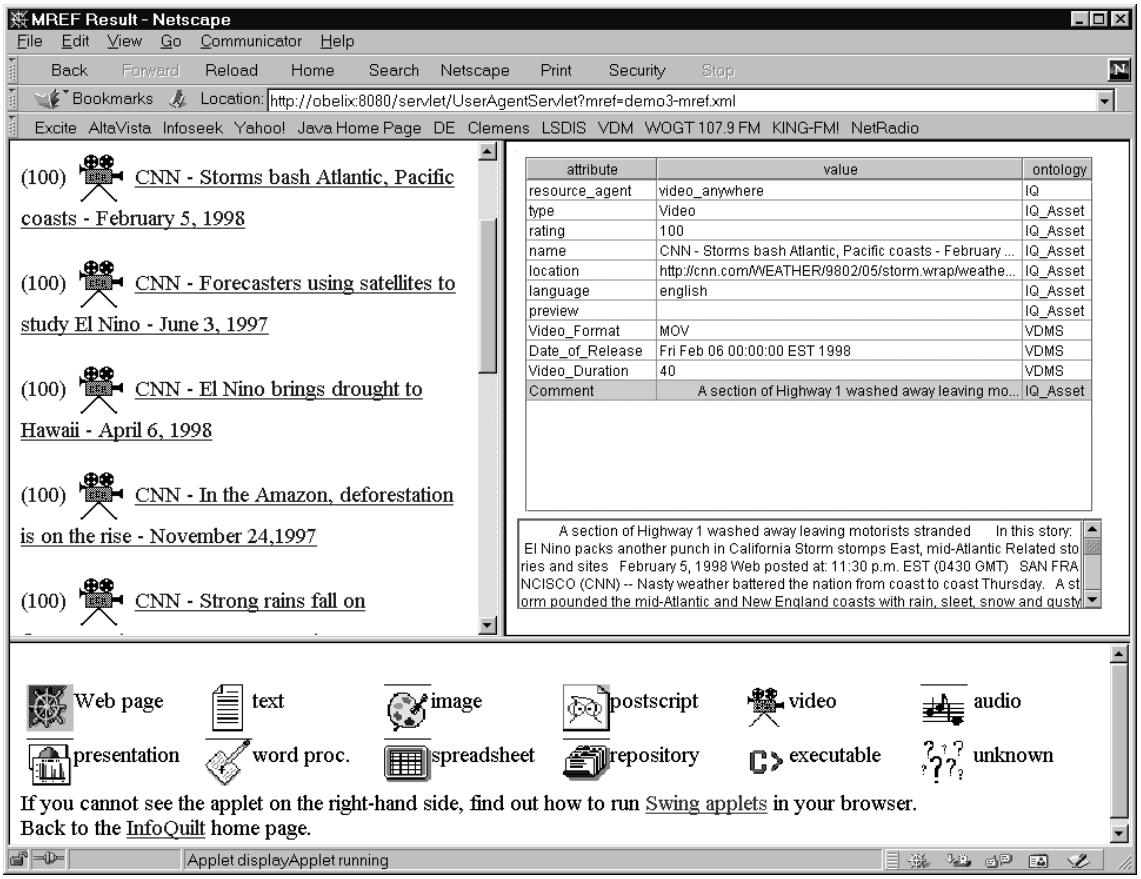

Figure 2.3: Result Display

only at initialization time and stores the required data internally. A scrollable text area below the metadata–table displays the whole attribute–value of the currently selected table row. This is especially helpful when the text is too long to be fully displayed in the table.

The info frame on the bottom contains a legend, useful information and links.

Chapter 3

Ontology Design

3.1 Motivation

The Metadata Reference Link (MREF) has been designed to correlate information across the Internet. While the common HTML HREF[1] establishes a simple correlation between information entities (in most cases, Web pages), it has some major shortcomings due to the static nature of a hard–coded link:

- The target page or asset might be taken off its server, and the relation between the source page and the target that the link used to represent breaks.
- If the name of the target changes, the HREF becomes invalid, too. Other than in the former case, the information target is still there and could be referenced, but the HREF is not able to update itself dynamically to maintain the link.
- A (common) HREF cannot lead to a collection of related information assets unless it is a link to a hand–crafted page that contains a number of other links that are again static and suffer from the above mentioned shortcomings.

On the other hand, MREF is dynamic in that the information it leads to is not a single, hard–coded Web page but rather a collection of related assets that is generated at runtime. It may therefore be considered an information request that can be expressed as a combination of keyword, attribute, and content–based search.

[1] that is, not a link to an executable script

Experience with common search engines shows that keyword search alone cannot guarantee that the results are semantically related because a) the words are indexed out of context, and b) single words can never capture semantics adequately. Thus, the power of MREF stems mainly from its attribute and content–based search capabilities. In both search types we describe properties (metadata) of certain entities by assigning values to attributes and look for those artifacts that meet the given requirements. For instance, we might look for information on red roses by describing the entity "rose" with the property "color=red". One must be careful, however, because the terms that describe entities can be ambiguous, e.g., a "map" means something different to a Geologist than to a tourist! In order to eliminate ambiguity, information beyond the names of the correlated terms is required. Ontologies provide such additional information.

According to [Gru], an ontology can be considered a vocabulary for representing and communicating knowledge about some topic plus a set of relations and constraints that hold among the terms in that vocabulary. For our purposes, the vocabulary terms model the entities and their metadata, which are used in the attribute and content search. The relations establish semantic correlations between entities within the same ontology as well as across ontology boundaries. For example, it is possible to talk about an "El Niño" — contained in the ontology "Climate" — and how it influenced the fish population — in the ontology "Organisms"— at the coast of Peru.

Experience with creating and managing a single huge ontology [CYC] suggests that this approach has not succeeded. Instead, multiple ontologies that are topic specific are much more likely to be the way of describing the universe. In some cases, even ontologies that cover narrow, well defined domains, are fairly large, and it takes a considerable amount of time and research to develop them. For this reason, it is highly desirable to make them reusable and accessible to other parties,

so that applications can integrate existing ontologies in their systems, building on what experts have painstakingly created. The Open Knowledge Base Connectivity standard [OKBC] and the OBSERVER project [OBSERVER] are two mostly complementary approaches to address the problem of interoperability between ontologies.

The use of ontologies in InfoQuilt is twofold. They are used to describe not only the entities we are interested in, but also the retrieved assets, which are of heterogeneous nature. One set of query result may contain Real Videos, WordPerfect documents, an MS Excel document, and bitmap images. The user agent must be able to distinguish between the different asset types to display them in an adequate manner and act appropriately when the user retrieves them. For instance, a word processing document could automatically launch the right application while most graphics will simply be loaded and displayed in the browser.

A classification of all possible assets not only supports this behavior, but has also additional advantages: Advanced attribute queries can be made — such as "retrieve only streaming videos", and the results can be displayed in groups of related asset types.

3.2 Design Considerations

Several steps have to be taken when designing ontologies for an application. This section describes the considerations and actual work that has been done for this thesis. In order to demonstrate how InfoQuilt in general and the MREF design and processing in particular work, we use the scenario "El Niño" for the following main reasons:

- A large number of heterogeneous assets on El Niño is available through the WWW, comprising video, audio, animation, and textual assets.

- Domain experts willing to cooperate with us (or at least share their opinions) are accessible on the campus, in newsgroups, and on mailing lists.

A major goal for the design of the needed ontologies has been to model the universe of discourse as scientific as possible within a reasonable time constraint. Consequently, more consideration has been given to the hierarchy of entities, their attributes, and inter–ontology links than to careful definitions of the terms and constraints, which would have required a substantially greater amount of research involving the coordination of domain experts to reach definitive ontological commitments. This thesis aims to show the use of ontologies to semantically correlate information on the Internet using MREFs. The example ontologies we have been created, can later be replaced by existing or more refined ontologies that meet the high standards of domain experts. The designer of a domain ontology has to be able to speak the language of the experts in that area. Beside the slew of information that is available through the WWW, the "Glossary of Meteorology" [Hus59], a standard reference among meteorologists, proved to be a rich fountain of domain knowledge. The major source of information, however, was the communication with domain experts: Professor Vernon Meentemeyer of the Climatology Research Lab of the Department of Geography at the University of Georgia, the newsgroup sci.geo.meteorology, and the mailing list met-ai@MCS.VUW.AC.NZ helped in the ontology design by providing valuable feedback.

Having acquired sufficient domain knowledge, it is possible to determine the number of needed ontologies. In our "El Niño" scenario we have to be able to talk about oceanic and atmospheric phenomena, climate and weather; it turned out later that storms, which are part of the weather, deserve an ontology on their own. In addition, we need an ontology of organisms to be able to talk about consequences El Niño has on animals.

The type of application determines the universe of discourse, which in turn determines the level of granularity. For instance, since we are only concerned about general characteristics of fish, but not their biological details, it is sufficient to keep the "Organism"–taxonomy at a very general level (see Section 3.5.5). The development of taxonomies is the central part of the ontology design[2]. Once all the entities of the universe are identified, they need to be arranged in subclass-of and instance-of relations. Entity attributes and other relations and constraints are then added as appropriate.

3.3 Design Problems

During the whole design process, the ontology is refined and, if necessary, restructured according to the feedback from the domain experts. However, problems arise if the experts have different answers to certain questions. For instance, on the question whether storms are part of the weather, the answers of meteorologists ranged from "True, and also part of the weather" and "Storms ARE weather" to "False, they generate weather". Most of those problems stem from differing definitions of meteorological terms, as there seems to be no definitive reference to which all experts subscribe. In our case, we tried to gather as many opinions as possible before deciding which ones to follow in the final versions of the taxonomies.

Although our interactions with many domain experts clearly showed how important precise definitions of the involved terms are, they also made clear that a taxonomy is only a part of the ontology as it cannot fully reflect the way terms are related to and constrained by other terms. For instance, one meteorologist suggested that "the definition of drought include the idea that 'drought is the repeated failure of EXPECTED rainfall"'. This is clearly not expressible by "is-a" relationships.

[2]That is true for most applications according to [JP98].

The necessary depth and precision of an ontology that most meteorologists would approve of is beyond the scope of this thesis. Rather, we demonstrate how existing ontologies (that might have been defined by domain experts) can be used in MREF and the whole InfoQuilt system. For that purpose we believe it is sufficient to use "toy ontologies", which are kept simple but still aim at resembling professional ontologies to a certain extent.

We were not able to find a taxonomy of atmospheric phenomena that all meteorologists agree upon; furthermore, some classification criteria cannot simply be taken as entity names because one usually prefers short names like "precipitation" of very few words over a whole sentence as in "Liquid or solid water particles formed and remaining suspended in the air." [Hus59] While it is often possible to formulate constraints for the entity definitions that model the original definition very well, it is still desirable for the entity names to capture as much of that definition as possible. Many meteorologists that we consulted about this problem were reluctant to suggest such names, nevertheless, they contributed some terms that we could use for our taxonomy.

3.4 Ontology Implementation

When deciding on a format to use for the implementation of the ontologies, it is important to consider the application for which the ontologies are going to be used. The Knowledge Interchange Format (KIF), a standard format for defining ontologies, is widely used among knowledge–based systems. For a first prototype of InfoQuilt, however, we have chosen the Resource Description Format Schema (RDF Schema) [BGE98] for the following reasons:

- The RDF Schema definition has a class concept and the property types subClassOf and instanceOf, enough to implement a class hierarchy.

- It allows to define other property types such as "affects" or "isPartOf", which serve as links between entities across ontologies.
- RDF is built upon XML, an evolving Internet standard language, which we use extensively in our InfoQuilt implementation.
- MREF itself (see chapter 4) is specified in RDF, hence it is very easy to reference entities that are defined in RDF–ontologies.

As long as only simple relations and descriptions, but no constraints are needed, RDF is fully sufficient. For instance, a hurricane is defined in RDF (in the "Storm" ontology) as

```
<rdfs:Class ID="Hurricane">
  <rdfs:subClassOf resource="#Tropical_Cyclone" />
  <rdfs:label xml:lang="en">Hurricane</rdfs:label>
  <rdfs:comment>12 <= wind speed on the Beaufort scale
      (Glossary of Meteorology, 1959)
  </rdfs:comment>
  <storm:area>North Atlantic, Caribbean Sea, Gulf of Mexico,
      off the west coast of Mexico
  </storm:area>
</rdfs:Class>
```

Once precise and more formal definitions of relations and constraints become more important, or if MREFs are created dynamically by a graphical ontology navigator, KIF will be the better choice. MREF does not require a certain format, as long as entities and their properties can be referenced. The referencing mechanism in MREF is described in detail in Section 4.3.

3.5 InfoQuilt and "El Niño" Ontologies

This section describes in detail the ontologies that are used for InfoQuilt in general and the "El Niño" scenario in particular.

In order to demonstrate the capabilities of MREF and InfoQuilt, we decided on a few sample MREFs that show how entities of different ontologies are semantically correlated in one compound information request (see Section 4.3 on MREF construction). Many reports describe the consequences of the ENSO (El Niño Southern Oscillation) on the weather, climate, and fauna of certain countries in videos, sound clips, images, and Web pages. MREFs that want to retrieve those documents use the following ontologies:

- Weather
- Climate
- Storm
- Atmospheric Phenomenon
- Organism
- IQ_Asset
- Video Data Management System (extension ontology)
- InfoQuilt

While the first five ontologies are domain specific, the last three deal with domain–independent information artifacts and terms that are used regardless of the scenario. We describe the domain specific ontologies first.

In the visualization of the ontologies we have adopted the Entity–Relationship diagram conventions, with the following additions: Solid links mean "subClassOf", dashed arrows denote the aggregate "isPartOf" relation, solid arrows mean "affects" with the arrow pointing towards the affected entity, and links that stand for other relations are annotated. Dashed entities are defined in another ontology.

3.5.1 Weather

The "Glossary of Meteorology" [GOM] defines weather as

> "1. The state of the atmosphere, mainly with respect to its effects upon life and human activities. As distinguished from climate, weather consists of the short–term (minutes to months) variations of the atmosphere. Popularly, weather is thought of in term of temperature, humidity, precipitation, cloudiness, brightness, visibility, and wind.
>
> 2. As used in the making of surface weather observations, a category of individual and combined atmospheric phenomena which must be drawn upon to describe the local atmospheric activity at the time of observation. [...]"

The weather ontology (see Figure 3.1) accounts for both aspects of this definition by assigning the attributes mentioned in the first part to the entity "weather" and making the whole class of atmospheric phenomena a part of the weather. Storms and weather phenomena like droughts are also part of the weather. Meteorologists disagree on where to insert a drought and how the obvious link to precipitation can be modeled. Some experts reason that a drought is not really a phenomenon per se, but rather the lack of a particular phenomenon. Others have raised concerns about the assumption that precipitation (or rather the lack of it) causes a drought, arguing that "flames do not cause fire, they are fire" or that the *expectation* of rainfall plays a major role in the definition of a drought. However, many experts agreed to our model, which lead us finally to use it for our prototype.

3.5.2 Climate

Climate can be defined as the "long–term manifestations of weather" or the "statistical collective of the weather conditions of a particular region during a specified

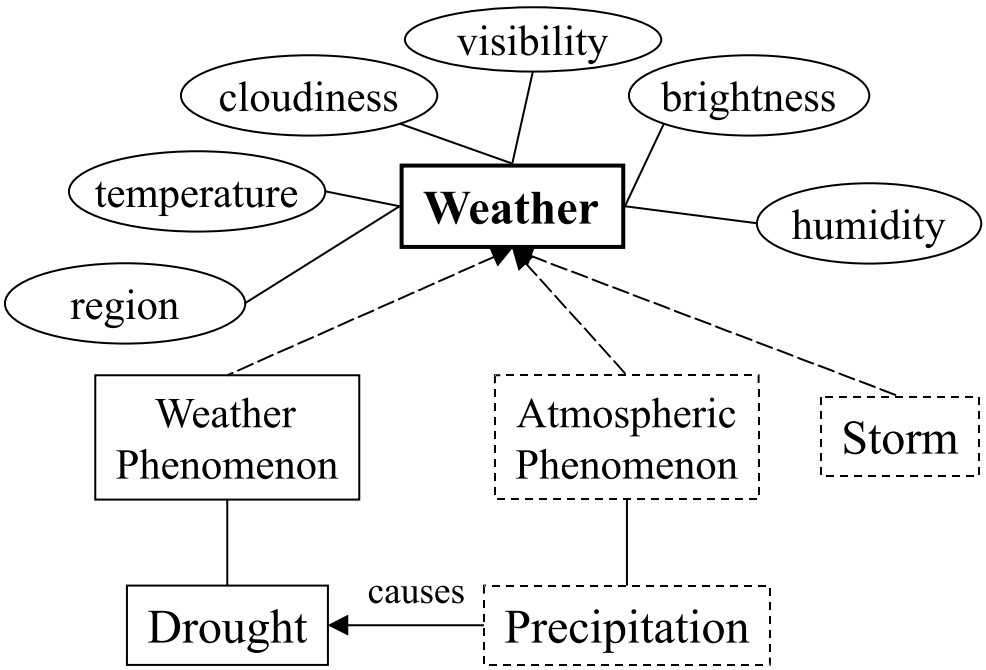

Figure 3.1: Weather Ontology

interval of time (usually decades)." [Hus59] As such, the weather affects the climate. Some meteorologists argue that the climate can in turn affect the local weather by setting certain boundaries for it. Our climate ontology (see Figure 3.2) has therefore a double–headed arrow between weather and climate. Many weather attributes and phenomena apply to the climate as well, but they refer to the statistical pattern rather than local instances. The same applies to storms, which are part of both weather and the climate.

El Niño is, according to the official El Niño theme page, "a disruption of the ocean-atmosphere system in the tropical Pacific having important consequences for weather around the globe" [NOA]. It occurs about every seven years and is sometimes followed by its counterpart La Niña. Together they form the so-called El Niño Southern Oscillation (ENSO). Many domain experts classify ENSO as an oceanic

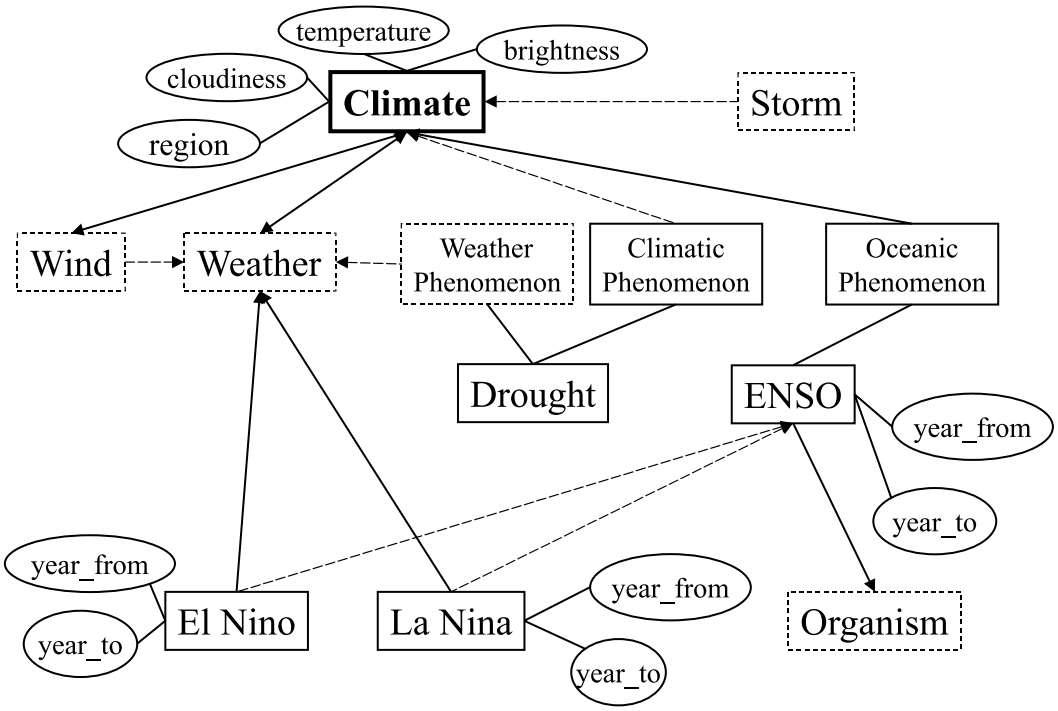

Figure 3.2: Climate Ontology

phenomenon that has a strong impact on the weather and, since it occurs regularly, on the climate as well.

3.5.3 Storms

Storms (see Figure 3.3) are part of the weather and the climate. The Glossary [Hus59] defines them as "any disturbed state of the atmosphere, especially as affecting the earth's surface, and strongly implying destructive or otherwise unpleasant weather". Three types of storms are of interest to us, namely tropical cyclones, thunderstorms, and tornadoes. The latter two ones sometimes count as atmospheric phenomena, too. Lightning as well as the thunder that it causes[3] are

[3]Thunder itself is not considered an atmospheric phenomenon.

essential to thunderstorms. Both tornadoes and tropical cyclones are atmospheric vortices, but are otherwise very different from each other. For instance, tornadoes are usually several hundred *meters* in diameter and occur only over the continent, while tropical cyclones originate only over tropical oceans and are often several hundred *kilometers* wide [Lan98].

Tropical cyclones are categorized according to intensity and region. Following a classification scheme that is widely used in the United States[4], there are *tropical depressions* with wind less than or equal to Beaufort force 6, *tropical storms* with wind stronger than 6 and less than 12, and *hurricanes* and *typhoons* with even stronger wind. The only difference between a hurricane and a typhoon is the region in which they occur; hurricanes originate in the North Atlantic, the Carribean Sea, the Gulf of Mexico, or off the west coast of Mexico, while people speak of typhoons in the western North Pacific and most of the South Pacific [Hus59].

3.5.4 Atmospheric Phenomena

When we want to model an MREF that leads to information on how El Niño affects the precipitation in Peru or drought periods in the western Pacific, it is necessary to develop an ontology that contains those phenomena (see Figure 3.4). Having identified the four main groups *Lithometeor*, *Hydrometeor*, *Luminous Meteor*, and *Igneous Meteor*, the main task was to classify the hydrometeors other than precipitation into meaningful categories (we mentioned the naming problem in Section 3.3). Note that clouds are an exception[5]: although they are hydrometeors, they are not considered atmospheric phenomena.

[4] the requirements *wrt.* isobars are omitted here

[5] and are hence shown in italics

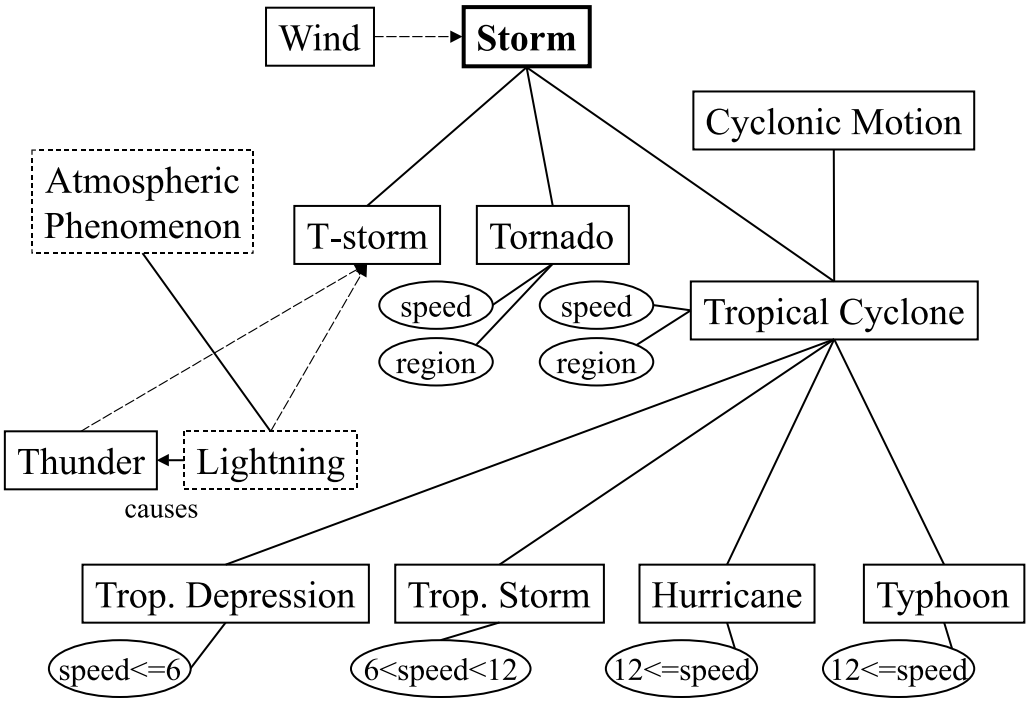

Figure 3.3: Storm Ontology

3.5.5 Organisms

In order to reference plants and animals, we developed a very general and small ontology that contains only the types of organisms that we need for our scenario, including fish and different types of grains (see Figure 3.5).

3.5.6 InfoQuilt Ontology

The InfoQuilt ontology defines only a few entities that are needed for MREF query and result processing, such as a *Resource Agent* and various other query related terms, and a set of logical connectors that are used for both entity descriptions and the MREF description (see Chapter 4 on the MREF specification). This ontology currently serves mainly as an XML namespace to make element names unambiguous and has therefore no graphical representation.

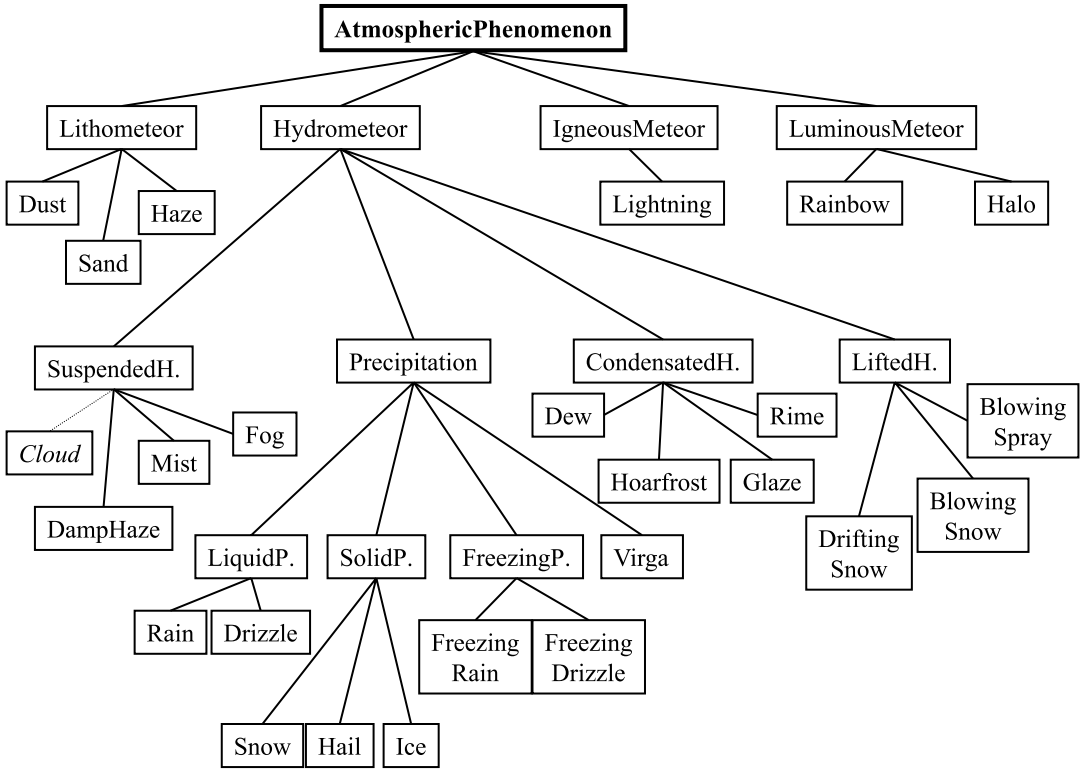

Figure 3.4: Atmospheric Phenomena Ontology

3.5.7 IQ_Asset Ontology

Our asset taxonomy does not resemble the only conceivable data classification schema; some applications might require a finer classification, some might not need certain distinctions such as "binary" vs. "ASCII", or the distinction between executables and documents might be much more important and would be made on a higher level of abstraction. The proposed ontology has been designed to support the way InfoQuilt queries and handles heterogeneous data. The more important a distinction, the higher its level of abstraction.

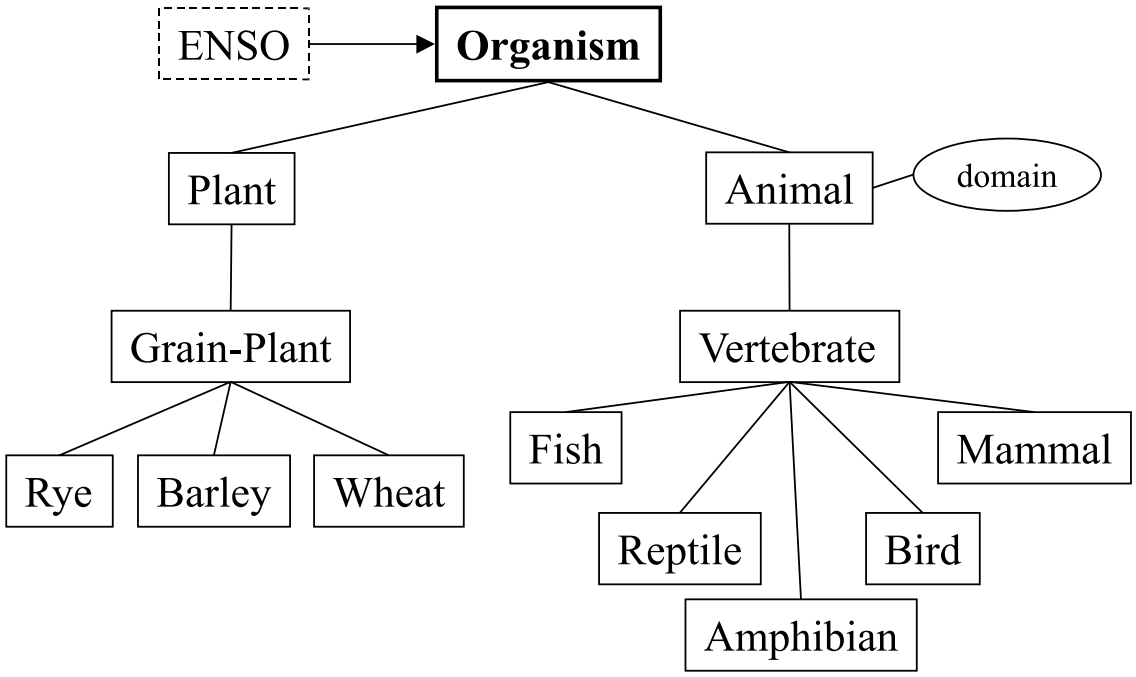

Figure 3.5: Organisms Ontology

How coarse-/close-grained should the ontology be?

One important ontology design issue is how detailed the ontology should be. That question has two sides to it:

1. *How coarse–grained?* As outlined above, InfoQuilt handles many types of heterogeneous media, therefore the ontology should for instance be able to tell streaming videos from C++ code in order to be useful to the user. In particular, we found it useful to distinguish between

   - different types of documents such as presentations, videos, images, and spreadsheets;
   - binary and textual (ASCII) documents;
   - different types of markup documents such as LaTeX, XML, and HTML;
   - executables and non–executables.

2. *How fine–grained?* The ontology should not become less useful because one major application is thrown out of the market or another important media type is invented. Rather, it should be coarse–grain (or abstract) enough to remain useful despite such events. Another thing to consider is that different data resources classify their assets in different ways; keeping the leaves of the asset classification at a certain high level of abstraction simplifies the correlation of their metadata schema with the InfoQuilt asset schema. Resources are free to detail one or more leaves with their own ontology (see Section 3.5.7 on extending ontologies). For example, *VideoAnywhere* classifies videos according to their respective sources — the Web, TV, or the home collection. A video server might distinguish between videos solely based on their formats. Instead of modeling complicated relationships between multiple viewpoints, InfoQuilt simply talks about "videos" at the leaf level and lets different resources add their additional attributes by supplying their namespaces.

The final decisions about which asset types to consider leaf nodes made a compromise between necessary specialization and simplification of ontology correlations. While the simplification is meant to help in the translation of the MREF into an actual query, a certain level of specialization is necessary to formulate meaningful queries without deploying possibly resource–dependent terminology. For instance, in an attribute query we want to talk about assets of different media types (audio, video, text, ...). While we cannot assume that all metadata resources support the distinction between vector and bitmap graphics, it is much more likely that they distinguish between images and textual data.

In particular, the following media types are distinguished: binary executables, audio files, videos, images, word processing documents, spreadsheets, presentations,

databases, archives, source code, scripts, TEX files, MREFs, XML documents, and HTML documents.

The IQ_Asset Ontology in Detail: Terms and attributes

This section explains the classification schema and describes the terms that are used if the IQ_Asset ontology (see Figure 3.6).

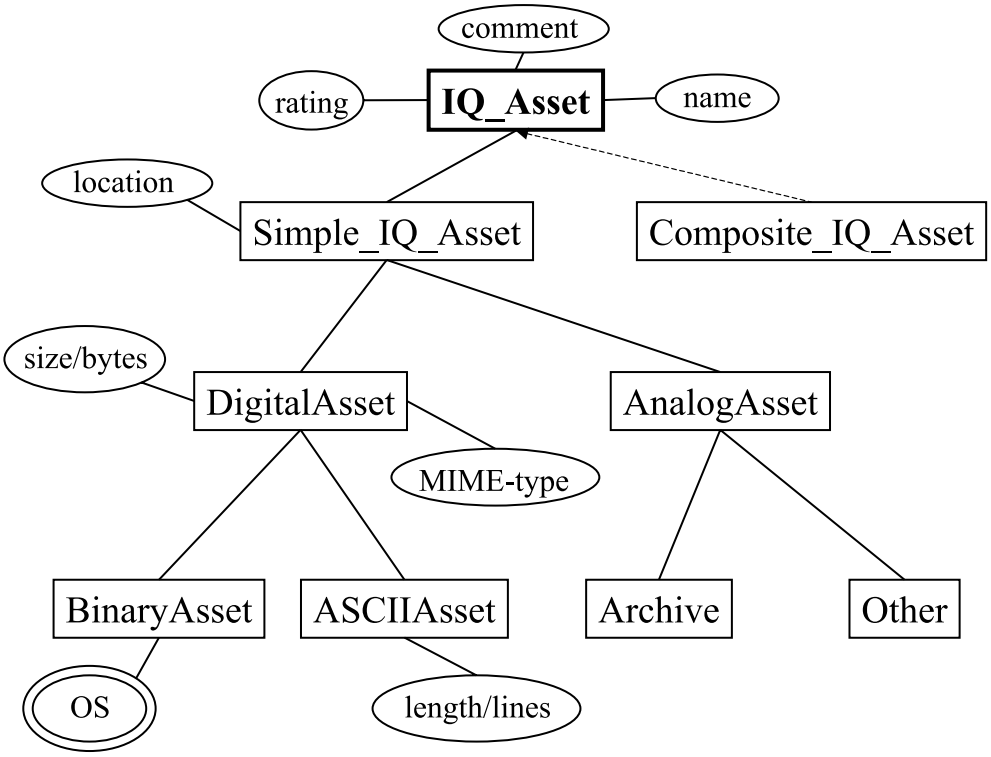

Figure 3.6: IQ_Asset Ontology (1)

The class "IQ_Asset" is the abstract superclass of "Simple_IQ_Asset" and "Composite_IQ_Asset". Composite_IQ_Assets are collections of related IQ_Assets, e.g. a Web page and the images within that page. Collections of collections are also conceivable, for instance a whole Web site. A Simple_IQ_Asset contains anything that

is retrievable in some way, from digital audio files to common hardcover books. All IQ_Assets have the following attributes in common:

- *name:* a distinguishing text, for instance a book title
- *comment:* a descriptive, human-readable text, maybe much longer than the name
- *rating:* a number between 0 and 1 intended to signify how closely this asset matches the MREF description

Simple_IQ_Assets have the additional attribute

- *location:* a description of the location of this asset.

Since we are most concerned with digital media retrievable through the WWW and are less interested in "analog" assets, the distinction between digital and non-digital assets is the top–most classification criterion. Digital assets are (in most cases) assigned a MIME–type [RFC] that helps the user agent to determine appropriate actions when the user clicks on a link to a particular asset. Furthermore, they have a certain size that can be measured in bytes[6]. Digital assets are divided into binary and textual/ASCII assets.

Binary assets (see Figure 3.7) are often restricted to a few operating systems; for instance, an executable that runs on Solaris is of little use to a user who runs Windows95. Information on the required operating system(s) is therefore valuable metadata. There are three types of binary assets: executables, documents, and repositories. Repositories are collections of data (databases) and assets (ZIP files) in one or more files. This distinguishes them from documents like audio files or MS

[6]The reason that the size is not an attribute of IQ_Asset is that there is no clear understanding of what "size" means for non-digital media, or how it can be measured in a reasonable way that is unique for all such assets.

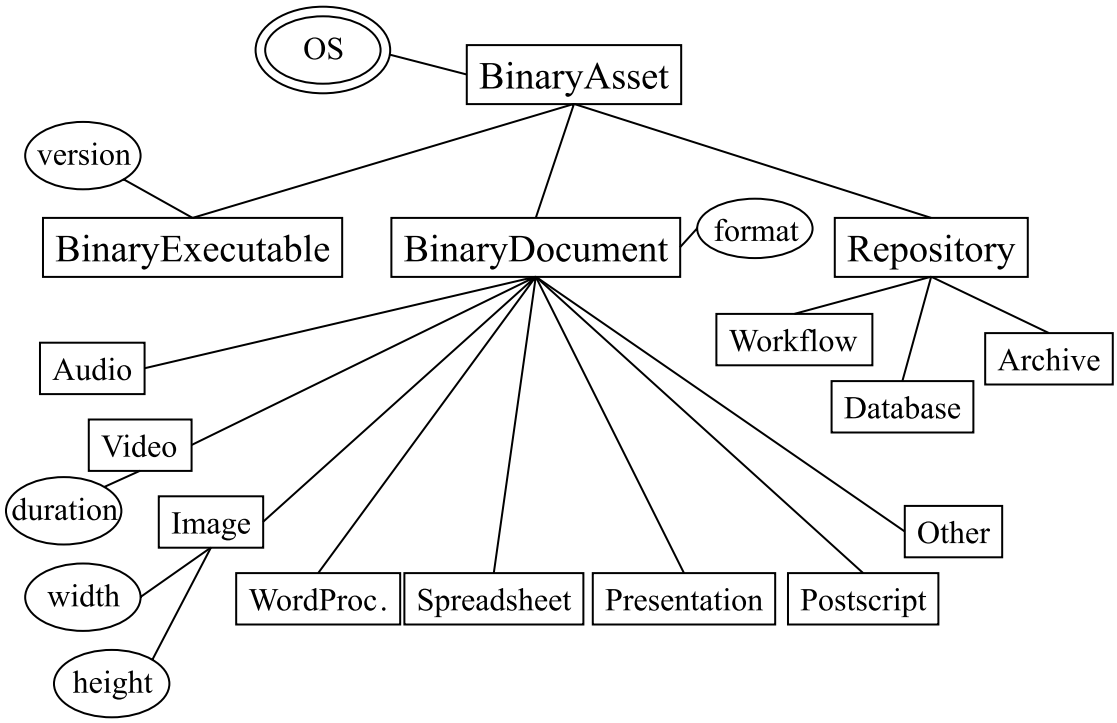

Figure 3.7: IQ_Asset Ontology (2)

PowerPoint presentations. We define documents to be non–executable, hence the third category of executables.

Documents are further classified into audio files, video files, images, pdf and postscript documents, word processing documents, spreadsheets, presentations, and other documents. While a more detailed specialization would be possible — for example, images can be divided into vector and bitmap graphics — we stop at this point for the reasons mentioned in Section 3.5.7. However, content–independent attributes such as height and width for images, and duration for videos are associated with the respective document types.

While binary files are usually not understandable for the human reader, textual — or ASCII — assets are. In addition to their size, which is measured in bytes,

the attribute length in lines makes sense, too. In general, ASCII assets (see Figure 3.8) are not bound to an operating system, with the exception of the interpretation of some special formatting characters like linefeed or carriage-return. ASCII assets are unstructured, semi–structured, or structured. Examples for unstructured ASCII assets are simple text documents and emails. The term "structured" in the IQ_Asset ontology describes documents such as ASCII–databases, which store for instance one record per line and use a vertical bar to separate the fields. Many assets exhibit some internal structure, but not as strict as found in the mentioned databases. They are commonly referred to as semi–structured assets. By far the most of those assets are either code or use some markup language. Code files belong to a certain language and are classified as scripts if they are executable (for instance, Perl scripts), or otherwise as source.

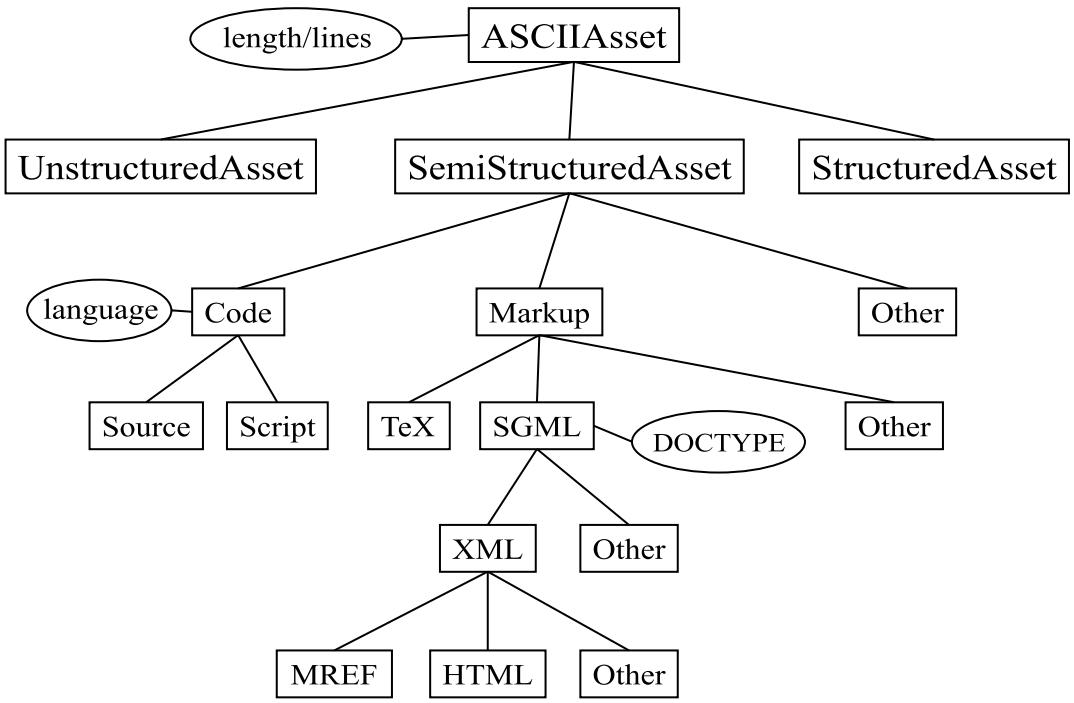

Figure 3.8: IQ_Asset Ontology (3)

Most markup documents belong to the SGML document family. XML is a subset of SGML with MREFs and HTML pages as its most prominent examples. Other markup documents include TEX and LATEX assets.

Extension Ontologies

Often, metadata resources will have information available that is more detailed than the general IQ_Asset ontology can and should be (for the above mentioned reasons). However, this additional information can be captured and displayed using extension ontologies. Those ontologies are referred to by a Uniform Resource Identifier (URI) and detail a particular IQ asset. One example is the Asset hierarchy of *VideoAnywhere*, shown in Figure 3.9.

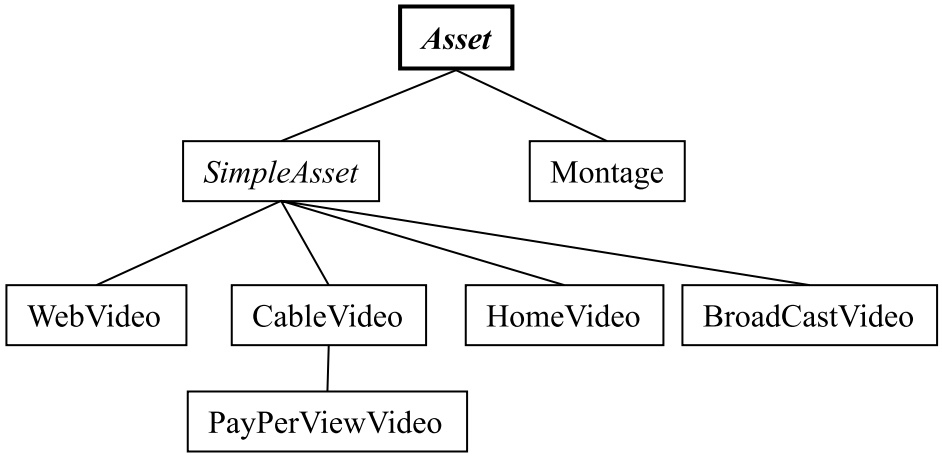

Figure 3.9: *VideoAnywhere* Extension Ontology

The user agent, upon reading attribute values that pertain to such an extension ontology, displays that information in a suitable manner (see Section 2.1.1 on the User Agent). In general, extension ontologies are considered not only when displaying asset metadata, but also when creating an MREF. However, the part of an

MREF referring to that extension ontology can only be resolved by resource agents that "know" or "understand" this ontology by means of correlating terms between that ontology and local resource ontologies; otherwise, the agents will have to ignore the respective XML elements.

Chapter 4

The Metadata REFerence Link (MREF)

4.1 MREF and HREF

A *hypertext* reference (HREF) in HTML is a Uniform Resource Identifier (URI), specifying which resource should be displayed by the browser when the user follows it. A *metadata* reference (MREF) is similar in principle, but goes beyond it. When the user clicks on an HTML link containing the MREF, the browser displays a set of assets that are specified by the MREF. With this behavior in mind, an MREF can be seen as a description of each of the assets in the result set.

4.2 RDF, XML, and XML Namespaces

RDF has been designed to "define a mechanism for describing resources that makes no assumptions about a particular application domain, nor defines (a priori) the semantics of any application domain." [LE98] We think of an MREF as a description of assets, or, in RDF terminology, "resources", of interest. That is the main reason why RDF has been chosen as the framework for constructing MREFs. RDF uses XML as its underlying syntactic model, thus relying on a standardized language that is likely to replace HTML in the near future.

XML [XML98a] is a subset of the Standard Generalized Markup Language (SGML); it retains most of SGML's expressive power while being much less complicated. XML ensures interoperability on a syntactic level, not only because of

its similarity to the widely–used HTML, but also because it provides standards for many encoding issues such as internationalization and character sets.

At the core of RDF are so–called "descriptions" that describe resources in terms of their possible property types. For instance, a Web page is a resource, and "author" and "date of creation" are property types that could be associated with that Web page. The attribute–search part of an MREF is constructed by means of descriptions of involved entities taken from different domain ontologies by specifying values for their attributes (a complete example is explained below).

The RDF Schema Specification allows to specify schemata in terms of entities and properties that are defined for a given set of so-called "classes". It partially uses the object–oriented paradigm but is property–centric rather than class–centric in that properties are defined "in terms of the classes they may connect" [BGE98]. This methodology can be used to easily define and refer to domain ontologies. Although KIF [GF92] is the prevailing format for defining ontologies, we found it useful to deploy RDF instead for its simplicity and ease of use when referring to entities and attributes (properties) of the defined ontologies. A later version of InfoQuilt may migrate to KIF without substantial changes in the MREF specification. The ontologies that we have used are described in more detail in Section 3.5; their RDF definitions can be found in the appendix.

XML namespaces enable the re–use of predefined XML elements in multiple software applications. Rather than redefining often–used elements (such as "person" or "book") it makes more sense to store such element definitions in a common place and refer to them from the applications that use them. Furthermore, a very important achievement of namespaces is that they avoid ambiguity of attribute or element names within an XML document by explicitly mentioning the respective namespace together with attributes and elements. If no namespace identifier precedes

an element or attribute name, then a default namespace is assumed. Consider the following example of an airline reservation [XML98b]:

```
<RESERVATION>
  <NAME HTML:CLASS="largeSansSerif">Layman, A</NAME>
  <SEAT CLASS="Y" HTML:CLASS="largeMonotype">33B</SEAT>
  <DEPARTURE>1997-05-24T07:55:00+1</DEPARTURE>
</RESERVATION>
```

Here the "Class" attribute refers one time to the seat class of the plane and otherwise to font classes that are relevant to the display of the element. If the respective namespaces (here: "HTML" and the default namespace for reservations) were not used in this example, it would be hard to correctly process the above reservation element. In InfoQuilt, namespaces are used to uniquely identify the domain ontologies of the described entities and properties.

4.3 MREF Structure

Informally, each MREF consists of three parts[1]:

1. Prefix
2. Entity Descriptions
3. MREF Description

4.3.1 Prefix

Following the RDF syntax, an MREF starts with the `<rdf:RDF>` tag, which encloses all descriptions. The ontologies that are going to be used in the descriptions are specified in the opening tag of the `rdf:RDF` element[2]. They take the

[1] A formal specification is given in EBNF in Appendix A.

[2] Note: Although namespaces could in theory be introduced in any element, it makes the MREF easier to understand when all used namespaces are defined in one place. By

form `xmlns:IQ_Asset="http://infoquilt.com/iqasset.xml#"` where the identifier after "xmlns:" introduces a so–called "prefix", a short convenience abbreviation for the usually long URI that identifies the source of the namespace. In the above example, an RDF would accept the namespace prefix `IQ_Asset` as an abbreviation for `http://infoquilt.com/iqasset.xml#`, and an element named `IQ_Asset:Video` would refer to the resource `http://infoquilt.com/iqasset.xml#Video`. At the time of this writing, there is no standard for naming such a URI. It is highly desirable that the URI be unique, but the URI need not always help to retrieve an actual schema [XML98b]. In InfoQuilt, however, the references to the domain ontologies do point to the RDF schema definitions, with the exception of the "RDF" and "RDFS" URIs, which are verbally taken from the examples of the defining documents [BGE98] and [LE98].

4.3.2 Entity Descriptions

Attribute and content queries can be regarded as a description of the resulting assets in terms of their properties and relations. In many cases that means specifying values for an entity's attributes. The set of possible attributes consists of the attributes that the entity inherits from its superclass(es) plus the ones defined for the entity itself. If necessary, comparison operators and units can be specified as well.

Relationships to other entities are obtained by following the different links that emerge from either the entity itself or one of its superclasses (see chapter 3 on Ontology Design). Thus, it is possible to describe logical relationships between different entities. If a description of entity involves such a reference to another entity, two cases are distinguished:

---

choosing the root element of the MREF for this purpose we ensure that all namespaces are visible throughout the whole MREF.

a) The referenced entity is not further described. In this case, it can be referenced using the `rdf:resource` attribute with the value being a URI of that entity. For instance, when talking about rain in Peru, one can write

```
<rdf:Description
     about="http://infoquilt.com/weather.xml#weather"
     id="wea1">
  <weather:region>Peru</weather:region>
  <part
     rdf:resource="http://infoquilt.com/atm_phenomena.xml#rain"
     />
</rdf:Description>
```

Here, we recall that there is an attribute `region` associated with the entity `weather` in the `weather` ontology (see Figure 3.1). There is also a link `isPartOf` from the entity `rain` (or rather its superclass `precipitation`) that is followed using the `part` element.

b) Some attributes of the referenced entity must be specified. In this case, there must be either a separate description of that entity and the referring entity refers to the description's unique id, or that entity is described in an inner description. Extending the above example, if we want to talk about El Niño and its consequences on the rainfall in Peru, we could do so by writing

```
<rdf:Description
     about="http://infoquilt.com/climate.xml#El_Nino"
     id="elnino">
  <climate:affects rdf:resource="#wea1" />
</rdf:Description>
```

Alternatively, it is possible to use nested descriptions:

```
<rdf:Description
     about="http://infoquilt.com/climate.xml#El_Nino"
     id="elnino">
  <climate:affects>
    <rdf:Description
         about="http://infoquilt.com/weather.xml#weather"
         id="wea1">
      <weather:region>Peru</weather:region>
      <part rdf:resource=
        "http://infoquilt.com/atm_phenomena.xml#rain" />
     </rdf:Description>
  </climate:affects>
</rdf:Description>
```

By default, all the attributes and relations in a description are considered to be connected by the logical operator `AND`. However, it is sometimes desirable to describe *alternative* attribute values. For instance, one could be looking for red, yellow, and white roses. While it is possible to use three different descriptions and appropriate boolean connectors in the MREF–Description, it is more intuitive to use just one description specifying alternative attribute values. There are two possible syntactic ways of denoting alternatives: one is using RDF's aggregate element `Alt`, the other is to use elements that model boolean connectors (in prefix notation). Since those boolean elements are deployed for the MREF–Description, we found it consistent to stick to this way of expressing alternatives throughout the whole MREF. Thus, the above mentioned query for different types of roses would look like this:

```
<rdf:Description
     about="http://infoquilt.com/organisms.xml#Rose"
     id="rose">
  <OR>
    <organisms:color>red</organisms:color>
    <organisms:color>yellow</organisms:color>
    <organisms:color>white</organisms:color>
  </OR>
</rdf:Description>
```

The following guidelines apply to the MREF design:

- all XML elements and attributes are by default case sensitive.
- The format `<namespace-name>:<attribute-name>` implicitly requires that attribute names be unique within one ontology. If that were not the case, the MREF processor would have to perform costly checks to find out which attribute was actually referenced in the MREF.
- Every description must have a (unique) id for referencing purposes. Although RDF supports nested descriptions and forward–references, our current implementation does not support them, so the MREF constructor has to make sure that every entity that is referenced has already been described earlier.
- Even if an inner description might not be referenced later on, it is recommended that *every* description is assigned an id regardless, so that a later redesign of the MREF can be done more easily; also, RDF–parsers will not have to use additional internal identifiers to deal with separate entities.
- If a logical expression is used in an attribute description, the `AND` operators should be explicitly mentioned in order to enhance readability.

4.3.3 MREF Description

The actual MREF description follows the entity descriptions. Technically, it does not have to be the last description of the whole MREF, but since it references the entity descriptions and we do not allow forward–referencing at this point, it has to be put at the end. As mentioned at the beginning of this chapter, the MREF logically describes the assets that the user wants to retrieve. Therefore, it always describes the entity `IQ_Asset` of the IQ_Asset ontology.

Having described all the involved entities in terms of their attributes and relations, it remains to logically connect those descriptions to complete the attribute and/or content-based query. The current specification of the MREF allows the logical connectors `AND`, `OR`, and `NOT`, which are applied in a prefix–like notation. For instance, the boolean expression `(NOT a) OR (b AND c)` is written in XML as:

```
<OR>
  <NOT>a</NOT>
  <AND>
    b
    c
  </AND>
</OR>
```

Entities are references using the `IQ:entity` element with the reference as the value of its `rdf:resource` attribute. The whole attribute query is embedded in the `attribute-query` element.

Content–based queries are written like attribute queries, but using the element name `content-query` instead.

Keyword queries are made of `terms` that are logically connected like entities in the attribute queries. The use of prefix notation and XML elegantly eliminates the need for modeling parentheses and quotes explicitly:

- Every term that contains more than one word will be enclosed in double quotes by the resource agents.
- Parentheses are implicitly expressed by the prefix notation.

The complete search expression is embedded in the `keyword-query` element.

The (up to) three queries — keyword, attribute, and content query — are run separately by the Resource Agents, and the result sets are merged later. Thus, it is possible to assign weights to each query that are considered when the final result set

is calculated. These weights take a value between 0 and 1 and are specified as values of the respective `weight` attributes. Attribute queries lead in general to more precise results while keyword queries will often have better recall values but significantly lower precision. MREF designers will therefore tend to assign higher weights to attribute–search results than to keyword–query results. The default weight is 1.

Many search engines return a rating value that is intended to tell how well the result matches the keywords. The Resource Agents map those ratings to values ranging from 0 to 1, so that results from different search engines are better comparable[3]. The optional `threshold` attribute allows to discard all query results with a rating of less than the specified value. Considering the very different rating strategies of the various search engines, this threshold should mainly be used to filter out very low ranking results, e.g., below 40%. The default threshold is 0, that is, all results are retrieved.

4.3.4 MREF Example

To clarify the above specification, let us go through a comprehensive MREF example. The MREF represents a request for short broadcast videos about El Niño and its consequences on sea animals. Here is the complete MREF, followed by block–by–block comments:

```
(00) <!-- How does El Nino affect sea animals?
(01)      Look for broadcast videos of less than 2 min
(02) -->
(03) <rdf:RDF
(04)   xmlns="http://infoquilt.com/iq.xml#"
(05)   xmlns:IQ_Asset="http://infoquilt.com/iqasset.xml#"
(06)   xmlns:climate="http://infoquilt.com/climate.xml#"
(07)   xmlns:rdf="http://www.w3.org/TR/WD-rdf-syntax#"
(08)   xmlns:organisms="http://infoquilt.com/organisms.xml#"
(09)   xmlns:VDMS="http://infoquilt.com/video_anywhere/vdms-rdf.xml#">
```

[3]The issue how to compare search result values across different search techniques and rating strategies is a matter of continuing research. For more information see [GP98]

```
(10)
(11)   <rdf:Description
(12)       about="http://infoquilt.com/organisms.xml#Animal"
(13)       ID="Animal">
(14)     <organisms:domain>sea</organisms:domain>
(15)   </rdf:Description>
(16)
(17)   <rdf:Description
(18)       about="http://infoquilt.com/climate.xml#El_Nino"
(19)       ID="elnino">
(20)     <climate:affects rdf:resource="#Animal" />
(21)   </rdf:Description>
(22)
(23)   <rdf:Description
(24)       about="http://infoquilt.com/iqasset.xml#Video"
(25)       ID="video">
(26)     <IQ_Asset:duration op="lt" unit="seconds">120
(27)     </IQ_Asset:duration>
(28)     <VDMS:type>BroadcastVideo</VDMS:type>
(29)   </rdf:Description>
(30)
(31)   <rdf:Description
(32)       about="http://infoquilt.com/iqasset.xml#IQ_Asset"
(33)       ID="MREF_0">
(34)     <attribute-query weight="0.8">
(35)       <AND>
(36)         <entity rdf:resource="#elnino" />
(37)         <entity rdf:resource="#video" />
(38)       </AND>
(39)     </attribute-query>
(40)     <keyword-query weight="0.5" threshold="0.8">
(41)       <!-- "El Nino" AND (fish OR sea OR penguin) -->
(42)       <AND>
(43)         <term>El Nino</term>
(44)         <OR>
(45)           <term>fish</term>
(46)           <term>sea</term>
(47)           <term>penguin</term>
(48)         </OR>
(49)       </AND>
(50)     </keyword-query>
(51)   </rdf:Description>
(52)
(53) </rdf:RDF>
```

Lines	Description
00-02	There should always be a comment preceding the RDF element, not only for maintenance reasons, but also for providing a quick understanding of what the MREF is about.
03-09	The root element contains the namespace declarations for the used ontologies. This example uses the ontologies InfoQuilt (default namespace), IQ_Asset (`IQ_Asset`), Climate (`climate`), RDF (`rdf`), Organisms (`organisms`), and *VideoAnywhere* (`VDMS`).
11-15	Sea animals are identified by referring to the entity "Animal" in the Organisms ontology and specifying the value "sea" for their attribute "domain". The whole description can later be referenced by assigning the id "Animal" to it.
17-21	To ask for the consequences that El Niño has on sea animals, the entity "El_Nino" in the Climate ontology is described with the relation "affects" that links "El_Nino" with the Organisms ontology. Since relations hold for the class including all subclasses (unless otherwise specified in the ontology), it holds in this example for all organisms, including "animal". Here, the target of the "affects" link is the sea animal description above, hence the value for the `rdf:resource` is the id of that description. Note that it is necessary to specify the RDF namespace in the resource attribute, because according to the XML namespace convention, the namespace of the element (InfoQuilt) is the default namespace for its attributes.

Lines	Description
23-29	In this description we detail which properties our resulting videos should have. For that purpose, we specify attribute values of the entity "Video" in our IQ_Asset ontology. In this case, the duration of the video has to be less than 120 seconds, hence the comparison operator `op="lt"` and — to avoid ambiguities — the unit `unit="seconds"`. The type "Broadcast Video" is unknown to the generic IQ_Asset ontology (and probably many local metabase ontologies). We therefore use an extension ontology, in this case the *VideoAnywhere* ontology that classifies video assets according to their source (Web, Broadcast, TV, Home). Those resource agents who understand this ontology will return only broadcast videos, others will deal with the `VDMS`–entries in an appropriate manner.
31-51	The final MREF description is the heart of the whole query. It contains both the attribute and keyword query (content–based queries are not yet implemented), which are considered to describe the `IQ_Asset` entity of the IQ_Asset ontology. The attribute query is assigned a higher weight than the keyword query, giving results of the former priority over those of the latter.
34-39	The attribute query connects the above specified descriptions of El Niño and the video by referencing their id's and linking them with the logical operator `AND`.

[3]One reasonable reaction is to disregard the unknown element and to lower a local confidence value, which is used to calculate the overall rating of an asset.

Lines	Description
40-50	The keyword query complements the attribute query by specifying search terms that resemble (parts of) the intended query. In this MREF, we chose the query expression `"El Nino" AND (fish OR sea OR penguin)`. The threshold value of 0.8 means that only results of at least 80% should be returned. The MREF designer is advised to include the complete search expression as a comment to make the MREF easier to read.

4.4 Embedding MREF in HTML

Like HREF, MREF is embedded in HTML anchors, which makes the difference between the two transparent to the user. The link itself points to the User Agent servlet; the MREF location is passed as a parameter using the HTTP GET method, for example

```
<HTML><BODY>
Find information on El Nino and its consequences on sea animals
<A HREF="http://infoquilt.com/UserAgentServlet?mref=elninosea.mref">
here!</A>
</BODY></HTML>
```

Chapter 5

An Example Metabase: *VideoAnywhere*

VideoAnywhere is a system that searches and manages distributed heterogeneous video assets. It can be seen as a vertical slice of InfoQuilt that is exclusively concerned with video assets. Originally built as a home–based system that stores home, Web, and TV assets, it can — with minor modifications — also be deployed as a video search engine for the Web alone. We describe here only the metabase and related parts. A complete overview can be found in [BSS98].

The object–oriented metabase (POET 5.1) stores a wealth of video metadata that are organized according to the video classification shown in Figure 3.9. Because the system was originally intended to be used by a home user, the set of metadata does not include content–dependent attributes such as "camera angle" or "zoom", but rather "actors", "producer", "time of event", "review", and the like. The complete set of metadata is listed in Appendix B.

In order to keep the metabase up–to–date, the Resource Agent and the Encapsulator Agent cooperate to add new assets to the metabase and remove expired assets from the metabase. To find new assets, the Encapsulator Agent[1] has to go to the known Web sites to collect the metadata that describe the available assets. However, a number of issues and challenges arise in the gathering of metadata for the assets:

- Each video provider structures the metadata in a different way.

[1] This agent performs to a great extend the same functions as the original Video Content Agent.

- The amount of provided metadata varies widely.
- Different sites have different naming conventions.
- The structure of a particular Web site may change over time.
- New providers come on the market with interesting video offers.

The Encapsulator Agent handles these issues in the following ways. It maintains a list of extractors that are specialized for each Web site that provides video metadata. The extractors analyze the Web pages that belong to the respective Web site and send the metadata in an XML stream to the Resource Agent, which creates a new asset according to the XML elements read from the stream.

The Resource Agent uses an internal conversion table to map the XML tags used at the asset source to the tags that are used in the metabase. This need not be a complete mapping: some source tags and system tags might turn out not to be used. The former case does not constitute a problem, in the latter case default values must be assumed, for instance, if no value for the "color" attribute is given then it is assumed that the respective video is colored.

As of now, the developer of the extraction software has to provide the mapping. For that reason the tags that *VideoAnywhere* uses need to be made publicly accessible. In a later version, the Resource Agent will consult the Ontology Agent to do the mapping automatically.

New content providers who would like to advertise their assets can develop new extractors (and mappings if necessary) for their Web sites and register with the Encapsulator Agent.

As long as the contents of a Web page do not change, there is no need to re–run the extractors on that video source. Since the update check is site–specific, the Encapsulator Agent routinely asks the extractors whether the sites have been

updated before it tells them to extract new data. For that purpose, extractors are capable of fast–checking whether a new extraction would yield any new results. Only if that is the case, a complete extraction (of the new assets) is performed.

No matter how carefully an extractor is designed, it will have to be adjusted if the Web site changes its structure. In many cases these corrections are minor, but they can take considerable effort if the site undergoes a major change. In our prototype, we had to re–write the Foxnews extractor twice. In the immediate future, possible ways to catch changes early and quickly re–write extractors include

- on the developer's side:
  - writing extractors so that they alert the VideoAnywhere administrator when they encounter a non–recoverable problem.
  - developing an ExtractorToolkit that provides powerful graphical tools to quickly extract metadata from any Web site. "Extraction by Example" is one method we believe is worth investigating further. Such a toolkit would also enable Web site owners to develop their own extractors and make them available to Web portals through the *VideoAnywhere* system.
- on the Web administrator's side:
  - informing the developers of extractors whenever the site is about to change.

Obviously, human interaction is needed to keep such an infrastructure up and running. But this is also true for the very successful Web portal Yahoo, which hires a host of librarians to catalog and assess Web pages. Junglee, a company that was recently bought by Amazon.com, provided comparison shopping across multiple

online shopping providers through a Web portal site. They also deployed extractors that gathered data on various goods that were then displayed at the buyer's convenience.

With the advent of XML, however, the need for specialized extractors may become obsolete. Video content providers can structure their metadata using XML. The Resource Description Format and XML namespaces provide a useful and standardized framework for that purpose. A single robot could then crawl the Web and accomplish what multiple extractors are doing now. We are in the process of specifying an XML and RDF based framework (not discussed here for brevity) that can support automatic extraction of content from cooperating content providers, obviating the need for human generated or customized extractors in the future.

For the El Niño scenario we developed an extractor that is capable of extracting results from the CNN–Mediasearch engine. The extractor runs a sample query with the keywords "El Nino" against CNN's video archive and analyzes the results. Further extractors for other news sites and movies have been developed in the context of the *VideoAnywhere* project, which is funded by the private industry; however, they contribute only marginally to our scenario.

Chapter 6

Conclusions and Future Work

A prototype for InfoQuilt with the ontologies discussed in Chapter 3 and a multi–agent system [Par98] has been implemented. We have written a number of MREFs for our "El Niño" scenario, which can be found and tested on the InfoQuilt home page at `http://lsdis.cs.uga.edu/proj/iq/iq.html`.

The current implementation was aimed at providing the framework for a system that effectively performs the semantic correlation of information across multiple ontologies and data models as outlined in this thesis. However, some of the components that are necessary to provide this functionality are merely implemented as "dummies" that future work on this project can extend:

- The Ontology Agent will make use of the *OBSERVER subsystem*, which has not yet been integrated.
- The Query–Planning Agent will be enhanced to perform sophisticated *query planning and optimization.*
- One agent type is likely to be added to the current infrastructure to further enhance the query processing part: A *Correlation Agent* can aid in assembling the results before they are sent back to the requesting User Agent. This agent would deal with the problem of how to compare asset rating values that come from different Resource Agents.

- To speed up the process of writing Encapsulator Agents, an *ExtractorToolkit* is being created, which supports the rapid development of extractors for various resources.

- It has been a research issue how *user profiles* for such a powerful and heterogeneous system can look like. In the case of *VideoAnywhere* the domain was fairly limited so that it made sense to include asset–specific attributes like "actor" or "channel" in the profile. However, it is not easily possible to account for such specifics when dealing with *all* types of assets, whose attributes vary widely; and if extension ontologies are taken into consideration, even more come into play. While it is clearly desirable to deploy user profiles in general, one has to decide on a more or less generic set of properties that can be personalized. For instance, domain–independent preferences like the minimum rating of returned assets or the ordering algorithm ("by rating", "by asset type", a.s.o.) are rather easy to manage. The storage and maintenance of user profiles will be done on a central server, managed by an autonomous Profile Manager (see Section 2.2).

- At this point, there exist only pre–defined MREFs that the user can take advantage of. Such MREFs can be very useful — especially when they are designed by domain experts, so that we entertain the idea of building an *MREF–repository* that can be queried — possibly using MREFs again! At times, the user will want to create his or her own MREF. For that purpose, a *graphical user interface* can greatly simplify the development by letting the user navigate through a set of ontologies that the user has subscribed to. The tool allows to specify the desired entities, their properties and relations to others on a higher level, so that the casual user won't have to know the details of RDF, XML, or MREF at all.

In this thesis, we showed how the InfoQuilt system overcomes the limitation of today's simple HTML links and search engines and provides the Internet world with a mechanism that allows to search for and retrieve assets on a semantic level. This is achieved by

- deploying domain–specific as well as domain–independent ontologies that relate semantically "near" entities,
- combining keyword–, attribute–, and content–based search in one query description, the *Metadata Reference Link*, and
- using a multi–agent system that intelligently manages and accesses distributed heterogeneous resources.

Bibliography

[ACHK] Y. Arens, C. Y. Chee, C. Hsu, and C. Knoblock. Retrieving and integrating data from multiple information sources. to appear in the International Journal on Intelligent and Cooperative Information Systems.

[BBB+97] R. Bayardo, W. Bohrer, R. Brice, A. Cichocki, G. Fowler, A. Helal, V. Kashyap, T. Ksiezyk, G. Martin, M. Nodine, M. Rashid, M. Rusinkiewicz, R. Shea, C. Unnikrishnan, A. Unruh, and D. Woelk. Semantic integration of information in open and dynamic environments. In *Proceedings of the 1997 ACM International Conference on the Management of Data (SIGMOD)*, Tucson, Arizona, May 1997.

[BGE98] D. Brickley, A. V. Guha, and A. Layman (Eds.). Resource Description Framework (RDF) Schema Specification. http://www.w3.org/TR/1998/WD-rdf-schema/, August 1998. World Wide Web Consortium Working Draft.

[BSS98] C. Bertram, A. Sheth, and K. Shah. VideoAnywhere: A system for searching and managing distributed heterogeneous video assets. submitted for publication, 1998.

[CGMH+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. In *Proceedings of the 10th IPSJ Conference*, Tokyo, Japan, October 1994.

[CYC] CYC. http://www.cyc.com.

[DEFS98] S. Decker, M. Erdmann, D. Fensel, and R. Studer. How to use ontobroker. In *Proceedings of KAW'98*, Alberta, Canada, 1998.

[FLM95] T. Finin, Y. Labrou, and J. Mayfield. KQML as an agent communication language. In Jeff Bradshaw, editor, *Software Agents*. MIT Press, Cambridge, to appear (1995). invited chapter.

[GF92] M. R. Genesereth and R. E. Fikes. Knowledge interchange format, version 3.0, reference manual. Technical report, Computer Science Dept., Stanford University, 1992.

[GP98] L. Gravano and Y. Papakonstantinou. Mediating and metasearching on the internet. *IEEE Data Engineering Bulletin*, 21(2):28–36, June 1998.

[Gru] T. Gruber. What is an ontology? http://www-ksl-svc.stanford.edu:5915/doc/frame-editor/what-is-an-ontology.html.

[HS98] M. N. Huhns and M. P. Singh. *Readings in AGENTS*. Morgan Kaufmann Publishers, San Francisco, California, 1998.

[Hus59] R. E. Huschke, editor. *Glossary of Meteorology*. American Meteorological Society, Boston, Massachusetts, 1959.

[ISW] InfoSleuth Home Page. http://www.mcc.com/projects/infosleuth.

[JP98] D. M. Jones and R. C. Paton. Some problems in the formal representation of hierarchical knowledge. In N. Guarino, editor, *Formal Ontology in Information Systems*, pages 135–147, Padova, Italy, 1998. IOS Press.

[Lan98] Ch. W. Landsea. FAQ: Hurricanes, typhoons, and tropical cyclones. http://www.aoml.noaa.gov/hrd/tcfaq/tcfaqA.html, 1998.

[LE98] O. Lassila and R. Swick (Eds.). Resource Description Framework (RDF) Model and Syntax Specification. http://www.w3.org/TR/1998/WD-rdf-syntax-19980819, August 1998. W3C Working Draft.

[Mag] Magnifi. CNN mediasearch. http://cnn.com/SEARCH/media.

[MIKS98] E. Mena, A. Illarramendi, V. Kashyap, and A. Sheth. OBSERVER: An approach for query processing in global information systems based on interoperation across pre-existing ontologies. *Distributed and Parallel Databases Journal*, pages 1–49, 1998.

[NOA] NOAA. El Nino theme page. http://www.pmel.noaa.gov/toga-tao/el-nino/nino-home.html. National Oceanic and Atmospheric Administration.

[Par98] K. Parasuraman. InfoQuilt agent architecture. Master's thesis, University of Georgia, Athens, 1998.

[Pro] Informedia Digital Video Project. http://www.informedia.cs.cmu.edu.

[RFC] Multipurpose Internet Mail Extensions, RFC 2045-2049. http://www.oac.uci.edu/indiv/ehood/MIME/2046/rfc2046.html.

[Sco] Scour.Net. http://www.scour.net.

[She98] Amit Sheth. *Changing Focus on Interoperability in Information Systems: From System, Syntax, Structure to Semantics*, chapter 0, pages

1–22. Kluwer, 1998. M. F. Goodchild, M. J. Egenhofer, R. Fegeas, C. A. Kottman (eds.).

[SIMW] SIMS Home Page. http://www.isi.edu/sims.

[SK98] A. Sheth and W. Klas, editors. *Multimedia Data Managing, Using Metadata to Integrate and Apply Digital Media.* McGraw-Hill, March 1998.

[SS98] K. Shah and A. Sheth. Logical information modeling of web-accessible heterogeneous digital assets. In *Proc. Of the Forum on Research and Technology Advances in Digital Libraries*, pages 266–275, Santa Barbara, CA, 1998.

[TSIW] TSIMMIS Home Page. http://www-db.stanford.edu/tsimmis.

[Vir] Virage. http://www.virage.com.

[XML98a] Extensible markup language (XML) 1.0. http://www.w3.org/TR/REC-xml, February 1998. W3C Recommendation.

[XML98b] Namespaces in XML. http://www.w3.org/TR/WD-xml-names, August 1998. T. Bray, D. Hollander, A. Layman Eds., World Wide Web Consortium Working Draft.

Appendix A

Formal Specification of MREF

Some of the following definitions are taken from [LE98] and [XML98b]. A unit name can be any standard unit like "bytes" or "seconds". The required whitespaces between the declaration parts have been omitted to enhance readability.

```
         MREF ::= '<rdf:RDF' nsDecl* '>'
                  entityDescr* mrefDescr
                  '</rdf:RDF>'

       nsDecl ::= prefixDef '="' URI-reference '"'

    prefixDef ::= 'xmlns' (':' nsName)?

       nsName ::= (any legal XML namespace prefix)

URI-reference ::= (see RFC1738, RFC1808, [URI])

  entityDescr ::= '<rdf:Description' aboutAttr idAttr '>'
                  (relDescr | attrDescr)+
                  '</rdf:Description>' |
                  '<rdf:Description' resourceAttr idAttr '>'

    aboutAttr ::= 'about="' URI-reference '"'

 resourceAttr ::= 'resource="' URI-reference '"'

       idAttr ::= 'ID="' IDsymbol '"'

     IDsymbol ::= (any legal XML name symbol)

     relDescr ::= '<' relName resourceAttr '>'
```

```
   attrDescr ::= '<' attrName operatorExpr? unitExpr? '>'
                 logicalExpr
                 '</' attrName '>'

     relName ::= (prefix ':')? qName

    attrName ::= (prefix ':')? qName

       qName ::= (nsName ':')? name /* qualified name */

        name ::= (any legal XML name symbol)

operatorExpr ::= 'op="' operator '"'

    operator ::= 'lt' | 'le' | 'ge' | 'gt'

    unitExpr ::= 'unit="' name '"'

   mrefDescr ::= '<rdf:Description about=
                    "http://infoquilt.com/iqasset.xml#IQ_Asset"
                 idAttr '>'
                 keywordQuery? attrQuery? contentQuery?
                 '</rdf:Description>'

keywordQuery ::= '<keyword-query' weightExpr? threshExpr?>
                 expression
                 '</keyword-query>'

   attrQuery ::= '<attribute-query' weightExpr? '>'
                 expression
                 '</attribute-query>'

contentQuery ::= '<content-query' weightExpr? '>'
                 expression
                 '</content-query>'

  weightExpr ::= 'weight="' number '"'

  threshExpr ::= 'threshold=' number

      number ::= (any valid number expression)

  expression ::= logicalExpr | primExpr
```

```
logicalExpr ::= '<AND>' expression+ '</AND>' |
                '<OR>'  expression+ '</OR>' |
                '<NOT>' expression  '</NOT>'

   primExpr ::= keywordTerm | attrTerm |
                contentTerm | valueTerm

keywordTerm ::= '<term>' name+ '</term>'

   attrTerm ::= '<entity' resourceAttr '>'

contentTerm ::= (to be defined in a later version)

  valueTerm ::= name | number
```

Appendix B

VideoAnywhere Metadata

VideoAnywhere uses the following attributes (ordered by asset types in the hierarchy):

Asset	
assetID	unique descriptor for this asset, assigned by the metabase
timeOfCreation	the time when this asset was registered with the metabase
title	... of the asset
contents	a searchable description of this asset length
length	... of the whole asset
SimpleAsset	
color	black/white or colored
category, subcategory	classification of this asset according to a fixed schema
producer	the studio, company, maker, ... of this video
director	the name of the director
actors	a list of actors and actresses
rating	the TV or Movie rating, depending on the asset type
reviews	a number of reviews (e.g., by Siskel & Ebert, New York Times, ...)
format	the video format (AVI, MPG, MOV, ...)

dateOfRelease	the date when this video was released, applies especially to movies
dateOfVideoCreation	the time of recording
expirationDate	the time when this video is to be deleted from the metabase; the main reason for this is that URLs may change; applies especially to news clips
annotations	close captions or user defined additional information
BroadcastVideo	
location	the URL from where the clip can be downloaded
timeOfEvent	the time or date when the event that this clip is about actually happened
HomeVideo	
location	a description where to find this asset
WebVideo	
location	the URL from where the video can be downloaded
CableVideo	
channel	the channel that broadcasts this asset
PayPerViewVideo	
price	the price for receiving this particular program