Improving Access to Environmental Data using Context Information

Anthony Tomasic
INRIA Rocquencourt
Anthony.Tomasic@inria.fr*

Eric Simon
INRIA Rocquencourt
SRAE - Ministère de l'Environnement
Eric.Simon@inria.fr

Abstract

A very large number of data sources on environment, energy, and natural resources are available worldwide. Unfortunately, users usually face several problems when they want to search and use environmental information. In this paper, we analyze these problems. We describe a conceptual analysis of the four major tasks in the production of environmental data, from the technology point of view, and describe the organization of the data that results from these tasks. We then discuss the notion of metainformation and outline an architecture for environmental data systems that formally models metadata and addresses some of the major problems faced by users.

1 Introduction

Over the last years, governments have recognized that environmental information could have a profound impact on our ability to protect our environment, manage natural resources, prevent and respond to disasters, and ensure sustainable development. All these issues emphasize the need to circulate and exchange information and also to combine information across different disciplines. Unfortunately, when users want to search and use environmental information, the following problems occur: (1) Data do not exist or are insufficient; sometimes this may require synthesis or reproduction of data. (2) Data is not referenced by data suppliers and therefore hard to locate, or data is referenced under specific classification criteria that are domain-specific. (3) Data is hard to access: either private, or of a too high cost, or requiring costly pre-processing (e.g., data must be re-entered manually from paper documentation) or format translation, or administrative procedures to acquire data are too long, etc. (4) Accessed data sets are hard to use because they are inconsistent or non-compatible (e.g. access to long time series but standard data collection techniques have not been applied, thereby making adjacent time series not compatible). This may entail detailed data identification such that corrections can be made (either in-house or by the data supplier); however, such data identification is often not present. (5) The quality of retrieved data is hard to assess (accuracy, "first-hand" versus derived, timeliness, etc). It is often hard to compare data produced using different scientific models because of a lack of documentation about the underlying computational process.

In this paper, we analyze some of these problems. Section 2 describes our conceptual analysis. Section 3 discusses the notion of metainformation and Section 4 outlines an architecture for scientific data systems that formally models metadata and addresses the above problems. Section 5 concludes the paper.

2 Conceptual Analysis

We distinguish between three main categories of users based on the data each user needs from an environmental data system [6].

End Users (e.g., general public, policy-maker) need to locate and extract data that matches their interest, or appropriate data servers to retrieve data of the desired level of quality. For example, Joe accesses the rating of beaches in his town. Then, he asks why his town is not considered a safe beach. As a result, he gets a definition of a safe beach that is understandable to him, i.e., at the appropriate level of detail. For instance, safety may be defined as a collection of criteria such as the expected height of waves, and presence of sharks. Then Joe may want to find out who, and when, collected the data about the presence of sharks near his beaches.

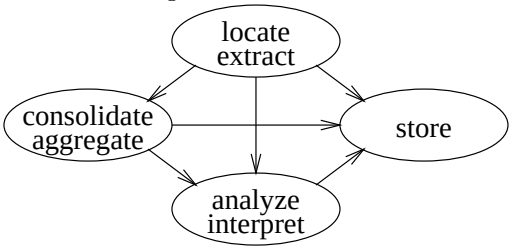
Brokers (e.g., environmental scientist, public administration) construct the servers for end users. For instance, Alice writes programs which read measurement databases, administrative enquiries, and geographical databases, to construct a map of France that indicates the quality of beaches. Also,

*INRIA Rocquencourt, 78153 Le Chesnay, France; http://rodin.inria.fr/.

she writes programs to improve the reliability of data using consolidation techniques. Generally, Alice must find the data required for each new program she writes. In addition, each new program uses multiple data sources. Each data source requires a unique program to extract the data for the new program.

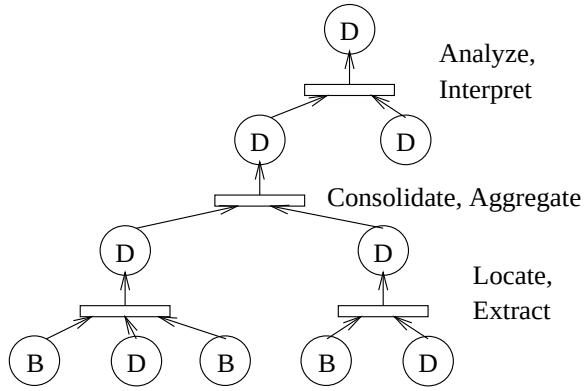
Data Providers (e.g., biologist, geologist) collect data and want to distribute them as widely and as easily as possible. For instance, Bob may manually enter his data to an existing database through a standard form-based entry system. Data can also be collected using automatic sensors that directly transmit their data to an associated system. In this case, Bob has to verify the quality of data and eliminate erroneous measurements. To do this, Bob needs to use specific programs for data analysis and interpretation and access other data systems for comparing his data with other related data.

In general, any individual in the real world may play the role of multiple generic users described here. From a task point of view, Joe's principal activity is to locate data servers and extract relevant data. Alice's and Bob's principal activity is to generate data by consolidation, aggregation, analysis and interpretation. These activities are interrelated, as shown in this figure:



Each arrow in the figure indicates a precedence relationship between the tasks. The tasks are defined as follows: (1) to *locate* and efficiently *extract* relevant and accurate information from a possibly very large number of information sources; (2) to *consolidate* data by creating new sets of data (e.g., approximate missing data in raw data sources); to aggregate data at a higher level of abstraction (e.g. aggregate measurement data into larger measurements); (3) to *analyze* and *interpret* data using simulation models and other complex analytical tools, thereby generating new "value-added" data; and (4) to *store* data that is either defined and supplied by data providers, or produced as a result of the three previous tasks.

Each iteration around the four tasks produces new data that is available for further iterations. The iterations naturally organize the data into a graph based on the tasks used to produce the data, as shown in this figure:



Circles represent data and thin boxes, or transitions, represent the application of tasks. Directed arcs represent the transformational flow of data. On the bottom, base and derived data (circles marked by "B" and "D" respectively) are the input of two transitions implementing locate and extract tasks. The result of these transitions is stored as derived data, which in turn are the input of a transition implementing a consolidate and aggregate task.

3 Data vs. Metadata

The notion of metainformation is used in many different areas with a similar general goal: enable better data integration, interchange, access, and interpretation [4]. However, there is not a clear definition of what metainformation is, and its interpretation is application-domain dependent. In fact, a specific piece of information is data or metadata only with respect to a specific application-domain definition. Before precisely defining what metainformation is in this paper, we propose to distinguish two levels of description of environmental data based on semantic considerations.

3.1 Data Characterization

We describe our work with respect to a common framework – first order logic [5]. Every first-order logic consists of a syntax (a first order theory) and a semantics (an interpretation of the theory). To clarify the issues in this paper, we restrict ourselves to *facts*, that is, simple first-order logic formula consisting of a single ground predicate. Note that every fact has its corresponding predicate.

In the last ten years, several *standards* have been developed to establish the most general definitions about the syntax and semantics of environmental data objects (e.g., measurements, observations, maps, regulations, high-level environmental indicators, etc.) on which a very large group of environmental people agree. In our framework, each standard provides a collection of predicates and abstract data types used to model specific environmental data

objects, a physical implementation of these predicates and abstract data types (usually called a data exchange format), and a thesaurus to define the vocabulary used for abstract data type names and instances. For instance, the Spatial Data Transfer Standard (SDTS) provides a model for spatial objects (e.g., lines, points, polygons) [2]. These standards are primarily intended to facilitate the locate and extract task mentioned before: a set of common API interfaces that return standardized data can be defined over data sets, and common exchange data formats facilitate data transfer.

The French Sandre standard [1] provides a model for various kinds of measurements in the area of continental water resource management, such as hydro-biological samplings. For example, a simplified fact in this standard is

```
HB-sample(11.10.96, 18:00, Paris1, DDASS, ...)
```

`Paris1` is the name of the sampling station, and `DDASS` is the name of the institution that performed the measurement. The predicate `HB-sample` has the following structure

```
HB-sample(date,time,location,observer,val)
```

where `date`, `time`, `location`, `observer` and `val` are all abstract data types whose interpretation are defined in a thesaurus. For instance, `time` is the starting time of the sampling operation, and `val` is an encoding of the measurement of various biological constituents (which we do not list).

However, the abstract data types and the thesauri used to describe these facts are generally not sufficient to enable the consolidation, analysis, and interpretation of data. For instance, the interpretation of a sampling by an hydro-biologist requires the description of *contextual information* such as the measurement network in which the sampling takes place, the method used to perform the sampling, or the participant who did the sampling. Each of these contextual descriptions may entail the definition of new predicates and abstract data types, e.g., an `observer` predicate that describes the attributes of the entity that performed the observation. Each observer attribute value of `HB-sample` fact would then be a reference to an `observer` fact.

We call these added predicates and abstract data types *contextual*.[1] Note that the distinction between predicates for basic environmental data objects and predicates for contextual data is purely semantic and is indistinguishable from a syntactic point of view.

[1] Since the `observer` predicate is, in some sense, data *about* other data, some people label as metadata the facts of this predicate. In our view, it is not, it is simply more data.

The definition of contextual information is an end-to-end process. Data undergo various stages of processing, as shown in the previous section, with contextual information being appended at each stage. For instance, a broker may analyze and interpret data sets, and provide a contextual description of the interpretation that is understandable by an end user. Thus, we have a collection of data which we can classify as basic or contextual. We have the corresponding predicates and abstract data types to the data, or the *schema*. Schema predicates are then also basic or contextual.

In many current environmental systems, data is stored in files and contextual information is stored as free text in associated files (that is, contextual information does not have any formal language). Extraction of data consists of *ad-hoc* programs (that is, query languages are not used). Contextual information for the results of analysis is described in scientific papers. Thus, there is no formal language for describing the data itself and their meaning. The UDK project [3] provides an interesting framework for the definition of classes to describe both basic and contextual environmental data.

3.2 Metadata Characterization

What we call *metadata* in this paper is a *second-order* logic. A second-order logic is simply a logic with a theory and interpretation that describes a logic.

The notion of second-order logic is well-known in databases. For instance, metarelations are used to describe the schema of user-defined database relations (data dictionary). Similarly, metaclasses in object-oriented databases are used to describe the structure of user-defined classes and the methods to create and manage them. We can show a diagram for the metadata:

Meta-Level	Basic & Contextual
Meta-Data	a
Meta-Schema	b

For the table, instances of metadata a are schemas at the data level and b are metapredicates. (The distinction between basic and contextual is blurred here.) For instance, one can define the metapredicates `raw_measurements` and `estimated_measurements`, and have the following facts:

```
raw_measurements(HB-sample)
estimated_measurements(river-throughput)
```

where `HB-sample` is the predicate name defined before and `river-throughput` is another predicate name.

Metapredicates enable end users to locate useful data sources without having to examine the con-

tents of those data sources, which is typically a time-consuming activity. Determining the appropriate meta predicates associated with a data source is a modeling issue. The choice of metapredicates depends on which information about predicates can be effective in helping end users for locating, extracting, analyzing, etc. data sources. Metapredicates also permit views and constraints to be defined over predicates. For instance, one could define a metapredicate `HQ-estimated-measurements` that contains references of all predicates having "high quality" estimated measurements.

In any actual standard for data representation or system, many modeling short-cuts are taken for various reasons. One short-cut mixes in a single fact parts of facts from the data and parts of facts from the metadata. We call such facts *demi-metadata*.

For example, consider an extension of `HB-sample` that includes an attribute `normalized`.

```
HB-sample(date,time,location,observer,val,
          normalized)
```

The instances of normalized are the set of *attributes* that contain normalized values. Thus, an above data instance is

```
HB-sample(11.10.96,18:00,Paris1,DDASS,...,
          {date})
```

indicating that only the `date` value, `11.10.96` is normalized. Note that the normalized values can vary from fact to fact.

3.3 Contexts and Dyads

As explained before, the metapredicates necessary to understand the meaning of a data source depend on the user point of view. Thus, we introduce the notion of a *context* associated with a particular data source. A context is a set of metapredicate names (and the associated thesauri) that models the minimum required contextual information to enable an unambiguous interpretation of the data set by a group of users who share some common knowledge. Thus, typically, a context includes all the metapredicates that characterize the derivation of a given data set, such as metapredicates that describe the scientific hypothesis underlying the models.

Note that a *context* is a formal object. As such, it does not easily lend itself to location or comprehension by scientists. Thus, in addition to a context, each instance of data, schema, metadata or context can be *paired* with a piece of text that provides a natural language interpretation. We call each such pair a *dyad*. In particular, the dyads that pair metadata instances with text are very useful for searching over components of the architecture.

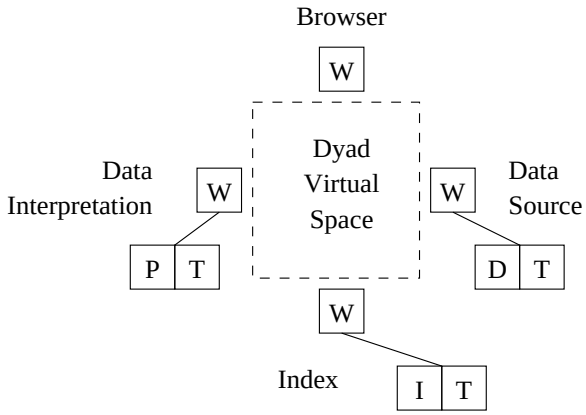


Figure 1: The proposed architecture. W stands for wrapper, D for data, T for text, I for index, and P for program.

4 A Mediator-based Architecture

We propose an architecture that directly supports the process-oriented view of data in Section 2 and the metadata framework of Section 3. Figure 1 shows a diagram of our proposed architecture. The architecture consists of four classes of components: browsers, data sources, indexes, and data interpretation programs.

Browsers are the user interface to the system. The user can view data from data sources, search for objects in the system through the indexes and launch programs for data interpretation. Data sources, such as a database system, export data and metadata via query services. The information in the data sources includes the metapredicates necessary to define the interrelationship between data sources as described in Section 2. Indexes provide searching over all free text, data and metadata. Data interpretation programs comprise scientific models used in the consolidation, aggregation, analysis and interpretation.

All components are interconnected via the dyad virtual space, which is a collection of protocols. Each component interfaces to the system through a wrapper [7]. Each object in the virtual space is a dyad consisting of (i) free text and (ii) the formal object that represents the data, metadata, context, index or computation. The free text provides a means (through the indexes) for locating the corresponding objects.

The functionality of wrappers varies widely, depending on its purpose. Browser wrappers understand the structure of dyads and support browsing of the free text and formal objects contained in them. Browsing formal objects is supported by the invocation of data visualization programs. In addition, the browser supports the query interface of the data

sources and indexes, and it supports requests for wrappers to translate queries and responses between the dyad virtual space and data sources. The index wrappers provide the information retrieval queries and also web-crawling technology for the construction of indexes. Data interpretation browsers manage the invocation of programs.

Returning to the example in Section 2, we describe each part of Figure 1. A database of historical wave data, including wave height, force of a wave, etc. for every beach in France, is a data source. This data is generated by a government authority and exported via a wrapper. Meteorological data is another data source, also produced by a government authority and also exported via a wrapper. Both of these data sources are managed by Bob Data Provider. Alice Broker searches, using the index, for these two data sources and constructs an analysis of *predictions* of wave data based on the historical database and the predicted weather patterns. The model used to generate the predictions comes from a data interpretation. It consists of a program, the predicate `model1` that describes the parameters of the prediction model, and metapredicates. The predictions are a third data source whose associated context α consists of `raw_measurements`, `estimated_measurements`, and `forecast`. The metapredicate `forecast` indicates, for each forecast program, the predicates of its input, output and prediction model, i.e. `model1`.

Finally, a fourth data source represents shark attack reports in French newspapers. Another broker, Mary Broker accesses the dyad containing the α context via the index and constructs a fifth data source `safe_beach` that describes safe beaches for wind surfing, based on the predicted wave data and recent reported shark attacks. For Joe End User, the task of finding a safe beach consists of using a browser to examine `safe_beach`. Since the context of `safe_beach` contains the structure of the analysis, Joe can trace back through the structure to understand how a specific beach was determined safe or unsafe.

5 Conclusion

In this paper, we have presented a process-oriented model for the tasks involved in the generation of environmental data. Second, we modeled a hierarchy of environmental data. Third, we discussed the nature of data and metadata in this hierarchy. Finally, we proposed an architecture to support these activities.

Several ideas underly our proposed architecture. First, we propose *dyads*, a pairing of a natural language hyper-text description of information with the formal description of the corresponding data and metadata structures. This enables combining textual searching techniques with formal database query language techniques to help locate and interpret environmental data. At the same time we improve the quality of data delivered to users.

Second, we propose *contexts*, or collections of metapredicates that model the minimum information needed for unambiguous interpretation of a data set. Third, we propose organizing components of the architecture into browsers, data sources, indexing engines, and data interpretation programs. These components interact though a virtual space of dyads.

The following table summarizes the relationships between the tasks and metainformation.

Concept	Usefulness			
	Locate	Extract	Agg.	Analyze
Schema	dyad index	queries	queries	contextual predicates
Meta-Schema	dyad index	format exchg.	context	context

Acknowledgments The authors thank Oliver Günther for insightful comments on a previous version of this paper.

References

[1] Sandre: Dictionnaire de données sur l'eau. Technical report, Office International de l'Eau, rue Edouard Chamberlan, 87065 Limoges cedex, France, 1996.

[2] F. G. Fegeas, J. L. Cascio, and R. A. Lazar. An overview of FIPS 173, the spatial data transfer standard. *Cartography and Geographic Information Systems*, 19(5), 1992.

[3] O. Günther, H. Lessing, and W. Swoboda. UDK: A European environmental data catalogue. In *Proceedings of the Third International Conference on Integrating GIS and Environmental Modeling*, 1996. http://www.ncgia.ucsb.edu/conf/SANTA_FE_CD-ROM/santa_fe.html.

[4] O. Günther and A. Voisard. Metadata in Geographic and Environmental Data Management. In W. Klas and A. Sheth, editors, *Managing Multimedia Data: Using Metadata to Integrate and Apply Digital Data*. McGraw Hill, 1997. To appear. Also available as ICSI Technical Report No. TR-96049, http://www.icsi.berkeley.edu.

[5] E. Mendelson. *Introduction to Mathematical Logic*. Van Nostrand, Princeton, N.J., 1979.

[6] K. Millard. Personal Communication, 1996. HK Wallingford, Inc., United Kingdom.

[7] A. Tomasic, L. Raschid, and P. Valduriez. Scaling heterogeneous database and the design of DISCO. In *The IEEE International Conference on Distributed Computing Systems (ICDCS-16)*, Hong Kong, 1996.