A labeled-tree approach to semantic and structural data interoperability applied in hydrology domain

Nimmy Ravindran a, Yao Liang b,*, Xu Liang c

a Department of Electrical and Computer Engineering, Virginia Tech, United States
b Department of Computer and Information Science, Indiana University Purdue University Indianapolis, United States
c Department of Civil and Environmental Engineering, University of Pittsburgh, United States

ABSTRACT

The issues of data integration and interoperability pose significant challenges in scientific hydrological and environmental studies, due largely to the inherent semantic and structural heterogeneities of massive datasets and non-uniform autonomous data sources. To address these data integration challenges, we propose a unified data integration framework, called Hydrological Integrated Data Environment (HIDE). HIDE is based on a labeled-tree data integration model referred to as DataNode tree. Using this framework, characteristics of datasets gathered from diverse data sources - with different logical and access organizations - can be extracted and classified as Time–Space–Attribute (TSA) labels and are subsequently arranged in a DataNode tree. The uniqueness of our approach is that it effectively combines the semantic aspects of the scientific domain with diverse datasets having different logical organizations to form a unified view. Further, we also adopt a metadata-based approach for specifying the TSA-DataNode tree in order to achieve flexibility and extensibility. The search engine of our HIDE prototype system evaluates a simple user query systematically on the TSA-DataNode tree, presenting integrated results in a standardized format that facilitates both effective and efficient data integration.

1. Introduction

1.1. Problem definition

The advent of global observing systems (e.g., satellite remote sensing) and global field programs has enabled the generation of an unprecedented volume of multi-variate data that is critical for use in scientific studies. For example, in both hydrological and environmental research, massive amounts of data are used in studies of flood and drought predications; the information gained from such studies significantly impacts the areas of agriculture, industry, food production, and farming and can be then used to improve human lives. Effective examination and interpretation of this data is also pivotal in the improvement of hydrologic calibrations, weather prediction models, and in the fundamental understanding of critical environmental interactions.

Traditionally, the datasets used in such research are diverse and heterogeneous, both semantically and structurally [32]. Furthermore, the data publication methods and data management of these collection sources significantly complicate the data integration process [5]. For example, most organizations that collect such data, such as NASA and the United States
Geological Survey (USGS), publish their data over the internet with a user-friendly web interface for data retrieval, while at the same time maintaining their autonomous control over both the data models and data organization. This is complicated by the fact that the data access patterns utilized by these web-based interfaces often vary widely.

An example of such data integration problems can be found in the NOAA National Weather Service (NOAA NWS) River Forecast Centers (RFCs), which provide forecasts for rivers across the contiguous United States and Alaska. For these RFCs, data collection, re-organization and re-formatting have become a complex and time-consuming process. The forecast models developed and maintained by these NWS offices are calibrated for specific rivers and streams based on historical events. As a consequence, any delayed, inaccurate, inconsistent, incomplete or insufficient data used in calibrating the forecast models result in significant deficiencies in the forecasting process. In order to correct these deficiencies, particularly as new data sources continue to emerge, it is essential to progress in research with new models and develop corresponding tools for exploring, accessing and analyzing this data.

To further illustrate, consider this hypothetical scenario: a scientist needs to obtain information on precipitation trends across the United States. Assume that this information (e.g., precipitation data) is available from two independently managed data sources, A and B, which are each equipped with a web-based interface and organization-specific downloadable tools such as FTP methods for data access. In this scenario, the scientist must deal with two basic issues: first, how to identify pertinent data from each of the sources; and, second, how to issue queries based on the respective interfaces provided. Since the precipitation datasets extracted from sources A and B can vary in their data formats, semantics and models, it is necessary to first perform a manual transformation of the data to a common data model and a subsequent aggregation to make the data useful for concrete data analysis. If the number of data sources is increased beyond two, these operations have to then be repeated additional times. Clearly, this can become an extremely tedious and expensive task which requires a considerable expenditure of both time and effort. In fact, it has been estimated that such work could even consume as much as 80% of the overall effort [1]. To date, there are no effective tools to deal with such problems efficiently, although some advances have been made.

1.2. Existing approaches

Thus far, one of the most important advances in this area – within the Earth Science community – has been the development of metadata standards. Metadata has been introduced to address both the structural and syntactical heterogeneity associated with varied data sources. These efforts have led to the OpenGIS Standard [28], Federal Geographic Data Committee Standard [15], the Earth Science Markup Language [14], the Geography Markup Language [19], and the Ecological Metadata Language [13]. In the Geosciences domain, there have been similar efforts to develop a cyber infrastructure called GEON [18] for integration of resources (data, tools, visualization and computing capability) based on service oriented architecture (SOA). A similar effort is also currently being undertaken through CUAHSI (Consortium of Universities for the Advancement of Hydrologic Science Inc.) with support from the National Science Foundation. While these solutions are indeed proving useful, they are suitable only for newly collected datasets. For existing datasets in legacy systems, it is still an arduous task to transform them into a format which would even allow these new solutions to be applied in the first place. Therefore, it is now critically important to develop a data integration tool that can deal with existing heterogeneous datasets without requiring significant transformation work beforehand.

Currently, there are two commonly used approaches for such information integration: “data warehousing” (e.g., [20,22–24,29]), and “virtual integration” (e.g., [6,18,31]). Data-warehouse techniques require importing data from various data sources and maintaining them at a centralized data warehouse; such methods thus provide for centralized control over the data stored within and necessitate that queries are executed at the data warehouse itself. In all, there are a number of disadvantages inherent in the data warehousing approach, including data duplication, challenges in synchronizing data with autonomous data sources, data administration of the data warehouse, and significant wastage of resources.

In the virtual information integration approach, a mediator–wrapper architecture is often employed. This involves using a mediator as a semantic “wrapper” capable of transforming queries and data between the data integration system and data sources [6,18,36]. Such data integration systems attempt to enable data retrieval from existing heterogeneous and autonomous data sources, A and B, which are each equipped with a web-based interface and organization-specific downloadable tools with a unified query interface, which is significantly different from the data warehouse method. A majority of systems (e.g., [11,12,17,8,35,27,2,26,18]) using this approach emphasize the need for querying capabilities that can pinpoint the most appropriate datasets with a semantically rich query, although, a partially materialized method could also be implemented in mediator to speed up some more frequently used queries (e.g., [21,23]).

Various methods have been developed to facilitate such a semantically rich query function using ontology, including adopting a common ontology among registered information resources [11,17,8,35,18], mapping between ontologies of the registered information resources [27,2,26], and sharing ontology by defining similarity relations [25,3] using a common thesaurus. Such approaches which employ the virtual information space along with ontology to address semantic heterogeneity, work well only as long as data integration models can satisfy the constraints and rules of the domain.

One such ontology-based information integration system – Infosleuth ([2,27]) – is a collection of agents communicating via KQML (Knowledge Query Manipulation Language) that employs a common ontology translated by resource agents from individual data source-specific ontologies. In doing so, this approach enables users to identify a very specific dataset through a semantically rich query. Similar approaches can also be found in systems [11] utilizing declarative agreements between sources and system, and in the ODGIS framework [17]. Another such system – the OBSERVER [26] – utilizes IRMs (Inter
Nodes are organized into a source or a dataset to a basic set of hydrological primitives: Time, Space and Attribute. In this framework, individual DataNodes mentioned above by defining a simple, lightweight framework of hierarchically organized DataNodes which map a data structure or a dataset to a basic set of hydrological primitives: Time, Space and Attribute. In this framework, individual DataNodes are organized into a unified DataNode tree with domain view addressing ontology details and logical views representing the logical structure of datasets or data sub-repositories within individual data sources (i.e., a DataNode can represent a relational database, a collection of files or a single file).

Our approach is uniquely distinguished from existing approaches by enabling a tree-structured logic view of each individual data source. Furthermore, our method enables the unification of multiple tree-structured logic views of individual data sources into a global domain view through the DataNode tree modeling. Changes in any data source – even such a large change as a complete/partial migration from a data file-based system to a database-based system – would involve only the modification of the corresponding DataNode sub-tree (or part of the DataNode sub-tree) for that data source (i.e., the sub-tree representing the logic view of the corresponding data source) through XML metadata files.

While domain view does indeed represent the global ontology approach, the significance of our system lies in the unique ability of a DataNode tree to represent disparate data sources as a simplified network of nodes. Unlike other existing systems which mainly deal with global ontology, our approach takes into account the physical data organization of varied autonomous data sources (such as their hierarchical databases and/or file systems) and focuses on seamlessly integrating individual data organization schema (logical view) into an overall domain view representation of the underlying datasets to form a unified view.

To achieve a global view where local schema can be translated into global schema, as in [35], standard wrappers utilized for data access can be extended by source-specific derived wrappers capturing additional query capability. The dynamic nature of data sources and data sets as addressed in [35] highlights the need for systems with XML wrapping which can extend the various capabilities of data sources, however the global schema used in response in [35] is very specific to the datasets’ domains – namely geology. In contrast, our system employs a generic global schema of Time–Spatial–Attribute, while still utilizing the similar concept of extending querying capabilities of the data sources through XML metadata wrappers. It must be emphasized here that the wrappers used are XML files which describe how each information source is to be queried using their necessary language/API promoting protocol diversity. The necessity for creation of new wrappers is required only when a new data source with a different protocol interface (e.g., FTP) is registered into the system. Our integration solution is illustrated through implementation of the HIDE prototype system, using datasets from USGS and NWS River Forecast System (RFS) databases.

The remainder of this paper is organized as follows: we begin in Section 2 with a detailed description of our approach, including the data integration model, query evaluation, and metadata mechanisms. Section 3 then describes the architecture of the HIDE prototype system, as well as its implementation. Finally, Section 4 discusses conclusions.

2. Approach

2.1. System overview

A data integration system for distributed and autonomous data sources can be considered as a collection of independently managed resources/data that appears to its users as a single coherent data system, such that users and applications can interact with the distributed data in a consistent and uniform way. To support such a consistent view for varied data sources’ non-trivial structural data organizations and their heterogeneous datasets (unstructured, semi-structured, or structured), we propose a data integration framework, referred to as Hydrological Integrated Data Environment (HIDE), and develop a prototype HIDE system as shown in Fig. 1.

The primary function of our data integration system is to provide users with easy access to remote resources through the Internet in a controlled manner. These varied resources can be virtually anything: simple databases with SQL interfaces, web pages managing databases, or complex file systems with proprietary data standards. The lowest layer of the data integration system is formed by the data layer, which provides the core facilities enabling communication to the resources; this, in turn,
enables the translation of queries to more resource-specific queries and subsequent data management. The second layer is formed by a collection of facilities used in the creation and management of the DataNode tree, and also facilitates searching and querying on various resources. The highest layer – the presentation layer – consists of multiple web-based clients for data access, data manipulations, and visualization tools.

2.2. Problem domain

To further illustrate, consider this scenario: scientists working with the National Weather Service’s River Forecast System (NWS RFS) to calibrate forecast models need to also study and use datasets from other sources, such as the USGS and the National Climatic Data Center (NCDC), alongside datasets that are internally collected and maintained. In working with these data sources, the scientists will find that while the USGS maintains an exhaustive collection of water data, these datasets are controlled solely by the USGS, are maintained autonomously, and are accessible only through their web-based interface NWISWEB (National Weather Information System Web Interface) (http://waterdata.usgs.gov/nwis). In order to acquire the datasets, a user must traverse multiple web screens after filling in necessary querying information. The resulting datasets are then made available in various formats (e.g., graphs and ASCII).

In contrast, the datasets of the NWS RFS itself consist of a collection of various database files created periodically from a wide variety of information sources. These files contain current, forecasted and historical data and are specified using complex ‘C’ structures in multiple levels. Here, even a simply dataset (precipitation dataset covering a period of two days) is distributed across multiple files. In order to decipher such complex and variable files, the NWS has developed a wide range of independent ‘C’ based utilities to assist users in extracting the necessary information. In the course of working with only these two separate sources (USGS NWISWEB and NWS RFS) on one single project, it becomes apparent that a manual integration of data will be complex and time consuming.

The main idea behind our HIDE system is to provide scientists with a common portal to efficiently search and access varied datasets from sources such as the NWS, the USGS, the NCDC, and proprietary databases, thereby translating the data into common data models through the use of transformation layer. It then enables the use of various visualization techniques on...
the results obtained. Further, the HIDE framework maintains a consistent logical view irrespective of the nature of each dataset when migrating unstructured or semi-structured data files to structured databases. In such cases, the HIDE is only required to update its corresponding XML wrappers defining the access points associated with the external data sources. In addition, due to the modularity of our model, new tools can be seamlessly added at each layer with minimal effort.

In the following subsection, we will take a closer look at how we achieve the above mentioned data integration results using the DataNode tree model and search-query process. One note: throughout this paper, we use the terminology resource to represent a dataset which is independently owned and managed by a remote data source.

2.3. Data integration model – DataNode tree

The unstructured or semi-structured nature of the data typically used in scientific research has a tremendous impact on data integration approaches. Semi-structured data – often referred to as “self describing data” ([9,10]) – does not have any separate schema but, rather, its schema is embedded within the data itself. In order to develop a common model of integration which can be used effectively for both unstructured and semi-structured data, we have employed a “labeled graphs” model. This model, commonly used in data integration solutions and web information integration for conventional data [16], has been specifically tailored to address the issues of semantic and structural heterogeneity in a hydrological and environmental domain. We have also further simplified the “labeled graphs” model by defining an extendible DataNode which can then be used to specify datasets and data sources from the hydrological domain, while simultaneously being augmented by the concept of information hierarchy. The primary focus of our data integration model is to identify a DataNode tree to represent relevant data sources (such as the USGS, NCDC and NWS RFS) of interest.

DataNodes, the building blocks of the data integration model TSA-DataNode tree, facilitate the representation and organization of heterogeneous datasets residing at diverse, remote sources. A single DataNode, the smallest unit of information integration, can represent either an ontological concept or a structural element such as a database, a file (as in the NWS RFS system), or a particular data source. Due to the unique spatio-temporal nature of hydrological data, a DataNode associated with hydrological datasets can best be modeled as a Time–Space–Attribute (TSA) DataNode. Here, the “Time” aspect of the model corresponds to time units – for instance, hours, days, months, or even “recent” (e.g., “realtime” unit used by the USGS NWISWEB or a time series ‘C’ data structure used by the NWS RFS). The “Space” aspect can be characterized by spatial and geographical features such as states, watersheds, and latitude-longitude ranges and the “Attribute” aspect is used to define names for variables/features represented by the DataNode, i.e., to represent precipitation data, the variable name “precipitation” can be considered as an Attribute. Similarly for defining a data source such as the USGS, the name of the data source itself, i.e., “USGS”, can be expressed as an Attribute. These time, space and attribute aspects of the TSA-DataNodes are essentially labels which provide an identity to each DataNode. In the actual formation of DataNodes, we rely on researchers with domain-specific knowledge, such as hydrologists, for identification and labeling of nodes.

In the next step of the data integration model, we define “visibility” in the semantic and structural view of the data integration, using views in our model. The views in the system define hierarchical relations between the TSA-DataNodes thereby forming TSA-DataNode sub-trees. The associations between various TSA-DataNodes are designed to conceptualize either a generic ontology of the domain (domain view) or a logical organization of data (logical view). In the domain view, domain specialists organize TSA-DataNodes based on “concepts” in the domain. Hydrological concepts, such as “precipitation” and “windspeed”, essentially represent “attributes” which can be taxonomically organized (Fig. 2). In the logical view, data integrators organize TSA-DataNodes based on the logical organization of a given data source.

We note here that one unique feature of our approach is that the proposed integration model is capable of, in the logical view, representing an arbitrary tree-structured hierarchical organization of data for unstructured or semi-structured datasets for a given data source, as shown in Fig. 2a. In contrast, for unstructured or semi-structured datasets in the data source’s logic view, existing approaches treat such a tree-structured hierarchical organization of data as a set of individual datasets, as the data source of NWS RFC shown in Fig. 2b. Here, the information on any hierarchical organization of those datasets within the data source is basically unrepresented.

Logical views of individual data sources can later be combined or fused together, using Adaptation DataNodes, into a TSA-DataNode tree (black colored DataNodes in Figs. 2a and 2b). An adaptation DataNode acts as a root node for a logical view, while merging the logical view into the domain view. The choice of an Adaptation DataNode, while clearly of vital importance, largely depends on the data integrators’ intuition and knowledge of the organized datasets. Interestingly, domain views can be defined for different classes of users with various levels of permissions and accessibility, while a logical view is unique to a particular data source.

As discussed above, a domain view and the logical views of multiple data sources complete a TSA-DataNode tree. Generalization of the semantic and structural nuances of a particular data source through the use of TSA-DataNodes and the corresponding TSA-DataNode tree helps to present transparency to the users. As such, a user can pose high-level queries independent of the actual structure of the TSA-DataNode tree, since they will be automatically translated into TSA-DataNode sub-queries and executed. In addition, the flexibility in defining individual logical views ensures that the data sources maintain their design autonomy. The extendible nature of a TSA-DataNode permits users to define and elaborate DataNode trees and then “glue” them appropriately with datasets from other data sources, such as the different data sources of USGS, GPCC and NWS, as shown in Fig. 2a.
In logical view, a logical path to a TSA-DataNode is defined as a sequence of TSA-DataNodes traversing a path originating from an Adaptation TSA-DataNode and culminating at an intermediate node or a leaf node of the chosen data source. For instance, the logical path to TSA-DataNode “Alabama” in the TSA-DataNode tree of the logical view of USGS shown in Fig. 2a is
the sequence of USGS, USGS Water Data, realtime, Alabama. As logical views are different for individual resources joined by a domain view, the logical path information is necessary and critical in the query evaluation process.

The TSA definition of TSA-DataNodes and TSA-DataNode tree can also be applied to datasets with similar characteristics from other scientific domains. Alternatively, if the characteristics of datasets in a domain are significantly dissimilar, different model definition should be employed. Although similar approaches have been used in other domains [7] as well, our model takes into consideration both the structural (logical/physical) organization of the data and the autonomy of the data sources.

2.4. Model formalization

A DataNode tree $DT = (D, E)$, consists of a set of DataNodes $D$ and a set of unlabeled edges $E$ (Table 1). Let $T$ be a set of time labels $\{t_1, t_2, \ldots \}$. $S$ be a set of spatial labels $\{s_1, s_2, \ldots \}$. $A$ be a set of attribute labels $\{a_1, a_2, \ldots \}$, and $V$ be a set of synonyms of TSA labels, in the TSA modeling of the information space. A domain view sub-TSA-DataNode tree over $T$, $S$, $A$ is a tuple $DV = (D_o, E_o, \lambda, V_t)$ where $D_o \subseteq D$, $E_o \subseteq E$, $\lambda : D_o \rightarrow A$ is a node labeling function, and $V_t : \lambda(D_o) \rightarrow V$ is a controlled vocabulary matching function.

Every managed resource is equipped with a query interface. Assume that $M_d$ represents the interface and is defined as optional z-element tuple $M_d := (y_1, \ldots, y_z) \in T \cup S \cup A(1 \leq i \leq z)$, where $z$ is the number of query parameters to the managed resource. Accordingly, a logical view of a data source over TSA is a TSA-DataNode subtree, $LV := (D_o, E_o, \lambda, V_t, M_d)$ where $D_o \subseteq D$ is a set of logical TSA-DataNodes; $E_o \subseteq E$ is a set of edges; $\lambda : D_o \rightarrow (T \cup S \cup A)$ is a node labeling function; $V_t : \lambda(D_o) \rightarrow V$ is a controlled vocabulary translating function; and $D_o \subseteq D_o$, an adaptation node, is the root of the sub-tree $LV$. The function $M_d$ can also represent an aggregation model for an intermediate node. A node $D_i$ in the TSA-DataNode tree is said to be a domain node if $\lambda(D_i) \in A$ and to be a logical node if $\lambda(D_i) \in (T \cup S \cup A)$. Therefore, a DataNode tree, $DT := DV \odot LV$, $1 \leq i \leq m$, where $m$ is the number of data sources registered for unified access and potential integration, and the operator ‘$\odot$’ represents a tree join operation for all values of $i$. This operation represents attaching or “fusing” the domain view with the logical views of various data sources through “Adaptation” DataNodes.

A path over tree $DT$ is a sequence $P = (D_1, \ldots, D_n)$ where $D_i \in D(1 \leq i \leq n)$. Similarly, a logical path over $LV$ to a logical TSA-DataNode $D_i$ of a data source is an $n$-element sequence $P(LV, n)$ := $(D_1, D_2, \ldots, D_n)$; where $(D_i, D_{i+1}) \in E_i, 1 \leq i < n$. $D_i$ = $D_o$ is an adaptation TSA-DataNode, and $D_n = D_r$. That is, a logical path to a given logical TSA-DataNode of a data source starts with an Adaptation TSA-DataNode and ends with the TSA-DataNode $D_r$ in question. The label of the path of $P_i$, denoted as $label(P_i)$, is defined as $(\lambda(D_1), \ldots, \lambda(D_n))$.

2.5. Examples: constructing a logical view for NWS RFC data

As a first example, we will consider how database files consisting of a subset of time series data generated by the River Forecast Centers (RFCs) of the NOAA National Weather Service (NWS) can be specified using a logical view. The time series data from the operational forecast system at the RFCs are stored in flat files and consist of measurement variables such as precipitation and temperature. These time series files are generated using 45 days of data (composed of 30 days of observed data and 15 days of forecasted data), and subsequently collected and saved. Each file is then retrieved, zipped and stored in

---

**Table 1**

Summary of symbols and their meanings.

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>DT</td>
<td>DataNode tree</td>
</tr>
<tr>
<td>TSA</td>
<td>Time–Space–Attribute</td>
</tr>
<tr>
<td>$T$</td>
<td>Set of time labels</td>
</tr>
<tr>
<td>$S$</td>
<td>Set of space labels</td>
</tr>
<tr>
<td>$A$</td>
<td>Set of attribute labels</td>
</tr>
<tr>
<td>$DV$</td>
<td>Domain view</td>
</tr>
<tr>
<td>$LV$</td>
<td>Logical view</td>
</tr>
<tr>
<td>$D_o$</td>
<td>Domain node</td>
</tr>
<tr>
<td>$E_o$</td>
<td>Edges in domain view</td>
</tr>
<tr>
<td>$V_t$</td>
<td>Controlled vocabulary function</td>
</tr>
<tr>
<td>$M_d$</td>
<td>Query model (interface) of data source</td>
</tr>
<tr>
<td>$D_0$</td>
<td>Logical DataNode</td>
</tr>
<tr>
<td>$E_0$</td>
<td>Edges of logical view</td>
</tr>
<tr>
<td>$D_a$</td>
<td>Adaptation DataNode</td>
</tr>
<tr>
<td>$P_i$</td>
<td>Logical path to a DataNode</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>Node labeling function</td>
</tr>
<tr>
<td>$\odot$</td>
<td>Tree join operation</td>
</tr>
<tr>
<td>$Q(x)$</td>
<td>array of predicates</td>
</tr>
<tr>
<td>$\phi[]$</td>
<td>DataNode search operation</td>
</tr>
<tr>
<td>$\sigma[]$</td>
<td>DataNode query operation</td>
</tr>
</tbody>
</table>
the repository. The centers use complex software solutions to access and analyze this data, while a (partial) migration of the flat files into structured databases is also initiated.

In the example, we consider the construction of logical view for precipitation data which can be easily merged into the TSA-DataNode tree as shown in Fig. 2a. We begin by identifying appropriate TSA-DataNodes that accurately represent the information space. In this scenario – considering the structural information of datasets presented at a RFC – the nodes that are most readily identifiable are NWS, River Forecasting Center, States, Observed data group, and Forecasted data group. These DataNodes in the Time-Space-Attribute space can then easily be organized in a hierarchical manner.

At the NWS RFCs, the data files/tables are organized into years/months/days. Now suppose we are interested in the data from year 2000 to year 2004. By subsequently applying hierarchical relations between these nodes, one can construct a logical view as shown in the NWS node branch in Fig. 2a. An alternative simple logical view of the NWS RFCs is also shown in Fig. 2b, where the hierarchical organization of those datasets in the data source is not represented/considered. By maintaining a hierarchical logical view of the datasets as in Fig. 2a, rather than a set of individual datasets as in Fig. 2b, the HIDE model is able to achieve better efficiency in the searching and querying of those datasets. It must be noted here that the domain-specific knowledge of hydrological research scientists is essential in creating the TSA-DataNode tree with appropriate TSA time labels.

Let ‘’B’’ denote the NWSRFC dataset. Assume $T = \{t_1, t_2, \ldots, t_d, t_3, t_n, \ldots\} = \{\text{realtime}, \text{monthly}, \ldots, 2001, 2002, \ldots\}$, $S = \{s_1, s_2, \ldots, s^r_1, s^r_2, \ldots\} = \{\text{Washington}, \ldots, \text{Ohio, Arizona, Virginia,} \ldots\}$, and $A = \{a_1, a_2, a_1, a_2, \ldots\} = \{\text{precipitation, windspeed,} \text{ radiation,} \ldots, \text{NWS, RFC, observed, forecasted,} \ldots\}$. The DataNodes in logical view for the NWS shown in Fig. 2a can then be formalized as shown in Table 2, where $T^B = \{t^B_1, t^B_2, \ldots\}$, $S^B = \{s^B_1, s^B_2, \ldots\}$, and $A^B = \{a^B_1, a^B_2, \ldots\}$ and $T^B \in T$, $S^B \in S$, $A^B \in A$.

$M^B_0$ of DataNode “2000” represents the external query model of the data source and can be represented in this case as a $(T_d, S_d, A_d)$ as shown in the table. If a “join” operation of datasets is desired at the state level (e.g., Ohio), then an intermediate query model could be added along with the DataNode “Ohio”. In this case, the DataNode “Ohio” can be represented as illustrated in Table 3.

The logical path to DataNode “2000” can be represented by the blue arrows in Fig. 2a as follows:

$$P_3(D_x, 6) := (D^B_{11}, D^B_{12}, D^B_{13}, D^B_{14}, D^B_{15}, D^B_{16}, D^B_{17}, D^B_{18}, M^B_0) = (a^B_1, a^B_2, a^B_3, a^B_4, t^B_{11}(t_d, s_d, a_d)).$$

As demonstrated above, creating logical views for a structured data store (such as a relational database) is fairly straightforward since they can be built from the logical organization of the data indicated in existing database schemas. For an unstructured and semi-structured data source, one must abstract (or extract) information at each level of the tree to construct a TSA-DataNode tree.

### 2.6. Query evaluation mechanism

To better understand the importance of the DataNode tree used in our data integration approach, it is useful to consider the user query evaluation mechanism of various resources. We have devised a suite of algorithms for query evaluation: a UML sequential diagram is adopted to describe each algorithm, and a detailed pseudocode of each algorithm is developed.

UML sequence diagrams have two dimensions – a vertical dimension representing time and a horizontal dimension representing various instances of classes, modules or packages in the system. The messages sent between modules are represented by arrows. For example, the UML diagram shown in Fig. 3 can be summarized as follows: The query engine starts the

### Table 2
An illustration of the representation for NWS RFC DataNodes and DataNode tree.

<table>
<thead>
<tr>
<th>DataNodes</th>
<th>Formal representation</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precipitation</td>
<td>$D^B_{11}$</td>
<td>$a^B_1$</td>
</tr>
<tr>
<td>NWS</td>
<td>$D^B_{12}$</td>
<td>$a^B_2$</td>
</tr>
<tr>
<td>RFC</td>
<td>$D^B_{13}$</td>
<td>$a^B_3$</td>
</tr>
<tr>
<td>Ohio</td>
<td>$D^B_{14}$</td>
<td>$s^B_1$</td>
</tr>
<tr>
<td>Arizona</td>
<td>$D^B_{15}$</td>
<td>$s^B_2$</td>
</tr>
<tr>
<td>Observed Data group</td>
<td>$D^B_{16}$</td>
<td>$a^B_4$</td>
</tr>
<tr>
<td>Forecasted Data group</td>
<td>$D^B_{17}$</td>
<td>$t^B_{17}$</td>
</tr>
<tr>
<td>2000–2004</td>
<td>$D^B_{18}$</td>
<td>$t^B_{18}$</td>
</tr>
<tr>
<td>2000</td>
<td>$D^B_{19} M^B_0$</td>
<td>$t^B_{19}(t_d, s_d, a_d)$</td>
</tr>
</tbody>
</table>

### Table 3
A modified formal representation of the DataNode “Ohio” to add the intermediate query model supporting query split at the DataNode.

| Ohio            | $D^B_{19} M^B_0$       | $(\text{Ohio})(2000, 2001, 2003, \ldots, \text{“ohio”, “observed data group”, “forecasted data group”, “precipitation”})$ | $s^B_1(t \cdot s_d, a^B_1, a^B_2, a^B_3)$ |
evaluation by issuing a domain view search (DViewSearch) to the DataNode tree expecting results of various DataNode instances. Then, for each DataNode instance from the results, a logical path is evaluated (LPEval) based on the input search conditions (see Fig. 5). If a successful result for LPEval is indicated due to the presence of a matched DataNode instance, a DataNode Query operation (DataNodeQuery) is performed on the DataNode.

A query over a DataNode tree in HIDE allows users to obtain DataNode tree instances that satisfy the provided conditions. Given a tree $DT$, a query expression of $r$ in language $L$ over TSA, denoted as $L(r, DT)$, involves searching the appropriate DataNode(s) and then extracting the data from the respective data sources. In our model, we define query evaluation (as shown in Fig. 3) as a two-step process. It begins with a search () operation, followed by DataNode query () operation. The search () operation finds the appropriate TSA-DataNode(s) instances upon which the DataNode query () operation is performed.

Next, let $Q(x)$ be an $n$-array predicates of $x$ terms, where $(x \in T) \lor (x \in S) \lor (x \in A)$. We have $L(r, DT) := (\phi[Q_1(x)], \sigma[Q_2(x)])$ where:

- $\phi[Q_1(x)]$ is the search operation for the TSA-DataNode in the domain and logical view;
- $\sigma[Q_2(x)]$ is the TSA-DataNode query operation;
- the comma operator in the query denotes a sequential operation, i.e., search $\phi[Q_1(x)]$ is always followed by a TSA-DataNode query $\sigma[Q_2(x)]$;
- $Q_1(x) \subseteq Q(x)$ and $Q_2(x) \subseteq Q(x)$.

The search process $\phi[Q_1(x)]$ can be further divided into a search in the domain view followed by a search in logical view(s). The search in the domain view is performed by identifying the synonym relations associated with the aspects specified in the query and the aspect(s) associated with a TSA-DataNode, as shown by the UML sequence diagram in Fig. 4. The search in a logical view, shown by the UML sequence diagram in Fig. 5, involves the identification of a logical path to the targeted TSA-DataNode. Given a sequence of strings $w$ in $r$, we can say that $w$ spells a path in $DT$ if $w = \exists : D_i \rightarrow (T \lor S \lor A)$. Hence the search operation $\phi[Q_1(x)]$ will give us a list of the identified TSA-DataNode(s) $D_i$ (i.e., $[D_i|D_i \in D \land M_d(D_i) \neq \emptyset]$ for $1 \leq i \leq n$) on which DataNode query () operation $\sigma[Q_2(x)]$ can then be invoked. In other words, the search operation will list a subset of TSA-DataNodes that has a query model $M_d$ specified.
The search \( Q_1(\mathbf{x}) \) operation is highly critical in a query evaluation, which typically involves matching the user supplied keywords with labels of DataNodes. The match process in the search \( Q \) operation also takes into account the vocabulary \( V_r \) associated with each DataNode and the DataNode query model \( M_d \). The search \( Q \) operation can be further categorized into either a domain view search or a logical view search. In the domain view search, the search operation is performed only on Attribute DataNodes, while in the logical view search, it takes into account each type of DataNodes, namely Time, Space and Attribute. This search \( Q \) operation significantly reduces the expensive and time consuming nature of network operations by limiting the number of queries to resources. It should be noted that this level of query optimization depends to some extent on the structure of the TSA-DataNode tree itself; it may pose some limitations on users who do not have a complete knowledge of the TSA-DataNodes and the associated views, which is often the case for a logical view of unstructured/semi-structured data. To avoid such limitations, we provide the user with a guided query execution mechanism that manually traverses through the DataNodes. This permits the identification of necessary DataNodes and allows the user to execute a DataNode query \( Q \).

The DataNode query operation is performed on a logical DataNode, intermediate node or leaf, depending upon the model \( M_d \) of the TSA-DataNode. As illustrated by the UML sequence diagram shown in Fig. 6, DataNode query \( Q_2(\mathbf{x}) \) involves applying the user-defined query parameters onto the query model \( M_d \) of the TSA-DataNode. Referred to as “atomic query”, the DataNode query \( Q \) at a leaf node is of special interest, as it performs the actual query on the resource. The DataNode query \( Q \) at an intermediate TSA-DataNode of any logical view is an “aggregation query,” which is decomposed into multiple sub-queries and delegated to the TSA-DataNode's immediate children. Each child TSA-DataNode evaluates a sub-query with the continuation of this procedure until a leaf node is reached, where an “atomic query” will be issued to the corresponding resource. Upon the receipt of responses for all issued atomic queries, the results of all sub-queries will be aggregated at the corresponding intermediate TSA-DataNode(s).

Pseudocodes describing the algorithms of the Query Evaluation, Domain View Search, Logical Path Evaluation and DataNode query discussed above are provided as follows:
Algorithm 1. (Query Evaluation QueryEval)
Procedure QueryEval (Q(x): an array of predicates, D: root DataNode in the DataNode tree)

{Notations:
  DViewSearch: A function for searching an appropriate DataNode in the domain view. Returns the most appropriate DataNode.
  numChildren ( ): A function for retrieving the number of children of a DataNode.
  childi( ): A function for retrieving ith child of a DataNode.
  tuples[ ]: A 2-dimensional array of retrieved tuples.
  LPEval: A function for evaluating logical paths.
}
Algorithm 2. (Domain View Search DViewSearch)
Procedure DViewSearch (Q1: Predicates, D: DataNode)
{Notations:
isAdapt(D): A function to check whether the node is an Adaptation DataNode.
numChildren(): A function for retrieving the number of children of a DataNode.
child(i, D): A function for retrieving ith child of a DataNode.
Vj(D): The Controlled Vocabulary function for the DataNode.
Dtemp: A temporary DataNode}
1. if isAdapt(D) = 0
2. for i := 1 to numChildren(D)
3. Dtemp := child(i, D)
4. Dtemp := {Vj(Dtemp) | Q1}
5. if Dtemp ≠ 0
6. DViewSearch(Q1, Dtemp)
7. End
8. End

Algorithm 3. (Logical Path Evaluation LPEval)
Procedure LPEval (Q1: Predicates, D: DataNode)
1. Initialize the temporary variables, Counter and MC
2. for $i := 1$ to numLPs($D$)
3. \[ DLP := LP_i(D) \]
4. Counter $:= 0$
5. for $j := 1$ to numDNs(DLP)
6. \[ Dtemp := \{DLP(j)\} \]
7. if $Dtemp \neq \emptyset$
8. increment Counter
9. if Counter $\geq MC$
10. MNode $:= Dtemp$
11. MC $:= Counter$
12. End
13. End
14. $D := MNode$

Algorithm 4. (DataNode query DNQuery)

Procedure DNQuery ($Q_2$: Predicates, $D$: DataNode, tupleArray: An array of strings)

1. if numChildren($D$) $= 0$
2. \[ D := \{Md(Q_2)\} \]
3. tupleArray $:= fill(D)$
4. for $i := 1$ to numChildren($D$)
5. \[ D := child_i(D) \]
6. if $Md \neq \emptyset$ then
7. \[ tupleArray := DNQuery(Q_2, D) \]
8. globalTupleArray $= join(tupleArray)$
9. End

Now, given a DataNode tree as shown in Fig. 2a with a DataNode query model $M_d$ of $(\text{ASsitesTseconds})$ for the USGS DataNode “Alabama”, consider this example of a possible user question: “What is the precipitation distribution of realtime water data for the state of Alabama from United States Geological Survey data source for a period of 30 days?” To find answers for this question, the user can search for relevant data by using key words, for example, the highlighted/bolded words above. Thus, the user-level query to HIDE can then be represented as:

Search (precipitation, USGS, realtime, Alabama), DataNode Query (precipitation, 214762, 30).

To illustrate this in further detail, let us assume (for simplicity) that each DataNode carries a simple vocabulary list of keywords to be used for matching the DataNode. Note that a vocabulary list could also be implemented using more complex data structures, for example, a hash dictionary for faster lookup. The search () operation in the query evaluation mechanism involves a domain view search and a logical path evaluation (refer Algorithms 1–3). In domain view search, starting with root DataNode, the query engine tries to match the conditions of the search () operation (precipitation, USGS, realtime, Alabama) with each DataNode. In other words, at each level, the query engine tries to match a DataNode by performing a linear search.
of user supplied keywords across the vocabulary list of children nodes, as denoted by rectangular boxes in Fig. 7a. If a match is found, that child DataNode is selected and the search is continued in the next level of the tree. If a collision is found, the algorithm chooses the first matched child node. For the query in question, the “precipitation” keyword matches the labels in the vocabulary list of child (“precipitation”) of root DataNode and hence is selected.

Now, let us assume that the vocabulary list of DataNode (“precipitation”) contains keywords (“precipitation”, “precip”, “rainfall index”) and the keyword “precipitation” in the user-supplied search query string is changed to “precip”. Based on the simple matching algorithm, in the first level of query, the query engine would still match and select the DataNode (“precipitation”). Subsequent search (involves similar matching and the selection of the matched paths, as shown in Fig. 7a, until an adaptation DataNode is reached where the domain view search is replaced by a logical path evaluation (refer to Fig. 7b).
to Algorithm 3). It is worth mentioning that if a partial match is found, the query engine stops the search at the partially matched DataNode and then enables the user to manually traverse the DataNode tree further.

In logical view, the query engine continues the keyword matching process on the pre-computed logical paths from each adaptation DataNode. Additionally, the engine records a matching counter which is updated with the percentage of match for each logical path used in the evaluation. This matching counter is later used by the engine to choose the best logical path. In addition to matching keywords with the vocabulary list in logical view, the search operation also performs a keyword matching against the query model \( M_d \) of the DataNode. On identifying the appropriate logical path, the engine traverses to the leaf DataNode “Alabama”, to continue with the DataNode query () operation. In completing this operation, the engine utilizes the query model \( M_d = T(\text{precipitation}), S(214762) \) and \( T(30) \) to issue a request to the external data source and extract the datasets. The DataNode query operation performed at a leaf DataNode hence is referred to as an “atomic query” (see query traces in Fig. 7a).

Consider another example: “What is the precipitation distribution of realtime water data from USGS data source for all states”. In this example, the user selected the highlighted/bolded words as his keywords to form his query to HIDE. Thus, the user-level query to HIDE is represented as:

\[
\text{Search (precipitation, realtime, USGS) DataNode Query (precipitation, all, all)}
\]

Assuming that “realtime” DataNode in Fig. 7 has a query model \( M_d = (T(\text{all}), S(\text{all}), A(\text{precipitation})) \), the search operation (domain and logical) stops at the “realtime” DataNode (Fig. 7b). At this intermediate DataNode, the query engine decomposes the original query and issues multiple sub-queries to each of the child DataNodes. Each sub-query is continued until the leaf DataNode is reached. Multiple requests are issued to the data source and – depending on load parameters – datasets are extracted and merged at the intermediate DataNode – “realtime”. This type of DataNode query is referred to as “aggregation query” where the query is decomposed into multiple sub-queries and delegated to the immediate children DataNodes. Upon receipt of responses for all issued atomic queries, the results of all sub-queries will be aggregated at the corresponding intermediate TSA-DataNode(s).

Fig. 8. XDMS specification for a USGS realtime DataNode.

```xml
1. <DataNode xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation="c:\IDAM\xml\xsd\funcModelSchema.xsd" >
2.  <identifiedAs>
3.       <name> USGS-RealTimeData </name>
4.       <label> daily </label>
5.       <index> realtime,daily </index>
6. </identifiedAs>
7. <canMeasure>
8.   <query_template1 />
9.   <measurements>
10.      <measurement name="gaugeHeight">
11.         <constraint>
12.           <type> GaugeHeight </type>
13.           <minValue> 10.0 </minValue>
14.           <maxValue> 20.0 </maxValue>
15.         </constraint>
16.       </measurement>
17. </measurements>
18. </canMeasure>
19. <supportOperations>
20.   <operation type="select">
21.     <select>
22.       <select_templates>
23.         <select_templates
24.           fileName="USGS/USGS_realtime_query.xoms"/>
25.       </select_templates>
26.     </select>
27.   </operation>
28. </supportOperations>
29. </DataNode>
```
2.7. XML schema for data integration model

In this research, the proposed data integration model is realized using an XML-based metadata approach. Our primary aim in designing the metadata was to provide a certain level of flexibility, extensibility and simplicity to the users in integrating heterogeneous datasets from diverse data sources. Once the user (as integrator) provides the metadata – an XML [4] instance document specifying the data source – the data source can be integrated/added into the TSA-DataNode tree without any modifications to the implementation (code) of the TSA-DataNode tree model. Additionally, using XML enables easy adoption of new datasets and thus addresses the requirements for achieving an effective and efficient data integration system. Our XML-based metadata standard can be classified into three different categories as follows:

1. Describe TSA-DataNode, its taxonomy/ontology nature, and TSA-DataNode tree.
2. Define DataNode query model and translation to the resource query model.
3. Define the syntactic and semantic nature of data.

---

```xml
1. <xml version="1.0"/>
2. <query xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3. xsi:noNamespaceSchemaLocation="c:\IDAM\xml\xsd\query_template.xsd" >
4. <internal>
5. <search_criteria>
6. <fields>
7. <field name="sites" displayName="Sites" type="choice" empty="no" description="List of sites for the state of Alabama"/>
8. <choice type="SingleChoice" input="external" value="fileNames.txt"/>
9. </field>
10. <field name="Days" displayName="Days" type="Entry" empty="no" description="Number of Days in the range 1-31">
11. <Entry type="SingleEntry">
12. <SingleEntry dataType="Integer">
13. <Integer rule="yes">
14. <ValidationRule type="RangeRule">
15. <RangeRule>
16. <from> 1 </from>
17. <to> 31 </to>
18. </RangeRule>
19. </ValidationRule>
20. </Integer>
21. </SingleEntry>
22. </Entry>
23. </field>
24. </fields>
25. </search_criteria>
26. <output_criteria name="AvailableParameters" display="Available Parameters">
27. <items order="no" dependsOn="sites" fileName="codes.txt"/>
28. <item name="item3" value="04" displayName="00045 Precipitation (DD04)"/>
29. </items>
30. </output_criteria>
31. <APDefinition type="http"/>
32. <http>
33. <baseURL>http://waterdata.usgs.gov/nwis/uv</baseURL>
34. <selection>
35. <parameters>
36. <parameter type="nodefault" param="dd_cd" value="$AvailableParameters.item1"/>
37. <parameter type="nodefault" param="dd_cd" value="$AvailableParameters.item2"/>
38. <parameter type="nodefault" param="dd_cd" value="$AvailableParameters.item3"/>
39. <parameter type="default" param="format" value="rdb"/>
40. <parameter type="nodefault" param="period" value="$Days"/>
41. <parameter type="nodefault" param="site_no" value="$sites"/>
42. </parameters>
43. </selection>
44. </http>
45. </APDefinition>
```

Fig. 9. XOMS specification USGS realtime DataNode.
Metadata describing TSA-DataNode: Each TSA-DataNode in HIDE, irrespective of views, describes itself in an XML description model specification (XDMS). As an example, Fig. 8 shows the XML definition of a USGS realtime DataNode. A DataNode referred to in line 2 consists of four elements – identity (identifiedAs), documentation (documentedBy), measurements (canMeasure), and operations (supportOperations). In line 4, an identity element is defined with a name of “USGS realtime data” and is indexed by controlled vocabulary of “realtime, daily”. This element is used by search () in the query evaluation process. Similarly, the documentation element can be utilized by resources to include additional information about the data-providing organization, purpose of the organization, its research goals, etc. A URL that directs a user to the corresponding organization’s website can also be specified here. A TSA-DataNode can also model any entity capable of producing and managing any amount of data. For instance, a sensor node measuring a physical variable can be best represented as a TSA-DataNode.

Taking into account this versatility of the specification, the “canMeasure” element, lines 7–8 in Fig. 8, can also be used to specify measurement details and the quality constraints of the node. Moreover, the module can also be used by other TSA-DataNodes (data sources) to specify a high-level picture of the measurements. In the element of operations, one can specify operations (i.e., query/update) that are permitted on the resource and URLs that link to the corresponding query interface of the resource. In this example, the DataNode can support query operations and the corresponding interface can be found at USGS_realtime_query.xoms.

Metadata defining the operational characteristics-query of a TSA-DataNode: Disparity of interfaces used by individual data sources makes locating and accessing data cumbersome. Hence, one of the primary objectives of this research is to provide a uniform representation of the user interfaces irrespective of the complexities of underlying data sources. This is achieved through the definition of operational model metadata XOMS (XML operational model specification). This interface is used by the DataNode query () operation in the query evaluation process. Fig. 9 shows an example of an XOMS file for USGS realtime DataNode.

A DataNode query operation can be considered as a combination of conditions and result parameters. A user can define a set of possible conditions on parameters as well as a set of result parameters in the query metadata for each node. Each condition can be characterized by a specific data type along with appropriate validation rules. This enables both simple

---

```xml
1. <?xml version="1.0"?>
2. <syntax type="Ascii" delimiter="tab-separated">
3.  <Ascii>
4.  <Comment value="#"/>
5.  <Header type="string" numLines = "10"/>
6.  <Dataset type="dataTable">
7.    <dataTable numRecords="unbounded">
8.     <dataFields>
9.       <dataField name="agent_cd" displayName = "Agency code" description="Agency code" type="string" default="yes"/>
10.      <dataField name="site_no" displayName = "Site no" description = "Site no" type="string" default="yes"/>
11.      <dataField name="date" displayName = "Date" description = "Date" type="date" default="yes"/>
12.      <dataField name="time" displayName = "Date" description = "Date" type="date" default="yes"/>
13.      <dataField name="Discharge" query_fldname="AvailableParameters.item1" displayName="Discharge" description="Discharge" type="integer" unit="cubit feet per second"/>
14.      <dataField name="Gauge Height" query_fldname="AvailableParameters.item2" displayName="Gauge Height" description="Gauge Height" type="float" unit="feet"/>
15.      <dataField name="Precipitation" query_fldname="AvailableParameters.item3" displayName="Precipitation" description="Precipitation" type="float" unit="inches"/>
16.     </dataFields>
17.   </dataTable>
18. </Dataset>
19. </Ascii>
```

Fig. 10. XFD specification for USGS realtime DataNode.
and flexible dynamic generation of the user interface. In this example, the lines 6–24 in Fig. 9 represent the query parameters “sites” and “days”. Each query parameter can be checked against validation rules as defined in the XOMS files (line 14). The result parameters in this example are “precipitation” as shown by line 28.

In addition to the DataNode query model of each TSA-DataNode, the standard also defines how the node's query model can be translated to the resource's query model. Here, we define an access point as a port of connection between HIDE and the resource. The access point manages the connections with the resource in addition to the translation of HIDE query to the query model of the resource. The lines 31–45 define an HTTP access point representing a web-based interface with the resource. Using the XOMS specification along with the XDMS specification, the user-defined query is applied to the appropriate TSA-DataNodes and then translated to a query of the resource.

**Metadata for describing the data (syntax and semantics):** The specification of the syntactic and semantic information of the data received from the resource is provided in XFD (XML format description) specification (see Fig. 10). This type of metadata describes syntax information such as (1) “What is the format of the file ASCII/binary?”; and (2) “What is the appropriate type of the file – comma-separated or tab-separated?”. Semantic information can also be described, i.e., (1) “What are the retrieved fields?”; and (2) “What are the data types of the retrieved fields?”

Multiple TSA-DataNodes can share an XFD file if they have similar syntactic and semantic characteristics. A syntax element referenced in line 2 of the example in Fig. 10 is of type Ascii with tab-separated file. Each field specified in the retrieved dataset is defined subsequently in lines 8–16. This information is then utilized by the data transformer (see Fig. 1) to understand and transform the data into the internal data models of the HIDE system.

### 3. HIDE implementation

To validate and demonstrate our proposed data integration approach, we have developed a web-based prototype system of HIDE. In developing our system, we have built upon other ongoing efforts that have focused on designing strategies to address the extensibility of such a system. One such current approach is to employ a modular architecture in which a system is composed of small and autonomous components that collaborate to achieve a common goal. The power of this type of

---

**Fig. 11.** A snapshot of a time series display (2D plot) for a range of 31 days. The left panel shows a time series for each field (e.g., gage height, precipitation and discharge). The two vertical lines in the left panel specify a range with which a zoom-in time series is displayed on the right panel. The statistics of each field shown within the two vertical lines are listed below the plots on the left. The single vertical line in the right panel of the time series probes the values of each field it points to.
architecture is that one can add/replace any component without affecting others. Using this notion of flexibility, we have developed a prototype HIDE system based on the modular/layered architecture as shown in Fig. 1.

The presentation layer is a user interface layer comprised of various tools, such as a search engine and a visualizer, which allows users to interact with HIDE. This layer enables users to search datasets, visualize them and perform statistical analysis on them. The search performed here would list all the possible datasets or resources defined as TSA-DataNodes which match the user query. Additionally, a user can also view the corresponding DataNode query interface in order to perform any further queries to the resource. Finally, users can also perform 2D/3D remote data visualizations of the received integrated data (Fig. 11). In all, the presentation layer acts as the main point of contact from which hydrologists can search data sources, access data sets and perform 2D/3D analysis.

As the name suggests, the query engine layer creates and manages the DataNode tree specified by the instances of the metadata files. It also performs the critical search and “query” operation on the external data sources. This layer is the main processing unit of the HIDE and is used by HIDE system administrators for data source registration as well.

The transform engine (data transformer) layer performs data transformation of the retrieved datasets into a common data model. This layer consists of various domain-specific transformation models, such as Time Model, Spatial Model and Time–Space Model. The usage of such a transformation layer allows the system to easily transform datasets that vary in their data formats, as represented by the XFD, to a common data model. This helps to resolve any measurement conflicts that may arise when performing data integration. For more details about the data models please see [30].

Finally, the resource manager layer contains a set of access control points which are used to interact with the external data sources. These access control points create and manage the physical connection of the external data sources with HIDE.

The success of the proposed model depends significantly on the effective generation of the TSA-DataNode tree. The crucial nature of this process affects the successful realization of the entire query evaluation itself. Hence, if a logical view of a data source is not properly constructed, erroneous results in the Search routines may very well be obtained. In addition, an instance of the XML standard XDMS defining a TSA-DataNode is used for the creation of the TSA-DataNode in system implementation. This may require a comprehensive understanding of the XML language by the user; although XML is fairly simple, domain scientists – such as hydrologists – may instead prefer a simpler “tool” for the generation of the TSA-DataNode tree.

To this end, we have designed and implemented a TSA-DataNode tree generator for creating domain view, logical views and the TSA-DataNode tree. The tool generates XDMS specification instances for each TSA-DataNode created by the user (Fig. 12). With this tool, it becomes evident that a significant amount of XML editing work can be minimized by inputting just a few parameters. In addition, because the generated XML strictly follows the schema, DataNode structure and XML validation errors can be avoided. Once a TSA-DataNode sub-tree is generated, the user can submit it to the HIDE system for its immediate use. This tool enables users to create TSA-DataNodes effectively and efficiently, and hides the complexities of the creation of XDMS instances for each TSA-DataNode.
Various types of users can be identified in the DataNode tree generator tool. Consequently, different types of users can also be equipped with different levels of permissions and accessibility. One set of users is comprised of “domain experts” who can define the domain view. Another type of user is the “integrator” who models the logical view for his/her data source to be later combined into the domain view. These changes in views are then exported to the system HIDE by the user administrator, and thus are reflected in the subsequent search and query operations.

For the current prototype system, we used JSP for developing user interface screens hosted on a Tomcat 4.x server. The query engine, data transformation and access engine layers are implemented in Java packages and thus promote an open environment. Presently, the data repository used is the local file system of the computer where the HIDE is hosted. In the future, it will be replaced by a relational database which can be accessed and updated by the HIDE.

Setup of the HIDE system involves the installation of a Tomcat server and the deployment of the HIDE modules in the server as a web component. The new data sources can then be incorporated through the metadata files namely XDMS, XOMS and XFD. This involves registration of data sources, which can be further elaborated by specifying a DataNode tree, defining query screens, data formats and semantics.

The modular nature of the proposed architecture ensures that each module is open and can be implemented using an alternate language and tools. In addition, any additional features required in a module can be easily plugged in without affecting other modules. Currently, the prototype system integrates datasets from data sources from the USGS and the NWS RFCs.

4. Conclusions and future work

This paper presents a unique and effective data integration solution for heterogeneous, autonomous, and distributed data sources using the concepts of TSA-DataNodes and the TSA-DataNode tree. The proposed TSA-DataNode tree integration model is realized using a metadata approach to facilitate flexibility and extensibility. Our system has the following features which distinguish it from existing tools and systems: (1) a unique data integration model – DataNode tree that employs the primitive and generic concept of Time–Space–Attribute to combine domain-specific capabilities with the varied logical organizations of datasets from diverse data sources in a unified framework; (2) an arbitrary tree-structured hierarchical organization of the logic view for a given data source to represent the structural information among its structured, unstructured or semi-structured datasets; and (3) a convenient and efficient metadata tool to add new data sources/datasets and a simple modular open architecture that can be further extended at each layer using multiple tools and languages.

Our approach enables data interoperability and integration without the constraints of a local data repository and re-organization, as is the case with data warehousing. The proposed model is also particularly useful for resolving data interoperability and integration problems involving autonomous data sources/organizations with massive data stores (e.g., USGS) where data duplication for data warehousing is prohibitive. Possible future work includes extending the DataNode tree generator tool and applying the TSA-DataNode tree integration model to other scientific domains.

Acknowledgements

This work was supported in part by National Science Foundation under Grant EAR-0413265, and National Aeronautics and Space Administration under Grant NNA07CN83A. The authors thank hydrologist Thomas Adams of the NOAA National Weather Service, Ohio River Forecast Center, for discussions related to the forecast center’s data formats and organizations.

References

[12] DODS, Distributed Oceanographic system, [http://www.opendap.org/] (checked 08.06.10).
[18] GEON, the Geosciences Network, <http://www.geongrid.org> (checked 08.06.10).