# An Adaptable Framework for Integrating and Querying Sensor Data

Shahina Ferdous, Sarantos Kapidakis, Leonidas Fegaras, Fillia Makedon

# An Adaptable Framework for Integrating and Querying Sensor Data

Shahina Ferdous[1], Sarantos Kapidakis[2], Leonidas Fegaras[1], Fillia Makedon[1]

[1]Heracleia Human Centered Computing Lab
University of Texas at Arlington
shahina.ferdous@mavs.uta.edu, fegaras@cse.uta.edu, makedon@uta.edu

[2]Laboratory on Digital Libraries and Electronic Publishing
Ionian University,
sarantos@ionio.gr

**Abstract.** Sensor data generated by pervasive applications are very diverse and are rarely described in standard or established formats. Consequently, one of the greatest challenges in pervasive systems is to integrate heterogeneous repositories of sensor data into a single view. The traditional approach to data integration, where a global schema is designed to incorporate the local schemas, may not be suitable to sensor data due to their highly transient schemas and formats. Furthermore, researchers and professionals in healthcare need to combine relevant data from various data streams and other data sources, and to be able to perform searches over all of these collectively using a single interface or query. Often, users express their search in terms of a small set of predefined fields from a single schema that is the most familiar to them, but they want their search results to include data from other compatible schemas as well. We have designed and implemented a framework for a sensor data repository that gives access to heterogeneous sensor metadata schemas in a uniform way. In our framework, the user specifies a query in an arbitrary schema and specifies the mappings from this schema to all the collections he wants to access. To ease the task of mapping specification, our system remembers metadata mappings previously used and uses them to propose other relevant mapping choices for the unmapped metadata elements. That way, users may build their own metadata mappings based on earlier mappings, each time specifying (or improving) only those components that are different. We have created a repository using data collected from various pervasive applications in a healthcare environment, such as activity monitoring, fall detection, sleep-pattern identification, and medication reminder systems, which are currently undergoing at the Heracleia Lab. We have also developed a flexible query interface to retrieve relevant records from the repository that allows users to specify their own choices of mappings and to express conditions to effectively access fine-grained data.

**Keywords:** Metadata, Schema Mappings, Query Interface, Digital Libraries, Pervasive Applications, Healthcare.

## 1 Introduction

Data generated from pervasive applications can be of many different types, such as sensor readings, text, audio, video, medical records, etc. One example of a common pervasive application is the "remote activity monitoring" in an assistive environment. Continuous monitoring of activities is very important in healthcare since it helps caregivers and doctors to monitor a patient remotely and take actions in case of emergency. Such a system may also be very useful for monitoring an elderly person who lives alone in an apartment and needs occasional assistance. A pervasive application deploys various non-invasive sensors as an integral part of a persons' assistive daily living to automatically monitor his activities. It may also deploy a less-intrusive audio and video recording system, based on the needs and privacy requirements of the patient. To be effective, an assistive environment needs to store the data generated from pervasive applications into a common repository and to provide the healthcare providers a flexible and easier access to this repository.

Current advances in sensor technology allow many different sensors that use different technology to be used interchangeably to generate and record similar information about an environment. For example, a person may wear a wireless wrist watch [1] as a heart rate monitor, which may also include a 3-axis accelerometer, a pressure and a temperature monitor. On the other hand, a Sunspot sensor [2] can be used as an accelerometer and a temperature sensor as well. Although, both devices can be used to generate the same acceleration and temperature data, the format of their representation can be very different. In fact, a sensor can be programmed in many different ways to deliver data in different formats. For example, a sunspot can be programmed to transmit either 3-axis accelerometer data or the angle of acceleration. Therefore, even for a simple sensor device, such as a sunspot, the data storage may contain data in various schemas and configurations. As a result, it is very hard for a caregiver or a doctor to understand and remember each such different schema to query the data repository.

A pervasive application may sometimes refine a sensor reading specific to its settings. For example, sensor devices, such as a smoke/heat detector, only detect and transmit results when the environment reaches a predefined threshold. On the other hand, a stand-alone temperature sensor, such as a sunspot, can be configured to indirectly generate a heat alarm when the temperature exceeds some prespecified heat threshold. Moreover, such thresholds and conditions can change dynamically depending on a user's query requirements, thus making it highly infeasible for a system to cache all possible answers related to a user's query beforehand.

This paper describes a Digital Library that consists of a repository of sensor data collected from various pervasive applications and a flexible query interface for the user that requires minimum knowledge about the real metadata schemas from a user's point of view.

### 1.1 Framework

In this paper, we describe our framework for the sensor data repository, which contains datasets, such as $C_{11}$, $C_{12}$, …, $C_{nm}$, derived from many different sensors, collected over a long period of time, possibly after years of experiments. We call each such dataset $C_{ij,}$ collected from various applications and contexts, a *collection*. Each collection is also associated with a metadata schema, which describes the format of that collection. For example, in Fig. 1, the collection $C_{11}$ is described by the schema $S_1$. A user can query over these different collections by simply using one of his preferred schemas, which we call a *virtual schema*, $S_v$ in our framework. In our framework, given a user-query $Q_v$ over the collections of different schemas, the user first provides the needed parts of "Map($S_i$ -> $S_v$)", the mapping for each individual schema $S_i$ of a selected collection to the specified virtual schema, $S_v$. Given that the user specifies all such required mappings, the system next applies these mappings to return the final query answers, denoted by $Q_1$, $Q_2$…$Q_n$. Thus, based on our framework, a user may query on any kind and of any number of collections and obtain fine-grained results without knowing details about the real schemas. In the future, we are planning to use this framework as the building block to obtain the background knowledge for automatic metadata mappings.

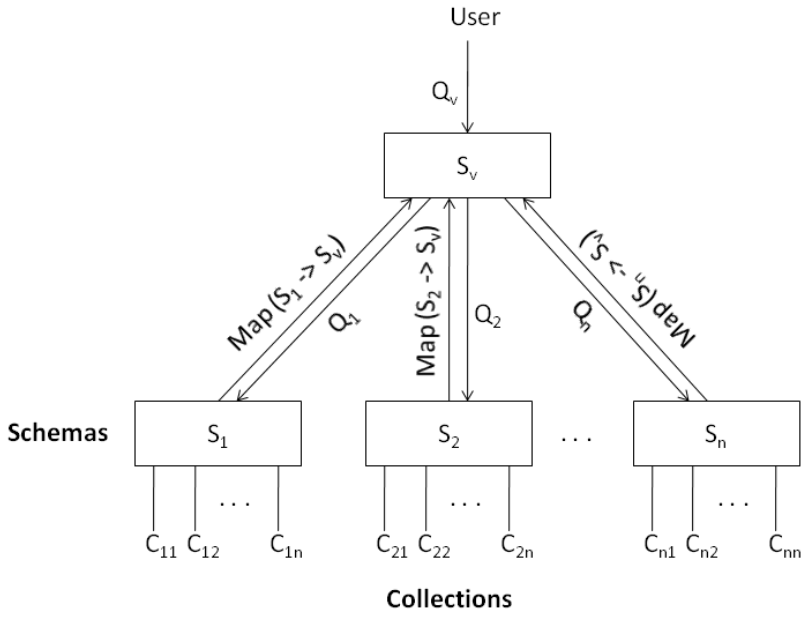


**Fig. 1.** Framework to query over the Digital Library

## 1.2 Challenges

Sensor records may not of the same format, since they could be generated from different types of sensors installed in various setups, or different versions of similar sensors, with small variations on the record formats. Besides, the values recorded from similar sensors can have different semantics too. The following example shows that even for a simple scenario, such as monitoring "whether a door of an assistive apartment is open or not", the system may require many different mappings to answer a user query.

A pervasive application can use variety of door sensors, which eventually produce door-"open/close" datasets in different formats. For example, the system may use a sensor based on door-mounted magnetic contacts, which denotes that the door is open (1) or closed (0) as an OPEN_STATUS (1/0) attribute, combined with the time of this event as an attribute named TIME_DETECTED. Any accelerometer sensor, such as a sunspot, can also be mounted on the door to obtain similar information. Since, a sunspot can be preprogrammed in different configurations, one configuration may transmit and store Cartesian (x, y, z) coordinates of the door, while the other may compute and transmit the angle of the current position of the door. Such a programmable sunspot may provide both TIME_RECEIVED and TIME_BROADCAST attributes as well.

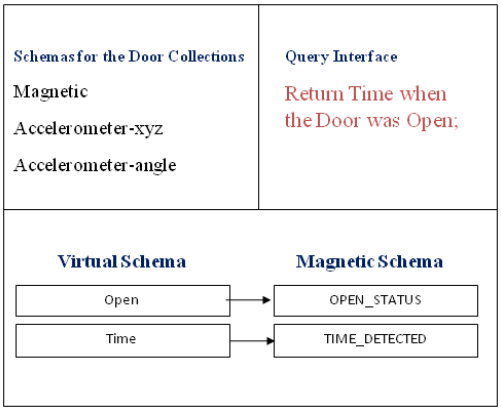


**Fig. 2.** Metadata Mapping between similar attributes with different names

A user may not know anything about the schema for the sensor being used to collect such door-data, but he still may ask a query such as "Give me the time when the door was open" over the collection of door datasets. Although answering such a simple query seems trivial, the mappings can be very different and complex. Fig. 2 describes the simplest mapping scenarios, where differently named attributes convey similar information, such as Open and OPEN_STATUS or Time and TIME_RECEIVED from the "Virtual" and the "Magnetic" schemas respectively.

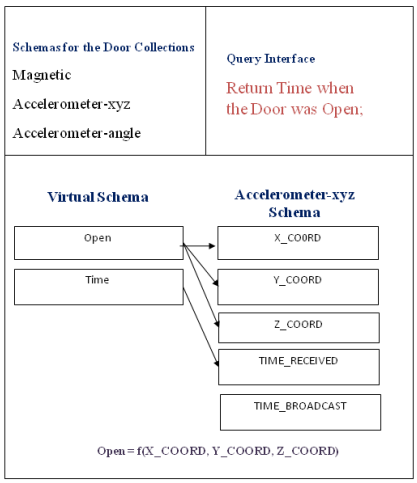


**Fig. 3.** Metadata Mapping from One-to-Many

However, a metadata field from one schema can be mapped to multiple metadata fields from the other by applying appropriate mapping function. Hence, if a user wants to select such an one-to-many mapping, he needs to specify the correct mapping function as well as to obtain meaningful results. Fig. 3 shows one example scenario, where the field "Open" in the virtual schema is mapped as a function of X_COORD, Y_COORD and Z_COORD fields of the Accelerometer-xyz schema. Fig. 4 shows another scenario, where a user may need to specify a condition even for a one-to-one metadata mapping. As shown in the example, a user may map the field "Open" to the field "Angle". But, since the attribute "Angle" does not directly specify the open status of the door, the user also needs to specify a condition, such as "the door is open, if the angle is greater than at least 4 degrees" and so on.

However, even if the metadata field describing an attribute is the same for two schemas, the associated units of their values can be completely different. For example, the temperature can be described in either Celsius or Fahrenheit, distances can be written either in feet or in meters, time can be a represented in milliseconds or as a general date/time expression. Therefore, in addition to field mappings, the system needs to be able to apply value mappings as well.

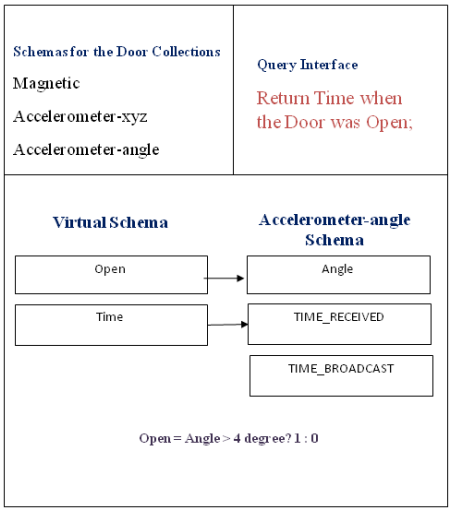


**Fig. 4.** Indirect Metadata Mapping

### 1.3 Contribution

In this paper, we describe a flexible query interface for searching relevant records from a Digital Library of sensor data. Our interface requires minimum background knowledge and returns results in a format chosen by the user himself. The Library also stores the history of mappings as part of a user profile. Thus, a user can re-use existing mappings to query over the same collections multiple times, thus providing minimal information. However, our system is flexible enough to allow the user to re-write some existing mappings or to add new conditions to the previously specified mappings. Our system may also suggest existing mappings from other user's profiles, which helps a new user to get some idea about the mapping between schemas. The flexibility of reusing and revising metadata mappings make the framework adaptive to different data management needs and different sensor metadata and formats.

## 2 Related Work

There is an increasing need for digital libraries to manage and share the vast variety of scientific data generated from different applications [3]. Even when sensor data are stored in clouds or used in grids, they need a mapping handling mechanism, like ours, to interoperate. DSpace@MIT [4] is an example repository that supports metadata creation and is built to store, share and search digital materials such as conference papers, images, scholarly articles and technical reports etc. Stanford Digital Library Project (SDLP) [5] proposes a metadata infrastructure to achieve

interoperability among heterogeneous digital data, thus providing a uniform access to such datasets or collections. SDSC (San Diego Supercomputer Center) Storage Resource Broker (SRB) [6] along with a metadata catalog component, MCAT provides a uniform interface to search over heterogeneous data sources distributed over the network. Although SRB is successful enough to store and query over a federated information system, it is not sufficient to manage and share real time data collected from wireless sensor networks [7]. In general, very little work can been found in literature on building digital libraries to store and search the data collected from pervasive applications.

## 3 Description of the System

### 3.1 Design

Based on our design, a user may have one of two different roles: He can either contribute or retrieve data to/from the repository. Whenever a user adds a new collection to the repository, he must also specify the metadata schema to describe that collection. Our system stores all such metadata schemas in the repository and stores the link between each collection and its corresponding metadata schema.

Our system provides a suitable query interface for the users who want to search over the repository for relevant data. The query interface consists of a suitable window to browse for different collections and metadata schemas. The user can either select a metadata schema from the list of collections or may use a "virtual" schema, which is not used in any collection. The user may either query over the entire library by selecting all collections or select some specific collections. As soon as the user asks to execute a query, the system first checks the stored metadata mappings to see whether some mappings already exist in the system from the schemas of the selected collections to the preferred virtual schema. Our interface collects and displays all such mappings and asks the user to select any of the following mapping choices:

1. Re-use an existing mapping.
2. Select the mapping used last time, which is the default choice for the preexisting mappings.
3. Specify a new mapping. This is the default option for the mappings that have not been specified yet. Since, initially, the system will not have any stored mappings; our interface will ask the user to enter the mappings first. A user may only provide mappings for the fields that he wants to query at that moment. However, he may enter a new mapping based on similar attribute names or RDF descriptions. He may also define a function to map one schema attribute to the others.
4. Select the recommended mapping from the system. Since our system stores mapping preferences into a user profile, it can identify the most commonly used mappings by different users for a pair of schemas and can recommend such mappings to the user.

Next, as soon as the user specifies his preferred mappings, the system retrieves the resulting records or data from the collections and returns those to the user. The user may either view all such records in a separate output window or browse individual collection manually and only view the results for that particular collection.

### 3.2 Implementation

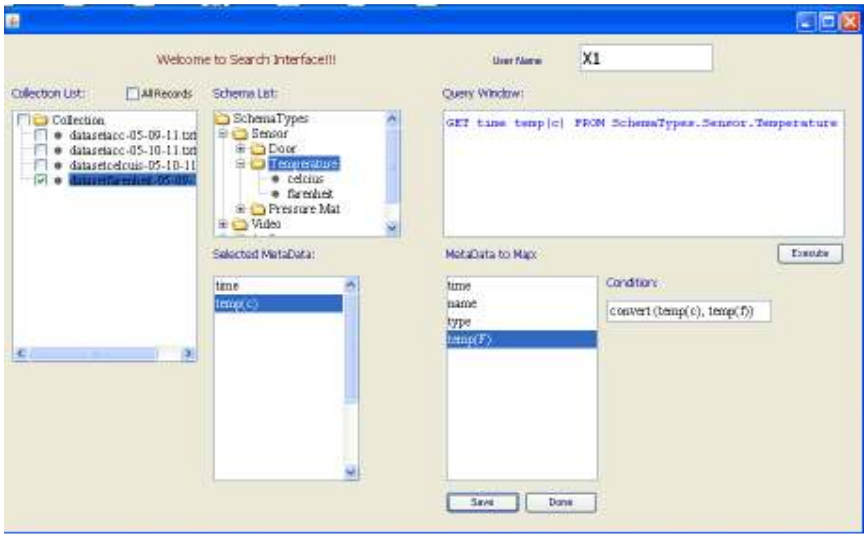


**Fig. 5.** A Screenshot of the Query Interface

We have implemented a prototype interface in Java to store and view mappings and collection data from the Library. A screenshot of our interface is shown in Fig. 5. From the interface, the user may express a query using the attributes listed under a particular metadata schema. Executing such a query is straightforward, as it does not require any mapping. However, the user does not need to use any such schema to search over the collections. Instead, the user can select any type and number of collections and can query using a virtual schema. In this case, he may be asked to provide the right pair-wise mapping for each selected dataset of new metadata schema to his preferred virtual schema, before the query could be executed. The user may also specify a mapping condition using the interface and as soon as he saves it, the mappings become part of the user profile. However, whenever he is done with all the mappings, the system executes the query and returns the results to the user.

### 3.3 Management of Mappings

In our framework, a mapping, $M_{ij}$, from schema $S_i$ to the schema $S_j$ is associated with a set of bindings $A_{jk}$:$f_{jk}(A_{i1},...A_{in})$ that derive the attribute value of $A_{jk}$ in $S_j$ from the attributes $A_{i1},...A_{in}$ in $S_i$ . The expression $f_{jk}$ may consist of simple

arithmetic operations, such as the comparison of an attribute value with a constant threshold, and string manipulation operations, such as string concatenation. These expressions are represented as abstract syntax trees and are stored in a mapping repository along with the rest of the mapping information. When a query is expressed in the schema $S_i$ and the mapping $M_{ij}$ is selected to query the data collections that conform to the schema $S_j$, then the query attributes are mapped to the $S_j$ attributes using the expression $f_{jk}$ and the derived query is used for querying the data collections that match $S_j$.

## 4 Conclusions and Future Work

In this paper, we develop a framework for the repository of heterogeneous sensor data. We also design and implement a prototype for a flexible query interface, which allows the user to search over the collections of different metadata schemas using minimal background knowledge. Our system collects and stores possible mappings from various users incrementally, which could work as a building block to derive commonly accepted mappings. As a future work, we are planning to use our mapping repository as a knowledge base to facilitate automatic metadata mappings.

## References


1. Temperature, Wireless, USB Data Logger or Watch Development Tool, http://focus.ti.com/docs/toolsw/folders/print/ez430chronos.html?DCMP=Chronos&HQS=Other+OT+chronos#technicaldocuments.
2. Sun SPOT World, http://www.sunspotworld.com/
3. Wallis, J.C., Mayernik, M.S., Borgman, C.L., Pepe, A.: Digital libraries for scientific data discovery and reuse: from vision to practical reality. Proceedings of the 10th Annual Joint Conference on Digital Libraries, Gold Coast, Queensland, Australia (2010)
4. DSpace@MIT, http://dspace.mit.edu/
5. Baldonado, M., Chang, C., Gravano, L., Paepcke, A.: The Stanford digital library metadata architecture. Intl. J. Digital Libraries **1**, pp. 108--121. (1997)
6. Storage Resource Broker, http://www.e-science.stfc.ac.uk/projects/storage-resource-broker/storage-resource-broker-.html.
7. Shankar, K.: Scientific data archiving: the state of the art in information, data, and metadata management. (2003)