UNIVERSITEIT VAN AMSTERDAM

METHODOLOGIES FOR RETRIEVAL, MANAGEMENT AND MEDIATION OF SENSOR DATA

Paul J. Manchego

Thesis submitted to the faculty of Science of University of Amsterdam in fulfillment of the requirements for the degree of

Master of Science
in
Computer Science

Supervisors:
Prof. dr. R.J. Meijer,
dr. G.D. van Albada,
University of Amsterdam,
The Netherlands

November 2005

Keywords: sensor data collection services, sensor data management, sensor data presentation services, applied telematics, internet services, Virtual School Lab.

METHODOLOGIES FOR RETRIEVAL, MANAGEMENT AND MEDIATION OF SENSOR DATA

Paul J. Manchego

Abstract

Collecting and processing data from heterogeneous sensor networks to make them available for education are some of the most challenging problems in management of sensor data. Environmental data which are heterogeneous are collected at various geographical locations for scientific studies and operational uses. The intrinsic problem of searching these scientific datasets is compounded by the heterogeneity of data and queries, thus limiting their availability for educational purposes. The data of sensor networks such as *Landelijk Meetnet Luchtkwaliteit (LML)* which measures the quality of the air at different places in the Netherlands or *Geluidsnet* which provides the sound landscape of many cities are diverse and complex. Moreover, sensor data formats are not uniform making them difficult to archive and query.

One of the current trends of data harvesting in scientific community is towards a distributed digital library initiative. Digital libraries allow organizations to join together in consortia projects by pooling their information into one resource or content portal. However, these approaches may not be adequate for organizations who do not want to upload data into a "data pool". In view of this, I present here a sensor data management system to address the issues of data collection and management for disparate sensor networks. The heterogeneities in these data are addressed based on the autonomy of data sources in a federation of sensor networks. A prototype implementation using web technologies that collect data from various sensor data sources has been developed for the *Virtual School Lab* project which aims to realise a federation of multiple sensor networks to make sensor data available for research and educational purposes.

Acknowledgments

I would like to thank my research advisor Dr. Rob Meijer for his guidance and support during my research. His advice, patience and optimism have helped me tremendously in the realization of this work. This has been an amazing learning experience. I would also like to thank Dr. Dick van Albada for providing valuable suggestions and comments.

I wish to express my gratitude to Microsoft Netherlands. Special Thanks go to Marcel Westra, Enterprise and Partner Group, Microsoft B.V., for providing useful development software tools. Thanks also to Bas van Oudenaarde, Advanced Internet Research Group, UvA, for his invaluable attention.

This thesis is dedicated to my colleagues at UvA. They have taught me to fight hard, face the adversities and be happy. It has been an honor to work with them in these past years.

Table of Contents

Table of Figures

Table of Tables

Chapter 1: Introduction

1.1. Motivation

With the advent of sensor networks that collect environmental data, such as *Landelijk Meetnet Luchtkwaliteit (LML[1])*, unprecedented amounts of multi-variant data are being published on the internet. They can help reveal the extent of the influence human activity has upon the environment and hopefully identify areas where changes can be made to prevent further damage. In combination, and without the inherent problems posed by heterogeneous data formats and inconsistent semantics, these new data offer great potentials for researchers to better understanding of the Earth's physical systems.

The demand for sensor data is extending beyond the scientific community to a broad range of users such as policy makers, educators, business people and the general public. For example, *Geluidsnet*[2] operates a network of sound level meters that measure the noise level in some urban areas like the Schiphol Airport in the Netherlands. The municipality of Amsterdam discusses the regulation of noise levels with the Air traffic control officials considering measurements provided by *Geluidsnet* [1].

However, users are confronted with some challenges and issues: they need to find out how to collect these data promptly and effectively. Sensor data are presented with different formats that require substantial efforts to process in order to be able to make any effective use of them.

In addition, the growing number of organizations owning sensor networks and their applications which process and publish sensor data is resulting in isolated sensor networks with incompatible data formats. Each sensor network has its own interface to access data values. In view of this, *Virtual School Lab (VSL[3])* aims to realise a federation of multiple sensor networks to make sensor data available for research and educational purposes.

1.2. Sensor Networks

A sensor network consists of a number of sensor nodes that combine physical sensing capabilities such as temperature, concentration, etc. with networking and some computation capabilities [2]. Conventionally, a sensor node has a general-purpose CPU to perform computation and a small amount of storage space to save

[1] LML is an environmental monitoring sensor network owned by The National Institute for Public Health and the Environment (RIVM) which is a recognized centre of expertise in the fields of health, nutrition and environmental protection in The Netherlands. Available: http://www.lml.rivm.nl.

[2] Geluidsnet operates a low-cost sensor network that measures noise levels. The measuring units are located in different places in The Netherlands such as the Schiphol Airport. The monitoring stations are housed by the local community; private people, schools and companies. Available: http://www.geluidsnet.nl.

[3] Virtual School Lab is an initiative of Virtual Laboratory for e-Science project (VLe). Available: http://www.virtualschoollab.nl

program code and data. Since nodes are usually not connected to a fixed infrastructure, they use batteries as their main power supply.

A sensor node has one or more physical sensors attached that are connected to the physical world. Examples sensors are temperature sensors, light sensors or passive infra-red (PIR) sensors that can measure the occurrence of events (such as object detections) in their vicinity.

Sensor readings are usually time-stamped. If an application cares about the current state of the network, readings from the network have to be updated relatively frequently since sensor data becomes out-dated fast if new events are happening in the network [3].

Sensor networks are distributed to measure and monitor a physical environment, such as tracking objects throughout an area or measuring environmental conditions in a large area. Due to the multitude of sensor nodes deployed, there is usually a huge number of data records generated. For example, in environmental monitoring applications, sensor readings are generated every few seconds (or even faster), thus the total volume of data generated is large.

However, not all sensor readings are of interest to users. For some sensor types, their data might change rapidly, and thus be outdated rather quickly, whereas for other sensors, their value changes only slowly over time. Example sensors of the first type are PIR sensors that sense the presence of objects; example sensors of the second type are temperature sensors that in steady state have a small bounded derivative. For applications that require only approximate results, buffering previous results for the second type of sensors and lowering the query update rate would be desirable.

Inherent to data from a physical measurement is uncertainty regarding the true value of the measured quantity. Conventionally, one of the sources of uncertainty in sensor readings is noise. In order to correct erroneous sensor readings, calibration is needed [4].

For many applications individual sensor readings are of minor importance, and users are usually interested in aggregates that combine a set of sensor data readings into a single, more robust statistic [5].

1.3. Management of Sensor Data

The management of sensor data involves techniques associated with acquisition, processing and presentation of data from different sources. In the scientific community several projects have focused on the management of sensor data. For example, *OPeNDAP*[4] has developed a software framework that simplifies some aspects of scientific data networking, allowing simple access to remote data. Local data can be made accessible to remote locations regardless of local storage format by using web servers. Existing, familiar data analysis and visualization

[4] OPeNDAP: Open-source Project for a Network Data Access Protocol. Available: http://www.opendap.org

applications can be transformed into clients that are able to access remote served data.

The Unidata Community Portal (Unidata[5]*)* helps researchers and educators acquire and use earth-related data. Most of the data are provided in real time, that is, the data are sent to the participants as soon as the observations are made. *Unidata* is a data facilitator, not a data archive center. They provide a mechanism named *Internet Data Distribution system (IDD)* that allow participating users to subscribe to streams (feedtypes) of current data that interest them.

The Science Environment for Ecological Knowledge (SEEK[6]*)* is a system designed to facilitate acquisition and archiving of ecological and biodiversity data. *SEEK* participants are building an integrated data grid (EcoGrid) for accessing a wide variety of ecological and biodiversity data and analytical tools for utilizing these data stores to advance ecological and biodiversity science. An intelligent middleware system (The Semantic Mediation System, SMS) will facilitate integration and synthesis of data and models within these systems.

In the educational community, Global Learning and Observations to Benefit the Environment (GLOBE[7]) is a worldwide hands-on, primary and secondary school-based education and science program that improve student understanding of science. GLOBE encourages students to utilize data to help answer questions about how the environment around them works. Through investigation projects, students do science, learning the importance of creating hypotheses, analyzing data, drawing conclusions, and reporting their results. GLOBE Student Investigations are scientific projects conducted by GLOBE students that include the use of GLOBE data or protocols. By publishing their investigation reports on the GLOBE Web site, students share their valuable findings with the rest of the world.

Although these approaches provide mechanisms to manage sensor data, they do not directly address the needs of collecting and querying data from heterogeneous sensor data sources for educational purposes. The goal of *Virtual School Lab* is to make data available and usable for educational purposes allowing school children to retrieve and query sensor data promptly and effectively.

Table 1 shows an example of sensor data collected from the *LML* sensor network. When properly processed and presented, these data can be used by school children to analyze the quality of the air at different places in the Netherlands.

[5] Unidata provides a broad array of data for use in geoscience education and research. It also provides a mechanism for remote access to a growing collection of geoscience data. Available: http://www.unidata.ucar.edu

[6] This SEEK cyber infrastructure is being developed to address the many challenges associated with data accessibility and integration of large-scale biocomplexity data in the ecological sciences. Available: http://seek.ecoinformatics.org/

[7] GLOBE is an interagency program funded by NASA and NSF. It is also a cooperative effort of schools in partnership with colleges and universities, state and local school systems, and non-government organizations. Internationally, GLOBE is a partnership between the United States and other countries.

Station	Date Time	SO2	PM10	NH3	CO	CO3	NO	NO2
111	2005-09-01 09:10:05	1	18	1	210	24	1	23
111	2005-09-01 10:10:05	1	15	1	210	20	1	20
111	2005-09-01 11:10:05	2	20	5	250	20	0	28

Table 1: Data sample from the *LML* sensor network. School children can analyze these data to determine the environmental consequences of releasing air pollutants to the environment. For example, acid rain occurs when emissions of sulfur dioxide (SO2) and oxides of nitrogen (NOx) in the atmosphere react with water and air pollutants to form acidic compounds.

1.4. Problem Definition

The main focus of this research is to design a sensor data management system that facilitates operational and administrative tasks of *Virtual School Lab*.

Virtual School Lab aims to generate a federation of sensor networks for research and educational purposes. It makes data from different sensor data sources available to school students and researchers to better understanding of the Earth's physical systems and other operational uses.

1.5. Approach

I've designed and implemented *Sensor Data Management System (SDMS)*, a set of software tools that assists *Virtual School Lab* to achieve their goals.

One of the aims of *SDMS* is to be data facilitator for school children allowing them to query sensor data from different sources with different schemas, access mechanisms and resource considerations. *SDMS* specifically addresses data collection, processing, and presentation of different sensor data sources. The system assumes that sensor data sources are autonomous sensor networks.

Virtual School Lab functional requirements are first analyzed and then specified. They are input to the design of the architecture presented in this thesis. A prototype implementation of the architecture is described at the end of this thesis.

Functional requirements describe what the system is supposed to do by defining functions and high-level logic. A prototype is a model of the system. It doesn't have to be fully functioning; it merely has to be illustrative of what the final system should look and feel like. It is constructed to depict concepts, design alternatives, and screen layouts [6].

In *Virtual School Lab*, federation is the organizational model used to group the participating autonomous sensor networks. Although federations have been used for centuries as an organizational model, the concept has rarely been applied to management of sensor data sources. A federation is an association of autonomous

partners that agree to abide by certain interface standards, business practices, and expectations of conduct to achieve a common goal.

Virtual School Lab deals with data from self-governing sensor data sources. The autonomous nature of the data sources with respect to the data collection, semantic nature and implementation plays a significant role in the architecture of the system. This autonomy yields persistent differences in formats and semantics of sensor data. In the vision of *Virtual School Lab*, sensor data consumers shouldn't be concerned about such differences. Additionally, *Virtual School Lab* aims to have persistent data stores with information about the physical processes that are being measured by the sensors. Details related to the collection and calibration of sensor data should be hidden from users.

A schematic representation of the context in which *Virtual School Lab* develops is shown in Figure 1.

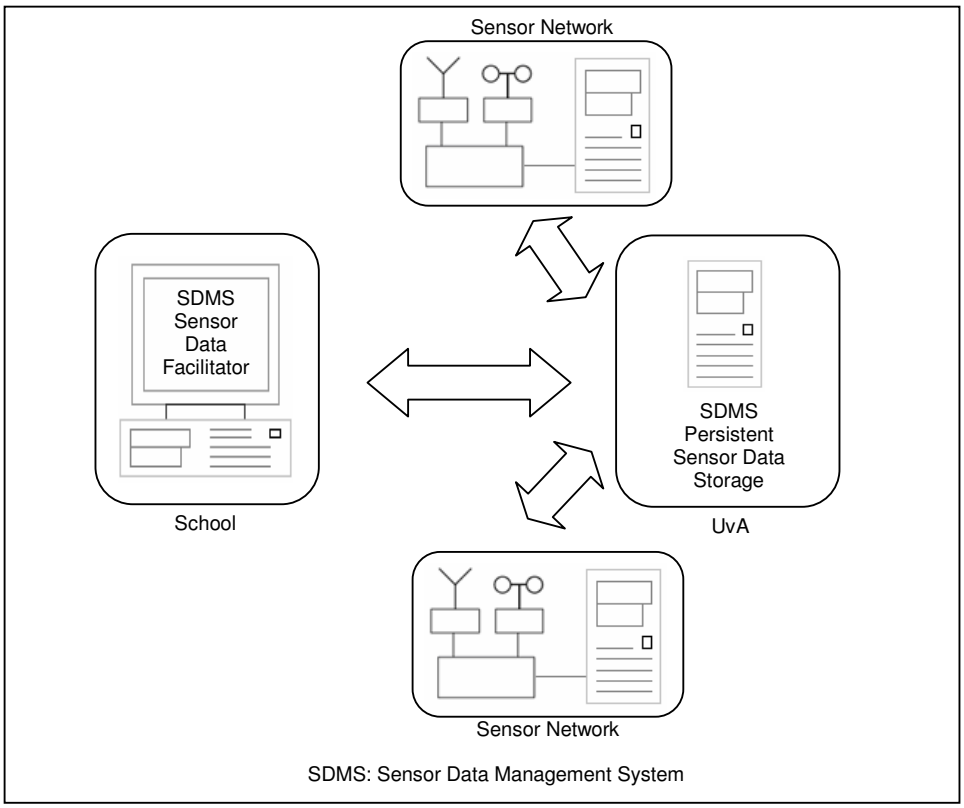

Figure 1: Schematic representation of the *Virtual School Lab* environment. The *SDMS* data storage accumulates data from heterogeneous sensor networks. Sensor data is processed and presented to school children by means of a web-based data facilitator provided by *SDMS*.

By means of the *Sensor Data Management System (SDMS)*, students will be able to select what they want to know and the system will present relevant sensor data in a user-friendly way. *SDMS*, in the context of *Virtual School Lab* project, could let schools educate children about the environment and the important role sensor data plays on it.

I have grouped participants in *Virtual School Lab* based on their roles. They are Producers, Consumers, Service Providers and Operators.

Producers are autonomous sensor networks wishing to participate in the Virtual School Lab project.

Sensors networks typically provide date, time and location stamps along with the observation data that their sensor nodes are designed to collect. Conventionally, sensor networks use the Web as the bridge to the outside world.

Nowadays, structured text schemas conforming to the Web's eXtensible Markup Language (XML) can be used to publish formal, standardized, machine- and human-readable descriptions of the sensor's capabilities, location and interfaces as well as the observations themselves. Web brokers, clients and servers then can parse and interpret XML data, enabling automated Web-based discovery of the existence of sensors and evaluation of their characteristics based on published descriptions [7].

Consumers are schools or other private organizations wishing to use sensor data for research and educational purposes. Operators are the administrators of *Virtual School Lab*. They could use *SDMS* to facilitate some of their tasks.

Additionally, the *SDMS* system will allow participants named Service Providers to add new functionality to the system. Service Providers consume raw sensor data, process them and produce useful information.

1.6. Outline

The remainder of the thesis is organized as follows. Chapter 2 describes the Virtual School Lab project and explains the importance of SDMS to facilitate operational and administrative tasks. Chapter 3 presents the architecture of SDMS. A set of functional and non-functional requirements are used as input for the system architecture presented in this chapter. This chapter also includes a set of UML diagrams used to represent the expected behavior of the system. Chapter 4 describes the prototype used to validate the architecture proposed in the previous chapter. Chapter 5 presents sample use scenarios for the system and includes a detailed description of the tools provided by the prototype to facilitate administrator and school children tasks. Chapter 6 summarizes the work and concludes with a summary of future enhancements.

Chapter 2: Virtual School Lab

2.1 Introduction

Virtual School Lab, an initiative to realise a federation of sensor networks for research and educational purposes, relies on the work of some self-contained research disciplines such as sensor networking.

Sensor networking has received much attention in recent years as a number of research groups have constructed small computational and communication devices, such as the Berkeley motes of the *Smart Dust Project*[8] which aims to build a self-contained, millimeter-scale sensing and communication platform for a massively distributed sensor networks [8]. The research in this area has not only focused on constructing small computational and communication devices but also developing an operating system named *TinyOS*[9] for wireless embedded sensor networks [9].

A common approach in sensor networking to collect data is that once data has been received by a base station, there is a direct path to desired consumers of that data [10].

In *Virtual School Lab* with a number of heterogeneous sensor networks, sensor data is first collected from different data sources into a persistent data store and then are made available to the consumers. No direct path between sensor data sources and consumers is needed since the persistent data store is always updated periodically. This data-centric storage approach will make sensor data highly available and enforce acceptable levels of quality of service (QoS) to the consumers. However, a different approach is needed if real-time sensor data are required. Real-time data retrieval would entail direct paths between consumers and sensor data sources.

Virtual School Lab envisions an extremely rich set of services providers that can process raw sensor data and produce more relevant information for educational purposes. Furthermore, *Virtual School Lab* should not be constrained to a small set of interfaces for specific sensor networks. It aims to evolve dynamically to support a wide range of sensor data sources.

2.2 Project Description

Virtual School Lab is a project with the ambition to exploit the potential of sensor data in education. Consequently, the project has started to propose a set of

[8] The goal of the Smart Dust project is to demonstrate that a complete sensor/communication system can be integrated into a cubic millimeter package. This involves advances in miniaturization, integration, and energy management. Available: http://robotics.eecs.berkeley.edu/~pister/SmartDust/

[9] TinyOS is an open-source operating system designed for wireless embedded sensor networks. It features a component-based architecture which enables rapid innovation and implementation while minimizing code size as required by the severe memory constraints inherent in sensor networks. Available: http://www.tinyos.net/

practical assignments for school children in which sensor data plays an important role.

In my vision, the most important use of sensor data is not only to teach children formal scientific concepts and theories but precognitive schemas. This would entail using sensor data to help children understand in an intuitive way some of the patterns that occur in the real world. An example would be the notion that the quality of the air can vary according to the levels of pollutants released to the environment such as sulfur dioxide (SO2). *Virtual School Lab* should assist school children to develop a qualitative understanding based on quantifiable information.

There are existing sets of sensor data that could be exploited for education. For example, the *LML* network generates data related with the quality of the air at different places in the Netherlands (see Table 1). *Virtual School Lab* makes these data available to school children for educational purposes.

Figure 2: *Virtual School Lab* aims to get full benefits of Information Technology (IT) to collect, process and present sensor data. Figure 2 shows a screen caption of Google Earth[10] interface used in conjunction with SDMS. This coupling would allow school children to retrieve geospatial information of stations where sensor data are collected.

[10] Google Earth combines satellite imagery, maps and the Google search engine to make world's geographic information available to internet connected computers. Available: http://earth.google.com/

In *Virtual School Lab*, collecting, processing and presenting sensor data for educational purposes are the major activities. *Virtual School Lab* aims to get the full benefits of information and communication technologies (ICT[11]) to accomplish its ambitions (see Figure 2).

Virtual School Lab consists of learning modules (practical assignments) and software modules (tools) which are linked together and work to achieve a common goal: to realise a federation of sensor networks for educational purposes. The software tools of *Virtual School Lab* facilitate operational and administrative tasks. For example, they help operators to register new members. They also help school children to access sensor data from disparate sensor networks. The learning modules consist of a set of practical assignments that exploit the use of sensor data and bring school children new opportunities to understand physical phenomena via experimentation. In this thesis, software tools of *Virtual School Lab* have been grouped together to create *Sensor Data Management System*.

Virtual School Lab aims to integrate its functionality into *VLAM-G*, the Grid-based Virtual Laboratory Amsterdam, which provides a science portal for distributed analysis in applied scientific research. Integration would offer scientists the possibility to carry out experiments with sensor data.

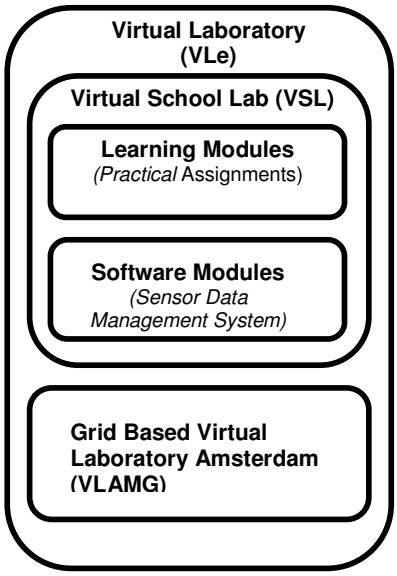

Figure 3: *Virtual School Lab* consists of two components that cooperate to achieve a common goal. *Tools* haven been grouped together to create *Sensor Data Management System*.

As seen in Figure 3, Virtual School Lab is an initiative of Virtual Laboratory for e-science (VL-E), a project supported through the ICES/KIS[12] fund. The VL-E project aims to contribute to spreading the e-Science paradigm and creation of the required e-Science infrastructure.

[11] Information and Communication(s) Technology (ICT) is the technology required for information processing. In particular the use of computers and software to convert, store, protect, process, transmit, and retrieve information.
[12] VL-E is supported by ICES/KIS fund. Available: http://www.minocw.nl/ices/

The focus of the next chapter is to propose the architecture of Sensor Data Management System. In order to accomplish this goal, I first present a detailed description of the different steps that the architecting process involves.

Chapter 3: The Architecture

3.1 Introduction

Software architecture describes the high-level structure of a software system, comprising software components and the relationships among them. The process to create the architecture includes the following steps: [11]

- Requirements: Establish and document the architectural requirements. It includes specification of functional requirements and system qualities or non-functional requirements.
- Specification: Define the architecture. It includes system's decomposition into components, definition of their responsibilities and interconnections between them.
- Validation: Validate that the architecture meets the requirements. It involves develop prototypes or proofs-of-concept. Validation is made to prove out critical aspects of the architecture.

These steps are best conducted iteratively, as shown in the Figure 4.

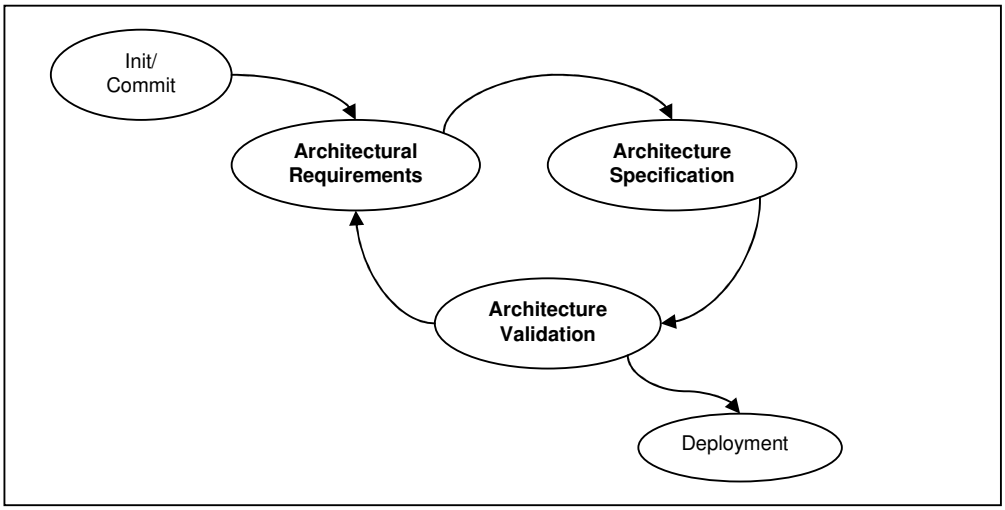

Figure 4: Architecting process steps for Sensor Data Management System (SDMS). As suggested in Visual Architecting Process (VAP) [11] they are best conducted iteratively.

3.2 Architectural Requirements

The level of success of the software architecture can be assessed based on its satisfaction of two complementary types of requirements: The first type is a set of user-centered requirements that concentrate on the functionalities provided to the system's user. They are referred as functional requirements (FRs). The second type is a set of context-centered requirements that impose restrictions that may affect greatly the operational environment and design choices. They are referred as non-functional requirements (NFRs) [12] [13].

3.2.1 Functional Requirements (FRs)

Definition of functional requirements for large scale systems such as *Sensor Data Management System* is non-trivial. Techniques are required to specify them. Furthermore, when building systems which are expected to have some longevity, it is important that people looking at the documentation can understand what is involved. Another important consideration is that these techniques should facilitate close analysis of the design to ensure, at early stages, that systems are complete in meeting their requirements and consistent in their operation.

In order to define the functional requirements of the system, a statement establishing high-priority goals of the system users is firstly issued. Then documentation of functional requirements is achieved by translating user goals into a set of *use-cases*[13].

Statement of high-priority goals:

In order to successfully create a viable and extensible federation of sensor networks for educational purposes, the high-level goals of the users of Virtual School Lab can be briefly described as follows:

Organizations owning sensor networks, schools wishing to participate in the *Virtual School Lab* initiative, and other interested organizations have to be registered as *members* of the federation. A *federated member* is an organizational unit (school, sensor network, etc.) or individual that plays a specific role in the federation. The possible roles in the federation are: sensor data producer, consumer, service provider and operator.

Policies of the federation determine activation or deactivation of the membership. A member who has not been active for a period of time, stated in the policies of the federation, will be removed from the membership list and it needs to re-apply for the reinstatement of its membership.

Producers are autonomous sensor networks that wish to participate in the federation as sensor data sources. Conventionally, producers publish relevant sensor data periodically. *Consumers* are schools or other private organizations wishing to use sensor data for research and educational purposes. Additionally, *Virtual School Lab* allows participants named *service providers* to consume raw sensor data process them and produce useful information for school children.

Operators are responsible for the supervision and administration of *Virtual School Lab*. Operators use a *web-based tool* that facilitates administrative tasks such as new member registration. *Operators* play an important role in the federation and *Virtual School Lab* aims to automate some of their tasks. Furthermore, *Administrators* of organizations participating as federated members of *Virtual School Lab* can be assigned the role of *operators* allowing collaboration among the participants.

[13] A use-case defines a goal-oriented set of interactions between external actors and the system under consideration.

The process of registration is initiated when interested organizations apply for membership by filling an electronic form provided in the *Virtual School Lab* website. Verification of membership details is made by the *operators*. A web-based tool assists *operators* in the process of the member registration. Selection of the appropriate role new member plays in the federation is required. After the operator has successfully completed the registration of the new member, a confirmation message is sent via email to the new federated member.

Virtual School Lab aims to exploit the potential of sensor data for educational purposes. *School students* are guided through the process of developing an experiment, gathering sensor data, and analyzing the results by means of practical assignments proposed by *Virtual School Lab*. The students use a *web-based tool* that *facilitates* access and analysis of sensor data from federated data sources. School students can also develop their own tools to document their procedure and to gather important data from internet. At the end of the experimentation process, they produce a final report describing their results.

Virtual School Lab facilitates data access to school students and researchers who have previously registered as *users* and authenticate themselves as members of one of the organizational units participating in the federation. For example, students from participating schools are *users* of Virtual School Lab. An automated administrative task named *authentication service* checks user credentials, determine access rights and data availability according to the *Virtual School Lab* policies and federated roles.

By means of the web-based tool that facilitates access to sensor data, users may query data from many sensor data sources including, but not limited to, producers of the federation.

The *data collection service*, an automated task, updates persistent data stores of the federation with new sensor data published by the federated data sources.

Virtual School Lab assists school children to select and query sensor data sources by means of data flow descriptors. Operators use a web-based tool that allows them to model and publish new descriptors enabling school children to use them to easily retrieve sensor data. Data flows descriptors define data paths that include data sources, service providers and consumers.

The notification service delivers notifications that are posted to the federation by operators. For example, posts stating remarkable changes in the normal values of sensor data would be desirable.

In order to allow school students to discover valuable data sources in the federation, *the discovery service* allows users to identify suitable sensor data sources. This capability offers school students a more flexible and rapid mechanism to find valuable information to complete the practical assignments proposed by *Virtual School Lab*. Although school students can use internet search engines to find relevant data sources, *Virtual School Lab* facilitates their access and retrieval.

Documentation of functional requirements:

To document functional requirements I will make use of use-cases diagrams. A use-case defines a goal-oriented set of interactions between external actors and the system under consideration. Actors are parties outside the system that interact with the system [14]. An actor may be a class of users, roles users can play, or other systems. Cockburn [15] distinguishes between primary and secondary actors: A primary actor is one having a goal requiring the assistance of the system. A secondary actor is one from which the system needs assistance.

A use-case is initiated by a user with a particular goal in mind, and completes successfully when that goal is satisfied. It describes the sequence of interactions between actors and the system necessary to deliver the service that satisfies the goal.

Thus, use-cases capture who (actor) does what (interaction) with the system, for what purpose (goal), without dealing with system internals. A complete set of use-cases specifies all the different ways to use the system, and therefore defines all behavior required of the system, bounding the scope of the system.

Table 2 presents the actors who interact with *Sensor Data Management System*. Table 3 specifies a set of use-cases that are used to represent the behavior of the system. Actors and use-cases where extracted from the statement of high-priority goals presented in the previous section.

Figure 5 represents the expected behavior of the system by means of a UML[14] use-cases diagram. This behavior implies interaction with primary and secondary actors. Secondary actors are presented as a set of services that assist the system to achieve its goals. In Figure 5 use-cases appear as ovals grouped according to the goal achieved. Actors are represented with stick figures.

[14] Unified Modeling Language (UML) is a semi-formal modeling language that provides a diagrammatic notation and semantics using mature object-oriented concepts. By using UML diagrams it is possible to design a well documented system. Furthermore, UML makes possible to present the designs in an almost universal language that is widely accepted within scientific community and IT industry [16].

Actor	Type	Description
Operator	Primary	Individuals responsible for the supervision and administration of Virtual School Lab.
Consumer	Primary	Schools and private organizations wishing to use sensor data for research and educational purposes.
Producer	Primary	Autonomous sensor networks wishing to participate in the federation as sensor data sources.
Service provider	Primary	Participants which consume raw sensor data process them and produce useful information for school children. Information are made available through Virtual School Lab.
School student	Primary	School children who register to use the system.
Authentication service	Secondary	Automated task that checks membership credentials.
Data collection service	Secondary	Automated task that updates persistent data store with data from sensor data sources.
Notification service	Secondary	Automated task that delivers notifications to the users of Virtual School Lab.
Discovery service	Secondary	Identify suitable sensor data sources in the federation
Data flow processing service	Secondary	Automated task that process data flow descriptors. This service access persistent data store.
Data presentation service	Secondary	Automated task that present sensor data to the school children in user-friendly formats.
Web proxy service	Secondary	Contact service providers.

Table 2: Actors of *Virtual School Lab*.

Use Case	Description
Apply for membership	Organizations whishing to become federated members initiate the registration process by fill the electronic registration form provided by Virtual School Lab.
Register member	New federated members (producer, consumer, service provider) are registered by Operators. A confirmation message is sent via email.
Update membership	Operator updates membership details.
Query sensor data	Registered users (e.g. school children from participating schools) can query sensor data from federated data sources.
Process data flow descriptor	The automated task "data flow processing service" process flow descriptors that specify data paths including sensor data sources, service providers and consumers.
Present sensor data	The automated task "data presentation service" presents data in user friendly formats.
Collect Sensor Data	The automated task "data collection service" updates periodically the persistent data store of Virtual School Lab.
Publish sensor data on internet	Sensor data sources publish data periodically on internet.
Check user credentials	The automated task "authentication service" checks user credentials, determine access rights and data availability.
User register	For example school children register in order to use the system.
Login System	Registered users (e.g. school children) login the system.
Post notification	Operator posts notifications.
Deliver notification	The automated task "notification service" delivers notifications posted by the operators.
Search for data source	School children identify suitable sensor data sources in the federation.
Discover suitable sensor data sources	The automated task "discovery service" facilitates searching of sensor data sources in the federation.
Provide useful information	Service provider facilitates new functionalities to the system such as provide useful information to school children.
Contact service provider	The automated task "web proxy service" contact service providers.

Table 3: Use-Cases of *Virtual School Lab*.

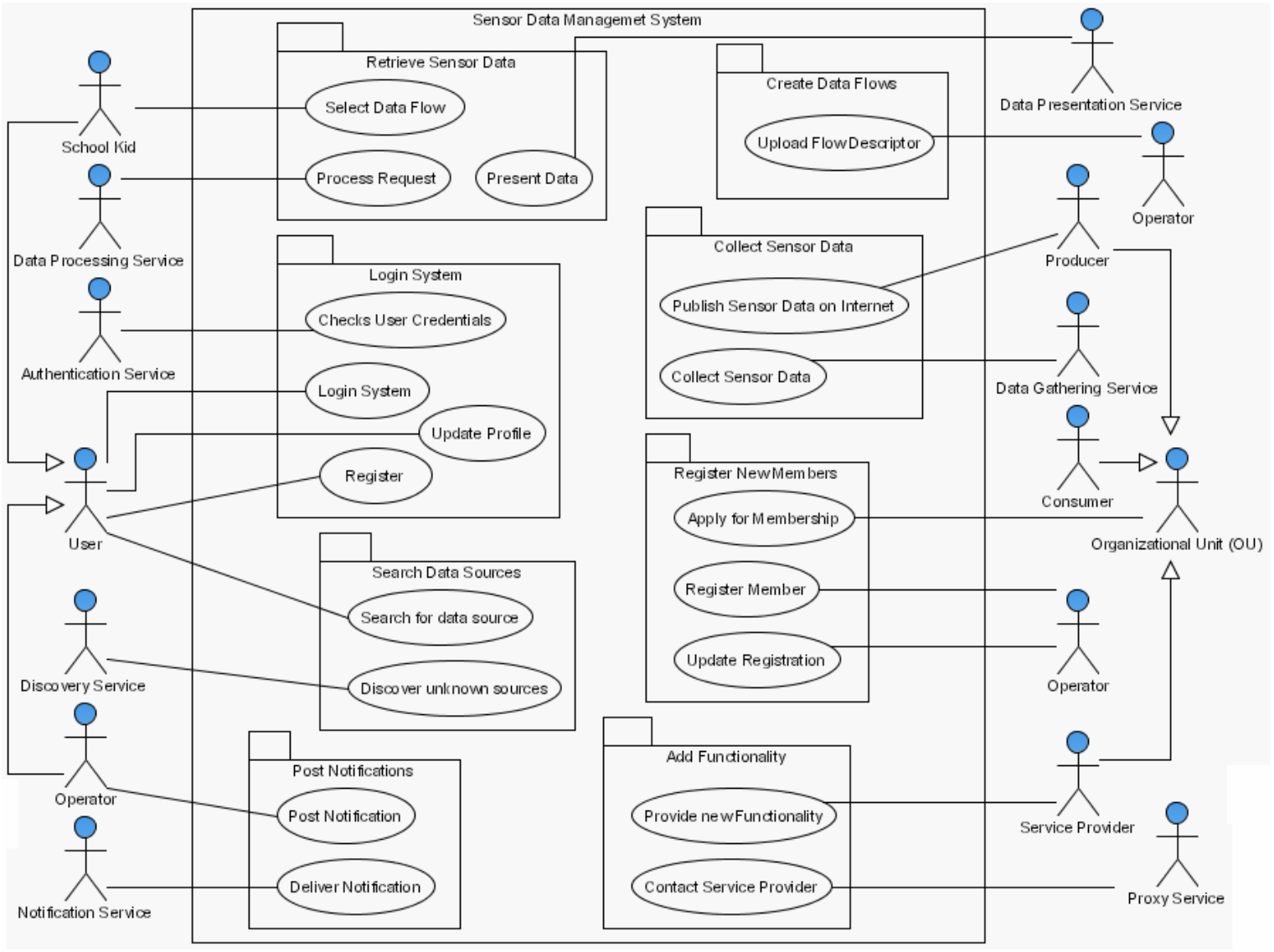

Figure 5: Use-Cases Diagram of *Virtual School Lab*.

3.2.2 Non-Functional Requirements (NFRs)

Non-functional requirements are properties and qualities the software system must possess while providing its intended functional requirements or services (see section 3.2.1). These types of requirements have to be considered while developing the functional counterparts. They greatly affect the design and implementation choices made for the architecture [12].

In the following, I briefly describe the four categories of non-functional requirements that may be imposed on Sensor Data Management System.

Operational requirements: these requirements specify the environment in which the software will be running, including, hardware platforms, software frameworks, and operating systems.

The client computers at the schools and the servers that collect and facilitate sensor data to school children should work with Microsoft operating systems.

The minimal operational configuration for system's clients can be described as follows:

- Personal computer with Windows XP Pro SP2 operating system. (.NET framework 1.1 is a component of the operating system).
- Microsoft Office 2003 installed.
- Office 2003 Web Components installed (OWC 11).
- Internet Explorer 6 SP1. Configure zone security settings to add Virtual School Lab as trusted website.
- ATOM feed aggregator installed. It allows monitoring notifications of the system.

The standard configuration for system's clients is as follows:

- Personal computer with Windows 2003 SP1 operating system. (.NET framework and Internet Information Server are components of the operating system).
- Microsoft Office 2003 installed.
- Office 2003 Web Components installed (OWC 11).
- Internet Explorer 6 SP1. Configure zone security settings to add Virtual School Lab as trusted website.
- ATOM feed aggregator installed. It allows monitoring notifications of the system.
- Microsoft Visual Studio 2005 Express Editions: Visual C# and Visual Web Developer, SQL Server. These tools are required to allow school children to deploy their own service providers.

The required configuration system's servers can be described as follows:

- Industry standard storage and web servers optimized for Windows 2003 SP1 operating system. (.NET framework and Internet Information Server are components of the operating system).
- Microsoft Office 2003 installed.
- Office 2003 Web Components installed (OWC 11).

- Internet Explorer 6 SP1. Configure zone security settings to add Virtual School Lab as trusted website.
- Microsoft Visual Studio 2005 Enterprise Edition.
- Microsoft SQL Server 2005 Enterprise Edition.

Scalability requirements: these requirements specify how many users and sensor data sources can be supported by Virtual School Lab.

- To make Virtual School Lab highly scalable, a dynamic and extensible architecture is needed. The architecture of the system should support an increasing number of sensor data sources. The sensor data should be collected from data sources through high-speed internet.

- Virtual School Lab should support many users concurrently active. The architecture should rely on high-performance storage mechanisms.

Conformance to standards requirements: these requirements specify the ISO standards that the system should comply with.

- Extensible Markup Language (XML) should be used as primary format for data transferring in Virtual School Lab. XML is a simple, very flexible text format derived from SGML[15] (ISO 8879). Originally designed to meet the challenges of large-scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere.

- Web Service Description Language (WSDL) should be used as description syntax for web services. The WSDL defines the parameters needed to interact and communicate with a particular Web service component. Formally, the WSDL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint. Related concrete endpoints are combined into abstract endpoints (services). WSDL is used in conjunction with HTTP[16] GET/POST, and MIME[17]

Usability requirements: these requirements specify criteria for judging the degree of usability and user friendliness of the system.

- User-friendly formats should be used to disseminate information in Virtual School Lab. Sensor data should be made available to be easily understood by school children. Virtual School Lab aims to produce alternate formats to enhance accessibility. School children should be able to choose the format

[15] The Standard Generalized Markup Language (SGML, ISO 8879:1986) gave birth to a profile/subset called the Extensible Markup Language (XML), published as a W3C Recommendation in 1998.
[16] Hypertext Transfer Protocol - HTTP/1.1
[17] Multipurpose Internet Mail Extensions - MIME

they prefer. The goal of dissemination of sensor data in Virtual School Lab is allow school children to utilize them to complete their practical assignments.

3.3 Architectural Specification

The intent of the architectural specification presented in this thesis is to direct attention at an appropriate decomposition of the system without delving into the details of interface specification. Key constructs are identified, including significant architectural elements such as components and relationships among them, as well as architectural mechanisms.

The architectural specification of the system is documented by means of UML component diagrams. The diagrams identify the components and their interconnections. They represent the conceptual architecture of the system.

Figure 6 represents the component diagram of the whole system. The system consists of two sub-systems: the student and administrator subsystems. Both of them use the database component. School children interact with the student subsystem and operators with the administrator subsystem.

Figure 7 represents the component diagram for the administrator subsystem. Figure 8 shows the component diagram for the student subsystem.

Many of the design decisions I have made stem from the functional requirements shown in the previous section.

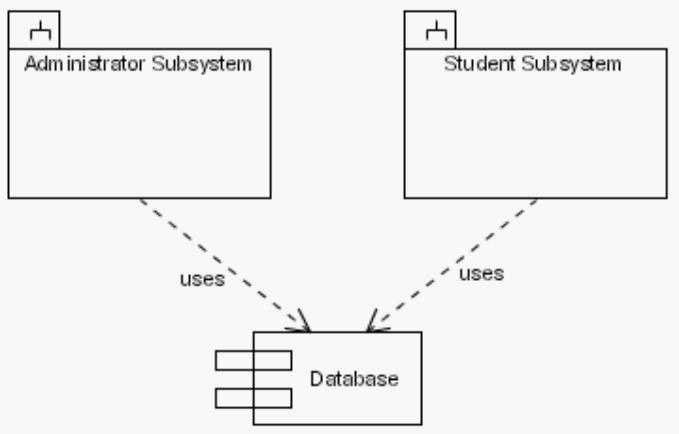

Figure 6: Component Diagram of *Sensor Data Management System (SDMS)*.

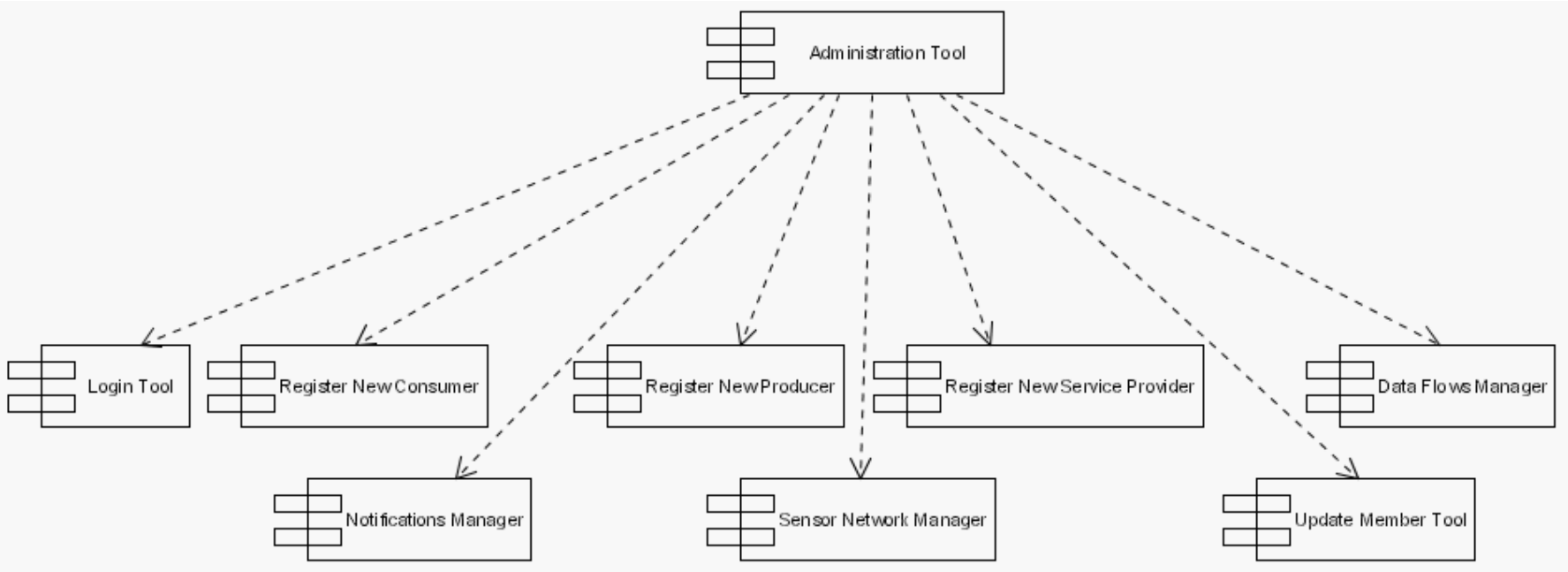

Figure 7: Component diagram of the administrator subsystem.

- **Login tool:** Includes the functionality needed to allow system users (e.g. school children) login the system.
- **Register new consumer:** It allows operators to register organizations as new consumers.
- **Register new producer:** It allows operators to register new sensor data sources. Sensor data are collected and made available to school children through the system.
- **Register new service provider:** It allows operators to register new service providers. Useful information produced by providers is made available to school children through the system.
- **Data flows manager:** Publishes new data flow descriptors allowing users to select them and retrieve sensor data.
- **Notifications manager:** It allows operators to automate generation of notifications.
- **Update member tool:** Modify membership details.
- **Sensor network manager:** This tool is used to administer low level sensors deployed according to the Virtual School Lab guidelines. (Not considered for the prototype).

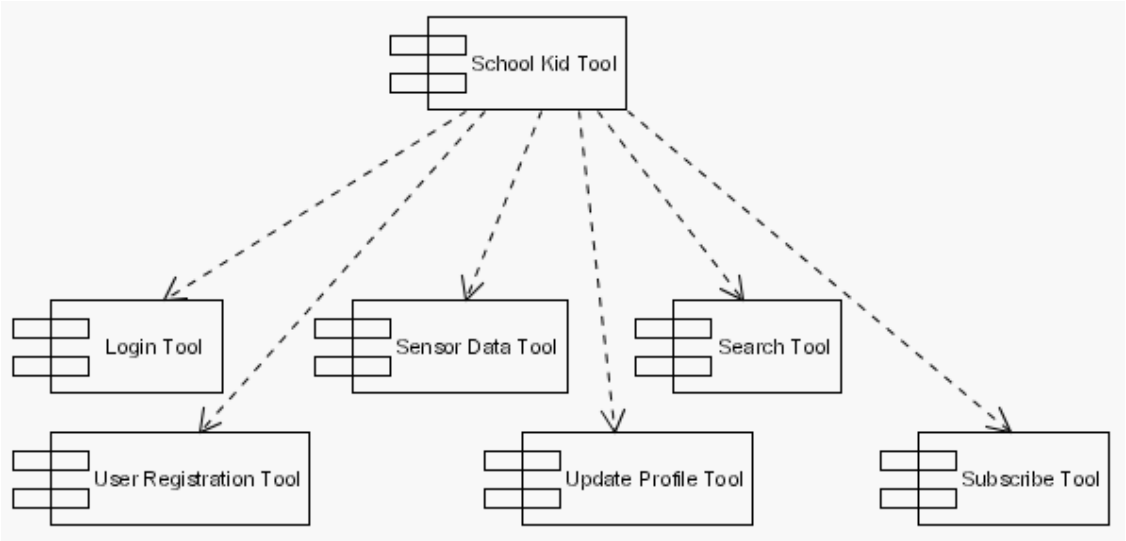

Figure 8: Component diagram of the student subsystem.

- **Login tool:** Includes the functionality needed to allow system users (e.g. school children) login the system.
- **Sensor data facilitator:** It allows users to select a data flow descriptor to retrieve sensor data. This tool presents sensor data in user-friendly formats.
- **Search tool:** It allows users to find suitable sensor data sources in the federation.
- **User registration tool:** Includes the functionality needed to register new system users.
- **Update profile tool:** Modify user registration details.
- **Subscribe tool:** It allows users to receive notifications delivered by the system.

Chapter 4: The Prototype

4.1 Introduction

The development of the prototype or proofs-of-concept is an important part of the validation process of the architecture presented in the previous chapter. The system should, according to current insights and functional requirements of *Virtual School Lab*, facilitate the student and operator tasks shown in Table 4.

Federated Role	Task
Student	Select the appropriate sensor data source through the data flow descriptors.
Student	Retrieve and query sensor data. Results are presented in user friendly formats.
Student	Register to use the system.
Operator	Register new producers. The system should be able to collect periodically sensor data from sources if they are published on internet. After collection, data is processed and made available to the users for educational purposes.
Operator	Register new service providers. The system should be able to contact provider when its functionality is required.
Operator	Register new consumers (schools).
Operator	Publish new sensor data descriptors. The system should be able to assist operator to create and publish data flow descriptors to be used by school children to easily retrieve sensor data.

Table 4: Tasks of administrators and students in *Virtual School Lab*.

Sensor Data Management System (SDMS) aims to facilitate quick access to sensor data from heterogeneous sensor networks. In order to accomplish this ambition, the prototype consists of a data collection service that support a wide range of sensor data sources which publish sensor data on the Web. Additionally, it supports other methods of sensor data collection such as push and pull mechanisms.

The rest of this chapter is organized as follows. I first introduce the core functionality of the prototype; I then explain the prototype's registration service. I explain the data collection service. I explain how useful information can be provided to school children via service providers. I then describe the push and pull mechanisms and the prototype's approach for real time sensor data retrieval. Finally, I explain the need for sensor data formats and metadata models to manage sensor data.

4.2 Core Functionality

This thesis argues that the prototype should be open and extensible. Consumers wishing to retrieve sensor data via the prototype should be free to use whatever network and system paradigm exists, while autonomous sensor data sources can always be used if they publish data in XML, HTML or XHTML formats.

Once a new sensor data source is registered via the prototype, it can be used by schools and private organizations at little additional hardware/software cost. The only possible infrastructure needed to retrieve data from a new sensor data source is a computational system and a web browser able to support the prototype client.

Figure 9 shows a schematic representation of the core functionality of the prototype.

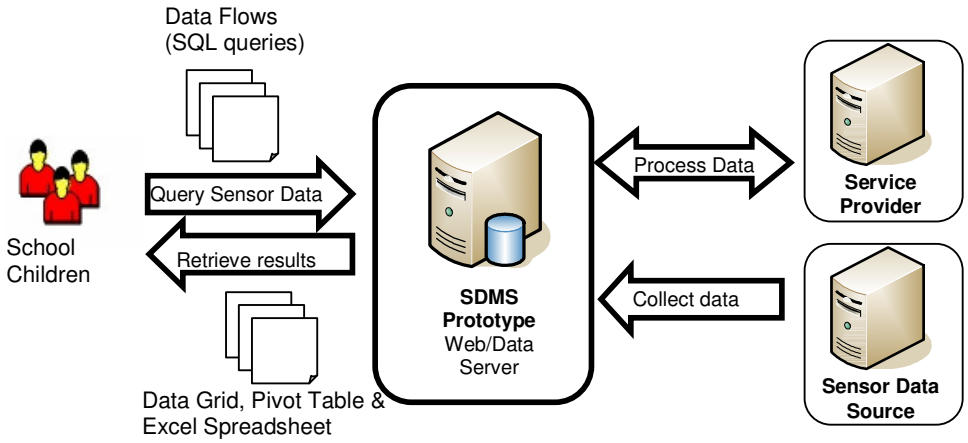

Figure 9: Schematic representation of prototype's core functionality.

As seen in Figure 9, school children query prototype's data storage and results are presented in user-friendly formats such as pivot table and excel spreadsheet. Although the prototype allows users to query and filter sensor data, other application-specific functionality can be offered by service providers. Service providers add new functionality to the system taking as input raw sensor data, processing them and producing useful information for school children.

4.3 Member Registration

The prototype allows operators to register new members. The registration process involves operators to specify membership details such as name, email, etc. Information regarding to the layout of the sensor data published by the data source is required. Sensor data layouts are defined by means of XML documents containing the schema needed to represent properties of the sensor data such as date, time, station, etc. Additionally, the semantics of the sensor data properties can be also defined using XML. The prototype takes sensor data layout as input to generate relational tables in the database that store collected sensor data.

Figure 10 shows a schematic representation of the member registration process.

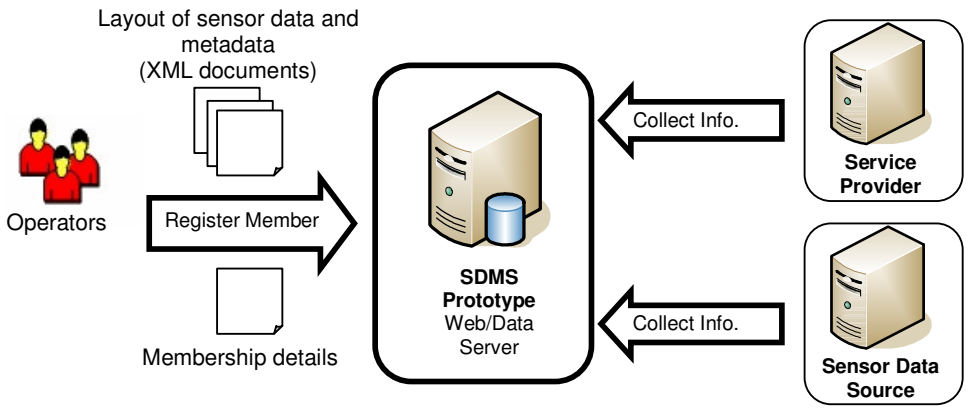

Figure 10: Schematic representation of the member registration process.

4.4 Sensor Data Collection

Conventionally, a sensor data consumer has to worry about collecting data from the sensors to a processing application. A common model is for individual sensors to detect data and send them to the consuming applications that run on powerful machines. This approach requires sensor network owners to deploy not only the sensors, but also the infrastructure to move collected data to interested applications. Each time a sensor network is deployed, owners must reinvent the design when considering how to collect data from sensors to the applications requesting information [17].

In Virtual School Lab with an increasing number of autonomous sensor data sources, data are made available to schools and private organizations via Sensor Data Management System (SDMS). School children should be able to easily query data from federated sensor data sources. Once a new sensor data source is registered, sensor data should be collected periodically, processed, and presented in user-friendly formats. This is possible because the prototype relies on a data collection service that allows interoperability with heterogeneous sensor data sources which publish data on the Web.

The data collection service features an adapter-based design to achieve its goals. Adapters are small programs that allow the system to fetch relevant sensor data from the web-based sources. Since most of the sensor networks publish sensor data in regular web pages, prototype adapters have to deal with parsing HTML web pages to collect numerical values. Figure 11 shows a schematic representation of the sensor data collection.

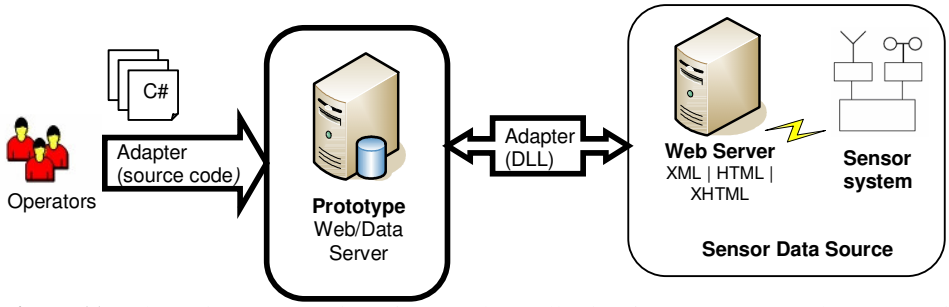

Figure 11: Schematic representation of sensor data collection from autonomous data sources in *Virtual School Lab.*

As seen in Figure 11, new adapters are injected to the system by operators. The prototype is able to use their functionality to collect sensor data from autonomous data sources.

The adapter-based design of the prototype relies on some software components that provide specific functionality to the prototype. These components have been implemented using the .NET framework as primary platform for development. Figure 12 shows the components of the adapter-based design of the prototype.

The dynamic adapter compiler:

The dynamic compiler component allows the prototype extends its functionality with code that is not compiled into the system. The dynamic compiler component takes as input the source code of new adapters injected (uploaded) by operators into the system and generates assemblies (DLL) with compiled code that can be easily invoked by the system. These Assemblies are stored locally and executed each time their functionality is required.

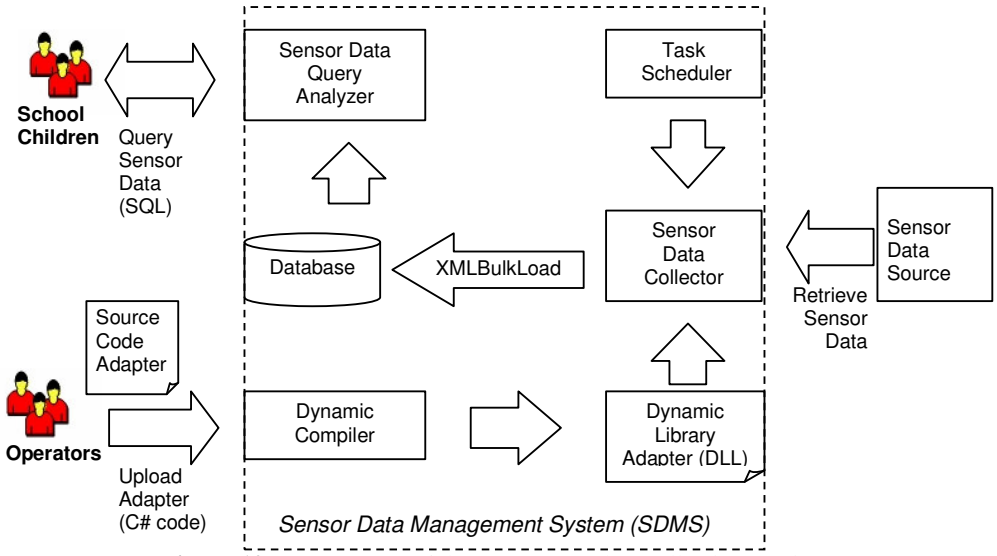

Figure 12: Components of the prototype's data collection service.

The process of compiling dynamically the source code of new adapters involves the following steps:

1. Read the source code of new adapter into a string representation.
2. Provide an implementation of the code compiler interface provided by .NET[18] framework to support dynamic compilation. The code compiler interface provides the capability to invoke compilation with specified parameters at runtime and access information related to compilation after compilation occurs, including the result code, and any errors or warnings the compiler returns. Each compile method accepts a compiler parameters object that indicates settings for the compiler, and returns a compiler results object that indicates the results of the compilation.
3. Compile the source code into an assembly (DLL[19]).
4. Check for errors on compilation.

The process of executing the new assemblies (compiled code of adapters) involves the following steps:

1. Reference the assembly (DLL of the adaptor)
2. Use the assembly reference to create an instance of the adapter object.
3. Call the execute method on the instance reference returned using reflection[20].
4. Handle the return value from the method call by casting into the proper type.

An example of an adapter program that allows sensor data collection from *LML* is given in appendix D of this thesis.

The collection tasks scheduler:

The task scheduler component allows the prototype to submit dynamically new jobs to the task scheduler windows service. The service schedules and automatically starts programs in the windows operating system.

Once new sensor data sources are registered in the prototype, the task scheduler component generates new tasks to collect sensor data periodically from registered sources. The component allows triggering data collection tasks hourly, daily and weekly.

The sensor data collector:

The sensor data collector is a binary program that is triggered according to the scheduled tasks of the prototype. Every scheduled task starts the collector program on a regular basis. The sensor data collector program invokes the functionality of available adapters (DLL) to collect sensor data from registered sources. The output of the collector program is a XML document containing the collected

[18] The .NET Framework provides the ICodeCompiler compiler execution interface. The CSharpCodeProvider class implements this interface and provides access to instances of the C# code generator and code compiler.
[19] DLL: Dynamic-Link Library.
[20] Reflection is the ability to find information contained in an assembly at run-time. .NET reflection is a mechanism which not only allows inspecting type information but also allows invoking methods on those types at runtime.

sensor data. These data is load into the prototype database through the XML Bulk Load component.

4.5 Service Providers Interoperability

Another concept in the prototype is allowing participants of Virtual School Lab to provide new functionality to the system. Service providers play two different roles in Virtual School Lab. They consume raw sensor data and produce useful information for school children that is collected and presented via the prototype.

Participants owning computational systems with different processing capacities can add new functionality to the system making it more extensible and distributed.

In order to allow participants to act as service providers, they should register via the prototype and describe the functionality they can offer to Virtual School Lab. Service providers are free to use any hardware or software technology to process raw sensor data and produce useful information for school children. However an important requirement is needed; they should deploy web services to expose their functionality to the federation. Guidelines to allow participants to act as service providers are given by Virtual School Lab.

As new service providers are registered via the prototype, they might rely on specific hardware and software technologies. To overcome this problem, the prototype should remain agnostic to the details of the service provider implementation. The approach in this thesis is to create a generic mechanism able to use the core functionality of distributed service providers which use web services as the bridge to the outside world.

The prototype consumes web services deployed by service providers by means of a mechanism that generate and invoke dynamically web service proxies[21].

A web service proxy is a class that encapsulates the complexity of calling a web service and exposes this complexity through a simplified interface. In order to create the proxy class, it is important to have access to the web service's WSDL document.

The WSDL document is an XML-formatted document that formally defines a web service. It specifies the data types of the messages, what protocols are accepted, and the web service's endpoint. .

In Virtual School Lab, once participants register as service providers via the prototype, their functionality are made available to the federation.

The process of calling service providers involves the following steps:

1. The web service proxy generation component makes an HTTP request to retrieve the functional description of the remote web service (WSDL). The

[21] A proxy is a special component that masquerades as the service. The proxy insulates the invoker from the location of the actual service.

WSDL document contains the information needed to invoke methods of the Web Service.
2. The web service proxy generation component receives the response and parses it to generate dynamically a proxy class for the web service.
3. The prototype calls the execute method including additional parameters of the proxy class.
4. The proxy class serializes the parameters into a SOAP[22] message, and sends it over HTTP to the web service.
5. The web service returns the result of the method call, an object containing processed sensor data, serialized in a SOAP message.
6. The proxy class deserializes the response, and returns the information to the system.

Figure 13 shows a schematic representation of dynamic generation and invocation of proxy classes for web services.

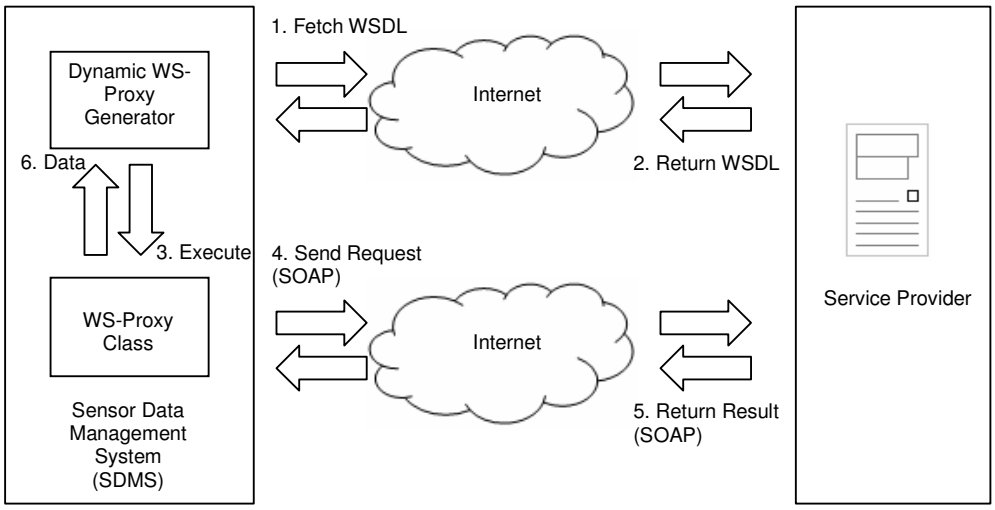

Figure 13: Dynamic web service proxy generation and invocation in SDMS.

4.6 Push and Pull Mechanisms

Although the prototype is mainly designed to address the issues of featuring an adapter-based design that enables sensor data retrieval from many autonomous sensor data sources which publish data on the Web, the system is also able to collect data from federated sensor networks using push and pull mechanisms.

In addition to pure data sets being sent from sensor data sources to the system, special query messages can be sent directly to sensor data sources via pull mechanism. Query messages will be limited to simple pre-defined tasks such as "get current status of the sensor". As illustrated in Figure 14, federated sensor networks wishing to use pull mechanism should deploy web services as the bridge to the outside world. Functionality of each individual low level sensor in the

[22] SOAP: Simple Object Access Protocol.

network is represented by a web service. Such approach is called virtualization of sensor networks.

On the other hand, if organizations owning sensor networks do not wish to deal with the complexity of deploying web services, they can use the push mechanism to send their data directly to the system. Although, sensor networks don't need to deploy their own web services, they should deploy a base station able to call remote web services provided by the prototype.

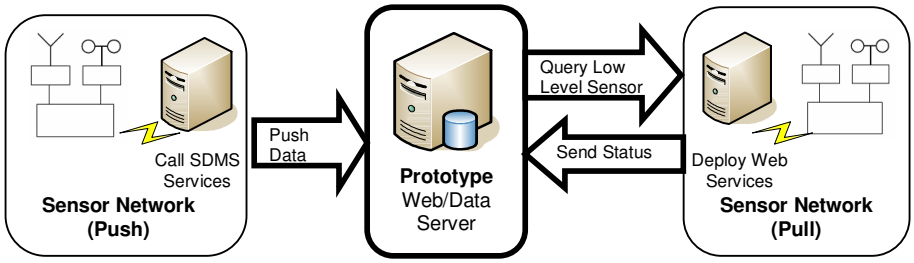

Figure 14: Schematic representation of sensor networks interacting with prototype push and pull mechanisms. The pull mechanism allows the prototype to send special query messages directly to federated sensors which then process the request and send response.

An example of pushing mechanism is illustrated in Figure 16 at the end of this chapter. It shows a cell phone based pollution sensor pushing data to the SDMS system. The idea is that cell phones equipped with carbon monoxide (CO) detectors and GPS systems would send location-specific data on concentration levels of carbon monoxide.

4.7 Real Time Sensor Data

In addition to sensor data retrieval from the Web and other methods such as the push or pull mechanisms, the system aims to offer users the possibility to retrieve real time sensor data. Although most of the important sensor data sources publish data on the Web such as the LML sensor network, the system uses the messaging and queuing paradigm (MessageQ) that makes possible to reliably transfer real time sensor data from federated sources.

In order to accomplish this goal, federated sources should deploy the core infrastructure needed to transfer real time data streams to the prototype. Guidelines for the required deployment process are provided by Virtual School Lab.

A schematic representation the different methods provided to retrieve sensor data is given in Figure 16.

4.8 Sensor Data Modeling

The need for formats comes from data heterogeneity between different sensor networks that publish sensor data on the Internet. To facilitate transferring and sharing sensor data on the Internet it is necessary a common vocabulary (format)

for the data encoding. XML is used in this thesis as primary format to publish structured sensor data on the Web.

XML documents are made up of storage units called entities, which contain either parsed or unparsed data. Parsed data are made up of characters, some of which form the character data in the document, and some of which form markup. Markup encodes a description of the document's storage layout and logical structure. XML provides a mechanism to impose constraints on the storage layout and logical structure. SDMS contains a software module called XML Bulk Load Component that is used to load XML documents containing sensor data into persistent data storages.

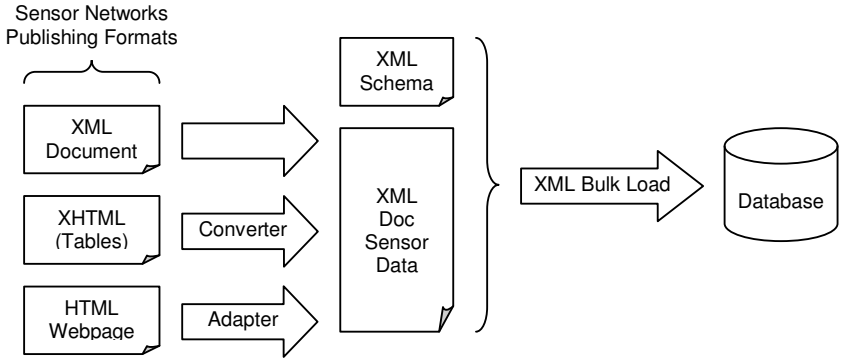

Figure 15: XML Bulk Load Component of *SDMS* loads sensor data into data storage. The component takes as input the XML representation of sensor data. Sensor networks publish sensor data in different formats. XML is the primary format for *SDMS*.

Figure 15 shows a schematic representation of sensor data loading into the persistent data storage of SDMS. Sensor networks publish data encoded in different formats such as XML, XHTML or HTML webpage. XML documents containing sensor data can be loaded directly into the data storage through XML Bulk Load Component of SDMS. The component takes as input the XML representation of the sensor data and the Schema representing their structure.

The following examples illustrate the use of sensor data formats in Virtual School Lab. Table 5 shows a sensor data sample from Geluidsnet sensor network. Each data element is represented in the table as a row containing the measured value dBmax (decibel maximum level) and some additional properties such as Date and Time.

Indicator	Date Time	Event ID	dB max
Mp111	2005-09-01 09:10:05	1	48
Mp111	2005-09-01 09:10:10	2	45
Mp111	2005-09-01 09:10:15	3	40

Table 5: Sensor data sample from *Geluidsnet* Sensor Network.

Table 6 contains the Schema that represents the basic structure for sensor data from Geluidsnet sensor network. The schema contains 3 different parts. The first one is called declaration and specifies data types for the measured value and the additional properties. In the schema shown in Table 6, the measured value (dBmax) is declared as float data type. The second part contains the layout of XML documents containing sensor data. The third part maps the measured value and properties to fields of a SQL relational table named VSL_Geluidsnet. The mapping schema is used by the XML Bulk Load Component to load the data into specific relational tables in the database of SDMS.

```
<?xml version="1.0" ?>
- <Schema xmlns="urn:schemas-microsoft-com:xml-data"
    xmlns:dt="urn:schemas-microsoft-com:xml:datatypes"
    xmlns:sql="urn:schemas-microsoft-com:xml-sql">
    <ElementType name="meetpunt" sql:datatype="int" />
    <ElementType name="datetime" sql:datatype="datetime" />
    <ElementType name="eventID" sql:datatype="int" />
    <ElementType name="dBmax" sql:datatype="float" />
  - <ElementType name="ROOT" sql:is-constant="1">
      <element type="Item" />
    </ElementType>
  - <ElementType name="Item" sql:relation="VSL_Geluidsnet"
      sql:key-fields="meetpunt datetime">
      <element type="meetpunt" sql:field="meetpunt" />
      <element type="datetime" sql:field="datetime" />
      <element type="eventID" sql:field="eventID" />
      <element type="dBmax" sql:field="dBmax" />
    </ElementType>
  </Schema>
```

Declaration

XML Data Layout

Mapping

Table 6: XML Schema for *Geluidsnet* Sensor Data.

Table 7 contains the XML representation of the sensor data sample shown in Table 5. The XML document layout is defined in the Schema shown in Table 6.

```
- <ROOT>
  - <Item>
      <meetpunt>mp111</meetpunt>
      <datetime>2005-09-01 09:10:05</datetime>
      <eventID>1</eventID>
      <dBmax>48</dBmax>
    </Item>
  - <Item>
      <meetpunt>mp111</meetpunt>
      <datetime>2005-09-01 09:10:10</datetime>
      <eventID>2</eventID>
      <dBmax>45</dBmax>
    </Item>
  - <Item>
      <meetpunt>mp111</meetpunt>
      <datetime>2005-09-01 09:10:15</datetime>
      <eventID>3</eventID>
      <dBmax>40</dBmax>
    </Item>
  </ROOT>
```

Table 7: XML document with sensor data of example given in Table 5.

Table 8 contains data from *LML* sensor network. Measured values are concentrations of Sulfur Dioxide (SO2), Particulate Matter (PM10), Ammonia (NH3), Carbon Monoxide (CO), Ozone (O3), Nitric Oxide (NO), Nitrogen Dioxide (NO2). Additional properties such as Date, Time and Station help to organize collected information into the relational database of *SDMS*.

Station	Date Time	SO2	PM10	NH3	CO	CO3	NO	NO2
937	2005-09-01 09:00:00	1	18	1	210	24	1	23
937	2005-09-01 10:00:00	1	15	1	210	20	1	20
937	2005-09-01 11:00:00	2	20	5	250	20	0	28

Table 8: Sensor data sample from *LML* Sensor Network.

Table 9 shows the Schema for sensor data from *LML* network. Table 10 shows a XML document containing the sensor data sample presented in Table 8.

```
<?xml version="1.0" ?>
- <Schema xmlns="urn:schemas-microsoft-com:xml-data"
    xmlns:dt="urn:schemas-microsoft-com:xml:datatypes"
    xmlns:sql="urn:schemas-microsoft-com:xml-sql">
    <ElementType name="station" sql:datatype="int" />
    <ElementType name="datetime"
      sql:datatype="datetime" />
    <ElementType name="SO2" sql:datatype="int" />
    <ElementType name="PM10" sql:datatype="int" />
    <ElementType name="NH3" sql:datatype="int" />
    <ElementType name="CO" sql:datatype="int" />
    <ElementType name="CO3" sql:datatype="int" />
    <ElementType name="NO" sql:datatype="int" />
    <ElementType name="NO2" sql:datatype="int" />
  - <ElementType name="ROOT" sql:is-constant="1">
      <element type="Item" />
    </ElementType>
  - <ElementType name="Item" sql:relation="VSL_LML"
      sql:key-fields="station datetime">
      <element type="station" sql:field="station" />
      <element type="datetime" sql:field="datetime" />
      <element type="SO2" sql:field="SO2" />
      <element type="PM10" sql:field="PM10" />
      <element type="NH3" sql:field="NH3" />
      <element type="CO" sql:field="CO" />
      <element type="CO3" sql:field="CO3" />
      <element type="NO" sql:field="NO" />
      <element type="NO2" sql:field="NO2" />
    </ElementType>
  </Schema>
```

Table 9: XML Schema for *LML* Sensor Data.

```
- <ROOT>
  - <Item>
      <station>937</station>
      <datetime>2005-09-01 09:00:00</datetime>
      <SO2>1</SO2>
      <PM10>18</PM10>
      <NH3>1</NH3>
      <CO>210</CO>
      <CO3>24</CO3>
      <NO>1</NO>
      <NO2>23</NO2>
    </Item>
  - <Item>
      <station>937</station>
      <datetime>2005-09-01 10:00:00</datetime>
      <SO2>1</SO2>
      <PM10>15</PM10>
      <NH3>1</NH3>
      <CO>210</CO>
      <CO3>20</CO3>
      <NO>1</NO>
      <NO2>20</NO2>
    </Item>
  - <Item>
      <station>937</station>
      <datetime>2005-09-01 11:00:00</datetime>
      <SO2>2</SO2>
      <PM10>20</PM10>
      <NH3>5</NH3>
      <CO>250</CO>
      <CO3>20</CO3>
      <NO>0</NO>
      <NO2>28</NO2>
    </Item>
  </ROOT>
```

Table 10: XML representation of sensor data sample given in Table 8.

4.9 Metadata

SDMS allows operators to manage manually-created metadata describing semantics of some properties of the sensor data collected. XHTML is used in this thesis as primary format to store and publish metadata on the Web.

eXtensible HyperText Markup Language, or XHTML, is a markup language that has the same expressive possibilities as HTML, but a stricter syntax. Because XHTML documents need to be well-formed (syntactically correct), they allow for automated processing to be performed using a standard XML library — unlike HTML, which requires a relatively complex, lenient, and generally custom parser. XHTML can be thought of as the intersection of HTML and XML in many respects, since it is both valid HTML and XML.

Table 11 contains metadata of different concentrations measured via LML sensor network. This information is made available to the users via SDMS. Metadata is encoded in XHTML format and presented to the users in regular HTML tables.

Property	Description	Type	Unit	Scale	Resolution
SO2	Sulfur Dioxide	integer	ug/m3	1	1
PM10	Particulate Matter	integer	ug/m3	1	1
NH3	Ammonia	integer	ug/m3	1	1
CO	Carbon Monoxide	integer	ug/m3	1	1
O3	Ozone	integer	ug/m3	1	1
N0	Nitric Oxide	integer	ug/m3	1	1
N02	Nitrogen Dioxide	integer	ug/m3	1	1

Table 11: Metadata describes semantics of concentrations measured via *LML* sensor network.

Table 12 contains metadata that describes the physical location of different stations of LML sensor network.

StationID	Location	Type	Coordinates
136	Heerlen-Looierstraat	Street	none
236	Eindhoven-Genovevalaan	Street	none
237	Eindhoven-Noordbrabantlaan	Street	none
240	Breda-Tilburgseweg	Street	none
433	Vlaardingen-Floreslaan	Street	none
445	Den Haag-Veerkade	Street	none
447	Leiden-Willem de Zwijgerlaan	Street	none
448	Rotterdam-Statenweg	Street	none
537	Haarlem-Amsterdamsevaart	Street	none
544	Amsterdam-Bernhardplein	Street	none
636	Utrecht-de Jongweg	Street	none
638	Utrecht-Vleutenseweg	Street	none
639	Utrecht-Erzeijstraat	Street	none
641	Breukelen-Snelweg	Street	none
741	Nijmegen-Graafseweg	Street	none
937	Groningen-Europaweg	Street	none

Table 12: Metadata describes physical location of some *LML* stations where concentrations are measured.

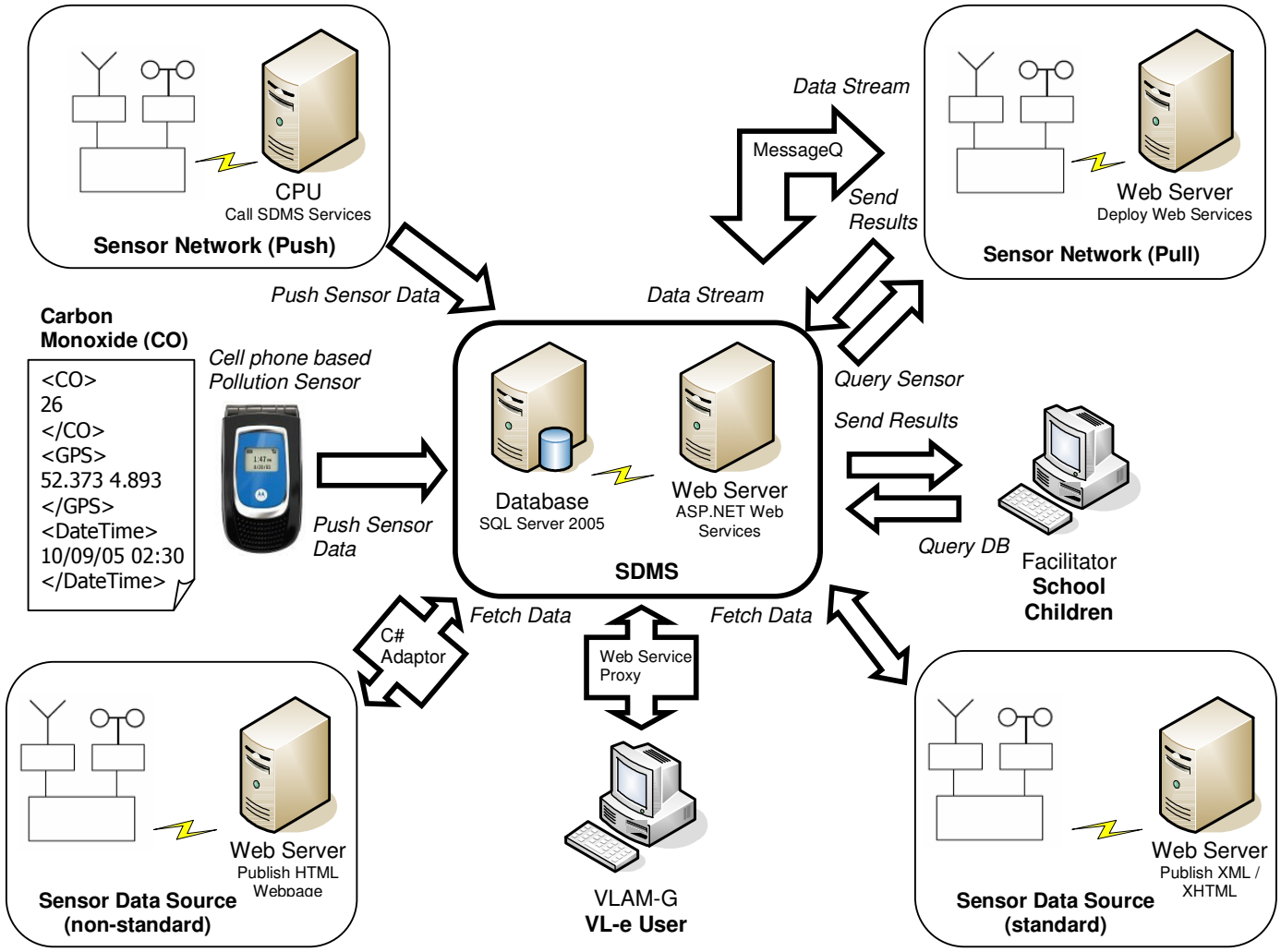

Figure 16: Schematic representation of 4 methods of sensor data retrieval envisioned by the SDMS prototype. Cell phone based pollution sensors use push mechanism to send sensor data back to the central database of *SDMS*.

Chapter 5: Sample Use Scenarios

5.1 Administration Tools

The Administration Tools of SDMS consists of a set of software tools that help operators to perform some tasks concerning the administration of Virtual School Lab. Tools are available to the users who authenticate themselves as operators. A detailed description of some of these tools will be given in the next sections.

5.1.1 Login the System

Operators and School children enter the system through a Login Tool Component. After successful authentication, operators can enter the administration site of SDMS selecting the appropriate address in the web browser as shown in figure 18. Administration site contains a menu of available tools for the administration of Virtual School Lab. Non-operator users are not authorized to enter the Administration site of SDMS and any attempt to violate security will be rejected.

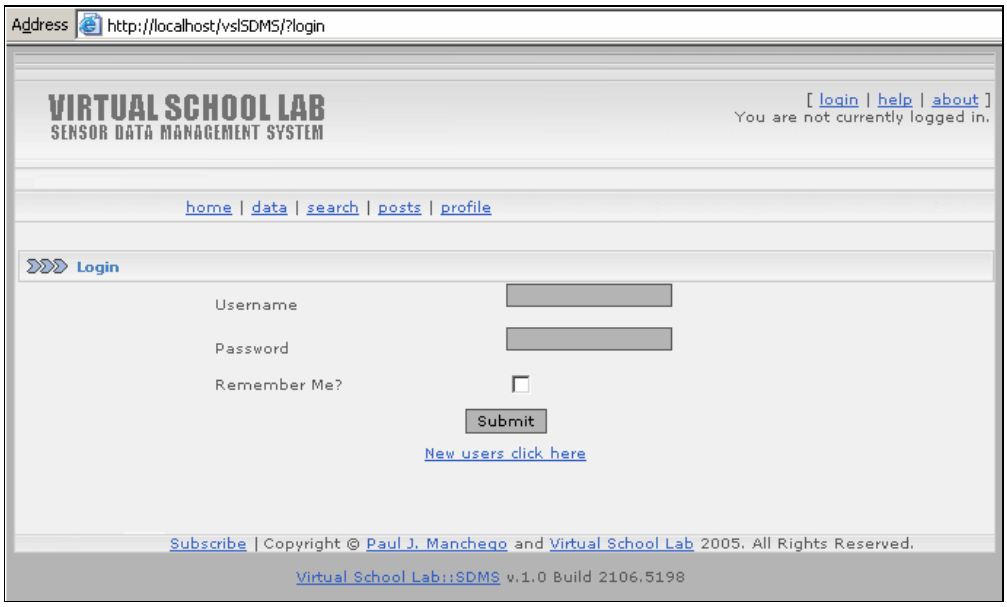

Figure 17: Screen caption of Login Tool of *SDMS*.

Figure 17 shows the screen caption of the login tool. Although, users can register themselves to use the system, Administration site is restricted to the operators. Non-operator users can be granted administrative privileges by operators according to the policies of Virtual School Lab.

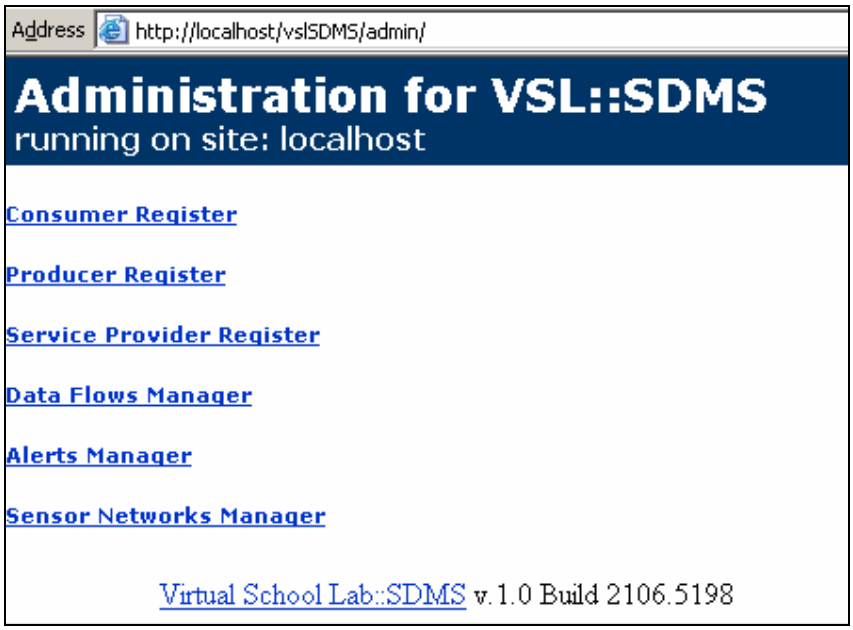

Figure 18: Screen caption of Tools Menu for Administrators.

Figure 18 shows the administrator tools available in the current prototype. This set of web-based tools allow Operators to perform some tasks such as register new members of Virtual School Lab. Organizational units can be registered as data producers, consumers or service providers. A detailed description of each of them is given in the next sections.

5.1.2 Register New Consumer

This tool helps Operators to register organizational units that participate in Virtual School Lab as Consumers. For example, Operators register Schools wishing to take part in the initiative that brings school children new opportunities to understand sensor data.

As seen in Figure 19, Consumer registration process consists on filling out a web-based form. Some details concerning the organizational unit to be registered should be entered by Operators. After form submission, a system message will confirm successful registration of the new member.

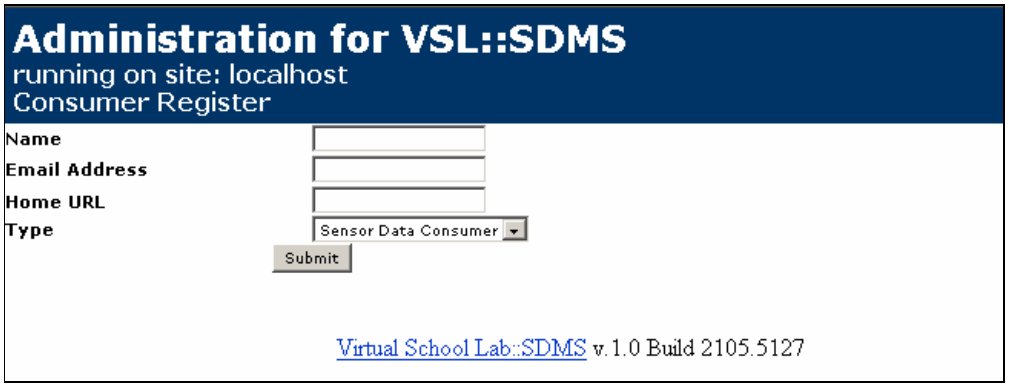

Figure 19: Screen caption of *Consumer Register Tool*. This web-based tool allows Operators to register new participants such as Schools.

5.1.3 Register New Producer

As seen in Figure 20, new producers are registered by means of a web-based form. The registration process consists on filling out some information concerning the new member.

A detailed description of details entered by Operators is given below:

1. Name: Organization's name to be registered as new Producer.
2. Email Address: Contact information.
3. Home URL: Producer's main Website.
4. Logo URL: Producer's logo that identifies organization.
5. About URL: Contains information concerning the measurements made by Producer (Sensor Network).
6. Type: SDMS allows Operators to register 4 different types of Producers:
    a. Sensor Data Producer (standard): Organizations that publish sensor data in XML or XHTML formats. No especial Adapters are needed to collect data from them.
    b. Sensor Data Producer (non-standard): Organizations that publish sensor data in regular HTML WebPages. Special Adapters are needed to collect sensor data from them.
    c. Sensor Network (Pull): Organizations that deploy sensor networks according to guidelines given by Virtual School Lab. They use web services as bridge to the outside world. Sensor data is collected via Pull mechanism. For example, SDMS is able to query individual low level sensors and to retrieve current status.
    d. Sensor Network (Push): Organizations owning sensor networks
7. Sensor Properties: XML document that specify properties of low level sensor. (Only used by Pull mechanism of SDMS).
8. Data Schema: XML document that describes the sensor data layout. (See chapter 3 for more details)
9. Metadata: XHTML document that contains semantics of sensor data.
10. Data URL: Site where sensor data is published. Operators should also select format used to publish sensor data. Three options are possible: XML, XHTML, or regular Webpage (C# Adapter).
11. C# Adapter: Adapter source code is compiled by SDMS and used to collect sensor data from organizations that publish sensor data in regular HTML WebPages.
12. Update Pattern: Sensor data is collected on a regular basis. Operator should select the appropriate pattern. Three options are available: Hourly, Daily and Weekly. Update Pattern is used by SDMS to generate sensor data collection tasks.
13. Start Date and Time: Operator should enter date and time to start sensor data collection tasks.
14. Description: Keywords that describes sensor network functionality.

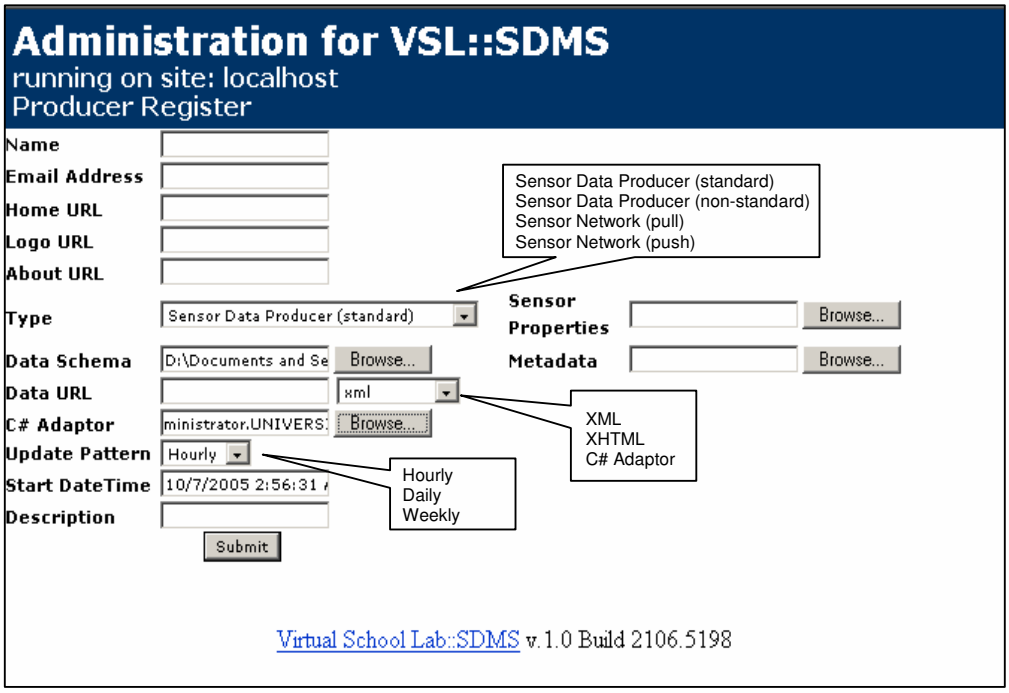

Figure 20: Screen caption of *Producer Register Tool*. This web-based tool allows Operators to register new Producers. Producer type should be selected during the process.

5.1.4 Register New Service Provider

New Service Providers are registered by means of this web-based tool. The process of registration involves Operator to fill out some information regarding Service Providers.

A detailed description of registration details is given below:

1. Name: Participant's name to be registered as Service Provider.
2. Email Address: Contact information.
3. Web Service URI: Uniform Resource Identifier of Web service deployed by Service Provider.
4. Service Provider Type: Operator should select the type of information provided by the new member. Two options are currently available: binary information and dataset.
5. XML Schema (Input): XML document that describes the layout of sensor data that is processed by Service Provider.
6. XML Schema (Output): XML document that describes the layout of resultant information generated by Service Provider.
7. Description: Keywords that describes Service Provider functionality.

Figure 21 shows a screen caption of Service Provider Register Tool.

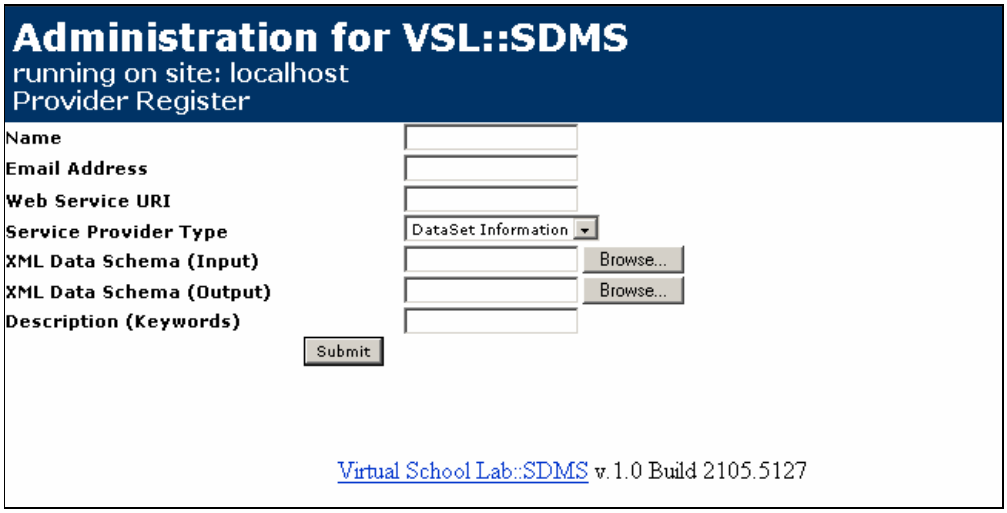

Figure 21: Screen caption of *Service Provider Register*. This tool allows operators to register new Service Providers.

5.1.5 Managing Data Flows

Data Flow Manager is responsible for the set-up and management of data flows in *SDMS*. A *data flow* is a fundamental abstraction that links data producers, data consumers, and service providers into a data path. This ensures that users receive the data in which they are interested. A data flow enables Operators to express user data needs at a high level and pass the responsibility for creating data flows to *SDMS*, thus simplifying selection and querying. Sensor data provided by federated sources and stored in the *SDMS* database can be processed by Service Providers and then delivered to interested users.

The layout of a data flow is specified in XML language. A schematic representation of a data flow is given in Figure 22. This schema can be used as template to define new data flows. Data flow descriptor consists of three nodes. The inner one named *producer* is tied to the sensor data source. It contains some attributes such as *caption* and *endpoint*. Both attributes are used to identify the sensor data source. Additionally to these attributes, a child node name *filter* can be used to specify how sensor data will be filtered before they are presented to the users.

The second node named *provider* is optional and it is tied to the service provider. It also contains the attributes *caption* and *endpoint*. The third node is only used to give data flow descriptors a well defined structure that is at the same time self-descriptive.

A well defined data flow descriptor should include at least the *Consumer* and *Producer* nodes.

```
- <dataflow>
  - <consumer>
    - <provider caption="Provider_CaptionName" endpoint="Provider_URI">
      - <producer caption="Producer_CaptionName"
          endpoint="Producer_HomeURL">
          <filter sql="select * from Producer_DataTableName" />
        </producer>
      </provider>
    </consumer>
  </dataflow>
```

Figure 22: Schematic representation of data flow descriptor. Notice that data flows allow the system to filter sensor data by means of SQL queries.

Figure 23 shows an example of a well defined data flow descriptor that selects sensor data from *LML*. In this example no intermediate service providers are used. Sensor data are retrieved directly from database and presented to the users.

```
- <dataflow>
  - <consumer>
    - <producer caption="lml" endpoint="http://www.lml.rivm.nl/">
        <filter sql="select * from VSL_LML" />
      </producer>
    </consumer>
  </dataflow>
```

Figure 23: Example of a data flow descriptor that allows users to retrieve sensor data from *LML*.

SDMS allows Operators to upload and publish new data flow descriptors. Figure 24 shows the web-based tool used by operators to submit new data flows to the system. Descriptors can be written in plain text files and uploaded using the Browse button shown in the figure.

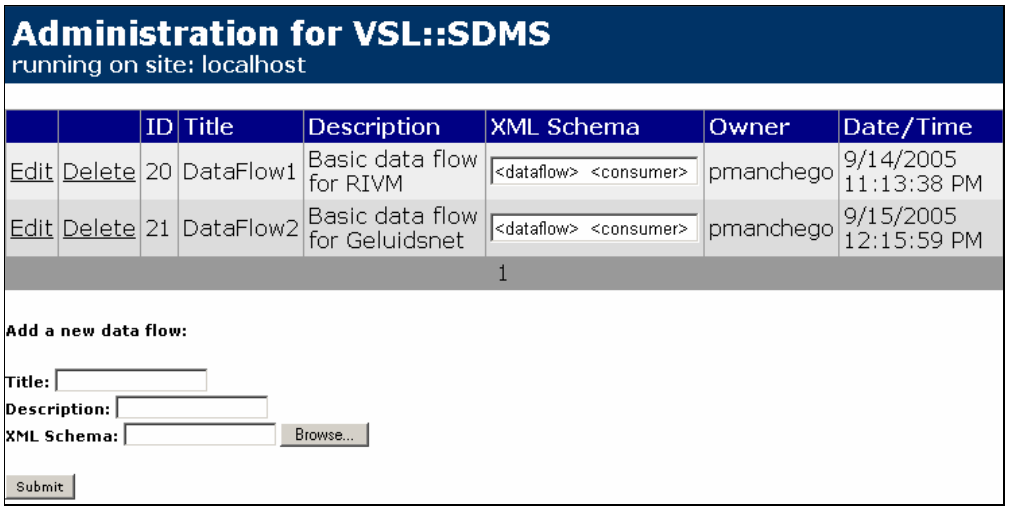

Figure 24: Screen caption of *Data Flows Manager*. This web-based tool allows operator to upload new data flows descriptors to the system.

5.2 School Children Tools

The School Kid Tools of *SDMS* consists of a set of software tools that help school children to complete practical assignments proposed by Virtual School Lab. Tools are available to the students who authenticate themselves as users of SDMS. A detailed description of some of these tools will be given in the next sections.

5.2.1 Register New User

School children enter the system through the Login Tool showed in Figure 17. New users should register themselves by means of a web-based registration form. Some details such as login name and password should be entered to complete user registration process. A confirmation message is sent to the user email address after successful registration. Figure 25 shows a screen caption of user registration tool.

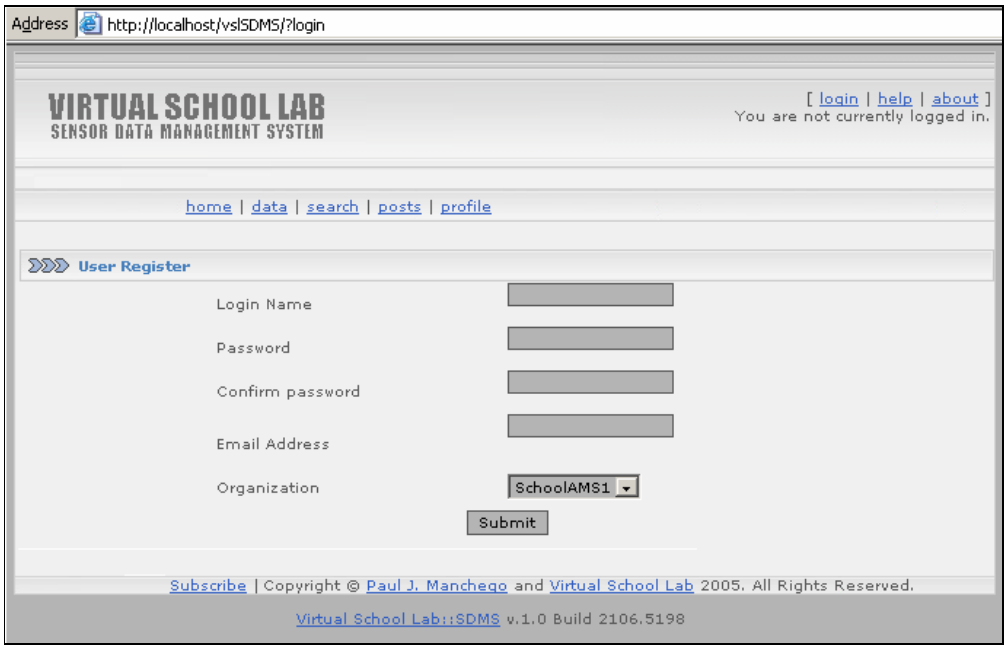

Figure 25: Screen Caption of User Registration Tool.

5.2.2 Retrieving Sensor Data

SDMS helps school children to retrieve and query sensor data from different sources by means of the sensor data tool. This web-based tool allows students to specify sensor data sources by selecting the appropriate data flow descriptor.

Data flow descriptors are made available by the Operators of Virtual School Lab. They contain information that SDMS need to query and present sensor data from specific sources to the users. Data flow descriptors are presented to the users with descriptions of their functionality. Figure 26 shows a screen caption of the data flow selection tool. As seen in the figure two data flows are available: DataFlow1

returns sensor data from LML and DataFlow2 presents data from Geluidsnet sensor network.

Figure 26: Screen Caption of Data Flows Menu.

Internally data flow descriptors contain SQL queries allowing Operators to customize data flows according to the school children's needs. After a data flow is selected, sensor data is presented into a HTML data grid. Figure 27 shows the different concentration fields from LML sensor network. Data grid is the primary data presentation tool of SDMS. There is however other presentation tool called Pivot Table. This is a web-based user-friendly tool that allows school children to manipulate sensor data values. Other capabilities of Pivot Table Tool includes: sensor data filtering, clipboard copying, exporting to text files and excel files, highly customized filtering capacities and sensor data aggregation functions.

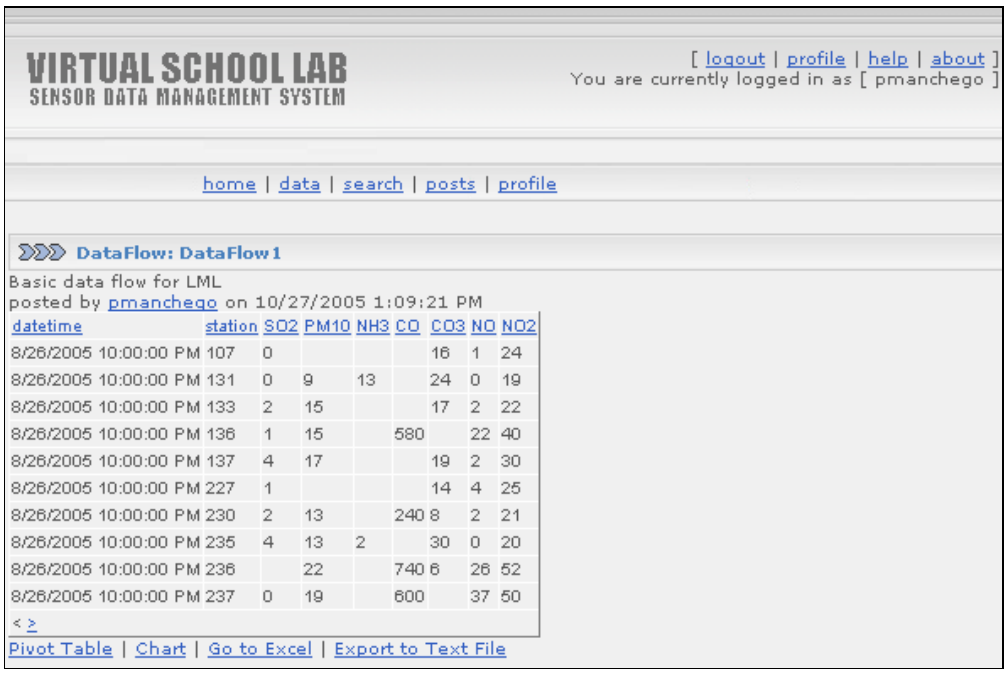

Figure 27: Screen Caption of Sensor Data Grid Tool. Data from LML sensor network are presented into a HTML data grid.

As seen in Figure 28, school children are able to select the appropriate concentration field from the Pivot table field list and drop it into the details area.

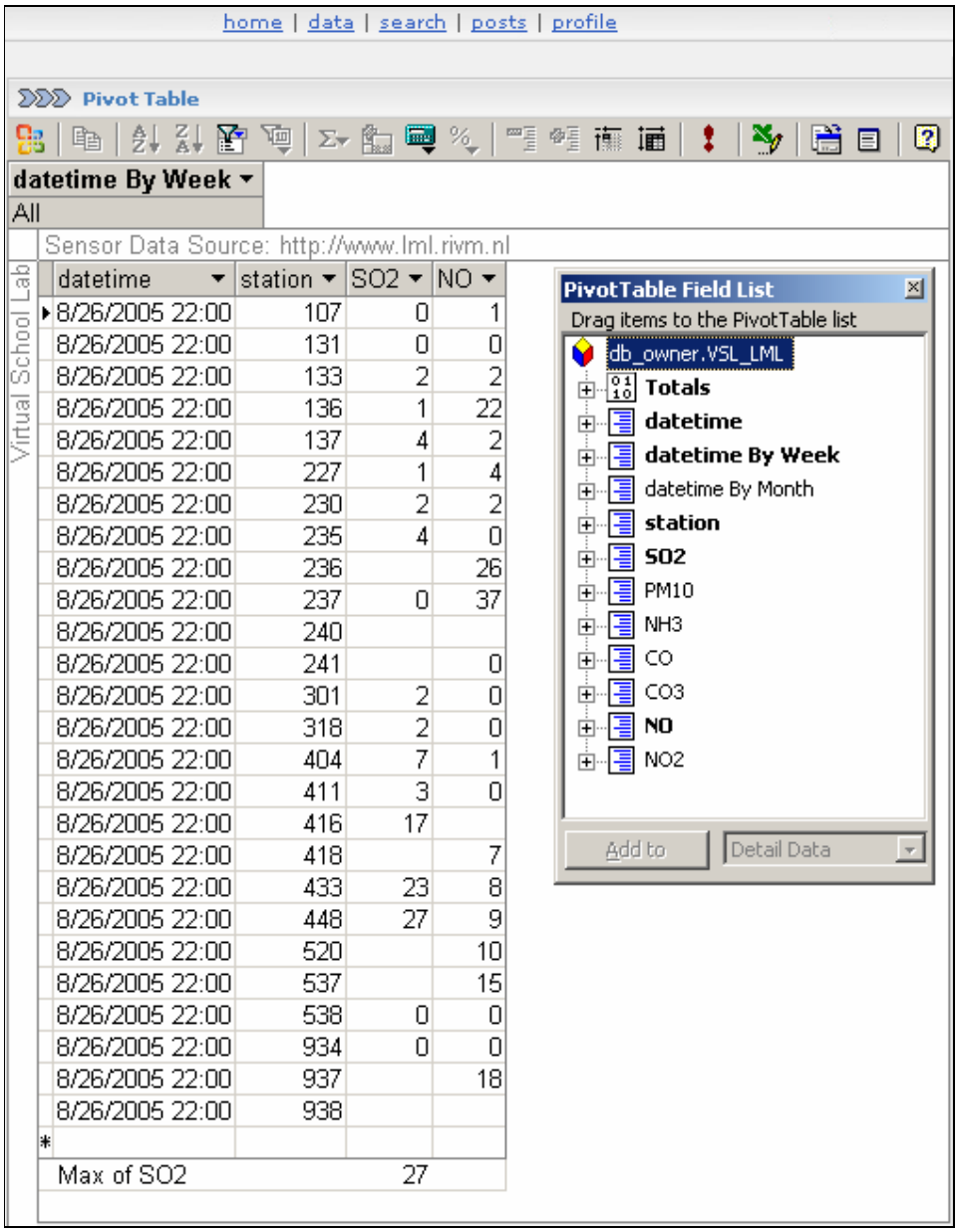

Figure 28: Screen Caption of Pivot Table Tool of SDMS. School children can drop concentration fields measured by LML sensor network into the Table.

5.2.3 Subscribing to Notifications

School children can subscribe to feeds delivered by SDMS. Notifications stating important announcements are made available by Operators. SDMS delivers these notifications to the subscribers by means of Atom feeds.

Atom is an XML-based file format intended to allow lists of information, known as "feeds", to be synchronized between publishers and consumers. Feeds are composed of a number of items, known as "entries", each with an extensible set of attached metadata. For example, each entry has a title.

The process of subscription involves the following steps:

1. Install and run the feed Reader. There are many feed readers available on Internet.
2. Click on the link Subscribe of SDMS to locate the web address (URL) of the Atom feed (XML file) to subscribe.
3. Insert the URL of the SDMS feed.
4. Set the interval that the feed reader will look for feed update.
5. The information in the feed will be updated when new notifications are posted by Operators of Virtual School Lab.

Figure 29 shows a screen caption of the configuration of a feed viewer to subscribe to notifications of Virtual School Lab.

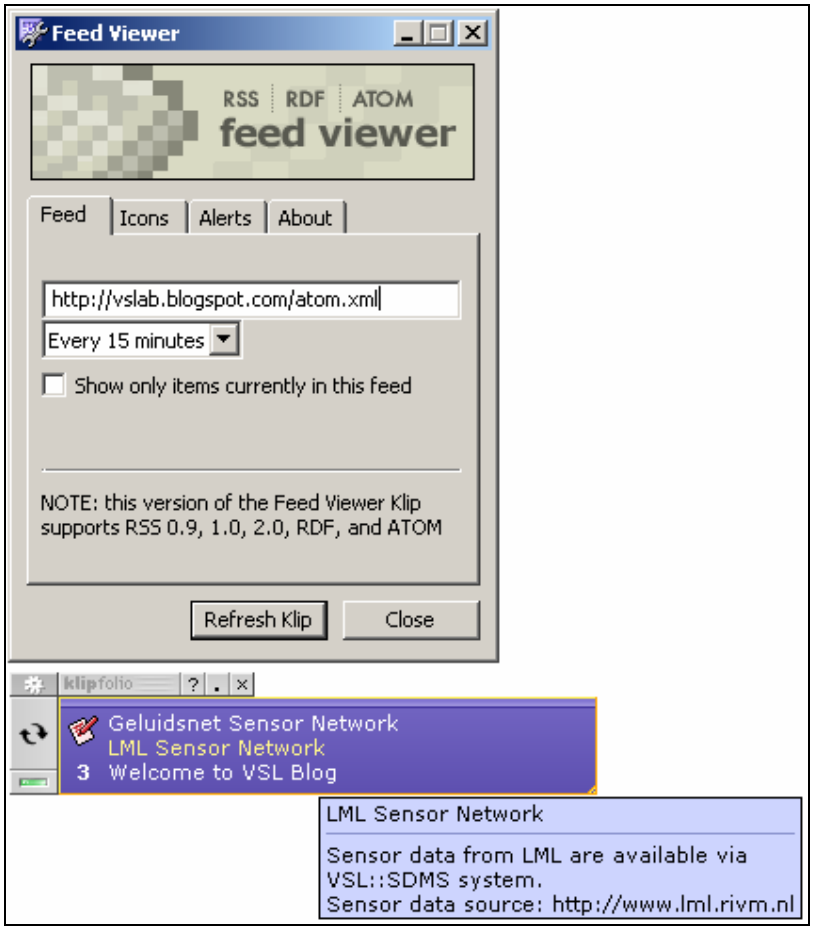

Figure 29: Screen Caption of ATOM feed viewer. Important announcements of operators are delivered through XML representation named ATOM.

Chapter 6: Conclusions and Future Work

The Virtual School Lab project imposes intriguing requirements on a system that manages data from heterogeneous sensor data sources. For example the system should be able to perform the complex parsing tasks required to make sensor data published on the Web available for educational purposes. This work presents a thorough analysis of requirements for Virtual School Lab; a methodological approach was used to design the architecture of the SDMS system that facilitates administrative and operational tasks in Virtual School Lab. A prototype was also constructed to prove critical aspects of the proposed architecture.

It can be argued that the more requirements and components the system specifies for Virtual School Lab, the more complicated its implementation becomes. SDMS does introduce a number of assumptions; however, its functional requirements are flexible and allow extensibility on many levels. For example, data traceability can be achieved defining appropriate layout schemas for sensor data to allow storing historical data. The current SDMS achieves its goals by allowing school children to retrieve and query data from a variety of sensor data sources. Development of adapters (interfaces to collect sensor data from the Web) is made straightforward and does not require any special knowledge except the necessary skills to develop C# programs. Moreover, the flexible structures governing dataflow descriptors allow for the construction of data paths that involve sensor data sources, service providers and consumers. SDMS is a useful tool for school children and a step further towards the realization of the Virtual School Lab goals.

Due to the sheer size of the project, the thesis has only focused on the core functionality of the system. A number of important further improvements are possible. They include:

1. Integrity of the collected sensor data through techniques of validation.
2. Traceability and calibration of collected sensor data. Although the system allows storing historical records and tracing changes in collected sensor data, further investigation is still needed.
3. Access control policies to govern relationships between consumers and sensor data sources. A necessary addition on the definition of dataflow descriptors is needed.
4. Improve Push and Pull mechanisms to facilitate real deployment of sensor networks.
5. Real time sensor data retrieval. Further investigation is still needed.
6. Fault-tolerant mechanism for SDMS. Real deployment of the system would involve a reliable sensor data storage system.
7. Virtualization of sensor networks allowing full control and management of individual low level sensor nodes.
8. Develop a framework for sensor network modeling. This would also include standards to process and publish sensor data and their semantics (metadata).

References

[1] Jasper Koolhaas, WTH conference, Liempde, The Netherlands (28-31 July 2005).

[2] Vijay Kr. Chaurasiya, Information Technology: How it can be helpful in the case of Natural Disaster, B-Cognizance: IIIT A Bi Monthly e-Magazine (Volume I Issue II).

[3] Yong Yao and Johannes Gehrke, The cougar approach to in-network query processing in sensor networks, ACM SIGMOD Record, Volume 31 Issue 3, September 2002, Publisher: ACM Press.

[4] K. L. Cheng and Da-Ming Zhu, On Calibration of pH Meters, Department of Chemistry and Department of Physics, University of Missouri-Kansas City, Sensors 2005, 5, pp. 209-219.

[5] Ameesh Pandya, Aman Kansal, Gregory Pottie and Mani Srivastava, Fidelity and Resource Sensitive Data Gathering, Department of Electrical Engineering, University of California, CENS Technical Report #42 , July 2004.

[6] Paul C. Clements, Constructing Superior Software, Software Quality Institute Series (SQI), Publisher: Sams, 1st edition (November 4, 1999), pp. 32, 70-71.

[7] Mark Reichardt, Sensor Data Are Spatial Data, Open Geospatial Connection, GeoWorld (April 2005).

[8] K. S. J. Pister, J. M. Kahn and B. E. Boser, Smart Dust: Wireless Networks of Millimeter-Scale Sensor Nodes, Highlight Article in 1999 Electronics Research Laboratory Research Summary.

[9] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler and Kristofer Pister, System architecture directions for network sensors, ASPLOS 2000, Cambridge, November 2000.

[10] Jeff Shneidman, Peter Pietzuch, Jonathan Ledlie, Mema Roussopoulos, Margo Seltzer and Matt Welsh, Hourglass: An Infrastructure for Connecting Sensor Networks and Applications, Harvard Technical Report TR2104.

[11] Ruth Malan and Dana Bredemeyer, The Visual Architecting Process, Architecture Resources for Enterprise Advantage, Bredemeyer Consulting, January 2005.

[12] K. Saleh and A. Al-Zarouni, Capturing Non-Functional Software Requirements Using the User Requirements Notation, The 2004 International Research Conference on Innovations in Information Technology.

[13] Lihua Xu, Hadar Ziv, Debra Richardson, and Zhixiong Liu, Towards Modeling Nonfunctional Requirements in Software Architecture, Aspect-Oriented Requirements Engineering and Architecture Design Workshop (Early Aspects 2005), held in conjunction with AOSD 2005, Chicago, Illinois, March 2005.

[14] Booch, G., I. Jacobson and J. Rumbaugh, The Unified Modeling Language User Guide. Addison-Wesley, 1999, pp. 219-241.

[15] Cockburn, Alistair, "Structuring Use Cases with Goals", Journal of Object-Oriented Programming, Sep-Oct, 1997 and Nov-Dec, 1997.

[16] J. Trowitzsch, A. Zimmermann and G. Hommel, Towards Quantitative Analysis of Real-Time UML Using Stochastic Petri Nets, ipdps, p. 139b, 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Workshop 2, 2005.

[17] Jeffrey Shneidman, Bryan Choi and Margo Seltzer, Collecting Data for One Hundred Years, Harvard Technical Report: Fall 2002 Work in Progress Description.

Appendix A
Source code of adapter dynamic compilation

```
public void BuildAssembly(string code, string adaptorName)
{
        string dllName = "\\" + adaptorName;
        Microsoft.CSharp.CSharpCodeProvider provider = new CSharpCodeProvider();
        ICodeCompiler compiler = provider.CreateCompiler();
        CompilerParameters compilerparams = new CompilerParameters();

        compilerparams.OutputAssembly =
        ConfigurationSettings.AppSettings["pathTmp"] + dllName;

        compilerparams.GenerateExecutable = false;
        compilerparams.GenerateInMemory = false;
        compilerparams.CompilerOptions = "/target:library /optimize";
        compilerparams.ReferencedAssemblies.Add("System.dll");
        compilerparams.ReferencedAssemblies.Add("System.Xml.dll");
        compilerparams.ReferencedAssemblies.Add("mscorlib.dll");

        TempFileCollection tfc = new
        TempFileCollection(ConfigurationSettings.AppSettings["pathTmp"], false);

        CompilerResults results  = new CompilerResults(tfc);
        results = compiler.CompileAssemblyFromSource(compilerparams, code);
        if (results.Errors.HasErrors)
        {
                StringBuilder errors = new StringBuilder("Compiler Errors :\r\n");
                foreach (CompilerError error in results.Errors )
                {
                        errors.AppendFormat("Line {0},{1}\t: {2}\n", error.Line,
                        error.Column, error.ErrorText);
                }
                throw new Exception(errors.ToString());
                }
        }
}
```

Appendix B
Source code of adapter dynamic execution

```
public object ExecuteAssembly(string adaptorName,  params object[] args)
{
        object objResult = null;
        object adaptorInstance = null;
        Type adaptorType = null;
        AppDomainSetup ads = new AppDomainSetup();
        AppDomain adaptorDomain = AppDomain.CreateDomain("adaptorDomain");

        string dllPath = ConfigurationSettings.AppSettings["pathTmp"]+ "\\" +
        adaptorName;

        byte[] rawAssembly = loadFile(dllPath);
        Assembly assembly = adaptorDomain.Load(rawAssembly, null);

        adaptorInstance = assembly.CreateInstance("VSLTools.Adaptors.myAdaptor");
        adaptorType = adaptorInstance.GetType();
        MethodInfo adaptorMethod = adaptorType.GetMethod("Execute");
        objResult = adaptorMethod.Invoke(adaptorInstance, args);

        adaptorInstance = null;
        adaptorType = null;
        assembly = null;

        AppDomain.Unload(adaptorDomain);
        adaptorDomain = null;

        return (objResult);
}
```

Appendix C
Source code of collection tasks scheduler

```
public void CreateTask(string strName, string strParameters, string strComment, string
strCreator, string strWorkingDirectory, string strTriggerPattern, short vsl_hour, short
vsl_minutes, TaskScheduler.DaysOfTheWeek vsl_days)
{
        TaskScheduler.Task t = st.CreateTask(strName);
        t.ApplicationName = "vsl_collector.exe";

        t.Parameters = strParameters;
        t.Comment = strComment;
        t.Creator = strCreator;
        t.WorkingDirectory = strWorkingDirectory;
        //string acct = Environment.UserName;
        t.SetAccountInformation("", null);

        t.MaxRunTime = new TimeSpan(30, 0, 0);
        t.Priority = System.Diagnostics.ProcessPriorityClass.High;

        t.Triggers.Add(new RunOnceTrigger(DateTime.Now +
        TimeSpan.FromMinutes(1)));

        if (strTriggerPattern.Equals("hourly")== true)
        {
                t.Triggers.Add(new DailyTrigger(vsl_hour, vsl_minutes));
                t.Triggers[1].DurationMinutes = 1440;
                t.Triggers[1].IntervalMinutes = 60;
        }
        else if (strTriggerPattern.Equals("daily")== true)
                t.Triggers.Add(new DailyTrigger(vsl_hour, vsl_minutes));
        else if (strTriggerPattern.Equals("weekly")== true)
                t.Triggers.Add(new WeeklyTrigger(vsl_hour, vsl_minutes, vsl_days));

        t.Save();
        t.Close();

}
```

Appendix D
Source code of Adapter that collect data from LML website

```
using System;
using System.Xml;
using System.Xml.Schema;
using System.Text.RegularExpressions;
using System.Globalization;

namespace VSLTools.Adaptors
{
	/// <summary>
	///	Adaptor for LML
	///	(c) Paul J. Manchego
	/// </summary>
	public class myAdaptor
	{	///</summary>
		///	'Execute' converts HTML published sensor data into XML
			readable format.
		///	Parameters:
		///	resultHTML: Published Sensor Data fetched from Sensor
			Network URL
		///	docXmlS: XML Schema (already defined)
		///	docXmlD: XML Data (empty documment to be filled)
		/// </summary>
	public XmlDocument Execute(string resultHTML, XmlDocument docXmlS,
	XmlDocument docXmlD)
	{
		int cn = 0;
		int elementCount = 0;
		String[] strFields;

		string strDate = resultHTML.Substring(resultHTML.IndexOf("
		",resultHTML.IndexOf("id=datum"))+1,10);

		string strTime =
		resultHTML.Substring(resultHTML.IndexOf("uur",resultHTML.IndexOf("
		",resultHTML.IndexOf("id=datum"))+1)-3,2);

		string strDateTime = strDate + " " + strTime+ ":00";
		int iniPosData = resultHTML.IndexOf("<pre>");
		int endPosData = resultHTML.IndexOf("</pre>");

	string strData = resultHTML.Substring(iniPosData,endPosData-iniPosData);

		string p1 = @"[0-9]+</a>";
		string p2 = @"\s(-[0-9]+|[0-9]+)";
		string strValue;
		int iniRow = 0;

		Match m1;
		MatchCollection mc1 = Regex.Matches(strData, p1);
		string[] station = new string[mc1.Count];
		string strCol;

		XmlNodeList ElementTypeNodes =
		docXmlS.GetElementsByTagName("ElementType");
		foreach(XmlNode node1 in ElementTypeNodes)
```

```
                if (node1.Attributes["name"].Value.Equals("Item"))
                        elementCount = node1.ChildNodes.Count;

                strFields = new string[elementCount];

                foreach(XmlNode node1 in ElementTypeNodes)
                if (node1.Attributes["name"].Value.Equals("Item"))
                        foreach(XmlNode node2 in node1.ChildNodes)
                        {
                                strFields[cn] = node2.Attributes["type"].Value;
                                cn++;
                        }

        XmlElement XmlElem1;
        XmlElement XmlElem2;
        XmlElement XmlElemData;
        XmlText XmlText1;
        XmlText XmlText2;
        XmlText XmlTextData;

        XmlElem1 = docXmlD.CreateElement("","ROOT","");
        docXmlD.AppendChild(XmlElem1);


        int ccn = 0;
        int icn = 0;

        IFormatProvider cultureNL = new CultureInfo("nl-NL", true);

        DateTime vslDateTimeNL =
        DateTime.Parse(strDateTime,cultureNL,DateTimeStyles.NoCurrentDate
        Default);

        IFormatProvider cultureUS = new CultureInfo("en-US", true);

        DateTime vslDateTime =
        DateTime.Parse(vslDateTimeNL.ToString(),cultureUS,DateTimeStyles.N
        oCurrentDateDefault);

        for (int i = 0; i < mc1.Count; i++)
        {
                station[i] = mc1[i].Value;
                XmlElem2 = docXmlD.CreateElement("","Item","");
                docXmlD.ChildNodes.Item(0).AppendChild(XmlElem2);
                iniRow = mc1[i].Index + 8;

                XmlElem1 = docXmlD.CreateElement("",strFields[0],"");

                XmlText1 =
                docXmlD.CreateTextNode(station[i].Substring(0,station[i].IndexOf
                ("<")).Trim());

                XmlElem1.AppendChild(XmlText1);

docXmlD.ChildNodes.Item(0).ChildNodes.Item(icn).AppendChild(XmlElem1);

                XmlElem2 = docXmlD.CreateElement("",strFields[1],"");
                XmlText2 = docXmlD.CreateTextNode(vslDateTime.ToString());
                XmlElem2.AppendChild(XmlText2);
```

```
docXmlD.ChildNodes.Item(0).ChildNodes.Item(icn).AppendChild(XmlElem2);

                    ccn = 2;
                    for (int j = 0; j < 7; j++)
                    {
                        strValue = strData.Substring(iniRow,8);
                        m1 = Regex.Match(strValue,p2);
                        if (m1.Success)
                        {
                            strCol = m1.Value;

                        XmlElemData =
                        docXmlD.CreateElement("",strFields[ccn],"");
                        XmlTextData = docXmlD.CreateTextNode(strCol.Trim());
                        XmlElemData.AppendChild(XmlTextData);


docXmlD.ChildNodes.Item(0).ChildNodes.Item(icn).AppendChild(XmlElemData);
                        }

                        iniRow = iniRow + 8;
                        ccn++;
                    }
                    icn++;
                }
                return (docXmlD);

                }
            }
}
```