# GeoDADIS: A Framework for the Development of Geographic Data Acquisition and Dissemination Servers

S. Villarroya, J.R.R. Viqueira*, J.M. Cotos, J.C. Flores

*Computer Graphics and Data Engineering Group (COGRADE), CITIUS, University of Santiago de Compostela. Centro Singular de Investigación en Tecnoloxías da Información, Campus Vida, 15782 Santiago de Compostela, Spain.*

**Abstract**

The homogeneous access to sensor data in data monitoring and analysis applications is gaining much interest nowadays. To tackle this problem from an application independent perspective, the design and implementation of a framework called GeoDADIS for the development of data acquisition and dissemination servers is discussed in the present paper. Those servers are of common use in monitoring applications as they perform as gateways between decision support and data visualization technologies used in application developments and the heterogeneous collection of protocols and interfaces available in the industrial area for sensor data access. To achieve its objective, the architecture of GeoDADIS consists of: i) a bottommost data acquisition layer that communicates with sensors, ii) a middle kernel layer that provides general purpose functionality related to data management and system control and iii) a topmost external interaction layer that enables the access from applications. The frameworks design does extensive use of the adapter (wrapper) design pattern to ease the incorporation of new data acquisition channels at the data acquisition layer and new data and remote control services in the external interaction layer. This makes GeoDADIS a very flexible and general purpose tool with broad application in many data monitoring domains.

*Keywords:*
Data Acquisition System, Data Logging, Sensor Data, SCADA, Distributed Control System, System Architecture

## 1. Introduction

The number of small devices devoted to the observation of different parameters of our environment is increasing exponentially nowadays. Many geographic information systems are being developed to take advantage of the huge amount of data produced every now and then by such devices. Many research and development areas are related to the efficient and intelligent exploitation of such information: Geographic Data Management, Data Logging, Environmental Observation Systems, Supervisory Control and Data Acquisition (SCADA), Sensor Networks, Distributed Control Systems. A common issue to all those areas is the need to access data produced by heterogeneous sensor networks, i.e, sensors (temperature sensors, GPS devices, etc.) with different software and hardware specifications that are accessed through different communication protocols (RS-232, ZigBee, Bluetooth, etc.).

If we focus on Data Acquisition and Monitoring applications (Kolar et al., 2009; Villarroya et al., 2009; Miller et al., 1995; Moore et al., 1988; Koutroulis and Kalaitzakis, 2003) and on non-real time supervisory control (Chimaris and Papadopoulos, 2007; Daneels and Salter, 1999), three main components use to be part of general system architectures, namely, *Sensing Devices*, *Data Servers* and *End-User Applications*. *Sensing Devices* are in charge of the observation process. Discussion is here restricted to in-situ devices, i.e., those that measure within an area surrounding the device and



*Corresponding author: Tel.:+34 881816463. Fax: +34 881816409.

*Email addresses:* sebastian.villarroya@usc.es (S. Villarroya), jrr.viqueira@usc.es (J.R.R. Viqueira ), manel.cotos@usc.es (J.M. Cotos), julian.flores@usc.es (J.C. Flores)

whose geographic location is used as geographic metadata for their measures. Device functionality and communication capabilities use to be highly heterogenous, even if we restrict to the environmental monitoring area. To enable homogeneous data access, one or more *Data Servers* are added to the system. These servers behave as a kind of gateways between the Information and Communication Technology (ICT) world and the heterogeneous collection of different field buses and serial communications available at the industrial world (Kolar et al., 2009). *End-User Applications* enable data visualization and analysis.

Programming *Sensing Devices* is highly conditioned by vendor specifications whereas the development of *End-User Applications* is highly dependent on the specific domain and user preferences, therefore generalization efforts for those components would tend to be fruitless. On the contrary, the functionality and architecture of *Data Servers* is not much different from one application to another. In spite of this, various important challenges have to be overcome during the design of a general purpose *Data Server*, all of them related to the flexibility required for the server to be used in different in-situ monitoring applications.

- New in-situ *Sensing Devices* should be incorporated in the framework with a minimum effort.
- New data dissemination services of any type should be incorporated in the framework with a minimum effort.
- Different persistent data storage technologies should be allowed for different measured parameters. Incorporation of such technologies should be done with minimum effort.

Based on the above, in this paper, an effort towards the generalization of the design of a data acquisition and dissemination server has been undertaken. More precisely, a framework called GeoDADIS has been designed whose generic functionality enables: i) in-situ data acquisition through both synchronous and asynchronous data communication channels and ii) data dissemination through data services that follow both client/server and publish/subscribe protocols.

The remainder of this paper is organized as follows. Related pieces of work are discussed in Section 2. In Section 3, generic framework requirements are identified through the generalization of a use case. Section 4 is devoted to the definition of the framework architecture. Further details on the design and implementation of the key components of the system are given in Section 5. Experimental deployments of the framework that enabled testing its flexibility are reported in Section 6. Finally, Section 7 concludes the paper.

## 2. Related Work

Many research and development efforts have been undertaken during the last decade related to the exploitation of sensor networks in monitoring and control applications. Distributed Control Systems (DCS) are composed of highly heterogeneous collections of physical devices using a wide variety of communication protocols (Colnaric et al., 2008). In general, the control functionality of a DCS is far beyond the data acquisition capabilities of GeoDADIS, including also real-time and fault tolerance constraints. In spite of this, similarities may be found between GeoDADIS and some typical DCS components. Thus, for example, the Virtual Field Bus layer of the CORFU framework (Thramboulidis and Tranoris, 2001) may be extended with new field bus wrappers to enable specific communication channels. Similarly, extensibility is also enabled in a TORERO device through the programming of device functions (Schwab et al., 2005).

Restricting to data acquisition and supervisory control (SCADA) (Daneels and Salter, 1999), typical architectures include a data server layer that behaves as a gateway between devices and applications, providing at the same time data recording and management capabilities. Devices are connected to data servers by different means, including direct links, networks and either standard or proprietary field buses. Homogeneous access to devices is achieved by data servers only when standards are implemented by device vendors (OPC Task Force, 1998; Goldschmidt, 1998).

The functionality and structure of GeoDADIS is similar to the one of *Remote Units* of Chimaris and Papadopoulos (2007), except for the fact that *Remote Units* may include control functionality that is out of the scope of GeoDADIS. However, the flexibility requirements imposed during the design of GeoDADIS for data dissemination, data storage and remote control are not present in Chimaris and Papadopoulos (2007).

In the remainder of this section, the discussion will restrict to data acquisition for monitoring purposes, which is the scope of the main functionality of GeoDADIS. Thus, in Horsburgh et al. (2011), the five components of an architecture for an environmental observatory information system are described. Related to GeoDADIS are the*Data Storage and Metadata* component and the *Data Publication and Interoperability*

component. The former enables persistent storage of both sensor data and metadata. An important value-added step in this component involves the mediation across the variety of software supporting sensor and communication systems. Such a mediation is achieved in GeoDADIS by the implementation of wrappers for different data acquisition channels. The latter provides data dissemination functionality, achieved in GeoDADIS by the implementation of data services.

Similar to the above approach is the GeoSWIFT distributed geospatial infrastructure for the Sensor Web proposed in Liang et al. (2005). The core of GeoSWIFT is the open geospatial *Sensing Service* that provides a web-enabled interface for sensor systems and their geospatial information. As in the case of GeoDADIS, the *Sensing Service* of GeoSWIFT acts as a gateway that hides the different communication protocols, data formats and standards of sensor systems and provides a standard interface for clients to collect and access sensor observations. The sensor data access standard used by GeoSWIFT is based on the specifications provided by the Sensor Web Enablement (SWE) initiative (Botts et al., 2007) of the Open Geospatial Consortium (OGC). Similarly, a data access gateway is also implemented in Musaloiu-E. et al. (2006) to gather data from a Wireless Sensor Network (WSN) for soil monitoring. None of the above approaches achieves the flexibility requirements imposed during the design of GeoDADIS.

The general architecture of the SPINE framework (Bellifemine et al., 2011) is composed of a collection of *Sensor Nodes* connected to one *Coordinator Node*, that manages the network, collects and analyzes the data and acts as a gateway to connect the sensors with wide area networks. The *Sensor Node* manages and abstracts the sensors, providing a standard interface to the diverse sensor drives and it is responsible for sampling the sensors and storing the sensed data in properly defined buffers. Two major differences may be found between SPINE and GeoDADIS. First, SPINE enables the incorporation of signal processing functionality that is out of the scope of GeoDADIS. Second, the flexibility in the incorporation of new data dissemination and remote control services of GeoDADIS is not present in SPINE.

To summarize, various components of systems and frameworks have similarities in either structure or functionality with GeoDADIS. Thus, dealing with heterogeneity in the access to data communication protocols and devices is a common problem solved in GeoDADIS that is also present in the areas of DCS, SCADA and generally in sensor networks for monitoring purposes. However, to the best of these authors knowledge, an effort for the generalization of data acquisition and dissemination servers as the present one is original and with great applicability in many scientific and industrial domains. Besides, the flexibility of GeoDADIS in the incorporation of different technologies for data acquisition, data persistence and data dissemination is not present in any of the approaches described in this section, what makes it unique. This is the main contribution of the present piece of work.

## 3. Generic Requirements: Generalization of a Energy Monitoring Use Case

Energy efficiency is a problem of great importance for economic and environmental reasons in many production areas. The characterization of the generation and consumption of energy in a fishing vessel during fishing activities was attempted in the "Green Fish" Project, founded by the Spanish and Galician public administrations. To achieve this, data acquisition systems for energy generation and consumption monitoring (Villarroya et al., 2009) were developed and deployed in various fishing vessels. Figure 1 depicts a typical architecture of one of those systems.

GeoDADIS was designed, as a generalization of the architectures used in the "Green Fish" project, to enable data acquisition and data dissemination in in-situ sensor platforms. Data acquisition must be supported from any kind of sensor through any kind of communication channel with a minimum effort. Data dissemination must be possible through a wide variety of means, ranging from manual transportation of physical storage to automatic delivering through communication networks. Most relevant challenges are now materialized in the following requirements.

**REQ 1.** It must support both synchronous and asynchronous data acquisition channels. In the former, the framework pulls measurements from the sensors. This is the case of the Modbus protocol over a RS-485 link shown in Figure 1. In the latter, measurements are pushed to the framework from the sensors. Examples of this are the NMEA-0183 channel over a RS-232 link and the Ethernet link based channel of Figure 1.

**REQ 2.** New sensors that measure new parameters and communicate through new data acquisition channels must be linked to the framework with a minimum effort. Notice that even in the present use case, sensor configurations may vary from one vessel to another.

**REQ 3.** The frameworks configuration must enable the specification of the data acquisition channels used to i)

obtain the geographic location of the platform and ii) synchronize the clock. GPS devices are generally used for those purposes.

**REQ 4.** The frameworks configuration must support the specification for each parameter of i) the range of historic measurements recorded and ii) the frequency of the sampling process. For example, registering values of temperature every ten minutes during the last year.

**REQ 5.** It must support both client/server and publish/subscribe data services. In the former, services pull data by querying the framework (*FTP Server* component in Figure 1). In the latter, data is pushed to the services from the framework (*Data Streamming* and *Data Export* components in Figure 1).

**REQ 6.** New data services must be implemented with a minimum effort. Clear candidates are the standard web services defined in the OGC-SWE initiative.

**REQ 7.** New remote administration services must be implemented with a minimum effort. The *Telnet Server* in Figure 1 is a simple generic solution, although careless administrators might damage the system by using wrong shell commands.

**REQ 8.** Different data persistence technologies must be implemented for different measured parameters with a minimum effort. Notice for example the different storage capabilities required by a simple temperature value and a complex image.

## 4. Frameworks Architecture

The general components architecture of GeoDADIS (see Figure 2) is organized into three main software layers. The *Data and Control Management* layer provides general purpose functionality related to data and configuration management and system control. More precisely, the *DataManager* provides persistent storage functionality to the *DataAcquisition* component and data query functionality to the *DataDissemination* component. The *ConfigurationManager* component enables the remainder components to access the configuration data set up through the *RemoteAdm* component. Further details on data management and systems configuration are given in Subsection 5.1. Finally, the *SystemControl* component provides functionality related to the start up, stop and restart of the various components of GeoDADIS.

At the bottom of the architecture, the *Data Acquisition* layer provides functionality related to the sampling of measures delivered by sensors through the available data acquisition channels. The time range during which measures are recorded and the frequency of the sampling process are configured by the administrator for each measured parameter of each sensor (see REQ 4 in Section 3). Therefore, sampling times are determined by the system following a time-driven (time-triggered) approach, in spite of the fact that some data acquisition channels may behave in an event-driven (event-triggered) manner. Issues related to the use of time-triggered and event-triggered concepts in the design and implementation of control systems are discussed in Chen and Eberlein (2003); Scarlett and Brennan (2006); Albert and Bosch (2004). As it is shown in Figure 2, each data acquisition channel must have a channel manager component, which is considered external to GeoDADIS. Asynchronous channels (*AsynchChannelManager* in the figure) are used to access event-triggered sensors whereas Synchronous channels (*SynchChannelManager* in the figure) enable the query of time-triggered sensors (see REQ 1 in Section 3). Every time an event-triggered sensor generates a new measure, its corresponding *AsynchChannelManager* uses the *iDataAcqInsert* interface to deliver the measured value to GeoDADIS. The last measure received from each such sensor is temporarily recorded in a buffer of the *DataAcquisition* component. On the other hand, measures of time-triggered sensors are sampled directly from the corresponding *SynchChannelManager*. Once a new measure is sampled, the *DataAcquisition* component uses the *iStamp* interface to get from the *StampManager* a spatio-temporal stamp for the measure, i.e., a time value together with the geographic location of the platform. The measure together with the stamp is finally delivered to the *DataManager* through its *iDataMan* interface. The design and implementation of the *DataAcquisition* component is discussed in more detail in Subsection 5.2. Periodically, a thread of the *StampManager* component uses the *iStRefresh* interface to get the current value of the spatio-temporal stamp. Location and time components of the stamp are obtained from specific data acquisition channels. Both the data acquisition channels used to get the stamp components and the refresh period are part of the configuration data supplied by the *ConfigurationManager* (see REQ 3 in Section 3).

Finally, the *External Interaction* layer provides general purpose functionality related to the interaction between GeoDADIS and its clients and remote administrators. Dealing with the possibly bidirectional communication between the available data services (*DataSubscriber* and *DataClient* components that are considered external to GeoDADIS) and the *DataManager* com-

ponent is the responsibility of the *DataDissemination* component. Queries over the recorded measures issued by a *DataClient* through the *iDataDisQuery* interface are delegated to the *DataManager* through its *iDataMan* interface. On the other hand, upon the insertion of a new measurement the *DataManager* component notifies the *DataDissemination* component through its *iDataDisPublish* interface. The notified measurement is next delivered to the appropriate *DataSubscribers*, which in response may also use the *iDataDisQuery* to do specific queries. To summarize, the communication between GeoDADIS and *DataSubscribers* follows a publish/subscriber model whereas the communication with *DataClients* is based on a client/server approach (see REQ 5 in Section 3). An in depth description of the internal design of the *DataDissemination* component is provided in Subsection 5.3. To complete the external interaction layer, the *RemoteAdm* component enables GeoDADIS administrators to deal with system configuration and control. Thus, configuration and control request issued by a *AdmClient* component (external to GeoDADIS) are delegated respectively to the *ConfigurationManager* component and to the *SystemControl* component (see REQ 7 in Section 3).

## 5. Frameworks Detailed Design and Implementation

Before getting into the detailed design of various important GeoDADIS components, some brief general descriptions are given related to three well known design patterns (Gamma et al., 1995).

- The *Singleton* pattern is used to restrict the instantiation of a class to just one object, to enable coordinated access to shared resources. Examples of such resources in GeoDADIS are the configuration data, data acquisition channels, data services and control clients.

- The *Adapter* (*Wrapper*) pattern is used to translate one interface for one class into a compatible interface, thus, enabling the cooperation between classes that otherwise would not be able to work together. Adapters are used in GeoDADIS to access the specific interfaces of data acquisition channels, data services and control clients in a uniform manner.

- The *Observer* (*publish/subscribe*) pattern is used to notify a list of objects, called *observers*, when some change occurs in the state of a *subject* object. In GeoDADIS, this pattern is used to implement the publish/subscribe communication between the *DataDissemination* component (*subject*) and the *DataSubscriber* components (*observers*).

### *5.1. Data and Configuration Managers*

The internal structure of the *ConfigurationManager* is shown in the UML class diagram of Figure 3(a). The singleton design pattern enables the *ConfigurationManagerFacade* to provide coordinated access to the available configuration data through its *iConfMan* interface. Starting, stoping and restarting the component may be done through its *iConfManControl* interface. Configuration data consists of the following three collections.

- *parameters* (*ParameterVO* class): Each measured parameter has *samplingInterval* and *recordInterval* properties. The former is used by the *DataAcquisition* component to determine its sampling frequency. The latter is used by the *DataManager* component to determine when a recorded measure is old enough to be deleted (see REQ 4 in Section 3). The sensor and therefore the data acquisition channel from which the parameter has to be sampled are also recorded, as it is shown in the figure. The *dataAccessClass* property records the name of the data access object class that the *DataManager* component has to use to access the parameters data persistence functionality (see REQ 8 in Section 3). Finally, it is noticed that both simple parameters (*simpleParameterVO* class) like temperature and compound parameters (*compoundParameterVO* class) like wind (it has speed and direction components) are supported.

- *dataChannels* (*DacqChannelVO* class): Each data acquisition channel has a *dacqChannelType* property that determines whether the channel is synchronous or asynchronous. Besides, the *dacqChannelAdapterClassName* property records the name of the adapter class that the *DataAcquisition* component has to use to interact with the external channel manager (see REQ 2 in Section 3). The data acquisition channels that are used as location and time sources are determined by associations with respective roles *locSource* and *timeSource* (see REQ 3 in Section 3).

- *dataServices* (*DataServiceVO* class): Each data service has a *dataServiceType* (either data client or data subscriber). The list of parameters about

which each data subscriber (*DataSubscriberVO*) has to be notified is also recorded. The *dataServiceAdapterClassName* property records the name of the adapter class that the *DataDissemination* component has to use to interact with the external data service component (see REQ 6 in Section 3).

The functionality of the *DataManager* component (see Figure 3(b) for a graphical representation of its internal structure) is accessed through the *DataManagerFacade* class. More precisely, its *iDataManControl* interface provides component control functinality and its *iDataMan* interface enables inserting and retrieving measurements from persistent storage. In particular, the *getMeasures* operation is used to obtain the recorded measures of the parameter identified by *paramId* for which the spatio-temporal filter $f$ holds. Such a filter enables the combined use of both the interval predicates defined by Allen (1983) and the spatial predicates defined by Clementini and Di Felice (1996). Retrieved measures include both measured value (various values in case of compound parameters) and spatial and temporal stamps. On the other hand, *insertMeasure* operation records a measure in persisten storage and delivers it to the *DataDissemination* component through its *iDataDisPublish* interface (in order to notify relevant subscribers). Data insertions and queries of measures of a given parameter are executed by its relevant *ParameterDAO* class, enabling this way different implementations for different parameters (see REQ 8 in Section 3). The pseudocode given in the figure for the *insertMeasure* operation of the *DataManagerFacade* illustrates the use of the collection of *ParameterDAO* classes and the *iDataDisPublish* interface.

### *5.2. Data Acquisition Component*

The functionality of the *DataAcquisition* component (see Figure 4 for a graphical representation of its internal structure) is accessed through the *DataAcqManager* class. Its *iDataAcqControl* interface provides component control functionality and its *iStRefresh* interface is used to obtain the current spatio-temporal stamp from the appropriate data acquisition channels. The implementation of the class follows a singleton design pattern in order to coordinate the concurrent access to both the sampling threads and the data acquisition channels. As it shown in pseudocode given in the figure for class *DataAcqManager*, the implementation of method *getStamp* of the *iStRefresh* interface access directly the data channels (class *AbstractDataAcqChannel*) configured as location and time sources (see REQ 3 in Section 3). Regarding the data acquisition process, each sensed parameter is sampled by a different *ParameterSampler* thread. This implementation is also illustrated with pseudocode in the figure. First, the thread sleeps during a given *samplingInterval* that is obtained from the configuration data of the specific parameter (see REQ 4 in Section 3). After waking up, the thread uses its data acquisition channel, which is also obtained from the configuration data, to obtain the next measure of the parameter. Next, the current stamp obtained from the *iStamp* interface is used to associate current time and location to the measure. Finally, the stamped measure is delivered to the *DataManager* component through its *iDataMan* interface.

A data acquisition channel (*AbstractDataAcqChannel*) may access measures of either a synchronous (*SynchDataAcqChannel*) or an asynchronous (*AsynchDataAcqChannel*) external channel manager component. Measures of synchronous channel managers are accessed directly through a relevant adapter class (*SynchDataAcqChannelAdapter*), which is obtained from configuration data and that implements the *iSynchDataAcqChannelAdapter* interface. On the other hand, measures of asynchronous channel managers are obtained from a buffer (*AsynchInputBuffer*), that is populated with the last measure of each parameter through the *iDataAcqInsert* interface. Coordinated access to the buffer is achieved through the use of the singleton design pattern for its implementation. Notice that an adapter class is still required for asynchronous channel managers in order to access their control functionality (start, stop and restart the manager) from GeoDADIS.

To summarize, the *DataAcquisition* component of GeoDADIS provides general purpose data acquisition functionality over both synchronous and asynchronous data communication channels (see REQ 1 in Section 3). The use of the adapter (wrapper) design pattern restricts the effort of adding a new channel to the implementation of only a new data acquisition channel adapter class (see REQ 2 in Section 3).

### *5.3. Data Dissemination Component*

The internal structure of the *DataDissemination* component is shown in the UML class diagram of Figure 5. The main class of the component is the *DataServicesManager* that implements the component interfaces and uses the singleton design pattern to coordinate the access to the available data services. The implementation of the operations of the interfaces that enable the querying (*iDataDisQuery*) and publishing (*iDataDisPublish*) of measures is described by relevant pseudocode in the figure. Thus, the methods *getParameters* and *getMeasures* of the *iDataDisQuery* interface,

which are used by external data service components to query the recorded data, are implemented by simply delegating, respectively, on the kernel configuration and data manager components of GeoDADIS. A combination of the observer and adapter design patterns is used for the implementation of the *publichMeasure* method of the *iDataDisPublish* interface, which is used by the *DataManager* component to deliver inserted measures to appropriate external data subscribers. More precisely, the *DataServicesManager* (subject class of the observer pattern) notifies each *DataSubscriberAdapter*(both observer class and adapter class) upon the reception of a new measure. The name of the adapter class for each data service is part of its configuration data as it is also the list of data subscribers that must be notified for each parameter. Notice that adapter classes are also needed for data clients in order to enable GeoDADIS to access their control functionality (start, stop and restart them).

As a resume, the *DataDissemination* component combines singleton, observer and adapter design patterns to provide flexible and general purpose functionality related to external data access. Such access may be either from data clients through the *iDataDisQuery* interface of the component, following a client/server approach or from data subscribers following a publish/subscribe protocol (see REQ 5 in Section 3). The use of the adapter (wrapper) design pattern restricts the effort of adding a new data service to the implementation of only a new either data subscriber or data client adapter class (see REQ 6 in Section 3).

## 6. Experimental Deployments

A couple of data acquisition and dissemination servers were developed in the context of the use case of Section 3, using specializations of a C++ implementation of GeoDADIS. A first *Light Weight Gateway* enables just data acquisition and delivering to an external data server, thus it does not incorporate data storage functionality. A second *Data Server* includes data storage, data export and data download functionality, avoiding the need of a permanent connection to an external data server. The components architecture of the latter is depicted in Figure 6. Both servers were successfully deployed in a Owa22I-ETH/b datalogger with a Linux operating system, enabling the testing of the three key features of GeoDADIS, as it is illustrated below. Beyond that, the two experimental deployments enabled also the validation of all the generic requirements specified in Section 3.

*Flexibility in the incorporation of new synchronous and asynchronous data acquisition channels.* Data acquisition channels in both developments are those of the use case of Section 3. More precisely, *NMEAChannelManager* and *EthernetChannelManager* are listening respectively to the RS-232 and ethernet buffers of the datalogger. They push measures to GeoDADIS through the *iDataAcqInsert* interface asynchronously. Contrary to the above, measures are pulled by GeoDADIS from the synchronous *MODBUSChannelManager* component. Communication between GeoDADIS and those channel managers is enabled through the implementation of the three adapter classes depicted at the bottom of the GeoDADIS component of Figure 6 (see also grey classes in Figure 4). Thus, to incorporate a new data acquisition channel in GeoDADIS it suffices with the implementation of an adapter class and entering configuration data for its sensors and parameters.

*Flexibility in the incorporation of new client/server and publish/subscribe data services.* The *Data Server* was designed having in mind vessels with WIFI and GSM communications enabled only when the ship approaches the harbor, requiring therefore persistent recording of measures. Data export is performed by a publish/subcribe *netCDFDataExport* component that records the received measures in standard netCDF (Domenico, 2011) files in removable storage. Data download is enabled through a client/server *SOSServer* component, that implements core operations of the OGC SOS interface (Na and Priest, 2007). SOS requests are transformed to operations of the *iDataDisQuery* interface of GeoDADIS (see Figure 5). On the other hand, the *Light Weight Gateway* was designed for vessels with permanent network link through satellite communications. Data storage is not required in the ship and data dissemination is performed through a data streaming publish/subscribe service. Measures published by GeoDADIS are transformed to invocations of the *insertObservation* operation of the transactional SOS interface of a remote server. Notice that to incorporate a new data service in GeoDADIS it suffices with the implementation of a new adapter class (see netCDFExportAdapter and SOSServerAdapter in Figure 6 and relevant grey classes in Figure 5).

*Flexibility in the incorporation of different data persistence technologies.* Persistent storage is only required in the *Data Server* deployment. GeoDADIS had to be extended with a new *FileSystemDAO* class that implements the *iParameterDAO* interface of Figure 3. All the measured parameters are assigned the same file sys-

tem based persistent storage technology, although different technologies (files, databases, etc.) implemented by different *ParameterDAO* classes (see grey class in Figure 3) could be included for different parameters.

## 7. Conclusions and Future Work

A generalization effort for the development of data acquisition and dissemination servers was undertaken during the design of the GeoDADIS framework. The scalable and extensible architecture of GeoDADIS solves the problem of dealing with heterogeneity in sensor data access and sensor data dissemination, which is a common problem in various research and development areas related to data acquisition and monitoring.

Various software design patterns have been used during the design of GeoDADIS components that helped in achieving nice flexibility features. A singleton pattern was chosen to implement classes that coordinate concurrent access to the resources provided by GeoDADIS (configuration data, data services, data acquisition channels, data buffers, etc.). The observer pattern is used by a data dissemination component to incorporate publish/subscribe data services. Finally, the adapter pattern enables the incorporation of new data services, remote control services and data acquisition channels with a minimum impact in the kernel components of the system. In fact, only configuration data has to be updated for the incorporation of those elements. The flexibility key features of the framework together with other important generic requirements were validated during the development of a couple of GeoDADIS based in-situ data acquisition and dissemination systems.

Future work on GeoDADIS is mainly related to the support of remote sensors (laser based measures, cameras, etc.) and their complex measures. Besides, a small variation of the GeoDADIS architecture is currently being considered as the base for the development of a sensor data storage and dissemination platform in the context of home automation.

## 8. Acknowledgments

The Green Fish project has been sponsored by the Spanish Ministry of Education and Science in the scope of Unique and Strategic Projects, ref. PSE-370300-2006-1. The work of J.R.R. Viqueira has been partially supported by the Spanish Ministry of Science and Innovation (TIN2010-21246-C02-02) and by Xunta de Galicia (PGIDIT 09MDS034522PR). The authors are grateful to the anonymous reviewers, whose valuable comments helped in improving the paper.

## References

Albert, A., Bosch, R., 2004. Comparison of event-triggered and time-triggered concepts with regard to distributed control systems, in: Proceedings Embedded World, Nürnberg, Germany, pp. 235 – 252.

Allen, J.F., 1983. Maintaining knowledge about temporal intervals. Communications of the ACM 26, 832–843.

Bellifemine, F., Fortino, G., Giannantonio, R., Gravina, R., Guerrieri, A., Sgroi, M., 2011. Spine: a domain-specific framework for rapid prototyping of wbsn applications. Software: Practice and Experience 41, 237–265.

Botts, M., Percivall, G., Reed, C., Davidson, J., 2007. OGC Sensor Web Enablement: Overview And High Level Architecture. Technical Report. Open Geospatial Consortium (OGC). URL: http://www.opengeospatial.org/projects/groups/sensorwebdwg (accessed 20 March 2012).

Chen, L., Eberlein, A., 2003. A framework of a web-based distributed control system, in: Canadian Conference on Electrical and Computer Engineering, Montreal, Canada, pp. 1351 – 1354.

Chimaris, A.N., Papadopoulos, G.A., 2007. Implementing a generic component-based framework for telecontrol applications. Software: Practice and Experience 37, 1087–1132.

Clementini, E., Di Felice, P., 1996. A model for representing topological relationships between complex geometric features in spatial databases. Information Sciences: An International Journal 90, 121–136.

Colnaric, M., Verber, D., Halang, W., 2008. Distributed Embedded Control Systems. Improving Dependability with Coherent Design. Springer, London, UK, 268pp.

Daneels, A., Salter, W., 1999. What is scada?, in: Proceedings International Conference on Accelerator and Large Experimental Physics Control Systems, Trieste, Italy, pp. 339–343.

Domenico, B., 2011. OGC Network Common Data Form (NetCDF) Core Encoding Standard version 1.0. Technical Report. Open Geospatial Consortium (OGC). URL: http://www.opengeospatial.org/standards/netcdf (accessed 20 March 2012).

Gamma, E., Helm, R., Johnson, R., Vlissides, J., 1995. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, NJ, USA, 416pp.

Goldschmidt, I., 1998. The development of bacnet. Strategic Planning for Energy and the Environment 18, 16–24.

Horsburgh, J.S., Tarboton, D.G., Maidment, D.R., Zaslavsky, I., 2011. Components of an environmental observatory information system. Computers & Geosciences 37, 207 – 218.

Kolar, H., Cronin, J., Hartswick, P., Sanderson, A., Bonner, J., Hotaling, L., Ambrosio, R., Liu, Z., Passow, M., Reath, M., 2009. Complex real-time environmental monitoring of the hudson river and estuary system. IBM Journal of Research and Development 53, 4:1–4:10.

Koutroulis, E., Kalaitzakis, K., 2003. Development of an integrated data-acquisition system for renewable energy sources systems monitoring. Renewable Energy 28, 139 – 152.

Liang, S.H., Croitoru, A., Tao, C.V., 2005. A distributed geospatial infrastructure for sensor web. Computers & Geosciences 31, 221 – 231.

Miller, R., Davis, K., von Zweck, O., Sprague, V.G., J., Sommers, J., 1995. Shipboard data acquisition, processing and analysis: an integrated software approach, in: Proceedings OCEANS 1995 MTS/IEEE "Challenges of Our Changing Global Environment", San Diego, CA, USA, pp. 1794 – 1799.

Moore, J., Charters, J., de Moustier, C., 1988. Multi-sensor real-time data acquisition and preprocessing at sea, in: Proceedings OCEANS 1988 MTS/IEEE "A Partnership of Marine Interests", Baltimore, MD, USA, pp. 509 – 517.

Musaloiu-E., R., Terzis, A., Szlavecz, K., Szalay, A., Cogan, J., Gray, J., 2006. Life under your feet: A wireless soil ecology sensor network, in: Third Workshop on Embedded Networked Sensors (EmNets), Cambridge, MA, USA. URL: http://www.eecs.harvard.edu/emnets/ (accessed 20 March 2012).

Na, A., Priest, M., 2007. Sensor Observation Service. Technical Report. Open Geospatial Consortium (OGC). URL: http://www.opengeospatial.org/standards/sos (accessed 20 March 2012).

OPC Task Force, 1998. OLE for Process Control Overview. Technical Report. OPC Foundation. URL: http://www.opcfoundation.org (accessed 15 March 2012).

Scarlett, J., Brennan, R., 2006. Re-evaluating event-triggered and time-triggered systems, in: IEEE International Conference on Emerging Technologies and Factory Automation, Prague, Czech Republic, pp. 655 – 661.

Schwab, C., Tangermann, M., Ferrarini, L., 2005. Web based methodology for engineering and maintenance of distributed control systems: the torero approach, in: 3rd IEEE International Conference on Industrial Informatics, INDIN 2005, Perth, Australia, pp. 32 – 37.

Thramboulidis, K., Tranoris, C., 2001. An architecture for the development of function block oriented engineering support systems, in: IEEE International Symposium on Computational Intelligence in Robotics and Automation, Banff, Alberta, Canada, pp. 536 – 542.

Villarroya, S., Otero, M., Romero, L., Cotos, J., Pita, V., 2009. Modular and scalable multi-interface data acquisition architecture design for energy monitoring in fishing vessels, in: Omatu, S., Rocha, M., Bravo, J., Fernández, F., Corchado, E., Bustillo, A., Corchado, J. (Eds.), Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living. Springer Berlin / Heidelberg. volume 5518 of *Lecture Notes in Computer Science*, pp. 531–538.

## Figures

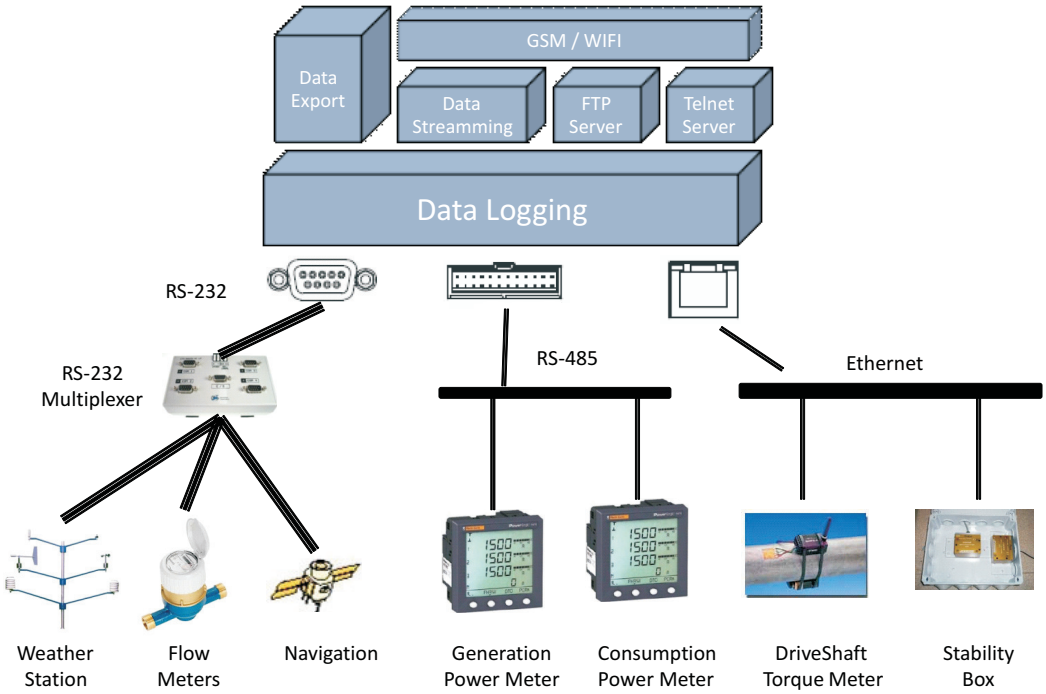



Figure 1: Example of architecture for Data Acquisition in the "Green Fish" Project

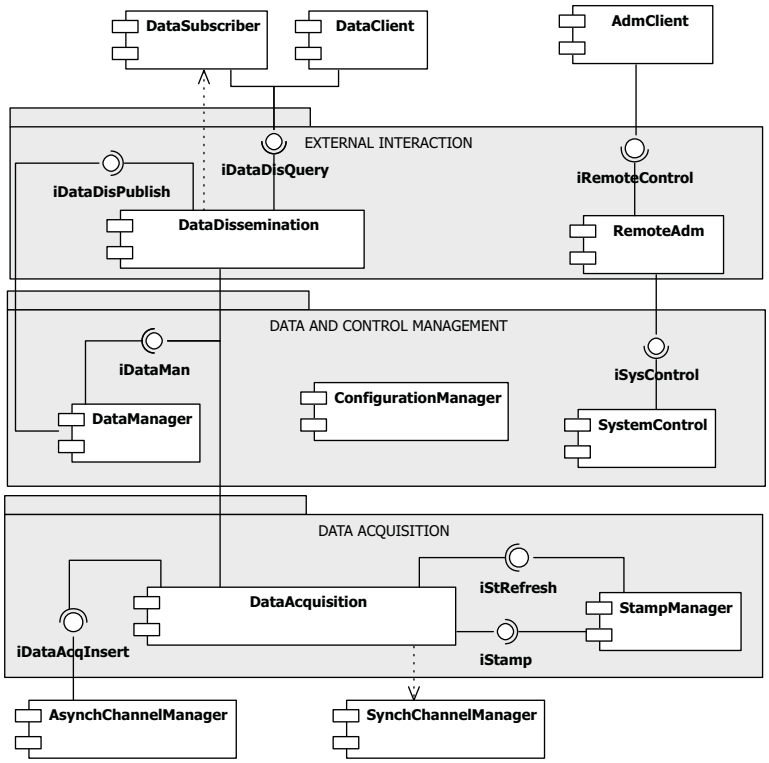
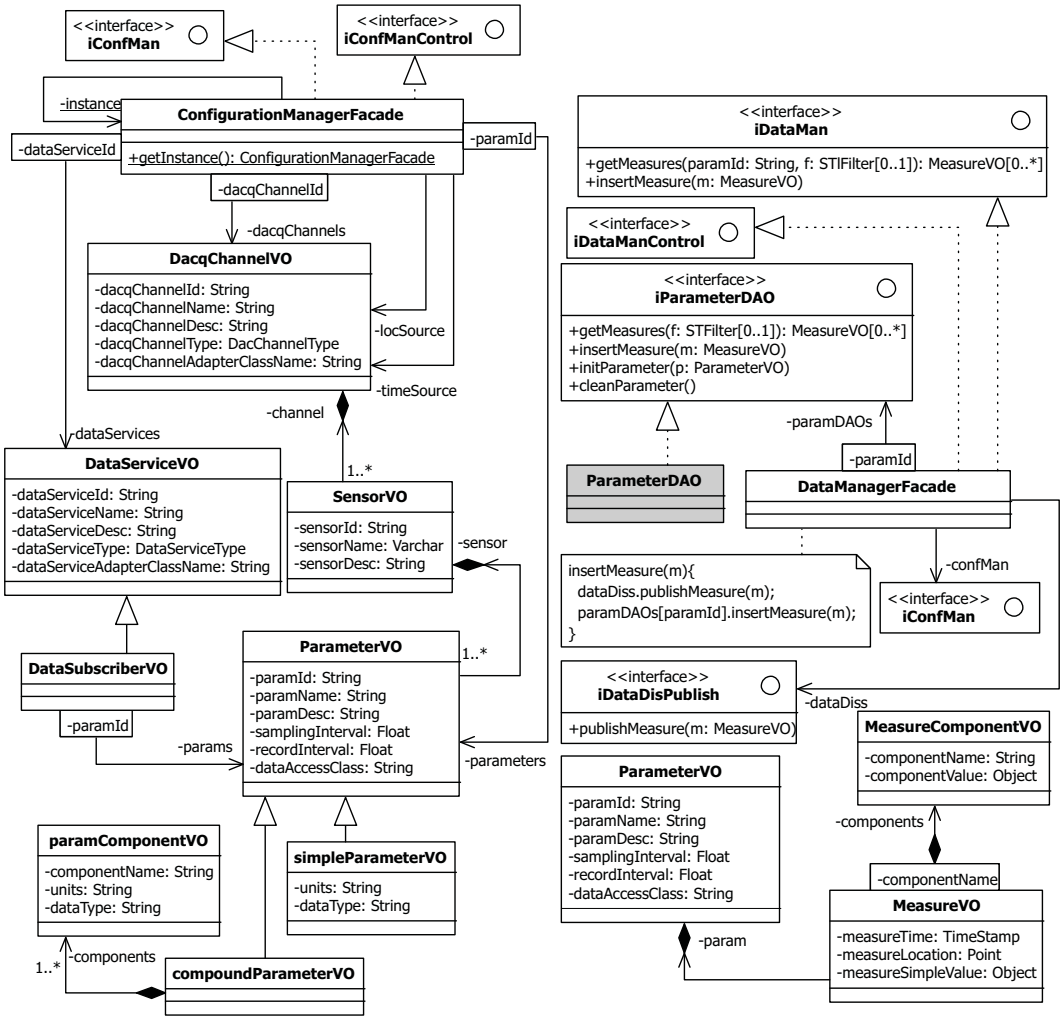



Figure 2: GeoDADIS General Components Architecture.




(a) ConfigurationManager (b) DataManager

Figure 3: Data and Configuration Managers.

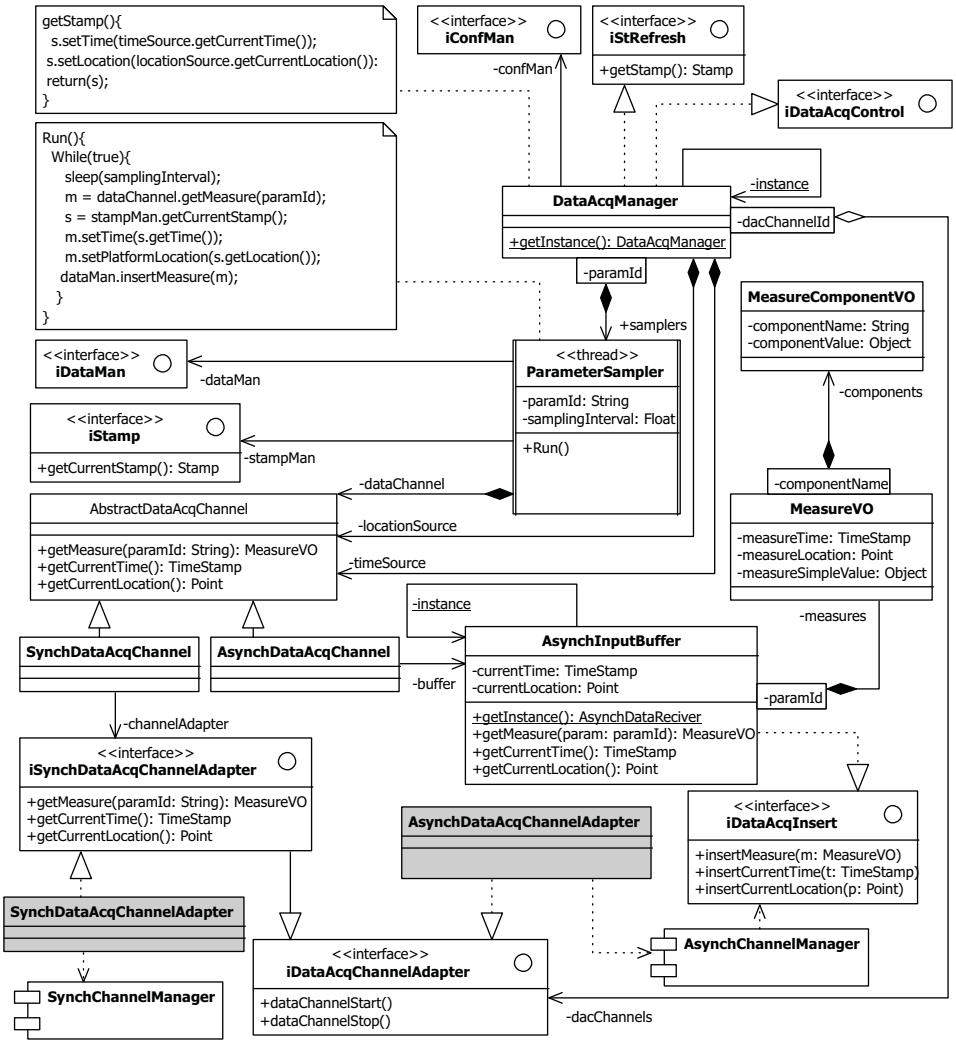



Figure 4: Data Acquisition Component.

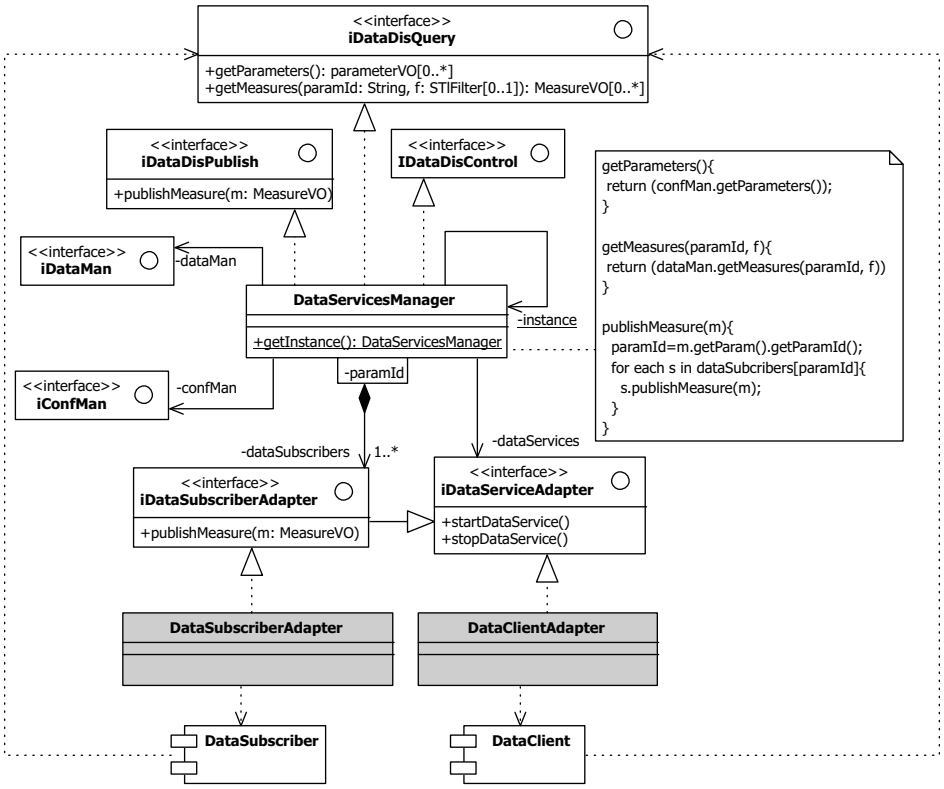



Figure 5: Data Dissemination Component.

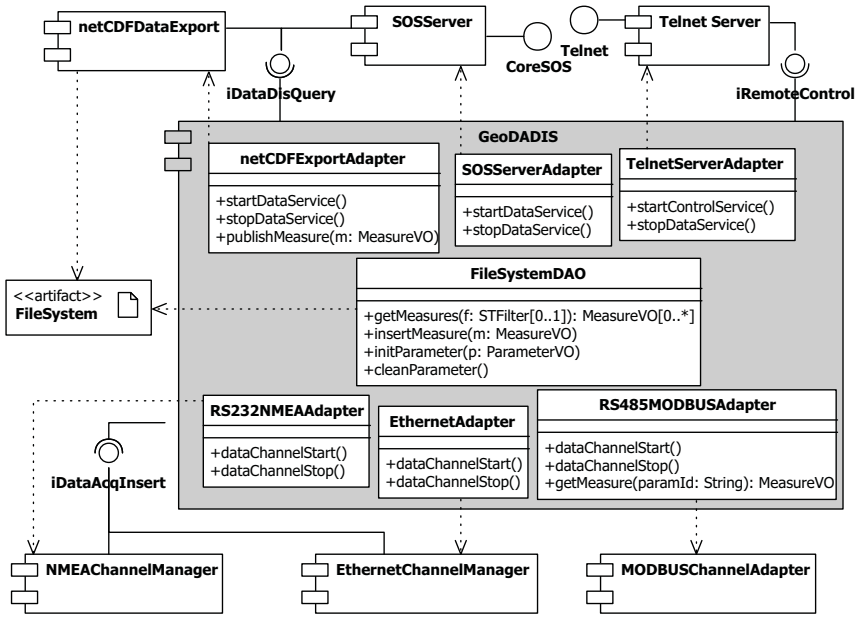



Figure 6: Data Acquisition and Dissemination Server Based on GeoDADIS.