

A Conceptual Model for Remote Data Acquisition Systems

Txomin Nieva*, Alain Wegmann

Institute for computer Communications and Applications (ICA), Communication Systems
Department (DSC), Swiss Federal Institute of Technology (EPFL),
CH-1015 Lausanne, Switzerland
{Txomin.Nieva, Alain.Wegmann}@epfl.ch

Abstract. Data acquisition systems (DAS) are the basis for building monitoring tools that enable supervision of local and remote systems. Unfortunately, DASs are commonly based on proprietary technologies. The data format usually depends on the industrial process, the fieldbus characteristics or the development platform. Currently, there are many standards of DASs, but none of them offer a well-accepted Application Programming Interface (API). However, all of them comply with the same conceptual model. Understanding this model allows for the significant improvement of the design of a specific DAS. In this paper, we propose a conceptual model of a generic DAS. This model gives researchers an abstraction of DASs and a quasi-formal specification of a generic DAS. It also enables developers to compare the existing standards and/or to propose a new open standard.

1 Introduction

In the last few years, companies from different business areas (such as power engineering and transportation) have become increasingly interested in maintenance management. Maintenance improves the reliability/availability of equipment and therefore the quality of service, which managers have found provides substantial benefits. However, maintenance management makes up anywhere from 15 to 40% of total product cost [1]. Consequently, improving maintenance management can also represent a substantial benefit to companies. Traditionally, there are two major maintenance approaches: Corrective Maintenance and Preventive/Predictive Maintenance (PPM). Corrective maintenance focuses on efficiently repairing or replacing equipment after the occurrence of a failure. Corrective maintenance aims to increase the maintainability of equipment by improving the speed of repair, or return to service, after a failure. PPM focuses on keeping equipment in good condition in order to minimize failures; repairing components before they fail. PPM aims to increase the reliability of equipment by reducing the frequency of failures. A successful management technique that can be applied for improving both techniques

* Corresponding author.

is the on-line supervision of the health of the equipment, which is usually known as condition monitoring. Condition monitoring is defined by *Davies* in [2] as:

“Condition monitoring is a management technique that uses the regular evaluation of the actual operating condition of plant equipment, production systems and plant management functions, to optimize total plant operation.”

Condition monitoring, applied to maintenance tasks, provides necessary data in order to schedule preventive maintenance and to predict failures before they happen. Condition monitoring is based on direct monitoring of the state of equipment to estimate its Mean Time To Failure (MTTF). DASs and monitoring systems provide condition monitoring systems with the necessary information about the state of equipment. Remote access to this information provides significant benefits by allowing for the collection of condition-related data from wherever equipment is located. Remote monitoring systems exist in different business areas such as building [3], power engineering [4] and transportation systems [5]. Some remote monitoring systems make use of the Internet network and Internet technologies.

Substantial benefits can also be obtained by the intensive use of Asset Management Systems (AMS). Asset Management is defined by the *Government of Victoria* [6] as:

“The process of guiding the acquisition, use and disposal of assets to make the most of their service delivery potential (i.e., future economic benefit) and manage the related risks and costs over their entire life.”

Asset management is a complementary task to maintenance. It provides support for the planning and operation phases. Similar to maintenance tasks, in AMSs access to utility data source is essential. *Draber et al.* proposed in [7] a data warehouse to homogenize data access from distributed and heterogeneous data sources.

In summary, DASs and remote monitoring systems build the infrastructure (see Fig. 1) needed to provide condition monitoring and AMSs with information about the state of equipment. Condition monitoring systems will analyze this data to estimate the MTTF. AMSs will propose or update preventive and predictive maintenance plans based on the information provided by the condition monitoring systems.

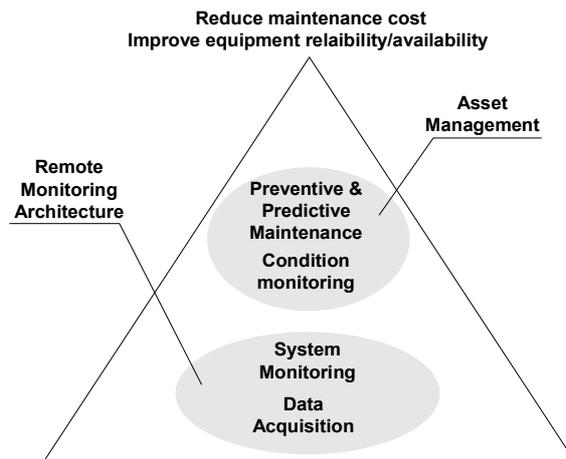


Fig. 1. Remote Monitoring & Asset Management

In this paper, we focus on the remote monitoring architecture and particularly on DASs. Unfortunately, DASs and monitoring systems are commonly based on proprietary technologies. The data format usually depends on the industrial process, the fieldbus characteristics or the development platform. Currently, there are many standards on DASs (see “OLE for Process and Control” (OPC) [8], “Interchangeable Virtual Instrument” (IVI) [9] and “Open Data Acquisition Standard” (ODAS) [10] among others). However, none of them provide a well-accepted API, which would enable for the communication of data between heterogeneous DASs. The “Object Management Group” (OMG) is currently addressing the issue of the lack of a universal API in [11]. Although there is not a well-accepted API for DASs, all DASs comply with the same information model. In [12] *Schenck* and *Wilson* propose the following definition of an information model:

“An information model is a formal description of types of ideas, facts and processes which together form a model of a portion of interest of the real world and which provides an explicit set of interpretation rules.”

Understanding the information model will make DASs easier to understand, because it only focuses on the main aspects of the DAS by hiding low-level details that renders it difficult to understand. *Bubenko et al.* noted in [13] that:

“An effective approach to analyzing and understanding a complex phenomenon is to create a model of it. By a model is meant a simple and familiar structure or mechanism that can be used to interpret some part of reality. A model is always easier to study than the phenomenon it models, because it captures just a few of the aspects of the phenomenon”.

In this paper, we present a conceptual model of a generic DAS. A conceptual model is an information model of a system from the object perspective that shows the relevant concepts of the system. Our conceptual model is the main result of a systematic review of DASs and software patterns, and from our practical experience with the development of a web-based monitoring tool applied to railway equipment [5, 14]. This model gives researchers a high level of abstraction of generic DASs. It also gives them a formal way to discuss about the main aspects of a DAS, and it enables them to compare existing standards and/or to propose a new open standard. We use Unified Modeling Language (UML) [15] as the modeling language to present our model.

Our conceptual model is inspired by several software patterns. A definition of software patterns is found in [16]:

“Patterns for software development are a literary form of software engineering problem-solving discipline that has its roots in a design movement of the same name in contemporary architecture, literate programming, and the documentation of best practices and lessons learned in all vocations”.

This paper is organized as follows: First, we give a definition of DAS. Secondly, we present a web-based monitoring system applied to railway equipment that we developed. Then, we propose a generic conceptual model for DASs, which is based on this practical experience and some software patterns. Finally, we draw conclusions from the actual work.

2 Data Acquisition System

The following definition of a DAS is found in [17]:

“A DAS is a set of hardware and software resources designed to compute the internal representation and then, to deliver to the user the external representation”.

Although this definition is appropriate, it does not reflect certain important aspects of a DAS. We postulate that a DAS is a system that gives:

- Means to *discover* and *access* system data.
- Means to *interpret* and *process* system data, in order to generate system information.
- Means to *publish* system information.

In order to clarify this definition we adopted the following definitions according to [18]:

- Discovering: “to obtain sight or knowledge of for the first time”.
- Access: “to get at”.
- Interpret: “to explain or tell the meaning of”.
- Data processing: “the converting of raw data to machine-readable form and its subsequent processing”.
- Publish: “to produce or release for distribution”.

Therefore, our definition of a DAS is:

“A DAS is a set of hardware and software resources that provides the means to obtain knowledge of a system, provides the means to access to system data, converts system data to more useful system information and distributes this information to the user”.

3 The RoMain System

We developed, in collaboration with *ABB Corporate Research* and in the frame of the *Railway Open System Interconnection Network (ROSIN)* European project, a web-based monitoring tool for trains that supports maintenance work. This monitoring tool was called Railway Open Maintenance tool (RoMain). The kernel of this tool is a DAS (developed on Java) installed on-board a train. The objective of this tool is not to replace the existing control network, but rather to enhance it with a parallel low-cost on-line data network for railways in order to support maintenance work. This data network will allow maintenance staff to supervise railway equipment from anywhere at anytime. It will also enable experts at different locations to collaborate and to get ahead of maintenance tasks. The user requirements for such a tool were: ubiquitous access, low cost, user friendly interface with textual and graphical views of the information and easy update of equipment documentation. Taking into account all these requirements, we decided to take an approach based on the Internet. The Internet has had a revolutionary impact on office automation, and now there is a clear trend towards using Internet technologies for industrial automation. The introduction

of Internet technologies for accessing embedded systems is mostly cost driven, thus bringing significant benefits:

- Reduction of the development cost of an application, by enabling the use of Common Off-The-Self (COTS) software components.
- Elimination of the cost of a proprietary communication network, by using the common Internet network.
- Reduction of the cost of development of a client application for each different platform, by using a standard web-browser as a single client interface for heterogeneous platforms.
- Elimination of the cost of installing proprietary client applications, as the client interface is a standard web browser usually pre-installed on the client machine.
- Reduction of the cost of maintaining up-to-date equipment documentation, by offering a simple way (hyperlinks) to publish documents accessible immediately from anywhere in the world.
- Reduction of maintenance personal traveling costs, by the possibility of ubiquitous access to the information.
- Reduction of maintenance scheduling costs, by the possibility of ubiquitous access to the information at any time.

The architecture of the RoMain system, shown in Fig. 2, is composed of:

- *Train Gateways* - connected to the train network gather actual train data.
- *Ground Stations* - automatically establish connections to train gateways over wireless networks.
- *Name and Directory Servers* - provide information about the train component models and train directory.
- *Manufacturer Servers* - provide on-line information about train components, for example fact sheets, user manuals, or installation instructions.
- *Maintenance Stations* - run a standard web browser to access train data.

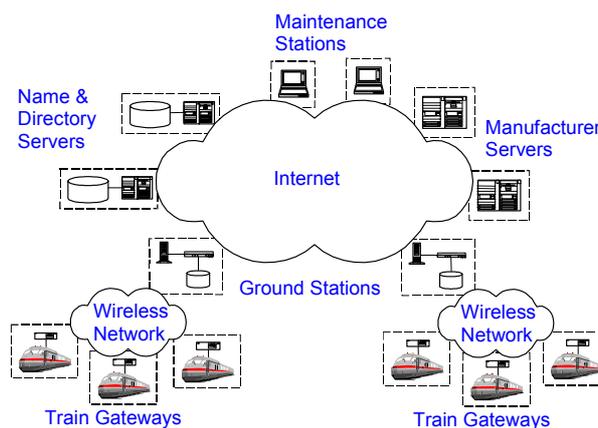


Fig. 2. The RoMain System

All the systems are interconnected by means of a secure TCP/IP network, usually the *Internet*, or eventually an *Intranet* or *Virtual Private Network*.

In this paper we expose a conceptual model for a generic DAS. This model has been developed as part of an iterative process consisting of the analysis, specification, implementation and deployment of a DAS for the RoMain system.

4 A Data Acquisition Conceptual Model

For a better understanding of the model, we group the concepts in four main packages, as shown in Fig. 3.

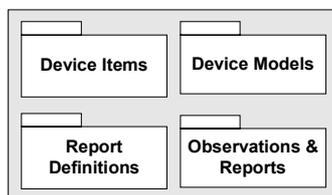


Fig. 3. Data Acquisition Main Packages

- *Device Models*: this package groups all the concepts regarding device models. A device model represents a model that characterizes a set of devices.
- *Device Items*: this package groups all the concepts regarding device items. A device item represents a real device that satisfies a device model.
- *Report Definitions*: this package groups all the concepts that allow the definition of criteria for generating monitoring reports.
- *Observations & Reports*: this package groups all the concepts regarding observations and reports taken on a system. Observations are classified as quantitative (measurements) or qualitative (category observations) according to the measurements and observations analysis pattern described by *Fowler* in [19]. Monitoring reports are classified as reports that record a snapshot of the system at a specific time (status reports) and reports that indicate a certain state of the system (event reports).

In the following sections, we describe in detail each of these packages. Finally, we conclude with a complete conceptual model diagram¹ representing the relevant concepts that make up any DAS, and their relationships.

¹ In the models, we distinguish between “operational” and “knowledge level” concepts. We adopt this idea from *Fowler* according to [19]. At the operational level the model records the day-to-day events of the domain, while at the knowledge level the model records the general rules that govern this structure. We represent knowledge level concepts by using a box with a thick border, and a box with a thin border represents operational level concepts.

4.1 Device Models

An instance of Device Model is the representation of a model, created in the design process, that characterizes a set of devices. In this section, we describe the concepts and relationships related to device models and how these models are organized. The conceptual model corresponding to device models is shown in Fig. 4.

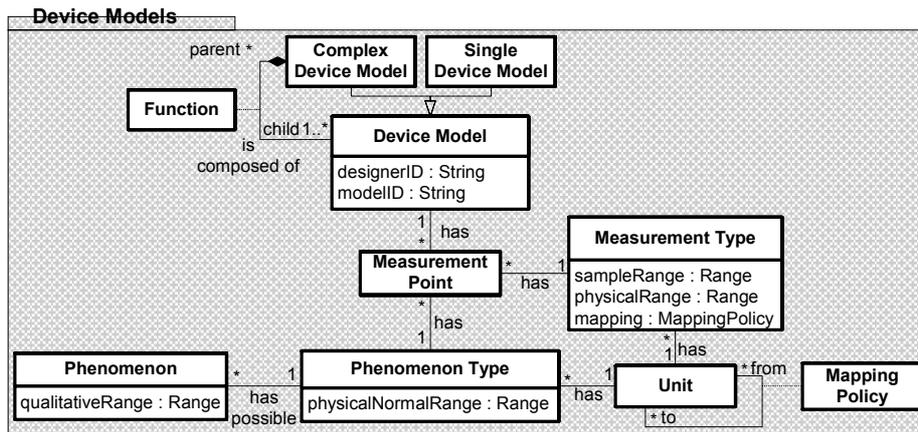


Fig. 4. Device Models

A device model defines many measurement points. An instance of “Measurement Point” defines a measurement point associated with a phenomenon type and a measurement type. An instance of “Phenomenon Type” represents something that can be quantitatively (e.g. “temperature”), or qualitatively (e.g. “door_status”), observed. A phenomenon type defines the units on which observations of this phenomenon type are expressed. We adopt the convention of using standard SI (metric) units, as proposed in [3], for phenomenon types. Eventually, a phenomenon type records the set of potential qualitative values that a measurement of such phenomenon type can take (e.g. “temperature_low”, “temperature_medium”, and “temperature_high”). Each of these values is an instance of “Phenomenon”. Phenomenon records the range of quantitative observations of a phenomenon type that corresponds to a qualitative observation. This enables the automatic recording of an occurrence of this phenomenon upon a quantitative observation, of the corresponding phenomenon type, with a value within the range of the phenomenon. An instance of “Measurement Type” is associated with a measurement point to give some semantic information about the measurements taken at this measurement point. A measurement type defines the permissible ranges of sampled and physical values of a phenomenon type in a measurement point. In industrial DASs, it is very common that the value actually measured (we refer to this value as “sampled value”) does not correspond to the physical value. Therefore, a measurement type also defines a mapping policy that makes it possible to calculate the physical value from the sampled value. A measurement type also defines the units in which a real

measurement is taken. These units are not necessarily the same as the units of the corresponding phenomenon type. In this case, a mapping policy between units defines the conversion from measurement type units to phenomenon type units. In the following sub-sections we describe the device model composition, how to define and assign a global unique identifier (GUID) to a device model, and we give more details about mapping policies.

Device Model Composition. In DASs, tree structures allow us to efficiently define an industrial system because an industrial system is usually composed of many parts, which can also be composed of many other parts in a part-whole hierarchy. For example, an HVAC (heating, ventilation and air condition) system is composed of subsystems such as heating coil, cooling coil, supply fan, etc., that can be composed of other subsystems such as temperature sensors, ventilation sensors and so on. Part-Whole relationships are commonly used to model the construction of composite objects out of individual parts. Part-Whole relationship categories and their application in object-oriented analysis are further discussed by *Motchnig-Pitrik et al.* in [20]. We used the “Composite” pattern, detailed in [21], to compose objects into tree structures to represent part-whole hierarchies. “Device Model” implements default behavior for a device model. “Complex Device Model” defines behavior for a device model that is composed of other device models. A device model that is part of a complex device model implements a function on this complex device model. The association class “Function” allows us to record this information. A function must be unique in the naming space of the complex device model.

Device Model Identifier. A GUID must be assigned to a device model. Device designers are responsible for assigning a designer specific model identifier to their device models. This identifier, which we named “modelID”, allows us to distinguish between two different models belonging to the same designer. A designer identifier, which we named “designerID”, allows us to distinguish between two different manufacturers. As a result, a device model GUID is obtained from the combination of “designerID” and “modelID”.

Mapping Policy. A mapping policy defines the conversion between two numerical values. “Function Mapping Policy” represents a mapping policy with a complex function ($y=f(x)$) while “Linear Mapping Policy” represents a mapping policy with a linear function ($y=Ax+B$); where “x” corresponds to the original value and “y” to the calculated value. The mapping policy model is shown in Fig. 5.

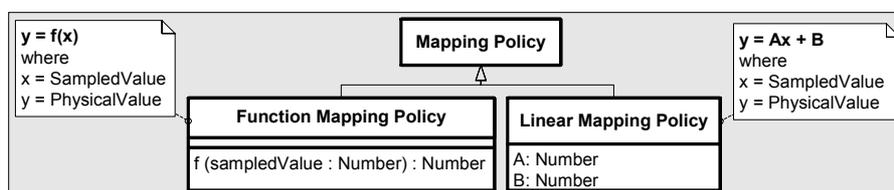


Fig. 5. Mapping Policy

4.2 Device Items

An instance of Device Item is a real item, created in the manufacturing process, that represents a real device. An instance of Device Item is described by an instance of Device Model. In this section we describe the concepts and relationships related to device items and how these items are organized. The conceptual model corresponding to device items is shown in Fig. 6.

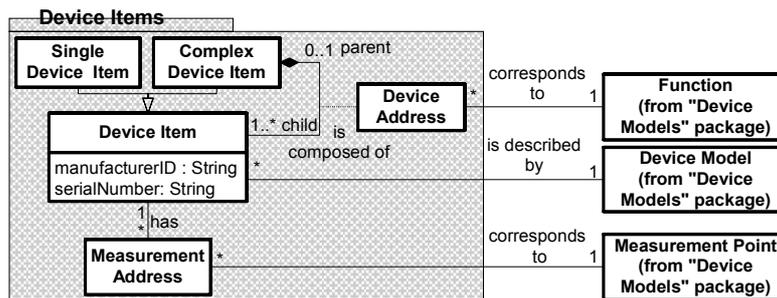


Fig. 6. Device Items

A device item is described by a device model. A device model can be associated with many device items. Each device item defines many measurement addresses. An instance of “Measurement Address” defines the actual location in a device item associated with a measurement point, where observations of a phenomenon type are taken. In the following sub-sections, we give more details about the device item composition and how to assign and define a GUID to a device item.

Device Item Composition. Similar to device models, device items are organized using the Composite pattern. An instance of Device Item is an occurrence of an instance of a Device Model. As a consequence, there is an analogous relationship between pairs of device items and pairs of the corresponding device models. A device item that is part of a complex device item is installed on a device address of the complex device item. “Device Address” allows us to record this information. A device address is unique within the naming space of a complex device item. A device address is associated with the corresponding function that a device item is implementing on a complex device item.

Device Item Identifier. A GUID must be assigned to a device item. Device manufacturers are responsible for assigning a unique identifier, which is named “serialNumber”, to each device item. “serialNumber” uniquely identifies device items of the same device model. In order to be able to globally identify a device item, it is necessary to include the device model GUID. As a result, a device item GUID is obtained from the combination of its corresponding device model GUID and a “serialNumber”.

4.3 Report Definitions

The ability to define reports, with a consistent status, of a set of data (the OMG DAIS RFP [11] refers to this as “DataSet”, and OPC [22, 23] as “OPC Group”) is one of the requirements of any DAS. Our approach for defining reports is inspired by *Mansouri-Samani* and *Sloman* [24]. The report model is shown in Fig. 7.

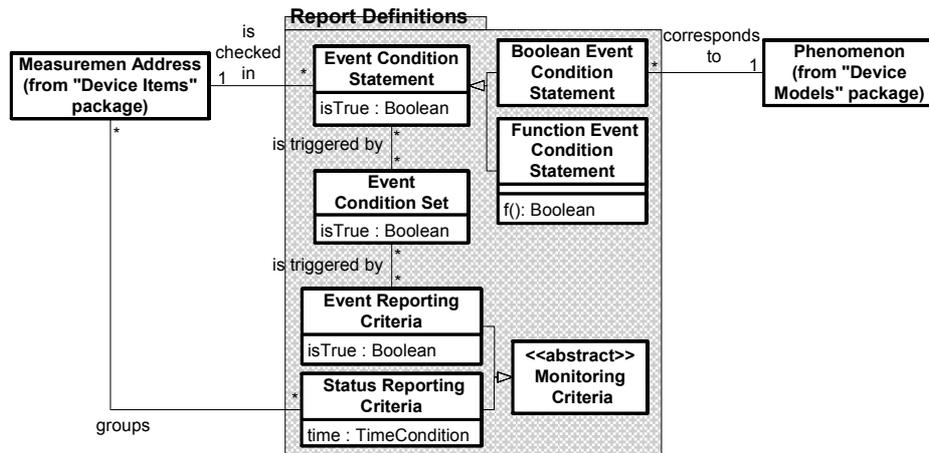


Fig. 7. Report Definitions

“Status Reporting Criteria” allows us to define a dataset with a list of measurement addresses where to observe periodically, or by schedule, phenomenon types. A report of this type is triggered by a time condition. “Event Reporting Criteria” allows us to define conditions in order to automatically generate reports when the system is under a certain state. In order to explain an event reporting criteria, we make use of the algebraic notation². Then, we state that “Event Condition Statement” allows us to record $X=A$ and $X=A'$ condition statements; where A means that a certain phenomenon has been observed in a measurement address, as recorded by “Boolean Event Condition Statement”, or that the result of a certain function “f()”checked in a measurement address returns true, as recorded by “Function Event Condition Statement”. “Event Condition Set” allows us to record conditions such as $X=A.B$ and $X=(A.B)'$; where A, B are “Event Condition Statements”. “Event Reporting Criteria” allows us to record reporting criteria such as $X=A+B$ and $X=(A+B)'$; where A, B are “Event Condition Sets”. This allows recording any even criteria, because any criteria can be expressed by means of an algebraic combination of “Event Condition Statements” with the “AND” logical operator and an algebraic combination of “Event

² “. ” corresponds to the “AND” logical operator; “+” corresponds to the “OR” logical operator; and “'” corresponds to the “NOT” logical operator

Condition Sets” with the “OR” logical operator. A transformation of any algebraic expression into these terms is possible by applying one of *De Morgan*'s law³.

Time Condition. “Time Condition” allows us to define periodical or by schedule time conditions to record status reports. The time condition model is show in Fig. 8. “Period” allows us to define a time condition as a period of time in milliseconds to enable taking periodical reports, while “Schedule” allows us to define a schedule when a status report will be generated. A schedule is defined by a recurrence time, a recurrence pattern and a recurrence range. “Recurrence Time” allows us to define a 24-hour period (with precision in milliseconds) when a status report will be generated. “Recurrence Pattern” allows us to define several ways to record a time pattern for occurrences of a status report; “Daily”, “Weekly”, “Monthly” and “Yearly” allow us to define a status report to be generated every certain number of days, every certain number of weeks on some specific days of a week, every certain number of months on a specific day of the month, and every certain number of years on a specific day of a certain month, respectively. Finally, “Recurrence Range” allows us to define a beginning time (just a “Begin Date”) to start recording status reports and an end time to stop recording status reports. “End Time” allows us to define the end time by a number of occurrences, by a specific end date or with no end (meaning that the status report will be generated “forever”).

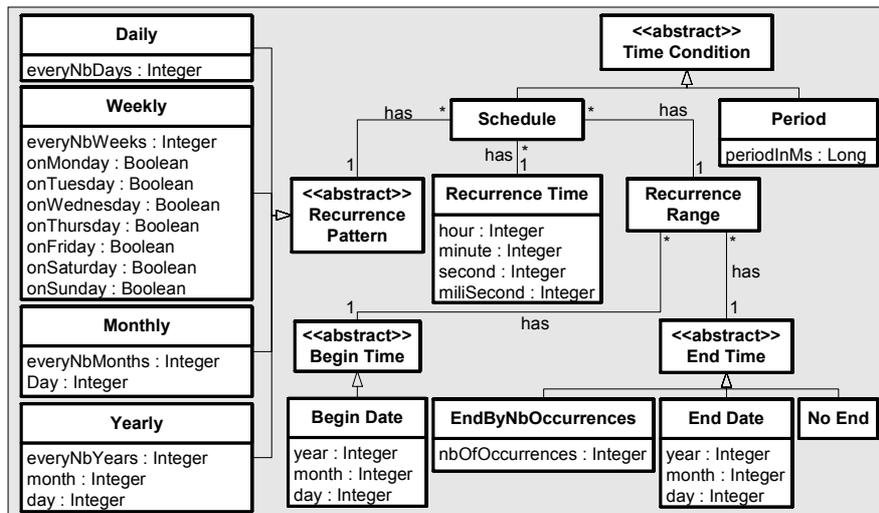


Fig. 8. Time Condition

3 The two laws, known as *De Morgan*'s, are: $(A+B)'=A'.B'$; and $(A.B)'=A'+B'$

4.4 Observations

The observation model, shown in Fig. 9, defines concepts that allow us to record observations taken on a device item. Our observation model is inspired by the “observations and measurements” analysis pattern described by Fowler in [19].

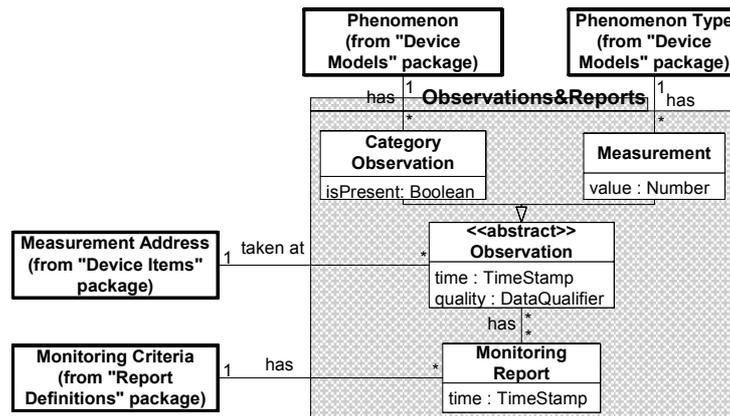


Fig. 9. Observations

In a device address we can record many observations. “Observation” is an abstract concept that represents both quantitative and qualitative observations. An observation records a timestamp to record the time an observation was taken. “Measurement” represents quantitative observations. A measurement records the physical value corresponding to the measurement, which is represented by the “value” attribute. A measurement is associated with a phenomenon type, while a phenomenon type can have many measurements. A “Category Observation” represents a qualitative observation. A category observation is associated with a phenomenon, while a phenomenon can have many category observations. Sometimes recording that a phenomenon is absent is as important as recording its presence. The “isPresent” Boolean attribute of category observation is added to enable recording the absence or presence of a phenomenon. In the following sub-sections we give more details about timestamps and data qualifiers associated with observations.

Timestamps. Recording the time an observation was taken is a key issue for enabling a subsequent analysis of observations. In order to avoid anomalies due to inconsistent time formats (e.g because of different time zones), we adopted the convention of storing all timestamps using UTC (Universal Coordinated Time) format. This, further discussed in [3], is a common practice in DASs.

Data Qualifiers. In DASs it is a common practice to include a data qualifier (see Fig. 10) with an observation. According to [11] a “Data Qualifier” includes information about the “Validity” (“valid”, “held” from a previous value, “suspect”, “not valid” or

“substituted” manually), the “Current Source” (“metered”, “calculated”, “entered”, or “estimated”) and the “Normal Value” (“normal” or “abnormal”) of an observation.

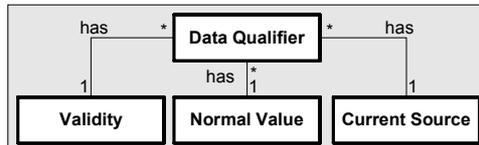


Fig. 10. Data Qualifier

4.5 Final Data Acquisition Model

The final conceptual model for DASs is shown in Fig. 11.

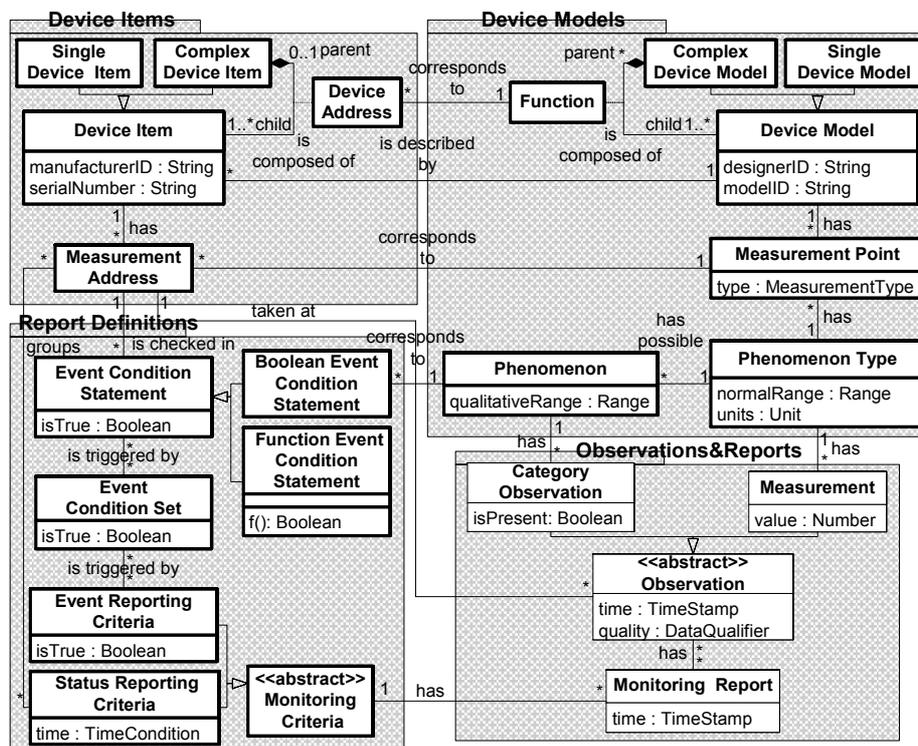


Fig. 11. Data Acquisition Conceptual Model⁴

⁴ To simplify the model we show only the main attributes of a concept. For the same reason, some concepts have been intentionally designed as attributes of higher-level concepts.

5 Conclusions and Future Work

In this paper, we propose a conceptual model of a generic DAS. This model gives researchers an abstraction of DASs and a quasi-formal specification of a generic DAS. It also enables developers to compare existing standards and/or to propose a new open standard. Understanding this model allows for the significant improvement of the design and development of a specific DAS. This model is generic enough to be applied for the design and development of similar DASs in many application domains such as building, power engineering and transportation systems.

This model has been developed as part of an iterative process consisting of the analysis, specification, implementation and deployment of a DAS for railway equipment. We use our own variation of the Catalysis [25] development process based on UML. The conceptual model specifies only the static aspects of a generic DAS. Currently, we are complementing this model with use case models that describe the dynamic aspects of a generic DAS. Conceptual and use case models are the foundation for specifying and designing specific DASs based on specific requirements of quality of service (QoS). We aim to provide some guidelines that, based on our generic conceptual and use case models, will help developer to make the right architectural choices depending on the specific requirements of a DAS for a specific system.

Acknowledgments

Thanks to all the members of the *ROSIN WP4* who brought a real framework to the discussions and the implementation of our hypotheses. To *Andreas Fabri* and *Hubert Kirrmann*, from *ABB Corporate Research*, for their valuable contributions to the research work of this paper. To *Guy Genilloud*, from *ICA*, for his reviews of the UML diagrams. And last but not least, to *Holly Cogliati*, from *ICA*, for proofreading this paper.

References

- [1] T.Wireman, “*Computerized Maintenance Management Systems*”, Industrial Press, Inc, 1994.
- [2] A.Davies, “*Handbook of Condition Monitoring - Techniques and Methodology*”, Kluwer Academic Publishers, 1997.
- [3] F.Olken, H.A.Jacobsen, C.McParland, M.A.Piette, and M.F.Anderson, “*Objects lessons learned from a distributed system for remote building monitoring and operation*” presented at Conference on Object-oriented Programming, Systems, Languages and Applications, Vancouver, Canada, October 18-22, 1998, <http://www.lbl.gov/~olken/rbo/rbo.html>.
- [4] R.Itschner, C.Pommerell, and M.Rutishauser, “*GLASS: Remote Monitoring of Embedded Systems in Power Engineering*” in *IEEE Internet Computing*, vol 2, 1998.
- [5] A.Fabri, T.Nieva, and P.Umiliacchi, “*Use of the Internet for Remote Train Monitoring and Control: the ROSIN Project*” presented at Rail Technology '99,

- London, UK, September 7-8, 1999,
<http://icawww.epfl.ch/nieva/thesis/Conferences/RailTech99/article/RailTech99.PDF>.
- [6] Victorian Government, “*Asset Management Series: Principles, Policies and Practices*”, November, 1995, <http://home.vicnet.net.au/~assetman/welcome.htm>.
- [7] S.Draber, E.Gelle, T.Kostic, O.Preiss, and U.Schluchter, “*How Operation Data Helps Manage Lifecycle Costs*” presented at International Conference on Large High Voltage Electric Systems - CIGRE'2000, Paris , France, August 27 - September 2, 2000.
- [8] OPC Foundation, “*OLE for Process and Control Standard*”, 1997,
<http://www.opcfoundation.org>.
- [9] IVI Foundation, “*Interchangeable Virtual Instruments Standard*”, 1997,
<http://www.ivifoundation.org/>.
- [10] Open Data Acquisition Association, “*Open Data Acquisition Standard*”, 1998,
<http://www.opendaq.org/>.
- [11] OMG, “*Data Acquisition from Industrial Systems (DAIS)*”, Request for Proposal (RFP), OMG Document: dtc/99-01-02, January 15, 1999,
http://www.omg.org/techprocess/meetings/schedule/Data_Acquisition_RFP.html.
- [12] D.A.Schenck and P.R.Wilson, “*Information Modeling: The EXPRESS Way*”, Oxford University Press, 1994.
- [13] M.Boman, J.A.Bubenko Jr., P.Johannesson, and B.Wangler, “*Conceptual Modelling*”, Prentice Hall, 1997.
- [14] T.Nieva, “*Automatic Configuration for Remote Diagnosis and Monitoring of Railway Equipment*” presented at IASTED International Conference - Applied Informatics, Innsbruck, Austria, February 15-18, 1999,
<http://icawww.epfl.ch/nieva/thesis/Conferences/ai99/article/ai99.pdf>.
- [15] J.Rumbaugh, I.Jacobson, and G.Booch, “*The Unified Modelling Language Reference Manual*”, Addison Wesley, 1999, <http://www.rational.com>, <http://www.omg.org>.
- [16] B.Appleton, “*Patterns and Software: Essential Concepts and Terminology*”, November 20, 1997, <http://www.enteract.com/~bradapp/docs/patterns-intro.html>.
- [17] J.Ehrlich, A.Zerrouki, and N.Demassieux, “*Distributed Architecture for Data Acquisition: a Generic Model*” presented at IEEE Instrumentation and Measurement Technology Conference - IMTC'97, Ottawa, Canada, May 19-21, 1997.
- [18] Merriam-Webster, “*Webster Dictionary*”, 2000, <http://www.m-w.com/>.
- [19] M.Fowler, “*Analysis Patterns: Reusable Object Models*”, Addison-Wesley, 1997,
<http://www2.awl.com/cseng/titles/0-201-89542-0/apsupp/index.htm>.
- [20] R.Motschnig-Pitrik and J.Kaasboll, “*Part-Whole Relationship Categories and Their Application in Object-Oriented Analysis*” in IEEE Transactions on Knowledge and Data Engineering vol. 11, pp. 779-797, 1999.
- [21] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, “*Design Patterns - Elements of Reusable Object-Oriented Software*”, Addison-Wesley, 1995.
- [22] Utility Communications Specification Working Group, “*IEC 60870-6-503, Telecontrol Equipment and Systems - Part 6: Telecontrol Protocols Compatible with ISO Standards and ITU-T Recommendations - Section 503: TASE.2 Services and Protocols*”, August, 1996, <ftp://ftp.sisconet.com/epri/iccp/iccp503.doc>.
- [23] Utility Communications Specification Working Group, “*IEC 60870-6-802, Telecontrol Equipment and Systems - Part 6: Telecontrol Protocols Compatible with ISO Standards and ITU-T Recommendations - Section 802: TASE.2 Object Models*”, August, 1996, <ftp://ftp.sisconet.com/epri/iccp/iccp802.doc>.
- [24] M. Mansouri-Samani and M.Sloman, “*Monitoring Distributed Systems*” in Network and Distributed Systems Management, Addison-Wesley, 1994.
- [25] D.F.D'Souza and A.C.Wills, “*Objects, Components, and Frameworks with UML - The Catalysis Approach*”, Addison-Wesley, 1999.