Annals of DAAAM for 2010 & Proceedings of the 21st International DAAAM Symposium, Volume 21, No. 1, ISSN 1726-9679
ISBN 978-3-901509-73-5, Editor B. Katalinic, Published by DAAAM International, Vienna, Austria, EU, 2010
Make Harmony Between Technology and Nature, and Your Mind will Fly Free as a Bird
DAAAM Symposium

SMART DATA ACQUISITION SOFTWARE USED IN INDUSTRIAL MONITORING SYSTEMS

GRIGORESCU, C[ostin] - M[arius]; MORARU, S[orin] - A[urel] & BADEA, M[ilian]

Abstract: *In general a monitoring software system has three main components: data acquisition, data processing and storage, and data display. This paper presents the architecture of a data acquisition component designed to communicate with devices using the serial, ethernet and other interfaces and various communication protocols. This component reads from an input structure the list of devices that are monitored with all the information needed for establishing the communication with them: the interface and its parameters, the protocol, the addresses of the monitored parameters.*
Key words: *data acquisition, monitoring software system*

1. INTRODUCTION

The most common architecture of a monitoring system is composed of the following items: a data acquisition component, a database (stores information about the monitored equipment, the measured values of the monitored parameters and other information needed to run the monitoring system) and a data display component (has the role of displaying the information stored in the database, allows users to view the state of the monitoring system and also raises alarms if specific events have occured) (Jestratjew, 2009; Kirubashankar et al., 2009). A representation of thir architecture can be seen in Fig. 1.

If we look at a monitoring system from the point of view of the three-tier architecture we can say that the presentation tier is represented by the data display component, the business tier is represented by the data acquisition component and the database represents the data tier of the architecture (Wang et al., 2007).

The goal of our research is to develop a monitoring system for energy consumptions in an offices and laboratories buildings campus.

Because the monitored data is used other research programs related to energy efficiency and considering the fact that the monitored parameters can change a lot during the lifetime of the system it is mandatory to develop a data acquisition component that can be adapted to the other researchers' needs without a lot of work and time effort.

This paper proposes a concept of a flexible data acquisition component (business tier) capable of communicating on multiple interface types, with a wide range of devices using user defined communication protocols.

The following sections of the paper present the need and the structure of this application concept.

2. THE NEED OF THIS APPLICATION

The application concept proposed by this paper offers its users a very flexible method of retreiving data from the desired monitored devices.

Most data acquisition systems are closely related to a specific type of equipment or require drivers and other interfaces to be able to communicate with a wider range of equipment types. Most of these drivers and interfaces are provided by the manufacturer of the equiment or third party companies at an extra cost.

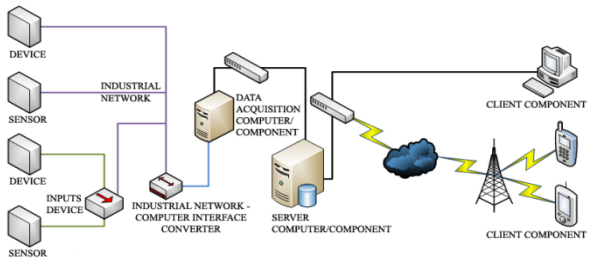

Fig. 1. Monitoring system architecture

The proposed concept solves some of these problems by offering its users the possibility of defining:

- The communication settings (interface(s) with settings).
- The list of monitored devices.
- The communication protocol(s) (the structure of each response and request telegram).
- The list of monitored parameters (name, address and size in bytes on the device, measuring unit, needed transformation, etc.) for each device.

All the information requested by the application from the users is usually available in the datasheets and manuals provided by the device's manufacturers.

The use of an application like this one gives the monitoring system flexibility in definig the number and type of monitored devices, allowing the users to add or remove devices with minimal effort and time loss for the functionality of the entire system.

3. GETTING THE INFORMATION FROM THE USER (CONFIGURING THE APPLICATION)

For receiving the information from the user, the application provides a graphical user interface (GUI) developed as a stand alone application using the Java programming language (Loy et al. 2002).

This GUI offers a wizard like method of inserting the information, using several steps.

After the last step, the application will save the information from the user in a XML file which is used later during the data acquisition process.

The wizard contains the following steps:

- Step 1: Definition of the communication interfaces. During this step the user will add the interfaces (serial, ethernet, etc.) and their settings (name, baud rate, parity, data bits, stop bits, etc.).
- Step 2: Definition of the monitored devices. For each device the following information is needed: name, interface on which it is connected, information to identify it on the interface (unique identifier on the network), communication protocol information (structure of the possible requests and responses issued by the application and the device), monitored parameters information (name, address, size in

bytes, measuring unit, telegram structures used for request and response, etc.).

- Step 3: Definition of the object encapsulation. For each piece of equipment the user has to provide a structure and class name for the object that will contain the values of the monitored parameters and other device information. These objects are also known as beans or transport classes.
- Step 4: Definition of the output location. After the application has filled an object with monitored data, this object has to be sent to the server component of the monitoring system. This component can be a database, another application, a web service, etc. The information for connecting to this component must be provided by the user.
- Step 5: Saving the information. During this step the application converts the information received from the user into a format that can be loaded and used by the data acquisition component. The chosen format is XML (McLaughlin & Edelson, 2006). A small capture of an output XML file is presented in Fig. 2. Also with this XML file, the application creates and compiles all user defined transport classes.

The output of this wizard is represented by an XML file and a number of class files (compiled Java classes) that will be copied into a predefined location so the data acquisition application will know from where to load and use them.

4. STARTUP PROCEDURE AND DATA ACQUISITION PROCESS

The startup procedure of the data acquisition application is composed of the following steps:

- Step 1: Loading the configuration. In this step, the application loads the XML configuration file into predefined memory structures (arrays, hashtables, bean classes, etc.). These memory structures are filled with information about the interfaces, devices, communication protocols, output parameters and settings, etc.
- Step 2: Communication threads creation. For each interface, the application creates a separate execution thread that establishes and makes the communication with the interface's monitored devices.
- Step 3: Output connection. During this step the application uses the connection parameters provided by the user to connect to the output (a database, another application, a web service, etc.).

After the third step is completed, the acquisition process starts and the monitored parameters' values are sent towards the application's output.

At a specified time interval, the application refreshes the configuration. This consists in reading again the XML file and modifying the memory structures according with the changes, if any, found in the new configuration.

This means that interfaces, devices, communication protocols, etc. are added, removed or modified, and the communication is restarted where these changes occurred.

```
<?xml version="1.0"?>
<configuration>
  <interfaces>
    <interface>
      <id>1</id>
      <type>serial</type>
      <name>COM2</name>
      <baud>9600</baud>
      <parity>even</parity>
      <databits>8</databits>
      <stopbits>1</stopbits>
    </interface>
  </interfaces>
  <protocols>
    <protocol>
      <id>1</id>
      <name>Modbus RTU</name>
```

Fig. 2. XML configuration file capture

The communication process made in each thread can be broken into the following steps:

- Step 1: Interface connection. During this step the thread tries to open the connection to the interface using the parameters available in the XML file. If the connection fails, an alarm will be raised so the users can check the interface's settings and integrity. This step is repeated until the connection is established with the interface.
- Step 2: Data acquisition. This is the most important step because during it the system's monitoring function is made. The application loops through the interface's devices and sends read requests for each of the monitored parameters. If all requests are followed by their corresponding responses, then the next step will be made, otherwise an alarm will be raised so the system's users can verify the device for faults. Also the device that did not respond according with the communication protocol will be added to a list and the communication with it will be tried only after a certain amount of time has passed. This ensures a good data acquisition speed and smaller intervals between successive reads.
- Step 3: Output fill. The bytes received in the responses are transformed into parameter and are added to the corresponding bean class. The resulting object is sent to the output of the application. If the output is a database, then the application creates the corresponding sql queries and executes them (Kline, 2008).

5. CONCLUSION

This paper proposes a smart data acquisition application designed for use in monitoring systems that have a flexible architecture (number of devices, etc.). The advantages of this application are represented by the flexibility of configuration for interfaces, devices, protocols and output, and by the fact that it can be added to an existing monitoring system only by properly configuring the output and the parameters for each monitored device.

Future research plans include building conceptual architectures and structures fot the other two tiers of the monitoring system (data and presentation).

6. ACKNOWLEDGEMENTS

This paper is supported by the Sectorial Operational Programme Human Resources Development (SOP HRD), financed from the European Social Fund and by the Romanian Government under the contract number POSDRU/88/1.5/S/ 59321.

7. REFERENCES

Jestratjew, A. (2009). *Improving Availability of Industrial Monitoring Systems through Direct Database Access*, Communications in Computer and Information Science, ISBN 978-3-642-02671-3_40

Kirubashankar, R.; Krishnamurthy, K. & Indra, J. (2009). *Remote monitoring system for distributed control of industrial plant process*, Journal of Scientific & Industrial Research

Kline, K.E. (2008). *SQL in a nutshell*, O'Reilly Media, ISBN 978-0-596-51884-4

Loy, M.; Eckstein, R; Wood, D; Elliot, J.; Cole, B. (2002). *Java Swing*, O'Reilly Media, ISBN 978-0-596-00408-8

McLaughlin, B.; Edelson, J. (2006). *Java and XML*, O'Reilly Media, ISBN 978-0-596-10149-7

Wang, C.; Xu, L. & Peng, W. (2007). *Conceptual design of remote monitoring and fault diagnosis system*, Information Systems 32, Elsevier