OPTIMIZATION OF WAMDAS: A WEB SERVICE-BASED WIRELESS ALARM MONITORING AND DATA ACQUISITION SYSTEM FOR PHARMACEUTICAL PLANTS

By

Edith Quispe Holgado

A thesis submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE
In
COMPUTER ENGINEERING

UNIVERSITY OF PUERTO RICO
MAYAGÜEZ CAMPUS
2008

Approved by:

Manuel Rodríguez Martinez, Ph.D. President, Graduate Committee	Date
Pedro I. Rivera Vega, Ph.D. Member, Graduate Committee	Date
Bienvenido Vélez Rivera, Ph.D. Member, Graduate Committee	Date
William Hernández Rivera, PhD Representative of Graduate Studies	Date
Isidoro Couvertier, Ph.D. Chairperson of the Department	Date

ABSTRACT

This thesis presents a novel infrastructure which allows the Web Service-Based Wireless Alarm Monitoring and Data Acquisition System for Pharmaceutical Plants (WAMDAS) to be a more portable, reliable, secure, and robust system. This infrastructure provides a reliable alarm management mechanism and a functionality to have a pool of available web services to respond the requests on time. Likewise, a Windows-based application is provided to configure dynamically all system information and parameters. Furthermore, the handheld device application was improved with a strong security mechanism to access the system, a system usage control, and a redesign of the user interfaces.

The results of the experiments show that the system protocols were optimized when using Stored Procedures instead of in-Line SQL Statements and Message Queues instead of TCP/IP Sockets. And the usability evaluation shows a high level of effectiveness and user satisfaction of WAMDAS at the pharmaceutical plant.

RESUMEN

Esta tesis presenta una novedosa infraestructura que permite a WAMDAS (el sistema inalámbrico basado en Servicios Web para monitoreo de alarmas y adquisición de datos para plantas farmacéuticas) ser un sistema más portátil, confiable, seguro y robusto. Esta infraestructura proporciona un mecanismo confiable de administración de alarmas y una funcionalidad para mantener los Servicios Web disponibles para responder peticiones de servicio oportunamente. Así mismo, se proporciona una aplicación Windows para configurar dinámicamente toda la información y parámetros del sistema. Además, la aplicación para dispositivos portátil fue mejorada con un mecanismo sólido de seguridad para accesar al sistema, un control de uso del sistema, y un rediseño de las interfaces de usuario.

Los resultados de los experimentos muestran que los protocolos del sistema fueron optimizados usando Procedimientos Almacenados en vez de Sentencias SQL dentro del código y Colas de Mensajes en vez de Sockets TCP/IP. Y la evaluación de usabilidad muestra un alto nivel de eficacia y satisfacción del usuario de WAMDAS en la planta farmacéutica.

To my great family

ACKNOWLEDGEMENTS

I would like to thank God for his blessings and for giving me the strength to accomplish this work.

Also, I would like to thank my advisor and president of my graduate committee, Dr. Manuel Rodriguez, for his support and guidance throughout my master studies and the development of the present thesis. As well, thanks to INDUSOFT for giving me the support for the development of this research project.

Thanks to Edilberto García for his valuable contribution and help throughout the development of this work. Also, thanks to Pfizer, Vega Baja Inc. for the collaboration in the development and evaluation of WAMDAS.

In addition, I want to thank the other members of my graduate committee, Dr. Bienvenido Velez and Dr. Pedro Rivera for their comments and suggestions.

Besides, my complete gratitude to the Department of Electrical and Computer Engineering of the University of Puerto Rico at Mayagüez for giving me the opportunity and support to pursue my master studies.

Finally, I would like to thank to Dr. Ingrid Padilla and the U.S Department of Energy – Savanna River for giving me the support to complete this work.

Table of Contents

Table List

Figure List

Abbreviation List

ADAS	Alarm and Data Acquisition Service
ADO	ActiveX Data Objects
ANS	Alarm Notification Service
API	Application Programming Interface
DBMS	Database Management System
DBR	Data Broker
COM	Component Object Model
HMI	Human Machine Interface
IP	Internet Protocol
LAN	Local Area Network
MAC	Media Access Control
MSMQ	Microsoft Message Queue
PDA	Personal Digital Assistant
PLC	Programmed Logical Controller
RDBMS	Relational Database Management System
RO	Reverse Osmosis
RS	Registration Service
SCADA	Supervisory Control and Data Acquisition
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
TCP/IP	Transmission Control Protocol / Internet Protocol
TKIP	Temporal Key Integrity Protocol
UDDI	Universal Description, Discovery and Integration of Web Services
URL	Uniform Resource Locator
Wi-Fi	Wireless Fidelity
WPA	Wi-Fi Protected Access
WSDL	Web Services Description Language
XML	Extensible Markup Language

1 INTRODUCTION

1.1 Overview

Highly regulated manufacturing plants require up-to-date equipment status information and operational readings, all of which are used to monitor the entire production process. This task is often achieved by connecting Programmable Logic Controllers (PLCs) to the equipment instrumentation. The data gathered from the PLCs are then extracted and visualized using Supervisory Control and Data Acquisition (SCADA) systems and Human-Machine Interface (HMI) systems (see Figure 1.1).

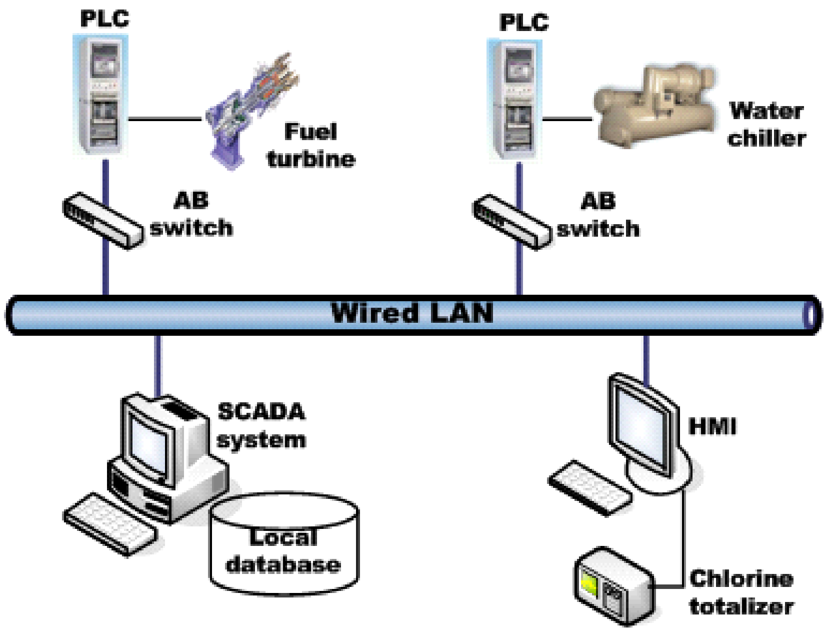


Figure 1.1 Pharmaceutical Plant Environment

Equipment in these environments must operate within normal operational set points and status levels. Any unmanaged deviation or equipment failure can result in larger malfunctioning events, loss of the active production in the plant, or even multi-million dollar fines from regulatory agencies. As a result, many manufacturers recruit operators and engineers whose main job is to watch over the equipment, as opposed to provide equipment maintenance services. Hence, these operators and engineers devote their time to watch equipment operational status over computer monitors, and to make periodic trips around the manufacturing plant, physically inspecting the instrumentation and logging data.

This operational scheme increases production costs and is prone to human errors. Clearly, an automated monitoring system is necessary to guarantee that the equipment status information and alarm messages are received in real-time to enable correcting actions. Furthermore, the solution should be wireless to simplify the deployment at the plant. Such system will enable operators to leave the secluded monitoring rooms and spend their time servicing equipment.

In response to this necessity, WAMDAS (Web Service-Based Wireless Alarm Monitoring and Data Acquisition System for Pharmaceutical Plants) has been developed [1] [2]. This system is designed and implemented with capabilities to monitor equipment status information and to manage equipment alarms. We define an *alarm* as an incident triggered by contingencies that occur while the equipment is in use under normal operational conditions; and this incident needs attention and must be corrected at a given time.

1.2 Problem Statement

During the testing and deployment of the previous version of WAMDAS in a real manufacturing environment, it was identified the need for WAMDAS to be dynamically configured to reduce the difficulty incurred in deploying the system. Since new equipment is frequently added and removed from the manufacturing plant, it became important to simplify the configuration of the alarms associated with each piece of equipment, and the delivery methods necessary to disseminate such alarms. Notice that even the personnel in charge of handling the alarms changes, thus the system must be able to cope with these changes on a daily basis. Clearly, timely delivery of alarms is critical to successful system operation, so it became necessary to have a mechanism to guarantee this process. Finally, we realized that it was necessary to seamlessly manage situations in which system web services were handling extreme workloads and system or network failures occurred.

Taking into account all the new requirements stated above, WAMDAS needed to be extended and optimized in order to completely satisfy these necessities of the dynamic and critical environments of pharmaceutical plants.

1.3 Objectives

The general objective of this thesis is to build an infrastructure which allows WAMDAS to be a more portable, reliable, and robust wireless web service-based middleware system.

This infrastructure will provide an alarm management mechanism that guarantees the alarm delivery according with a well-defined alarm configuration schema. Also it will provide a functionality to have a pool of available web services to respond to the requests on time. Likewise, this new infrastructure will provide a functionality to configure dynamically all system information and parameters. Moreover a security mechanism to access the system and a system usage control will be provided as well.

The specific objectives of this thesis which must be achieved to attain the general objective cited above are as follows:

1. Implement a mechanism to define a configuration schema for the alarms triggered by the equipment that is in use under normal operational conditions. This mechanism must consider the severity of each alarm and the set of users who will receive them.

2. Implement an alarm management mechanism that does not allow alarms to get lost, and guarantees the delivery of all the alarms to the users authorized to receive them.

3. Implement a dynamic configuration of all system Web Services' IP/URL to avoid recompiling their source code every time they are moved to other servers. Also implement an easy way to add new instances of these web services to the system when it is required.

4. Implement a functionality to provide all system Web Services the capacity to handle extreme workloads and be fault tolerant and consequently to allow the normal operation of the system under these circumstances.

5. Extend the functionalities of the system's handheld device module, so that it provides a security mechanism to access the system and a system usage control.
6. Provide to the system the necessary capacities for a smooth deployment task in the pharmaceutical plant environments.
7. Conduct a series of experiments to test the WAMDAS system in terms of performance and fault tolerance.

1.4 Contributions

In this thesis, we present a novel infrastructure which allows WAMDAS to be a more portable, reliable, secure, and robust wireless web service-based middleware system (see Figure 1.2). This infrastructure provides a reliable alarm management mechanism to guarantee that alarms are sent to and received by the handheld devices used by operators. Moreover, this alarm delivery process obeys a configuration scheme for alarms, which defines the mapping between the Real Alarms (alarms triggered by equipment) and the WAMDAS Alarms (alarms defined in the system). Likewise our new approach guarantees that alarm acknowledgments are sent to and received by the alarm notification server. It also guarantees that acknowledgement results are sent to and received by the handheld devices.

WAMDAS now provides a new functionality to have a pool of available web services to respond to requests on time, even under unexpected circumstances such as extreme workloads and system or network failures.

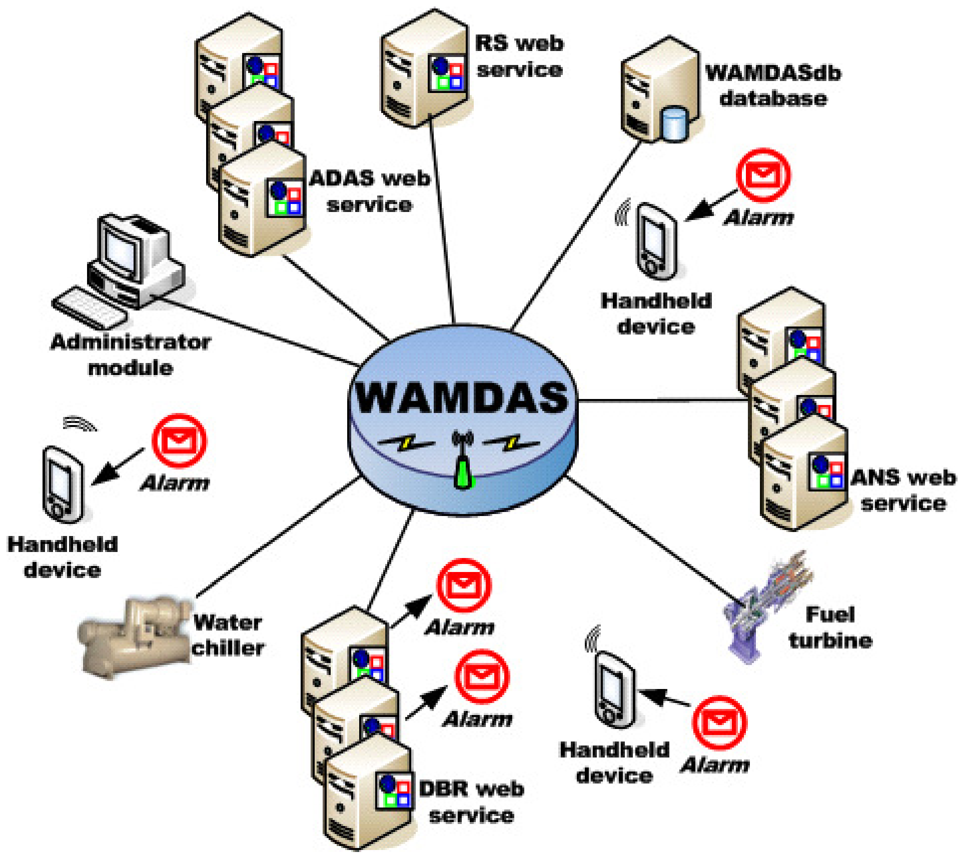


Figure 1.2 WAMDAS Solution

Likewise, a Windows-based application module is provided to permit system administrators to dynamically configure all system information and parameters, as well as define the configuration scheme for alarm management.

Furthermore, the handheld device application was improved with a strong security mechanism to access the system. WAMDAS can only run on authorized handheld devices, and only authorized users can log into the system. We also implemented a mechanism to control the system usage through operational shifts. Moreover, we redesigned the user interface in order to provide the users all the equipment status and alarm information in a very intuitive and feature-rich manner.

In addition, WAMDAS is a widely applicable system; it can be used in pharmaceutical plants and in any other type of manufacturing plants such as electric energy industry, metalworking, telecommunications, textile, construction, transportation, and others. In general, WAMDAS can be used in any environment where is necessary to monitor equipment status information and to manage equipment alarms.

1.5 Thesis Structure

This chapter has addressed the introduction of this thesis; the reminder of the document is organized as follows: Chapter 2 discusses the literature review that serves to develop this thesis. Chapter 3 presents the WAMDAS architecture and the Status and Alarms Services Protocols. Chapter 4 presents a detailed description of WAMDAS implementation. Chapter 5 presents the experiments and results of the WAMDAS evaluation. Finally, Chapter 6 presents the conclusions and future work related to WAMDAS.

2 LITERATURE REVIEW

This chapter provides a literature review of the previous work of this thesis, and the theoretical background of some related areas relevant to this research.

2.1 Previous Work

The previous work of this thesis is WAMDAS: A web service-based wireless alarm monitoring and data acquisition system for pharmaceutical plants [1] [2], which was developed as a master thesis at the University of Puerto Rico. WAMDAS is a prototype of a wireless web service-based middleware system, which monitors status information from the remote equipment (e.g., Turbines, Reverse Osmosis Systems and Water Chillers) deployed in a pharmaceutical plant. The information collected from this equipment includes equipment operational mode, component temperature, chlorine concentration, header pressure, and others. Also, WAMDAS processes critical alarms triggered by contingencies that occur while the equipment is in use under normal operational conditions.

2.1.1 Previous System Architecture

WAMDAS has a decentralized architecture (see Figure 2.1). It has four types of server components implemented as XML Web Services. The purpose of each web service is as follows:

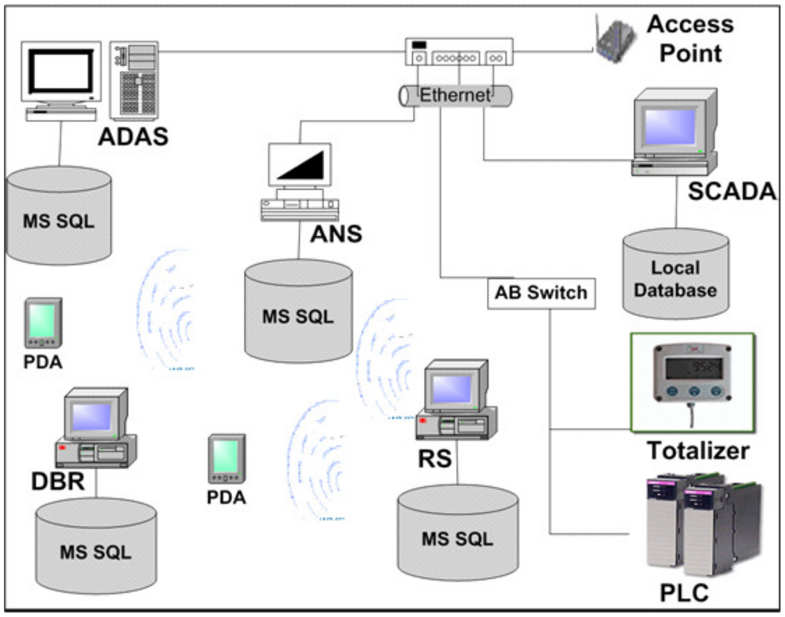


Figure 2.1 Previous WAMDAS Architecture

1. **Data Broker (DBR)**. – This web service was implemented to receive query requests about equipment status information from the PDA clients. Also, it receives the alarm messages from the ANS. The DBR processes an alarm message by searching in its local database for online PDA clients. After that, it proceeds to send the alarm message to all online PDA clients.
2. **Registration Service (RS)**. – This web service provides up-to-date status of any member of WAMDAS.
3. **Alarm and Data Acquisition Service (ADAS)**. – This web service monitors any possible alarm information from all system components. When an alarm message is generated, the ADAS broadcasts this message to the ANS for forwarding and acknowledgement actions.

4. **Alarm Notification Service (ANS)**. – This web service is responsible for forwarding the alarm messages and for processing the alarm acknowledgements.

Each web service has a local database (MS SQL Server) on which it stores the necessary information to accomplish its tasks.

The client side of the WAMDAS is a PDA client application, which helps the operators to request equipment status information and also to receive the alarm notification messages and acknowledge them.

2.1.2 Previous System Protocols

WAMDAS has two protocols, one to get status information from the equipment and one to process the alarm information. The description of the protocols is as follows:

1. **Status Services Protocol**. – This protocol works as follows (see Figure 2.2): The PDA client issues a query that the DBR will handle. The DBR will send a message to the RS to find the URL for each of the available data sources to resolve the query. Then the DBR will be responsible for sending the query to the ADAS running on the target data source site(s). The ADAS retrieves the information from the database where that data source information is stored through in-Line SQL statements. Finally the DBR collects the query results and forward these to the PDA client.

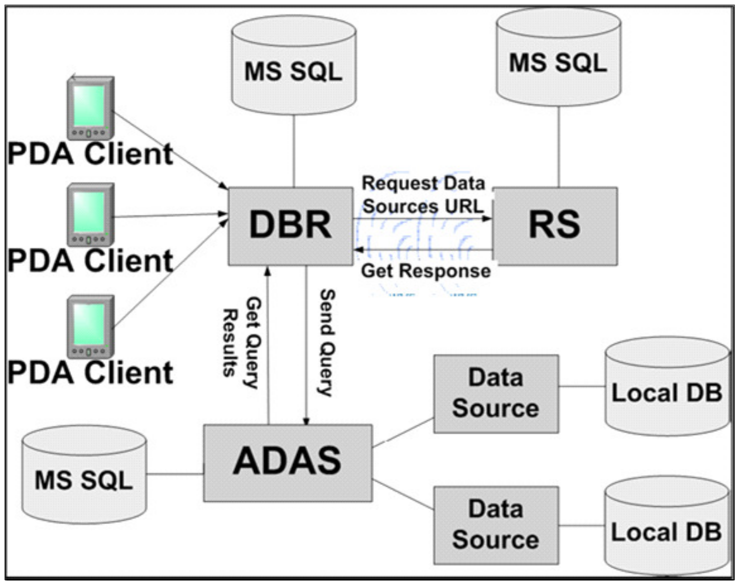


Figure 2.2 Previous Status Services Protocol

2. **Alarm Services Protocol**. – This protocol works as follows (see Figure 2.3): After the ADAS receives an alarm message from any equipment of a remote data source; it will broadcast this message to the ANS. The ANS will communicate with the RS to find out the URLs of the DBR's clients that are interested in the alarm. Then, the ANS will forward the alarm information to all DBR's client applications that are connected, and will continue to do so until an acknowledgment message is received back from one or more PDA clients. The DBR sends the alarm notification message to the PDA clients through client TCP/IP Sockets.

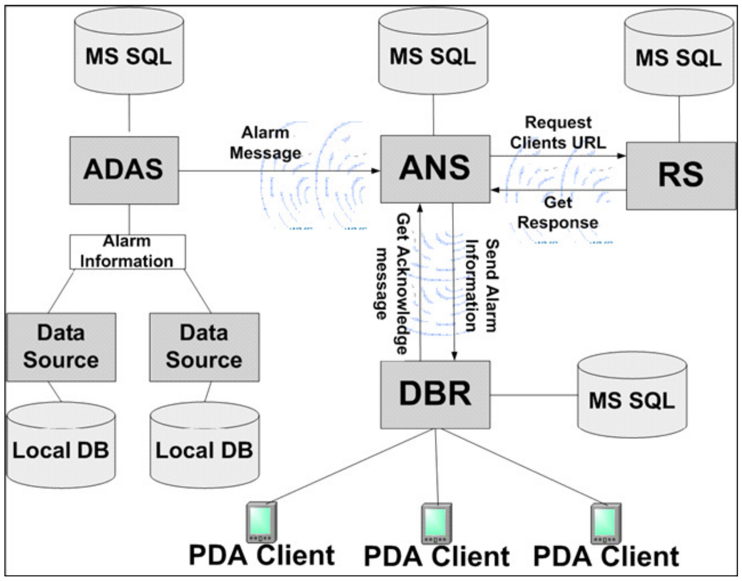


Figure 2.3 Previous Alarm Services Protocol

2.2 Theoretical Background

This research is related to the areas of Web Services, Relational Databases, and Messages Queues. Each of them will be described briefly in the following subsections.

2.2.1 Web Services

A web service is an application logic that is accessible using Internet standards. There have been many technologies for exposing application logic, but usually these were based on difficult to implement and often proprietary protocols. Internet standards, such as XML, have simplified the building of a distributed application [3].

Web services are another distributed technology; they allow us to create client/server applications. A web service exposes an interface to invoke a particular activity on behalf of the client; a client can access the web service through the use of Internet standards [4]. Information which is available through a web service will always be accessed by software, never directly by a human.

In the web services architecture, many kinds of distributed systems can be implemented. Examples include synchronous and asynchronous messaging systems, distributed computational clusters, mobile networked systems, grid systems, and peer-to-peer environments. The broad spectrum of requirements in program-to-program interactions forces the web services protocol stack to be much more general-purpose than the first web protocols [5]. Web services have certain advantages over other technologies; they are platform independent and language independent, because they use standard XML languages.

The baseline specifications for web services are SOAP, WSDL, and UDDI, which are described briefly as follows [3] [4] :

1. **SOAP (Simple Object Access Protocol)**. – It is a protocol that enables machine to machine communication over computer networks in very heterogeneous environments with different platforms and operating systems [6]. SOAP describes very simple XML-based packaging for exchanging messages between the web service and the client. In the WAMDAS context, we use SOAP transported atop the HTTP protocol.

2. **WSDL (Web Services Description Language). –** It is a XML-based service description of how to communicate and make a request to a web service [7]. It relies heavily on XSD schemas, and it is used to describe everything a SOAP service needs. This includes the operations, the schema for each message in an operation, the SOAP action headers and the URL end point of the service. An operation is a related set of messages (e.g., the message that a client sends to a server and the reply message that the client receives).
3. **UDDI (Universal Description, Discovery and Integration). –** It is an industry-standard centralized directory service that can be used to advertise and locate Web services. UDDI is designed to be interrogated by SOAP messages and to provide access to WSDL documents describing the protocol bindings and message formats required to interact with the web services listed in its directory [8]. The users can search for the web services using different criteria, including company name, category, and type of web service. UDDI is not just for the Internet, a UDDI server can be deployed within an enterprise. For companies that use different technologies and platforms internally UDDI can be a cost-effective and deployable solution.

In the WAMDAS context we do not implement a UDDI server; we use a discovery mechanism that consists of storing in and retrieving from the system database the URL of the WSDL document of the system web services.

2.2.2 Relational Databases

A Database is a structured collection of data stored in an organized way. A Database Management System (DBMS) is the software designed to assist in maintaining and utilizing a database. Another alternative, besides using a DBMS is to store the data in files and write application-specific code to manage it [9]. There are many types of DBMS but the dominant type is the Relational DBMS (RDBMS) that support relational databases.

The relational model was introduced by Edgar F. Codd in 1970, and nowadays, it is the most commonly used model in database applications [10]. This model is based in the essential assumption that all data are represented as mathematical relations. A database is a collection of one or more relations, and each relation is a table with rows and columns.

A relational database is a database that conforms to the relational model. It refers to a database's data and schema. The database schema is the structure of how that data is set. The relational database is the primary data model for commercial data-processing applications because of its simplicity compared to earlier data models such as the network model or the hierarchical model.

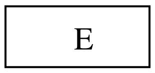
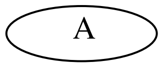
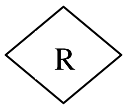
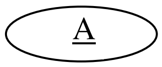
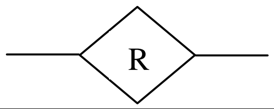
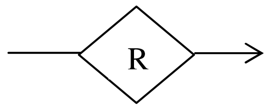
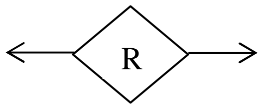
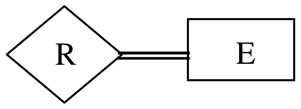
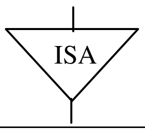
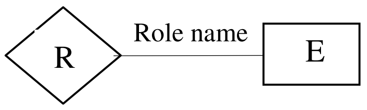
Relational database systems use a collection of tables to represent both data and the relationships among those data. Most commercial relational database systems employ the Structured Query Language (SQL); this language includes a Data Manipulation Language (DML) and a Data Definition Language (DDL). The DML enables users to retrieve, insert, delete, and update the information stored in the database. And the DDL allows defining tables, integrity constraints, assertions, authorizations, and others [11].

Middleware is defined as the connectivity software layer between the operating system and the applications on each site of a distributed system. Middleware is the technology that enables the enterprise application integration. The access mechanisms to RDBMSs used by middleware applications are JDBC [12], ODBC [13], and other vendor-specific APIs.

The Entity - Relation (E-R) data model allows us to describe the data involved in a real world that consists of a collection of basic objects, called entities, and the relationships among these objects. It is widely used to develop an initial database design [9]. The entities are described by a set of attributes; one of these attributes is called Primary Key, which is used to uniquely identify the entities. An E-R diagram consists of the following major symbols presented in Table 2.1 [11].

In a database design process, the initial phase is to fully characterize the data needs of the prospective database users. Then these requirements are translated into a conceptual schema (logical structure) for which the E-R model is typically used. The process continues moving from an abstract model to the implementation of the database with two final design phases: the logical design phase, which consists on mapping the conceptual schema into a relation schema; finally the physical design phase in which the physical features of the database are specified such as file organization and internal storage structures [11].

Table 2.1 Symbols used in E-R Diagrams

Symboll	Meaning
	Entity set
	Attribute
	Relationship set
	Primary key
	Many-to-many relationship
	Many-to-one relationship
	One-to-one relationship
	Total participation of entity set in relationship
	Specialization or generalization
	Role indicator

The necessity to protect the database is increasing; no matter what degree of security is put in place, sensitive data in database are still vulnerable to attack. To avoid the risk database encryption is an alternative; but encrypting all of database degrades the performance of the database system significantly. In [14] a solution is presented to encrypt a database that provides maximum security, while limiting the additional time cost of encryption and decryption. The scheme considers two levels (L1 and L2) for public data. All users have access to their own personal private data. And in addition, in L1 users have access only to unclassified (non-sensitive) data, while in L2 they have access to both unclassified and classified (sensitive) data. The database objects are classified into public (unclassified and classified) and private objects. Classification for public objects is done at attribute level while that for private objects is done at data element level. The proposed scheme makes use of three sets of encryption keys: for classified data, private data, and controller's master key. And the data elements are encrypted/decrypted using Data Encryption Standard (DES) technique.

2.2.3 Messages Queues

Microsoft Message Queue (MSMQ) is a Windows operating system service, and it is used as a transport mechanism. MSMQ is Microsoft's answer to the messaging middleware market, for developing scalable, reliable and asynchronous messaging based applications. MSMQ ensures guaranteed message delivery, efficient routing, security, fault-tolerance and recovery against communication losses when the network fails [15].

MSMQ supports the following scenarios [16]:

1. **Loosely coupled application**. – The sending application can send messages to a MSMQ without needing to know whether the receiving application is available to process the message. The rate of sending messages to a MSMQ is not dependent on the rate of processing the messages.
2. **Failure isolation**. – Applications sending or receiving messages to a queue can fail without affecting each other. That is, if the receiving application fails, the sending application can continue sending messages to the receiver's MSMQ. When the receiver is up again, it processes the received messages.
3. **Load leveling**. – The sending application could overwhelm the receiving application with messages. MSMQ can manage unequal message production and consumption rates so that a receiver is not overwhelmed.
4. **Disconnected operations**. – Operations of sending, receiving or processing can be disconnected when the communication is over high-latency or limited-availability networks (e.g. mobile devices). MSMQ allows these operations to continue even when the endpoints are unavailable.

There are two types of MSMQ: system queues and application queues. The system queues are used for administrative activities. And the application queues are created and used by applications for the purpose of communicating between them.

The application queues fall into two categories: public and private. The distinction primarily is their visibility. The public queues are registered in Active Directory so they can be located by any MSMQ application that can search an Active Directory. Therefore, the public queues work only on domain environments. On the other hand, the private queues are registered only on the local computer, so they are not explicitly visible throughout the network. Therefore private queues work on workgroup environments. Private queues can be accessed only if a MSMQ application has specific prior knowledge that the queue exists, or if a sending application provides the receiving application with the private queue name in a message property, as a queue to which responses should be sent.

Other distinction between queues is: local and remote. A local queue is a queue created in the local machine and a remote queue is a queue created in other machine in the network. In particular, MSMQ for Windows CE, the Windows operating system for handheld devices, only supports the creation of local private queues [15].

In the WAMDAS context, we work with application queues that are local and private, for both the server side and the client side of the system.

3 WAMDAS ARCHITECTURE

3.1 System Architecture

WAMDAS is based on a decentralized and distributed architecture (see Figure 3.1). This figure shows a deployment of WAMDAS on a hybrid network containing an Ethernet backbone and a wireless sub network.

The server side of the system is implemented with four XML Web services; each of them has the required web methods to provide WAMDAS an appropriate functionality and scalability. The first web service is the Data Broker (DBR), which works on behalf of the handheld device users to query equipment status information, to receive and acknowledge alarm notifications, and to query for historic alarm acknowledgment information.

Next is the Registration Service (RS), which manages and coordinates the communication between all other web services. The third web service is the Alarm and Data Acquisition Service (ADAS), which begins the alarms processing stage, and communicates with the databases where the equipment status information is stored.

The last web service is the Alarm Notification Service (ANS), which acts as a mediator for the alarm processing and it is in charge of the alarm acknowledgment process. For example when an alarm is triggered, it is first received by the ADAS, which requests to the RS to find any online ANS. The ADAS then requests to the ANS, found in the previous step, to forward the alarm message. The ANS in turn requests to the RS to find any online DBR. Once found, the ANS requests to that DBR to process the alarm message. The DBR sends the alarm message to all the handheld device users authorized to receive the alarm message.

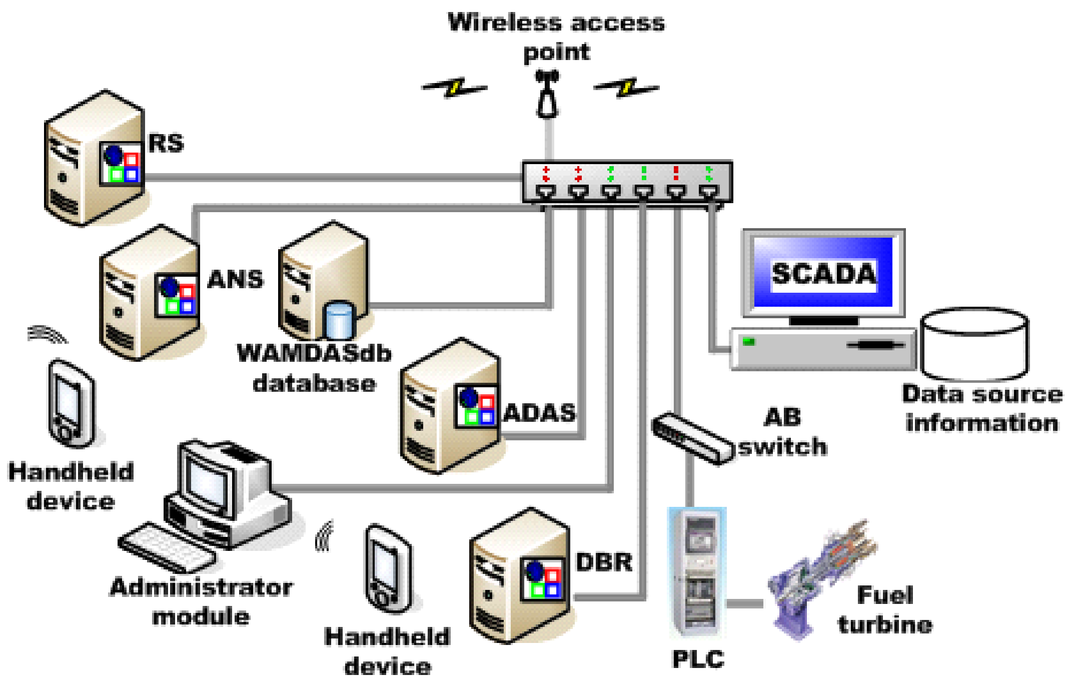


Figure 3.1 WAMDAS Architecture

In the previous version of WAMDAS, each web service had a different database to store the information needed to accomplish their tasks. In this new version we have a single, integrated, and unified database named WAMDASdb that stores all system information related to data sources, components, WAMDAS alarms, real alarms definitions, alarm events, received alarms, handled alarms, users, positions, operational shifts, web services, handheld devices, and parameters.

There can be multiple instances of each web service installed on different web servers in order to guarantee a pool of available web services to respond to the requests on time, even under unexpected circumstances such as extreme workloads and system or network failures. If a web service ‘A’ wants to communicate with another web service ‘B’, then ‘A‘ requests to the RS the Uniform Resource Locator (URL) of the first instance of the target web service that is online and can respond to requests.

The client side of WAMDAS consists of two modules. The first one is a device application that runs on handheld devices like Personal Digital Assistants (PDAs). This application provides the functionalities to query the equipment status information, to receive and acknowledge the alarm notifications, and to query for historic alarm acknowledgement information. The second module is the administrator module, which is a Windows-based application that runs on workstations. This application permits system administrators to configure all the system information and parameters needed to make WAMDAS work properly in a dynamic, scalable and flexible way.

We have implemented our new version of WAMDAS using Microsoft Visual Studio 2005 (.NET Framework 2.0 and .NET Compact Framework 2.0), the C# programming language, and Microsoft SQL Server 2000 / 2005 as the database management system.

3.2 System Protocols

WAMDAS has two protocols, one for processing equipment status information requests and another for managing alarms triggered by contingencies that occur while the equipment is in use under normal operational conditions. Their description is as follows:

3.2.1 Status Services Protocol

This protocol is used to query equipment status information. The protocol is described in terms of steps to denote the sequence of events in the process (see Figure 3.2). These steps match with the figure's enumeration, as we shall discuss next:

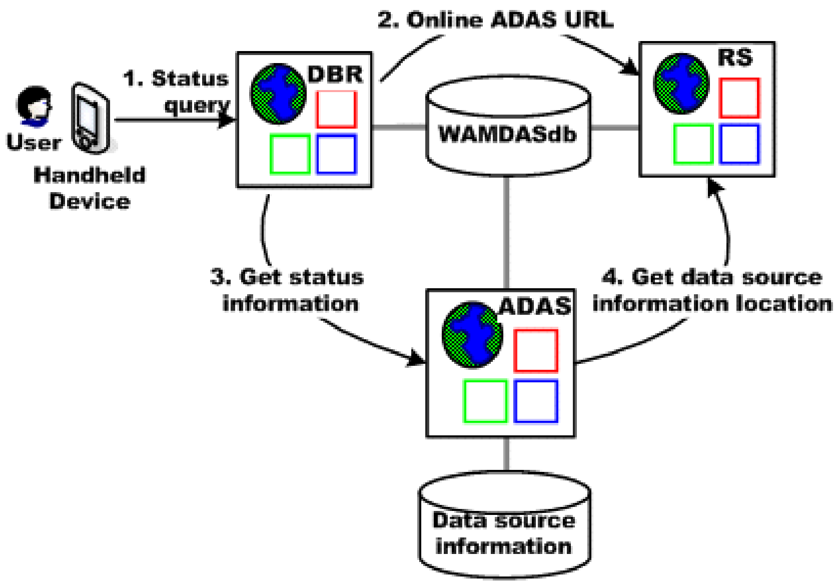


Figure 3.2 Status Services Protocol

- In Step 1, a user on the handheld device asks for equipment status information, and a request is sent to the DBR.
- In Step 2, the DBR requests to the RS to find any online ADAS.
- In Step 3, the DBR requests to the found ADAS to get the status information. The ADAS verifies if the target data source is currently registered in the system.
- Finally, in Step 4, the ADAS requests to the RS to get the data source information location (server name and database name). With this information the ADAS calls a stored procedure at the corresponding database to retrieve the status information. Then the ADAS returns to the DBR the status query result. The DBR, in turn, returns this result to the handheld device to be displayed to the user.

We optimized the status services protocol moving from In-Line SQL statements to Stored Procedures to exploit their great benefits in performance, security, readability, reusability and flexibility [17].

3.2.2 Alarm Services Protocol

This protocol is used to notify the alarms to the users. The protocol is described in terms of steps to denote the sequence of events in the process (see Figure 3.3). These steps match with the figure's enumeration, as we shall discuss next:

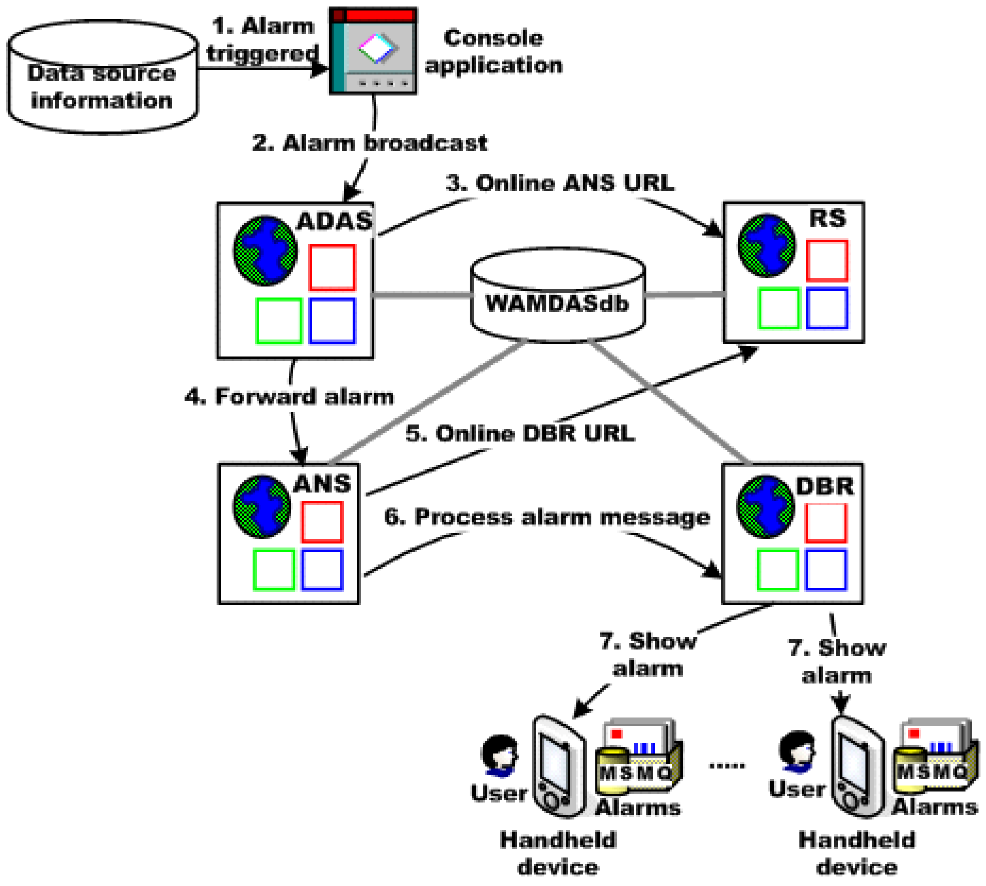


Figure 3.3 Alarm Services Protocol

- In Step 1, when an alarm is triggered by a piece of equipment, the console application (implemented as an interface between MS SQL Server 2000 and the system web services) is called to begin the process.
- In Step 2, the console application requests to the ADAS to broadcast the alarm. The ADAS first verifies if there is a WAMDAS alarm that maps to the received real alarm. If so, the ADAS registers a new alarm event in the system.
- In Step 3, the ADAS requests to the RS to find any online ANS.

- During Step 4, the ADAS requests to the found ANS to forward the alarm to the appropriate receivers.
- In Step 5, the ANS requests to the RS to find any online DBR.
- In Step 6, the ANS requests to the found DBR to process the alarm message, and the DBR then retrieves the IP addresses of the handheld devices on which users authorized to receive the alarm are currently logged into the system.
- Finally, in Step 7, the DBR sends the alarm to those handheld devices' message queues assigned to receive alarm messages. If the handheld devices are online they receive the alarm message immediately, if not, they will receive the alarm message when they recover their online status.

We optimized the alarm services protocol by changing the transport mechanism to deliver alarm messages from the DBR to the handheld devices. We switched from connection-oriented TCP/IP sockets in favor of connection-less message queues implemented via Microsoft Message Queues (MSMQ). With MSMQ, we get fault-tolerance and recovery against communication loss when the network fails, the handheld device is out of wireless network range, or the battery of the handheld device is discharged. Message queuing ensures guaranteed message delivery, efficient routing, security, fast and reliable asynchronous communication [15]. The types of message queues that were used in the implementation are application queues that are local and private

The Alarm Acknowledgement process is used to acknowledge the alarms that were received; it was also optimized using MSMQ. The way this process works is described also in terms of steps to denote its sequence of events (see Figure 3.4). These steps match with the figure's enumeration, as we shall discuss next:

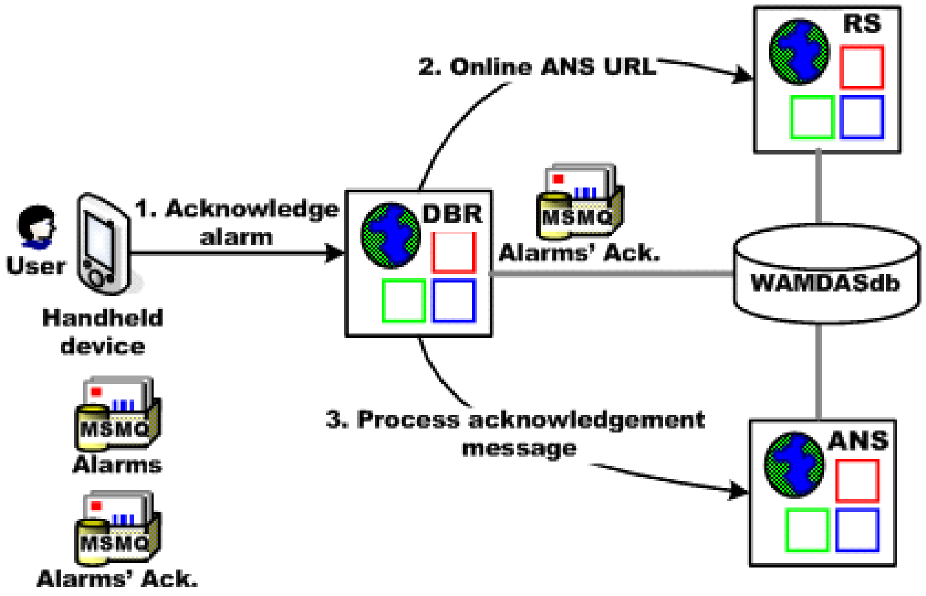


Figure 3.4 Alarm Acknowledgement Process

- In Step 1, a user acknowledges an alarm notification on the handheld device, and an acknowledgement message is sent to the MSMQ of a web server where a DBR is hosted.
- In Step 2, this DBR requests to the RS to find any online ANS.
- Finally, in Step 3, the DBR requests to the found ANS to process the acknowledgement message. The ANS retrieves information about the users who have already acknowledged the alarm, and then it verifies if the alarm has already been acknowledged from that handheld device. The ANS also verifies the number of acknowledgements that have already been received to determine if the alarm was

already acknowledged by the predefined number of users. Then, the ANS registers the acknowledgment message in the system. Finally, the DBR builds the acknowledgement result message with the messages notified by ANS. The DBR sends this result message to the handheld device's MSMQ assigned for alarm acknowledgement results. If the handheld device is online it receives the message immediately, if not, it will receive the message when it recovers its online status.

4 WAMDAS IMPLEMENTATION

This chapter presents a detailed description of the WAMDAS system implementation, both the server side and the client side of the system. First we describe the system web services implementation; next we describe the system protocols implementation, then WAMDAS's relational database is described, after that we describe the administrator module implementation, and finally, we describe the handheld device module implementation.

4.1 Web Services Implementation

We improved and extended the implementation of the previous version of system web services. For each web service, we describe its purpose and its relevant web methods implemented to accomplish its tasks.

4.1.1 The Data Broker Web Service (DBR)

This web service works on behalf of the users of the handheld device module to query equipment status information, to receive and acknowledge alarm notifications, and to query historic alarm acknowledgement information. Now, we describe the relevant web methods of the DBR as follows:

1. **ValidateOperatorDevice** (MAC_Address, IP_Address)

   This web method is called by the handheld device module to validate if a handheld device is correctly registered in the system to run WAMDAS. This validation is carried out through the handheld device's MAC and IP addresses, which are received by the web method as parameters.

2. **UserLogin** (UserName, Password)

   This web method is called by the handheld device module to validate if a user is correctly registered in the system. This validation is carried out through the user name and password, which are received by the web method as parameters.

3. **GetUsersInfo** ( )

   This web method is called to retrieve the information of all the users registered in the system. This information consists of the first name, last name, user name, email, telephone, and status of each user.

4. **RegisterShift** (UserID, HandheldDeviceID, StartDateTime)

   This web method is called by the handheld device module to register in the system a new operational shift. The parameters that the web method receives are the user ID, the handheld device ID and the start date time of the shift.

5. **GetClientRequest** (DataSourceID, ComponentID, QueryType, Date)

   This web method is called by the handheld device module to serve the equipment status information queries requested by users. The status information that this web method will get corresponds to the data source and component with IDs equal to the received parameters. The query type parameter specifies if the requested information is about Generator, Engine or Reverse Osmosis (RO) system. And the method will retrieve all the status information that was registered during the date received as parameter, ordered by time.

6. **ProcessAlarmMessage** (AlarmEventID)

   This web method processes the alarm messages forwarded by the ANS web service. First, the web method retrieves the alarm information corresponding to the alarm event with ID equal to the received parameter. Then it retrieves the information of the handheld devices to which the alarm message must be sent. Finally, the web method sends the alarm message to the handheld devices' MSMQ.

7. **RegisterReceivedAlarm** (HandheldDeviceID, AlarmEventID, DateTime)

   This web method is called by the handheld device module to register in the system the alarms that were received by handheld devices. The parameters that the web method receives are the handheld device ID, the alarm event ID, and the date and time when the handheld device received the alarm.

8. **AlarmAcknowledgementCompleted** (AlarmEventID)

   This web method is called by the handheld device module to verify if the required number of acknowledgements for an alarm event has already been completed, in order to conclude that the alarm event with ID equal to the received parameter has been fully attended.

9. **ClientAcknowledgement** (HandheldDeviceID, Date)

   This web method is called by the handheld device module to retrieve historic information of the alarms which were acknowledged on the handheld device with ID equal to the received parameter during the date equal to the received parameter.

10. **CloseShift** (ShiftID, DateTime, UserID, HandheldDeviceID)

    This web method is called by the handheld device module to close in the system an operational shift. The parameters that this method receives are the shift ID, the date and time when the shift is ending, the user ID, and the handheld device ID.

11. **VerifyIfDBRServiceIsOnLine** ( )

    This web method is called by the RS to verify if the DBR is online and it can respond to requests.

12. **UpdateWebConfigFileRSURL** (RSURL)

    This web method is called by the administrator module to update the value of the application setting in the 'web.config' file corresponding to the URL of the RS web service. This update is made with the value received as parameter.

4.1.2 The Registration Web Service (RS)

This web service manages and coordinates the communication between all other web services. Now, we describe the relevant web methods of the RS as follows:

1. **GetAllDBRInfo** ( )

   This web method retrieves the information of all instances of the DBR web service that are registered in the system. This information consists of the name, description, URL, server name, and the status of each instance.

2. **GetAllRSInfo** ( )

   This web method retrieves the information of all instances of the RS web service that are registered in the system. This information consists of the name, description, URL, server name, and the status of each instance.

3. **GetAllADASInfo** ( )

   This web method retrieves the information of all instances of the ADAS web service that are registered in the system. This information consists of the name, description, URL, server name, and the status of each instance.

4. **GetAllANSInfo** ( )

   This web method retrieves the information of all instances of the ANS web service that are registered in the system. This information consists of the name, description, URL, server name, and the status of each instance.

5. **GetConfigurationParameterValue** (ConfigParamName)

This web method is called by the system web services and the Administrator Module to retrieve the value of the configuration parameter which name is equal to the received parameter.

6. **GetURLFirstOnlineDBRService** ( )

This web method is called by the system web services to get the URL of the first element in a list of DBR web services which is available to respond to requests.

7. **GetURLFirstOnlineADASService** ( )

This web method is called by the system web services to get the URL of the first element in a list of ADAS web services which is available to respond to requests.

8. **GetURLFirstOnlineANSService** ( )

This web method is called by the system web services to get the URL of the first element in a list of ANS web services which is available to respond requests.

9. **GetWAMDASDatabaseConnectionString** ( )

This web method is called by the system web services to get the connection string to the WAMDASdb database. The connection string is built with the Server Name, Database Name, User Name, and Password values retrieved from the Application Settings section of the 'web.config' file of the RS web service.

10. **GetDataSourceDatabaseConnectionString** (DataSourceID)

This web method is called by the ADAS to get the connection string to the database where the status information of the data source with ID equal to the received parameter is stored. The connection string is built with the Server Name, Database Name, User Name, and Password values retrieved from the WAMDASdb database, where these values are stored as parameters.

11. **VerifyIfRSServiceIsOnLine** ( )

This web method is called to verify if the RS web service is online and it can respond to requests.

12. **UpdateWebConfigFile** (AppSettingName, AppSettingValue)

This web method is called by the administrator module to update in the 'web.config' file the value of the application setting with name equal to the received parameter.

The application settings stored in the 'web.config' file are the database server, database name, user name, and password values that are used to build the connection string to the WAMDASdb database.

4.1.3 The Alarm and Data Acquisition Web Service (ADAS)

This web service begins the alarms processing stage and communicates with the databases where the equipment status information is stored. Now, we describe the relevant web methods of the ADAS as follows:

1. **AlarmBroadcast** (RealAlarmIdentifier, AlarmValue, AlarmDateTime)

   This web method is called by the console application to broadcast an alarm. First, it verifies if a mapping to a WAMDAS alarm was defined for the real alarm with identifier equal to the received parameter. If so, a new alarm event of the found WAMDAS alarm is registered in the system with the alarm value and date time received as parameters. Finally, the ANS web service is requested to forward this alarm.

2. **GetDataSourceInfo** (DataSourceName, ComponentIdentifier, QueryType, Date)

   This web method is called by the DBR to retrieve the equipment status information. First, it verifies if the data source with name equal to the received parameter is registered in the system. If so, the connection string parameters of the database where the equipment status information is stored are retrieved from the WAMDASdb database. Then the corresponding database stored procedure is called to retrieve the status information for the component with identifier and query type equal to the received parameters. The result status information corresponds to the status readings made during the date equal to the received parameter, and they are retrieved ordered by time.

3. **VerifyIfADASServiceIsOnLine** ( )

   This web method is called by the RS to verify if the ADAS is online and it can respond to requests.

4. **UpdateWebConfigFileRSURL** (RSURL)

This web method is called by the administrator module to update the value of the application setting in the 'web.config' file corresponding to the URL of the RS web service. This update is made with the value received as parameter.

4.1.4 The Alarm Notification Web Service (ANS)

This web service acts as a mediator for the alarm processing and it is in charge of the alarm acknowledgment process. Now, we describe the relevant web methods of the ANS as follows:

1. **ForwardAlarm** (AlarmEventID)

   This web method is called by the ADAS to forward to the DBR the alarm event with ID equal to the received parameter.

2. **AcknowledgementsNumberVerification** (AlarmEventID)

   This web method is called by the DBR and the ANS to process the acknowledgements number verification. It first retrieves the number of acknowledgements already made to the alarm event with ID equal to the received parameter. Then it compares this result value with the configured Acknowledgement Quantity Number corresponding to the severity of the instanced WAMDAS alarm.

The method returns a value of "false" if the alarm event requires more acknowledgements; otherwise it returns a value of "true".

3. **ProcessAcknowledgement** (AlarmEventID, HandheldDeviceID, DateTime)

   This web method is called by the DBR to process the acknowledgement. It first verifies if the alarm event with ID equal to the received parameter has already been acknowledged on the handheld device with ID equal to the received parameter. If not, the acknowledgements number verification is carried out, and finally the alarm acknowledgement is registered in the system.

4. **RetrieveUsersWhoHaveAlreadyDoneAkn** (AlarmEventID)

   This web method is called by the DBR to retrieve the information of the users who have already acknowledged the alarm event with ID equal to the received parameter.

5. **GetAlarmMessageInfo** (HandheldDeviceID， Date)

   This web method is called by the DBR to retrieve the historic information about the alarm acknowledgements made on the handheld device with ID equal to the received parameter and during the date equal to the received parameter.

6. **VerifyIfANSServiceIsOnLine** ( )

   This web method is called by the RS to verify if the ANS web service is online and it can respond to requests.

7. **UpdateWebConfigFileRSURL** (RSURL)

This web method is called by the administrator module to update the value of the application setting in the 'web.config' file corresponding to the URL of the RS web service. This update is made with the value received as parameter.

4.2 System Protocols Implementation

In the implementation of the system protocols, we considered the following three layers:

1. **The client application layer**. – This layer comprises the handheld device module and the console application (which is an interface between MS SQL Server and the ADAS web service).

2. **The web services layer**. – This layer comprises the four system web services: the DBR, the RS, the ADAS, and the ANS.

3. **The data layer**. – This layer comprises the WAMDASdb database and the databases where is stored the equipment status and alarms information.

The client application layer communicates with the web services layer through messages using the SOAP communication protocol over HTTP transport protocol. And the web services layer communicates with the data layer through ADO .NET API version 2.0 [18], [19].

Now, we describe the implementation of the Status Services and Alarm Services protocols considering the three layers explained above.

4.2.1 Status Services Protocol Implementation

The implementation of the status services protocol is described in terms of steps to denote the sequence of events in the process (see Figure 4.1). These steps match with the figure's enumeration, as we shall discuss next:

- In Step 1, a ***user*** queries for equipment status information on the handheld device.
- In Step 2, the ***handheld device application*** calls the ***GetClientRequest*** web method of the DBR to get the requested information.
- In Step 3, the ***GetClientRequest*** web method calls the ***GetURLFirstOnlineADASService*** web method of the RS, to get the URL of the first online ADAS instance.
- In Step 3.1, the ***GetURLFirstOnlineADASService*** web method calls the ***GetWAMDASDatabaseConnectionString*** web method of the RS, to get the connection string to the ***WAMDASdb*** database.
- In Step 3.2, the ***GetURLFirstOnlineADASService*** web method connects to the ***WAMDASdb*** database to retrieve the information of all ADAS's instances registered in the system. Then, this web method finds the first ADAS that is online and available to respond to requests. If the finding is successful, the URL of the found ADAS is

returned and the process continues with the step 4; if not, the process terminates with a message noting that there is not any online ADAS.

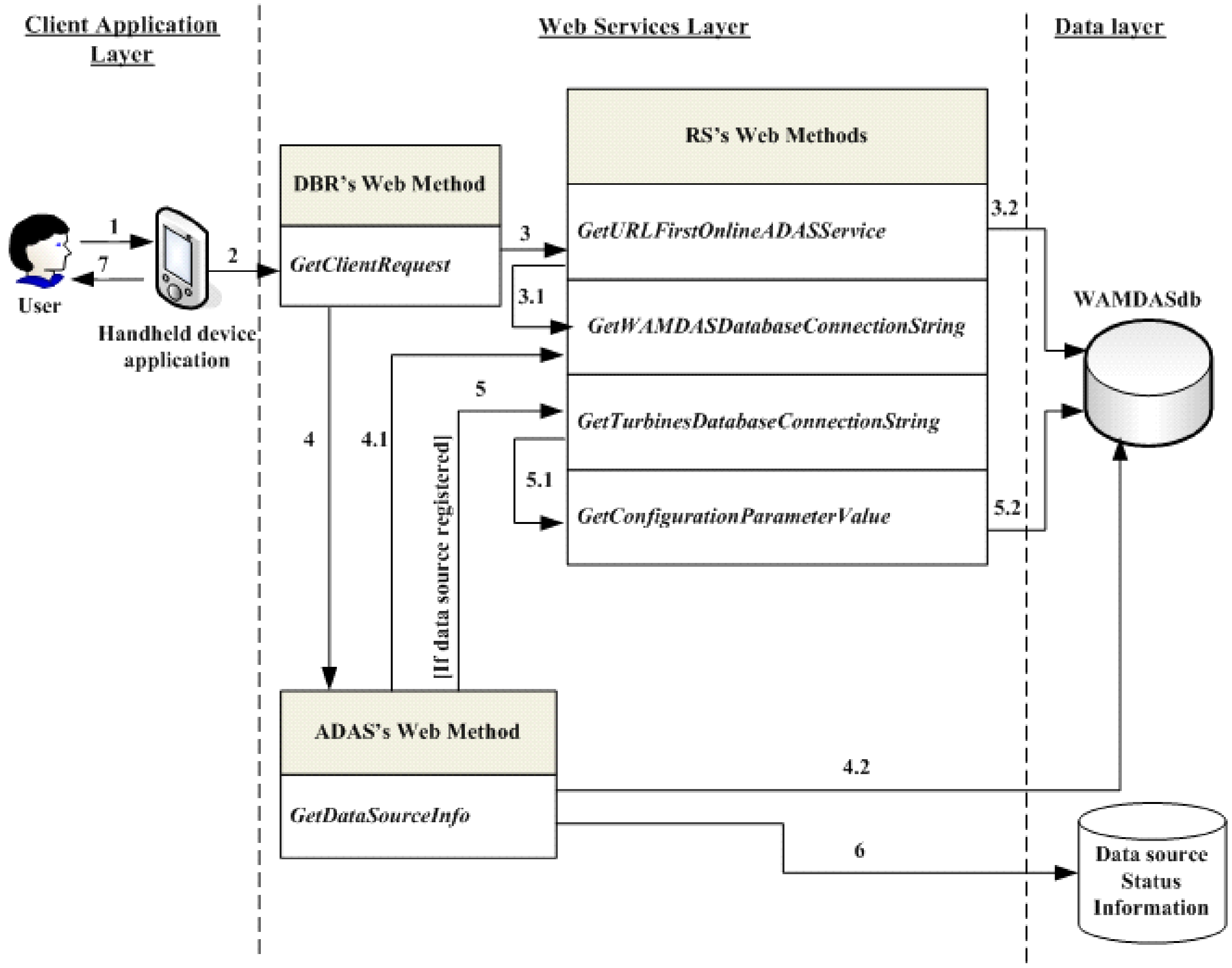

Client Application Layer
Web Services Layer
Data layer
RS's Web Methods
DBR's Web Method
GetURLFirstOnlineADASService
GetClientRequest
GetWAMDASDatabaseConnectionString
GetTurbinesDatabaseConnectionString
GetConfigurationParameterValue
WAMDASdb
User
Handheld device application
1
7
2
3
3.1
3.2
4
4.1
5
5.1
5.2
[If data source registered]
ADAS's Web Method
GetDataSourceInfo
4.2
6
Data source Status Information

Figure 4.1 Status Services Protocol Implementation

- In Step 4, the ***GetClientRequest*** web method calls the ***GetDataSourceInfo*** web method of the found ADAS instance.

- In Step 4.1, the ***GetDataSourceInfo*** web method calls the ***GetWAMDASDatabaseConnectionString*** web method of the RS, to get the connection string to the ***WAMDASdb*** database.
- In Step 4.2, the ***GetDataSourceInfo*** web method connects to the ***WAMDASdb*** database to retrieve the information of the data source, of which the status information was requested. If the data source is registered in the system, the process continues with the step 5, if not, the process terminates with a message noting that the data source is not registered in the system.
- In Step 5, the ***GetDataSourceInfo*** web method calls the web method implemented to get the connection string to the database where the data source status information is stored. For the specific case of turbines data source, the ***GetTurbinesDatabaseConnectionString*** web method is called.
- In Step 5.1, the ***GetTurbinesDatabaseConnectionString*** web method calls the ***GetConfigurationParameterValue*** web method to get the parameters to build the connection string to the database where turbines status information is stored.
- In Step 5.2, the ***GetConfigurationParameterValue*** web method connects to the ***WAMDASdb*** database to retrieve the value of parameters requested by ***GetTurbinesDatabaseConnectionString*** web method.
- In Step 6, the ***GetDataSourceInfo*** web method connects to the database where the data source status information is stored and calls the corresponding stored procedure to get the status information query results. Then these results are sent back to the

GetClientRequest web method, and in turns, this web method sends back the query results to the ***handheld device application.***

- Finally, in Step 7, the ***handheld device application*** displays to the user the resulting equipment status information.

4.2.2 Alarm Services Protocol Implementation

The implementation of the alarm services protocol is described in terms of steps to denote the sequence of events in the process (see Figure 4.2). These steps match with the figure's enumeration, as we shall discuss next:

- In Step 1, when an alarm is triggered, a record is inserted into the alarms table of the corresponding database where the data source alarms information is stored. In response to that event, an after insert statement trigger [20] is executed. This trigger calls the ***CallAlarmBroadcast*** console application to begin the alarm processing.
- In Step 2, the ***CallAlarmBroadcast*** console application calls the ***AlarmBroadcast*** web method of the ADAS, to broadcast the alarm.
- In Step 3, the ***AlarmBroadcast*** web method calls the ***GetURLFirstOnlineANSService*** web method of the RS, to get the URL of the first online ANS instance.
- In Step 3.1, the ***GetURLFirstOnlineANSService*** web method calls the ***GetWAMDASDatabaseConnectionString*** web method of the RS, to get the connection string to the WAMDASdb database.

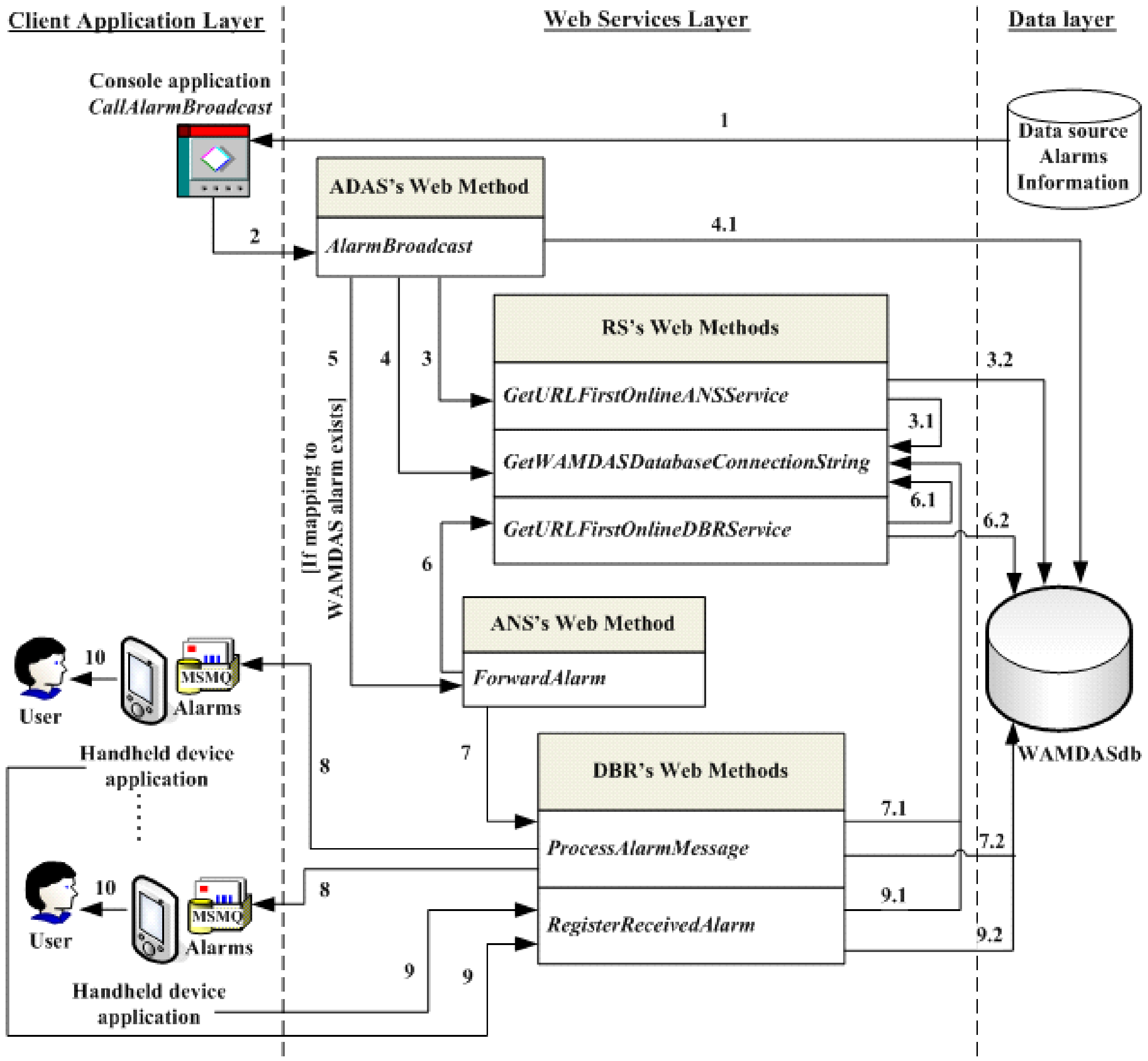


Figure 4.2 Alarm Services Protocol Implementation

- In Step 3.2, the ***GetURLFirstOnlineANSService*** web method connects to the ***WAMDASdb*** database to retrieve the information of all ANS's instances registered in the system. Then, this web method finds the first ANS that is online and available to respond to requests. If the finding is successful, the URL of the found ANS is returned and the process continues with the step 4; if not, the process terminates with a message noting that there is not any online ANS.

- In Step 4, the ***AlarmBroadcast*** web method calls the ***GetWAMDASDatabaseConnectionString*** web method of the RS, to get the connection string to the ***WAMDASdb*** database.
- In Step 4.1, the ***AlarmBroadcast*** web method connects to the ***WAMDASdb*** database to retrieve the information about the real alarm received to verify if it has a mapping to a WAMDAS alarm defined in the system. If the mapping is found, a new alarm event is registered in the system and the process continues with the step 5, if not; the process terminates with a message noting that a mapping for the real alarm has not been defined.
- In Step 5, the ***AlarmBroadcast*** web method calls the ***ForwardAlarm*** web method of the ANS instance found in the step 3 to forward the alarm.
- In Step 6, the ***ForwardAlarm*** web method calls the ***GetURLFirstOnlineDBRService*** web method of the RS, to get the URL of the first online DBR instance.
- In Step 6.1, the ***GetURLFirstOnlineDBRService*** web method calls the ***GetWAMDASDatabaseConnectionString*** web method of the RS, to get the connection string to the ***WAMDASdb*** database.
- In Step 6.2, the ***GetURLFirstOnlineDBRService*** web method connects to the ***WAMDASdb*** database to retrieve the information of all DBR's instances registered in the system. Then, this web method finds the first DBR that is online and available to

respond to requests. If the finding is successful, the URL of the found DBR is returned and the process continues with step 7; if not, the process terminates with a message noting that there is not any online DBR.

- In Step 7, the ***ForwardAlarm*** web method calls the ***ProcessAlarmMessage*** web method of the found DBR instance to process the alarm.
- In Step 7.1, the ***ProcessAlarmMessage*** web method calls the ***GetWAMDASDatabaseConnectionString*** web method of the RS, to get the connection string to the ***WAMDASdb*** database.
- In Step 7.2, the ***ProcessAlarmMessage*** web method connects to the ***WAMDASdb*** database to retrieve the information of the IP addresses of the handheld devices on which users authorized to receive the alarm are currently logged into the system.
- In Step 8, the ***ProcessAlarmMessage*** web method sends the alarm message to those handheld devices' ***Alarms MSMQ***. If the handheld devices are online they receive the alarm message immediately, if not, they will receive the alarm message when they recover their online status.
- In Step 9, once a handheld device receives the alarm message, the ***handheld device application*** peeks at the alarm message from the ***Alarms MSMQ*** and then calls the ***RegisterReceivedAlarm*** web method of the DBR to register in the system that the alarm message was received by that handheld device.

- In Step 9.1, the ***RegisterReceivedAlarm*** web method calls the ***GetWAMDASDatabaseConnectionString*** web method of the RS, to get the connection string to the ***WAMDASdb*** database.
- In Step 9.2, the ***RegisterReceivedAlarm*** web method connects to the ***WAMDASdb*** database to register the reception of the alarm.
- Finally, in step 10, the ***handheld device application*** displays the alarm notification to the ***user***.

For the specific implementation of the MSMQ in the alarm message processing see Appendix A.

Now, we describe the implementation of the alarm acknowledgment process. The way this process works is described also in terms of steps to denote its sequence of events (see Figure 4.3). These steps match with the figure's enumeration, as we shall discuss next:

- In Step 1, a ***user*** acknowledges an alarm notification on the handheld device.
- In Step 2, the ***handheld device application*** picks up the alarm message from the ***Alarms MSMQ,*** removing it from the queue. After that, the ***handheld device application*** sends an alarm acknowledgement message to the ***Alarm's Ack. MSMQ*** of a web server where a DBR is hosted. Once the alarm acknowledgement message is received by the ***Alarm's Ack. MSMQ***, the ***QueueReceiveCompleted*** method of the DBR is executed.
- In Step 3, the ***QueueReceiveCompleted*** method picks up the alarm acknowledgement message from the ***Alarm's Ack. MSMQ***, removing it from the queue. And then, the

QueueReceiveCompleted method calls the ***GetURLFirstOnlineANSService*** web method of the RS, to get the URL of the first online ANS instance.

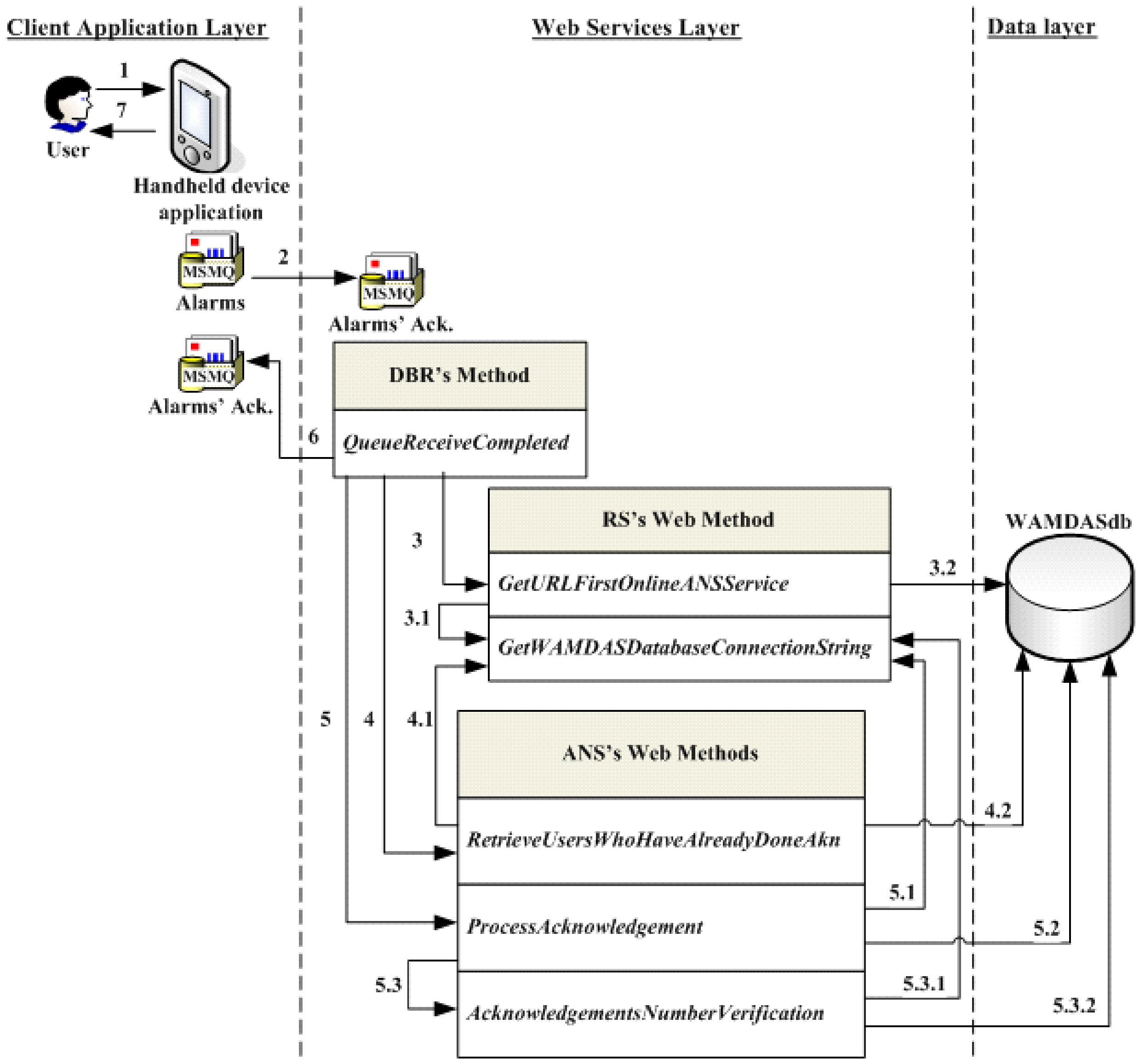


Figure 4.3 Alarm Acknowledgement Process Implementation

- In Step 3.1, the ***GetURLFirstOnlineANSService*** web method calls the ***GetWAMDASDatabaseConnectionString*** web method of the RS, to get the connection string to the ***WAMDASdb*** database.
- In Step 3.2, the ***GetURLFirstOnlineANSService*** web method connects to the ***WAMDASdb*** database to retrieve the information of all ANS's instances registered in the system. Then, this web method finds the first ANS that is online and available to respond to requests. If the finding is successful, the URL of the found ANS is returned and the process continues with the step 4; if not, the process terminates with a message noting that there is not any online ANS.
- In Step 4, the ***QueueReceiveCompleted*** method calls the ***RetrieveUsersWhoHaveAlreadyDoneAkn*** web method of the found ANS instance.
- In Step 4.1, the ***RetrieveUsersWhoHaveAlreadyDoneAkn*** web method calls the ***GetWAMDASDatabaseConnectionString*** web method of the RS, to get the connection string to the ***WAMDASdb*** database.
- In Step 4.2, the ***RetrieveUsersWhoHaveAlreadyDoneAkn*** web method connects to the ***WAMDASdb*** database to retrieve the information of the users who have already acknowledged the alarm.
- In Step 5, the ***QueueReceiveCompleted*** method calls the ***ProcessAcknowledgement*** web method of the ANS, to process the alarm acknowledgement.

- In Step 5.1, the ***ProcessAcknowledgement*** web method calls the ***GetWAMDASDatabaseConnectionString*** web method of the RS, to get the connection string to the ***WAMDASdb*** database.
- In Step 5.2, the ***ProcessAcknowledgement*** web method connects to the ***WAMDASdb*** database to retrieve the information about the handled alarms, and it verifies if the alarm has already been acknowledged from the handheld device. If so, the ***ProcessAcknowledgement*** web method terminates sending back a message noting that "An alarm can be acknowledged from a handheld device only once". If not, the process continues with Step 5.3.
- In Step 5.3, the ***ProcessAcknowledgement*** web method calls the ***AcknowledgementsNumberVerification*** web method of the ANS.
- In Step 5.3.1, the ***AcknowledgementsNumberVerification*** web method calls the ***GetWAMDASDatabaseConnectionString*** web method of the RS, to get the connection string to the ***WAMDASdb*** database.
- In Step 5.3.2, the ***AcknowledgementsNumberVerification*** web method connects to the ***WAMDASdb*** database to retrieve information about the number of acknowledgements to the alarm that have already been registered and the pre-configured number of acknowledgements needed according to the alarm severity; this is done in order to determine if the alarm has already been fully attended. If so, the alarm acknowledgement is registered in the system with status "Already Acknowledged" and a message is sent back noting that "The alarm was already

acknowledged by the configured number of users". If not, the alarm acknowledgement is registered in the system with status "Acknowledged".

- After Step 5.3.2, the process control returns to the ***QueueReceiveCompleted*** method, this method builds and acknowledgement result message with the messages notified by the ***RetrieveUsersWhoHaveAlreadyDoneAkn*** and ***ProcessAcknowledgement*** web methods.
- In Step 6, the ***QueueReceiveCompleted*** method sends the acknowledgment result message to the ***Alarm's Ack. MSMQ*** of the handheld device. If the handheld device is online it receives the message immediately, if not, it will receive when it recovers its online status.
- Once the handheld device receives the alarm acknowledgment result message, the ***handheld device application*** picks it up from the ***Alarm's Ack. MSMQ***, removing it from the queue.
- Finally, in Step 7, the ***handheld device application*** displays the alarm acknowledgment result message to the ***user***.

For the specific implementation of the MSMQ in the processing of alarm acknowledgement message, see Appendix A.

4.3 WAMDAS's Relational Database

WAMDAS needs to manage relevant information about the data sources, the components, the real alarms (which are triggered by contingencies that occur while these components are in use), and the WAMDAS alarms which are defined as part of the configuration scheme for alarm management. Before beginning the alarms processing, the real alarms are mapped to WAMDAS alarms. Also WAMDAS needs to manage data about the users who can use the system, the authorized handheld devices on which the users can log into the system, and the operational shifts to control the system usage. Furthermore WAMDAS requires managing information related to the alarms processing such as the alarm events, the alarms received by the users, the handled alarms and the unhandled alarms.

Finally WAMDAS needs to manage information of the system web services and all the configuration parameters. Hence the system includes a major database component to manage all this information, such database is modeled by the Entity-Relationship (E-R) Diagram shown in Figure 4.4

Now, we describe the E-R diagram for WAMDAS database, which is shown in the Figure 4.4. First, we describe the entities; in general, as we can see, the names of the entity attributes are auto descriptive, so most of them only are listed and we describe only those which need further explanation for their understanding.

1. **User**. – This entity set represents the users of the WAMDAS system; they can log into the two system modules: Handheld device module and the Administrator module. The User entity has the following attributes: User ID, First name, Last name, User name, Password, Email, Telephone, and Status. The status attribute is 'online' when the user is logged into the handheld device module; otherwise it is 'offline'.

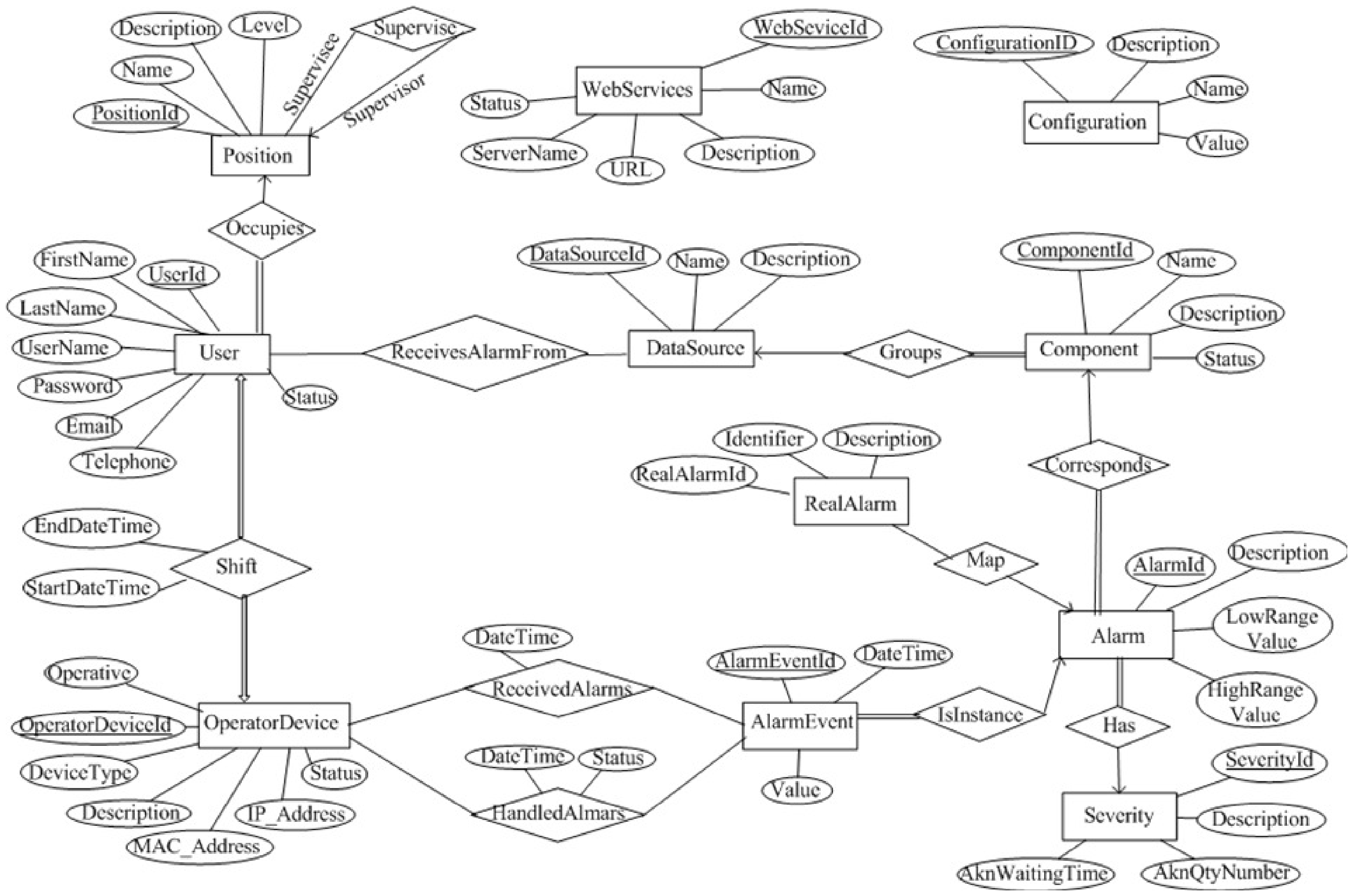


Figure 4.4 Entity-Relationship Diagram for WAMDAS Database

2. **Position**. – This entity set represents the positions of the system users. These positions are: Supervisor, Operator and System administrator. The supervisors and operators can use the handheld device module and only the system administrators can use the administrator module. The Position entity has the following attributes: Position ID, Name, Description, and Level. The level attribute describes the hierarchy of the position as follows: supervisor position has level equal to 1 and the operator position has level equal to 2. Furthermore the position entity has two roles: Supervisor and Supervisee.

3. **Data Source**. – This entity set represents the families of equipment such as Turbines, Chillers and Boilers. The Data Source entity has the following attributes: Data source ID, Name, and Description.

4. **Component**. – This entity set represents the members of an equipment family, for example Turbine A and Turbine B are members of the Turbines family. The components are the specific elements that WAMDAS will monitor to get their status information and process all the alarms triggered by them. The Component entity has the following attributes: Component ID, Name, Description, and Status.

5. **Real Alarm**. – This entity set represents the alarms that are triggered by contingencies that occur while the equipment is in use. The Real Alarm entity has the following attributes: Real alarm ID, Identifier and Description.

6. **Alarm**. – This entity set is also called WAMDAS Alarm, and represents the alarms defined by the system administrators as part of the configuration scheme for alarms

management. The Alarm entity has the following attributes: Alarm ID, Description, Low range value, and High range value.

7. **Severity**. – This entity set represents the values of the metric which measures how dangerous and important an alarm is. The Severity entity has the following attributes: Severity ID, Description, Akn Qty Number, and Akn Waiting Time.

   The Akn Qty Number attribute is the number of acknowledgements needed to conclude that an alarm has been fully attended. And the Akn Waiting Time is the maximum waiting time for an alarm to be fully attended.

8. **Alarm Event**. – This entity set represents the instances of the WAMDAS alarms. An alarm event is created after the real alarm is mapped to a WAMDAS alarm. The Alarm Event entity has the following attributes: Alarm event ID, Value, and Date time.

9. **Operator Device**. – This entity set represents the handheld devices on which the users can log into the system to monitor the equipment status information and to handle the alarms notifications. The Operator Device entity has the following attributes: Operator device ID, Description, Device Type, MAC Address, IP Address, Status, and Operative. The Device Type attribute can have the following values: PDA, Cell phone or Smart Phone. When the WAMDAS system is running on the handheld device the Status attribute has 'online' value, otherwise it has 'offline' value. And the Operative attribute can have the values 'true' or 'false'.

10. **Web Service**. – This entity set represents the deployed instances of the four system web services: DBR, RS, ADAS and ANS. The Web Service entity has the following

attributes: Web service ID, Name, Description, URL, Server Name, and Status. The Server Name attribute is the name of the web server where the web service instance is hosted.

11. **Configuration**. – This entity set represents the configuration parameters of the system. The Configuration entity has the following attributes: Configuration ID, Name, Description, and Value.

Following with the description of the E-R diagram for WAMDAS system database, which is shown in the Figure 4.4, now we describe the relationships among the entities. In general, as we can see, the names of the relationships attributes are auto descriptive, so most of them only are listed and we describe only those which need further explanation for their understanding.

1. **Occupies**. – This relationship is between the User and Position entities. A user must occupy only one position, and a position can be occupied for many users.
2. **Supervise**. – This relationship is between the Position entity and itself. A supervisor position can supervise many supervisee positions, and a supervisee position can be supervised only by one supervisor position.
3. **Shift**. – This relationship is between the User and Operator Device entities. One user must be involved only in an operational shift, and also one operator device must be involved only in that shift. The Shift relationship has the following attributes: Star Date Time and End Date Time.

4. **Groups**. – This relationship is between the Data Source and Component entities. One data source can group many components, and a component must be grouped into only one data source.

5. **Corresponds**. – This relationship is between the Alarm and Component entities. One alarm must correspond only to one component, and one component can have many corresponding alarms.

6. **Map**. – This relationship is between the Real Alarm and Alarm entities. One real alarm can map only with one alarm, and one alarm can map with many real alarms.

7. **Is Instance**. – This relationship is between the Alarm and Alarm Event entities. One alarm can have many alarm events, and one alarm event must instantiate only one alarm.

8. **Has**. – This relationship is between the Alarm and Severity entities. One alarm must have only one severity, and a severity can be owned by many alarms.

9. **Receives Alarm From**. – This relationship is between the User and Data Source entities. One user can receive the alarms generated by many data sources, and the alarms generated by one data source can be received by many users.

10. **Received Alarms**. – This relationship is between the Operator Device and Alarm Event entities. One operator device can receive many alarm events, and one alarm event can be received on many operator devices. The Received Alarm relationship has the Date Time attribute.

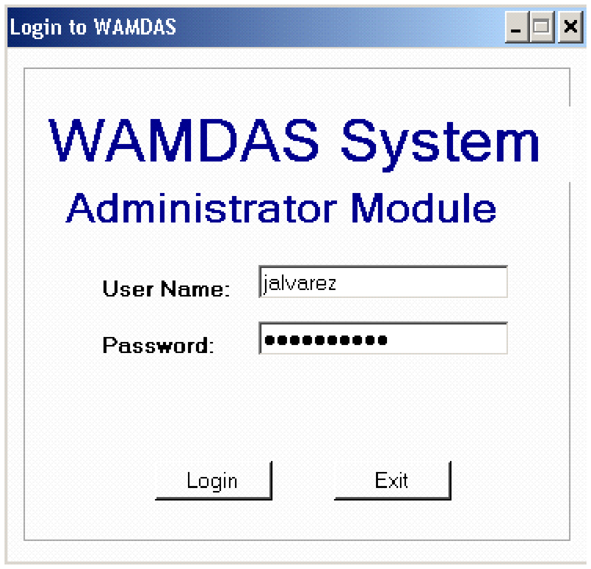
11. **Handled Alarms**. – This relationship is between the Operator Device and Alarm Event entities. One operator device can handle many alarm events, and one alarm event can

be handled on many operator devices. The Handled Alarms relationship has the Date Time and Status attributes.

4.4 Administrator Module

The Administrator Module is a Windows-based application that runs on workstations. This module permits the system administrators to configure all system information and parameters necessary that make WAMDAS run smoothly.

Only authorized administrators users can log into the module, which is validated through a user name and password (see Figure 4.5).


Login to WAMDAS

WAMDAS System

Administrator Module

User Name: jalvarez

Password: ●●●●●●●●●●●

Figure 4.5 Administrator Module - User Login

When the login is successful, the main window of the module is shown to the user (see Figure 4.6). This main window provides a menu with options, each one providing a particular functionality described as follows:

1. **User management**. – This menu option permits the system administrators to register the system users who will be authorized to log into the handheld device and administrator modules of WAMDAS. Only the users with ‘administrator’ position are authorized to log into the Administrator module. The status of a user is controlled automatically; the system administrators do not manage it. When the user is logged into the system, his/her status is ‘online’; otherwise it is ‘offline’.

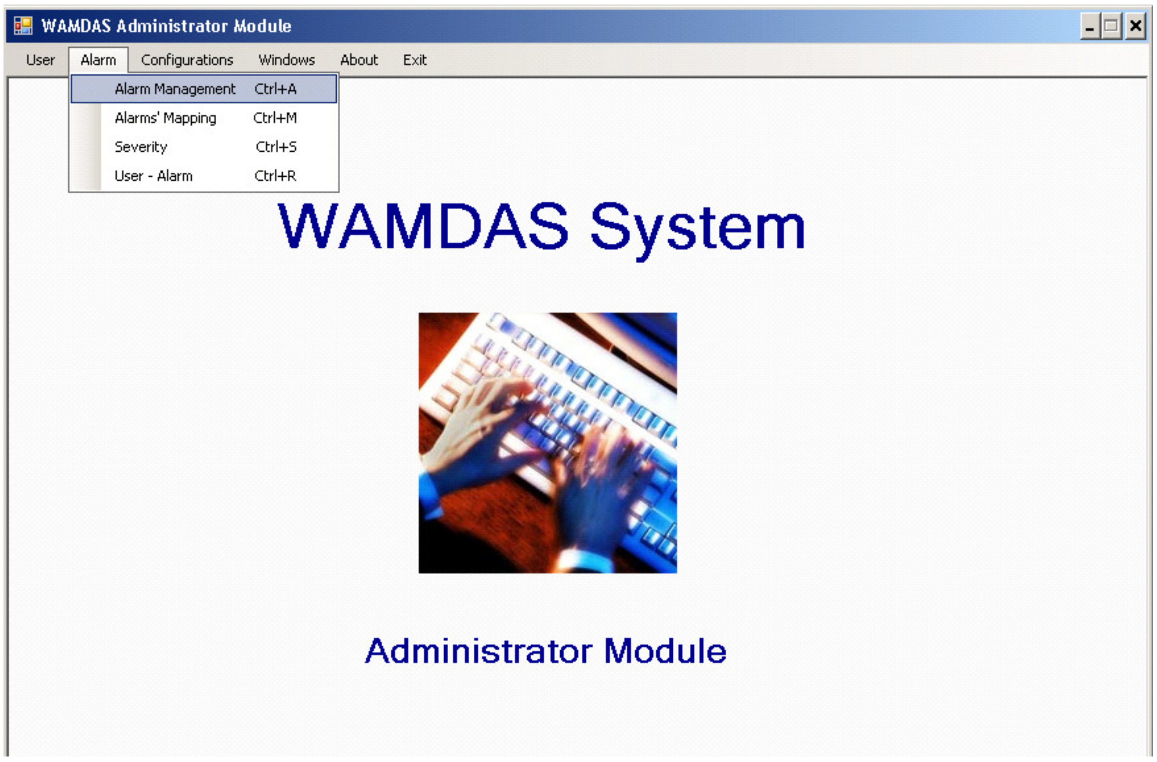
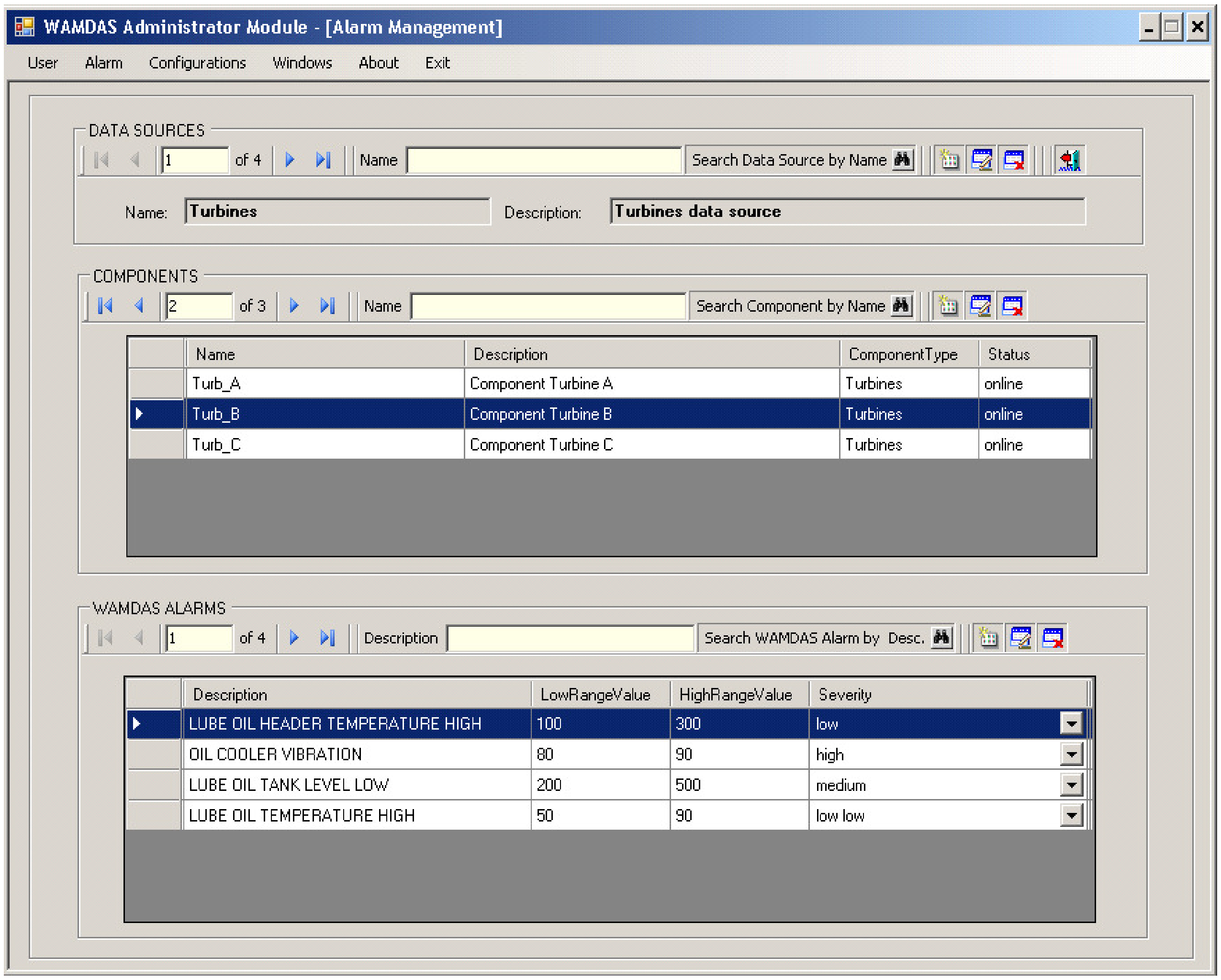


Figure 4.6 Administrator Module - Main Window

2. **Alarm management**. – This menu option permits the system administrators to define: 1) the Data Sources, which model families of equipment, for example Turbines, Chillers and Boilers; 2) the components, which are members of an equipment family, for example Turbine A and Turbine B are members of the Turbines family. These components are the specific elements that WAMDAS will monitor to get their status information and process all the alarms triggered by contingencies that occur while these components are in use under normal operational conditions. And 3) the alarms that WAMDAS will manage, each alarm corresponds to a specific component. For each alarm, it is necessary to specify a description; a low range value, a high range value, and its severity (see Figure 4.7). For example the 'Turb_B' component, which belongs to the 'Turbines' data source, triggers the alarms 'Lube oil header temperature high', 'Oil cooler vibration', 'Lube oil tank level low' and 'Lube oil temperature high'.

3. **Mapping between real alarms and WAMDAS alarms**. – Each piece of equipment has its own type of alarms, which we call *real alarms*. Since similar pieces of equipment might differ in the representation and name of their real alarms, WAMDAS permits system administrators to define their own configuration schema for alarms. This enables a more uniform alarm management. It is necessary to map each real alarm to a *WAMDAS alarm* (defined in the Alarm management menu option of the Administrator module). When a real alarm is detected by the system, the web service ADAS before beginning the alarm processing, first looks for the corresponding WAMDAS alarm. If the alarm is found, then this 'WAMDAS alarm' is delivered to the

proper handheld devices. If not, an error message is sent noting that a mapping for the real alarm has not been defined.



Name: Turbines
Description: Turbines data source

Name	Description	ComponentType	Status
Turb_A	Component Turbine A	Turbines	online
Turb_B	Component Turbine B	Turbines	online
Turb_C	Component Turbine C	Turbines	online

Description	LowRangeValue	HighRangeValue	Severity
LUBE OIL HEADER TEMPERATURE HIGH	100	300	low
OIL COOLER VIBRATION	80	90	high
LUBE OIL TANK LEVEL LOW	200	500	medium
LUBE OIL TEMPERATURE HIGH	50	90	low low

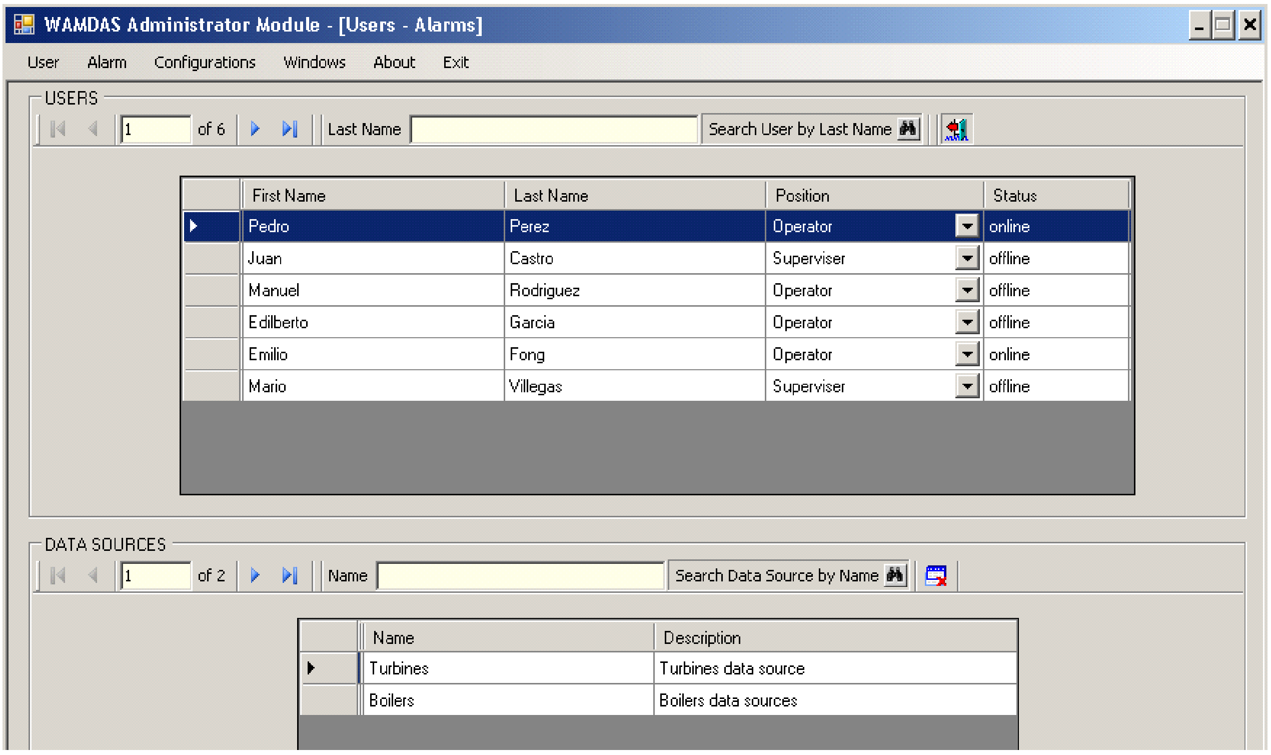
Figure 4.7 Administrator Module – Alarm Management

4. **Alarm severity**. – This menu option permits the system administrators to define the severities for WAMDAS alarms. The severity metric measures the danger and importance of the situation. For each severity, it is necessary to specify the number of

acknowledgements needed to conclude that an alarm has been fully attended, and the maximum waiting time for an alarm to be fully attended.

5. **User-to-Alarm Mapping**. – This menu option permits the systems administrators to assign to the users the WAMDAS alarms that they will receive, in order to provide a user-based configuration scheme for alarms. This enables the system to dynamically change who handles which alarms and when.

In figure 4.8, we can see that the user Pedro Perez, who has an operator position, is authorized to receive the alarms of the Turbines and Boilers data sources.

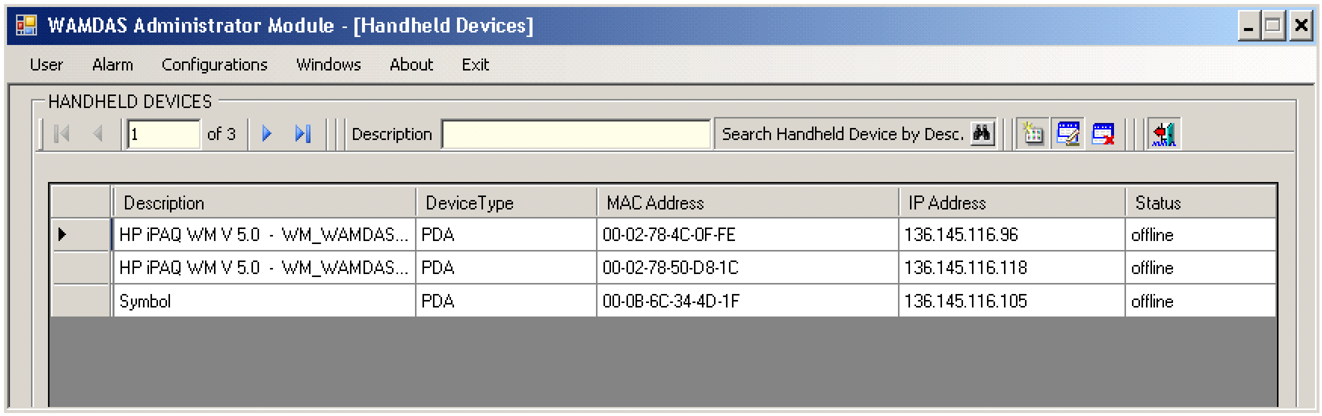


First Name	Last Name	Position	Status
Pedro	Perez	Operator	online
Juan	Castro	Superviser	offline
Manuel	Rodriguez	Operator	offline
Edilberto	Garcia	Operator	offline
Emilio	Fong	Operator	online
Mario	Villegas	Superviser	offline

Name	Description
Turbines	Turbines data source
Boilers	Boilers data sources

Figure 4.8 Administrator Module – User-to-Alarm Mapping

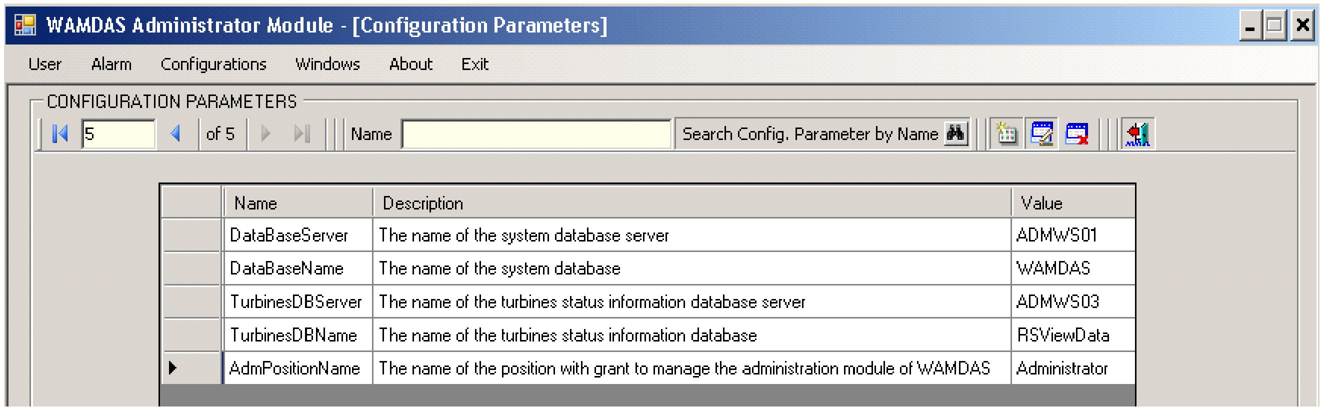
6. **Handheld Devices**. – This menu option permits the system administrators to configure the handheld devices authorized to run the handheld device module of WAMDAS (see Figure 4.9). For a handheld device, it is necessary to specify a description, the device type (e.g. PDA, Smart Phone), and its MAC and IP addresses. When a user executes the handheld device module, the handheld device's MAC and IP addresses is validated to determine if it is authorized to run WAMDAS. The status of a handheld device is controlled automatically; the administrator users do not manage it. The status is 'online' when WAMDAS is running on the handheld device; otherwise it is 'offline'.


WAMDAS Administrator Module - [Handheld Devices]
User
Alarm
Configurations
Windows
About
Exit
HANDHELD DEVICES
1
of 3
Description
Search Handheld Device by Desc.

Description	DeviceType	MAC Address	IP Address	Status
HP iPAQ WM V 5.0 - WM_WAMDAS...	PDA	00-02-78-4C-0F-FE	136.145.116.96	offline
HP iPAQ WM V 5.0 - WM_WAMDAS...	PDA	00-02-78-50-D8-1C	136.145.116.118	offline
Symbol	PDA	00-0B-6C-34-4D-1F	136.145.116.105	offline

Figure 4.9 Administrator Module – Handheld Devices

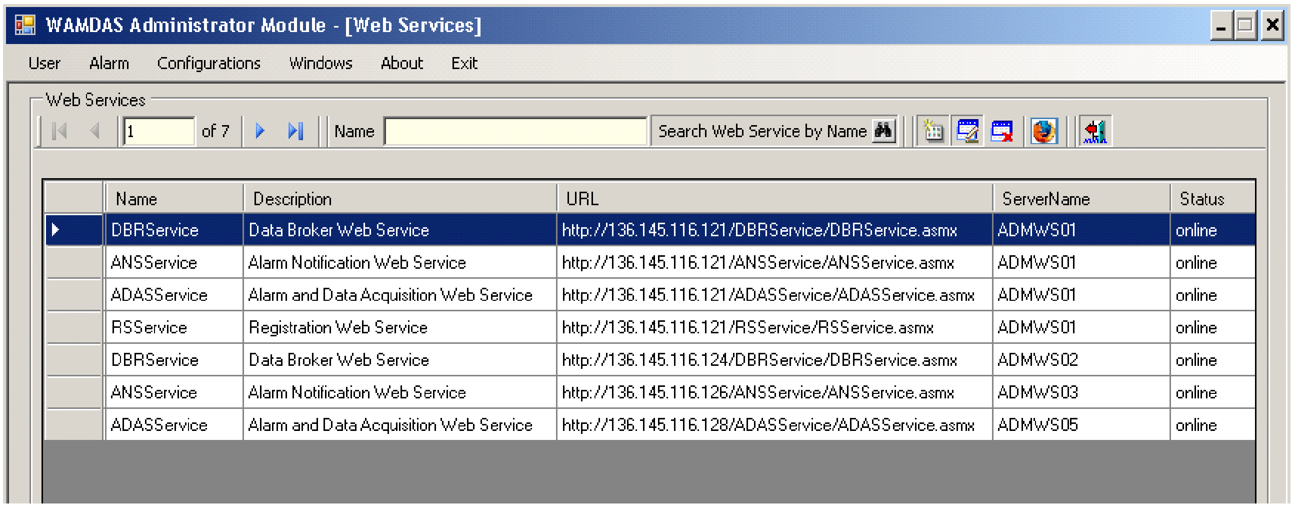
7. **Parameters** – This menu option permits the system administrators to configure parameters such as the name of the system database, names of databases storing the status information about the data sources, the names of the servers where these databases are hosted, and any other necessary parameters. In Figure 4.10, we can see that for each parameter, it is necessary to specify its name, a description, and its value.


WAMDAS Administrator Module - [Configuration Parameters]
User
Alarm
Configurations
Windows
About
Exit
CONFIGURATION PARAMETERS
5
of 5
Name
Search Config. Parameter by Name

Name	Description	Value
DataBaseServer	The name of the system database server	ADMWS01
DataBaseName	The name of the system database	WAMDAS
TurbinesDBServer	The name of the turbines status information database server	ADMWS03
TurbinesDBName	The name of the turbines status information database	RSViewData
AdmPositionName	The name of the position with grant to manage the administration module of WAMDAS	Administrator

Figure 4.10 Administrator Module – Parameters

8. **Web Services** – This menu option permits system administrators to configure dynamically the information of system web services, simplifying the cases in which web services are moved to another web server, or new web service instances are added to the system. In Figure 4.11, we can see that for a web service, it is specified its name, a description, its URL, the server name where it is hosted, and its status.


WAMDAS Administrator Module - [Web Services]
User
Alarm
Configurations
Windows
About
Exit
Web Services
1
of 7
Name
Search Web Service by Name

Name	Description	URL	ServerName	Status
DBRService	Data Broker Web Service	http://136.145.116.121/DBRService/DBRService.asmx	ADMWS01	online
ANSService	Alarm Notification Web Service	http://136.145.116.121/ANSService/ANSService.asmx	ADMWS01	online
ADASService	Alarm and Data Acquisition Web Service	http://136.145.116.121/ADASService/ADASService.asmx	ADMWS01	online
RSService	Registration Web Service	http://136.145.116.121/RSService/RSService.asmx	ADMWS01	online
DBRService	Data Broker Web Service	http://136.145.116.124/DBRService/DBRService.asmx	ADMWS02	online
ANSService	Alarm Notification Web Service	http://136.145.116.126/ANSService/ANSService.asmx	ADMWS03	online
ADASService	Alarm and Data Acquisition Web Service	http://136.145.116.128/ADASService/ADASService.asmx	ADMWS05	online

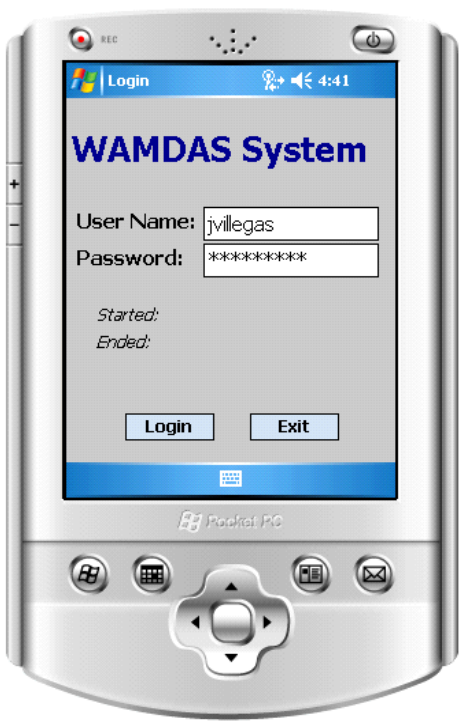
Figure 4.11 Administrator Module – Web Services

4.5 Handheld Device Module

The Handheld device module is a handheld device application that runs on handheld devices like PDAs. This module permits the users to query the equipment status information, receive and acknowledge the alarms, and query for historic alarms acknowledgement information.

This module was optimized to be more reliable, secure and robust. It provides a strong security mechanism to access the system. The handheld devices use for connectivity a wireless network with Wi-Fi Protected Access (WPA) for network authentication, and the Temporal Key Integrity Protocol (TKIP) for data encryption. This is done to guarantee that only authorized handheld devices can join the content network. When a user attempts to execute the module on a handheld device, the MAC Address and the IP Address of the device are verified to determine if the device is legally registered in the system. Thus, only registered handheld devices are authorized to run WAMDAS.

Likewise, only authorized users can log into the system, which is validated through a user name and password (see Figure 4.12). When the login is successful, an operational shift is created in the system, registering the beginning date and time (see Figure 4.13), and the main window of the module is shown to the user (see Figure 4.14). This main window provides a menu with two main options: the option “Equipments” to query the equipment status information, and the option “Event” to query for historic alarms acknowledgement information. Finally, when the user exits from the application (see Figure 4.15); the operational shift is closed in the system, registering the ending date and time (see Figure 4.16).

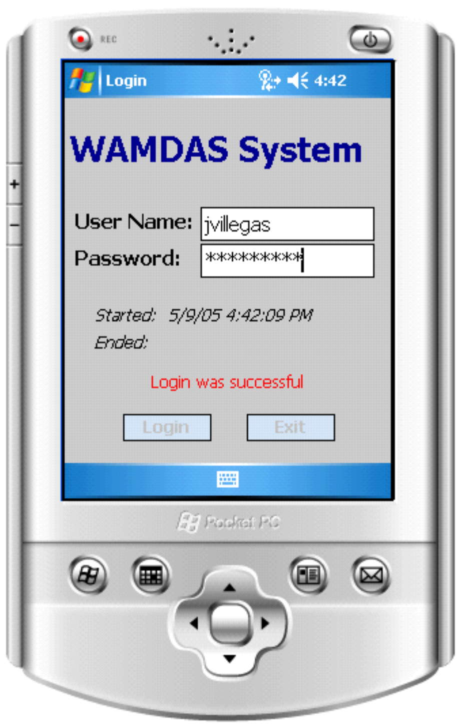


WAMDAS System

User Name: jvillegas
Password: **********

Started:
Ended:

Figure 4.12 Handheld Device Module - User Login



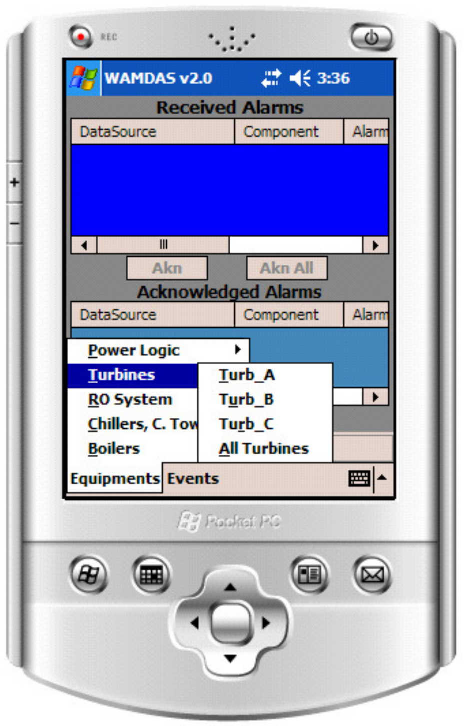
WAMDAS System

User Name: jvillegas
Password: **********

Started: 5/9/05 4:42:09 PM
Ended:

Login was successful

Figure 4.13 Handheld Device Module - Operational Shift Start



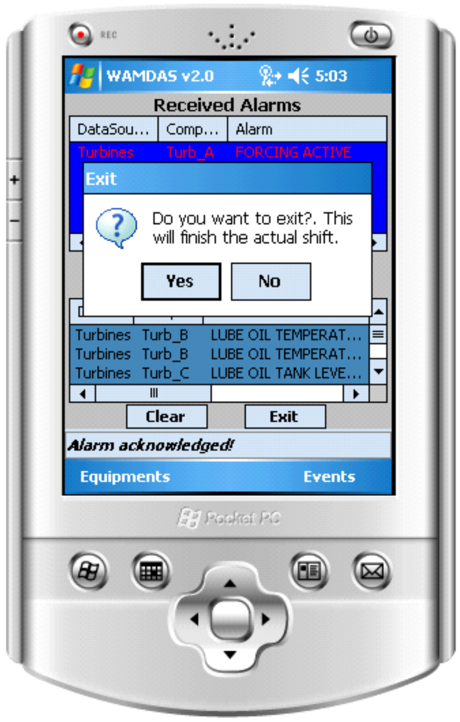
Received Alarms

DataSource	Component	Alarm

Acknowledged Alarms

DataSource	Component	Alarm

Figure 4.14 Handheld Device Module – Main Window



Received Alarms

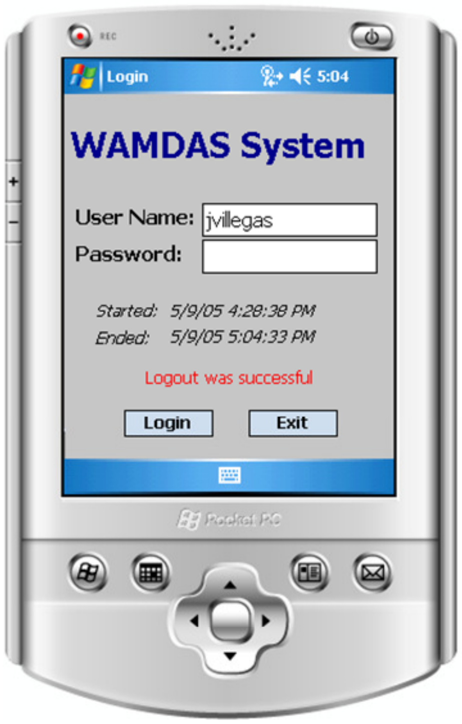
DataSou…	Comp…	Alarm
Turbines	Turb_A	FORCING ACTIVE

Do you want to exit?. This will finish the actual shift.

Turbines	Turb_B	LUBE OIL TEMPERAT…
Turbines	Turb_B	LUBE OIL TEMPERAT…
Turbines	Turb_C	LUBE OIL TANK LEVE…

Alarm acknowledged!

Figure 4.15 Handheld Device Module - Application Exit

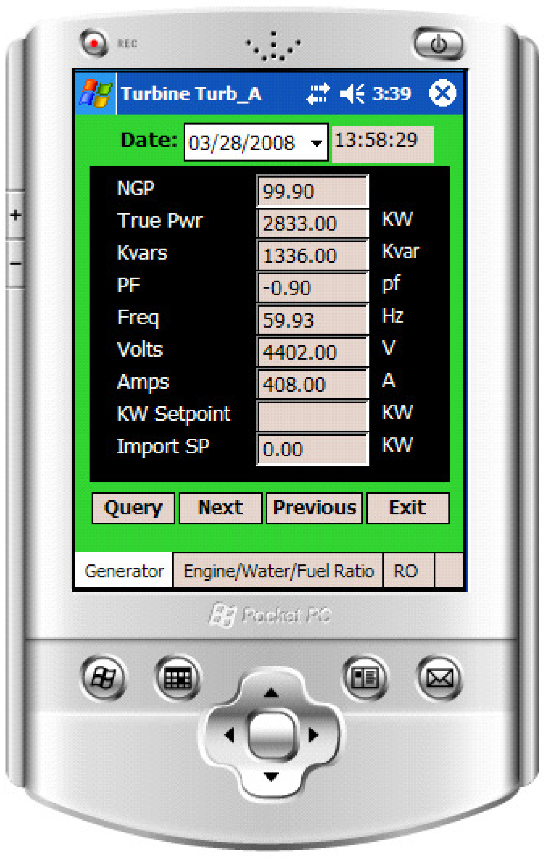


Started: 5/9/05 4:28:38 PM
Ended: 5/9/05 5:04:33 PM
Logout was successful

Figure 4.16 Handheld Device Module - User Logout

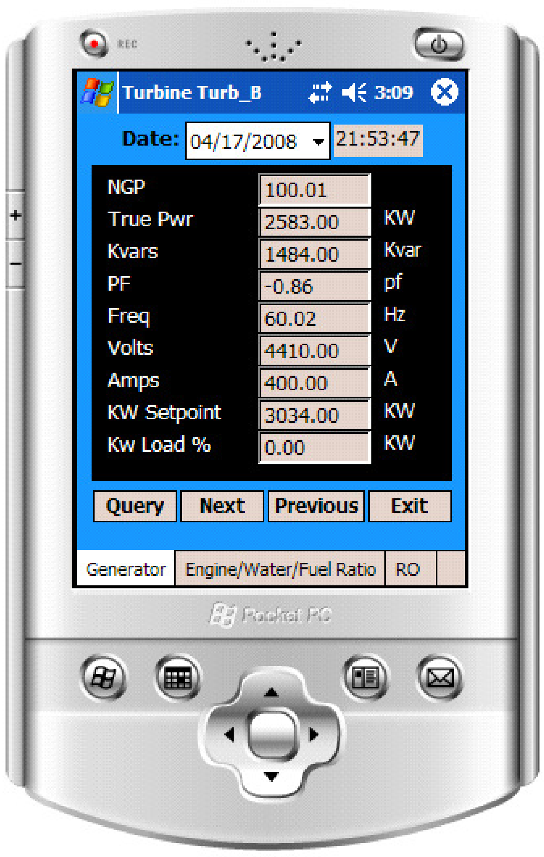
Likewise, the design of the interfaces of the handheld device module was improved to provide the users all the equipment status and alarms information in a very intuitive and feature-rich manner.

The user to query the equipment status information (see figure 4.17) must first select a date and then must push the button ‘Query’. The information retrieved is shown ordered by the reading time. Using the buttons ‘Next’ and ‘Previous’, the user can navigate through the readings. The Figure 4.17 shows the Generator status information from the turbine ‘Turb_A’ for the date and time: 03/28/2008 13:58:29. And the Figure 4.18 shows the Generator status information from the turbine ‘Turb_B’ for the date and time: 04/17/2008 21:53:47.


Turbine Turb_A 3:39

Date: 03/28/2008 13:58:29

NGP	99.90	
True Pwr	2833.00	KW
Kvars	1336.00	Kvar
PF	-0.90	pf
Freq	59.93	Hz
Volts	4402.00	V
Amps	408.00	A
KW Setpoint		KW
Import SP	0.00	KW


Turbine Turb_B 3:09

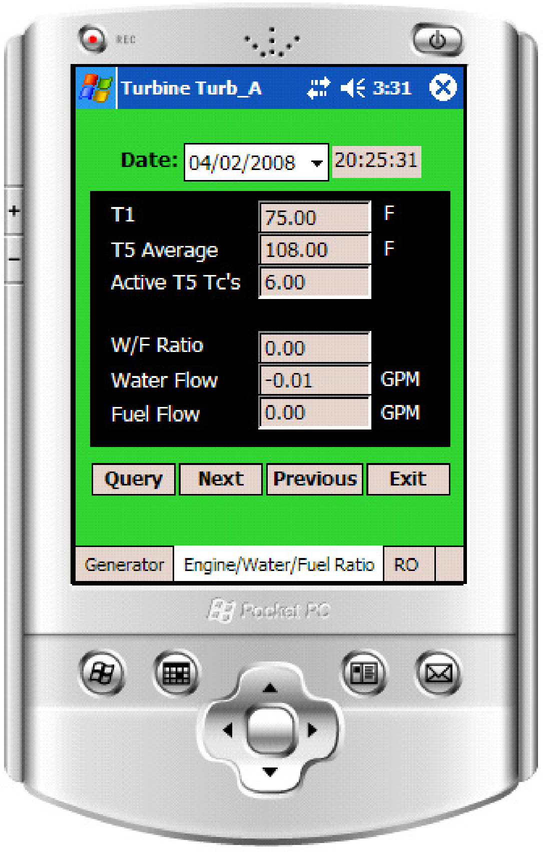
Date: 04/17/2008 21:53:47

NGP	100.01	
True Pwr	2583.00	KW
Kvars	1484.00	Kvar
PF	-0.86	pf
Freq	60.02	Hz
Volts	4410.00	V
Amps	400.00	A
KW Setpoint	3034.00	KW
Kw Load %	0.00	KW

Figure 4.17 Handheld Device Module - Turb_A Generator Status Information

Figure 4.18 Handheld Device Module - Turb_B Generator Status Information

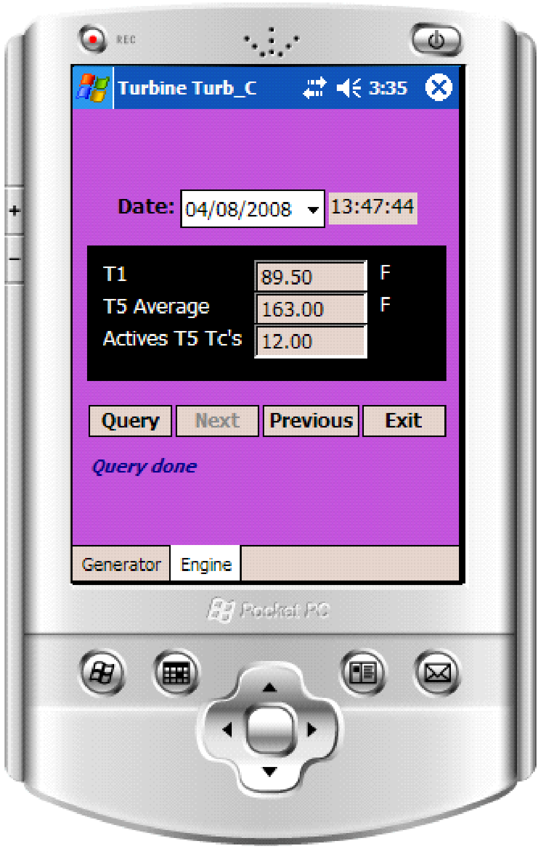
Likewise, the Figure 4.19 shows the Engine/Water/Fuel Ratio status information from the turbine 'Turb_A' for the date and time: 04/02/2008 20:25:31. And the Figure 4.20 shows the Engine status information from the turbine 'Turb_C' for the date and time: 04/08/2008 13:47:44.



Turbine Turb_A 3:31

Date: 04/02/2008 20:25:31

T1	75.00	F
T5 Average	108.00	F
Active T5 Tc's	6.00	
W/F Ratio	0.00	
Water Flow	-0.01	GPM
Fuel Flow	0.00	GPM



Turbine Turb_C 3:35

Date: 04/08/2008 13:47:44

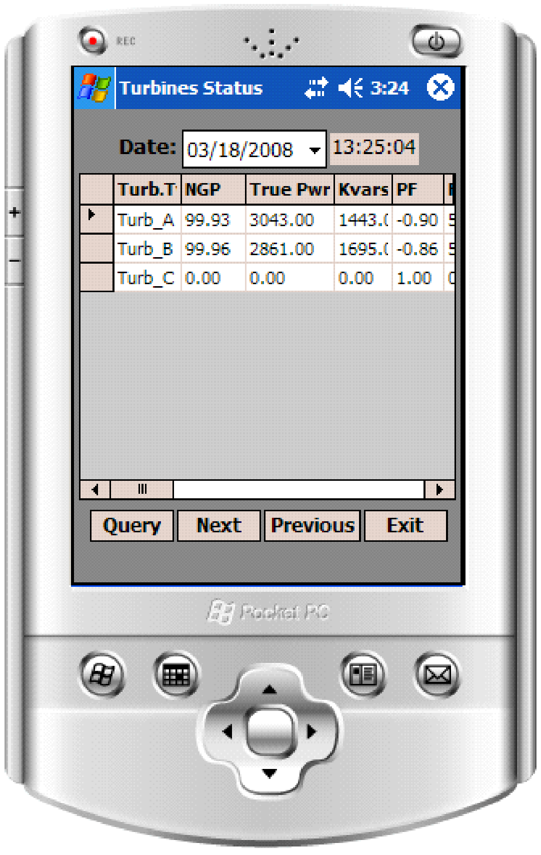
T1	89.50	F
T5 Average	163.00	F
Actives T5 Tc's	12.00	

Query done

Figure 4.19 Handheld Device Module - Turb_A Engine Status Information

Figure 4.20 Handheld Device Module - Turb_C Engine Status Information

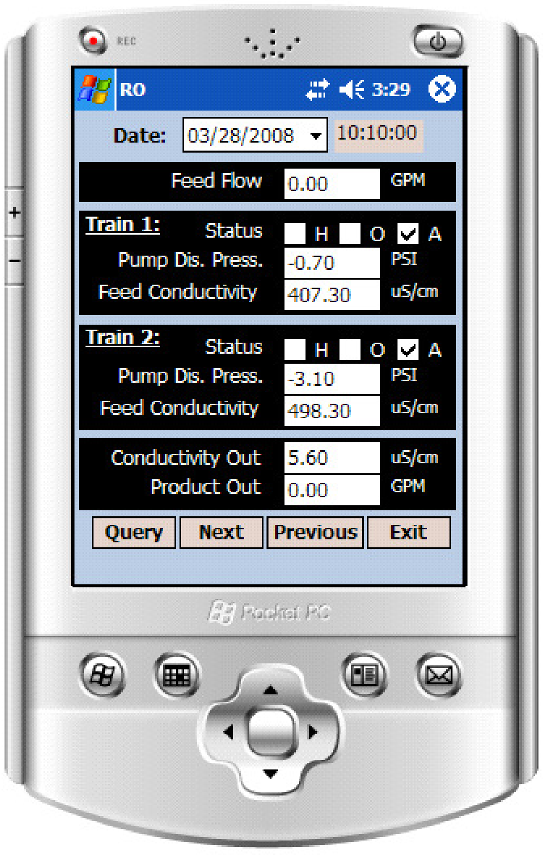
The Figure 4.21 shows the status information from the turbines data source (i.e. the status information from all its components), that is from 'Turb_A', 'Turb_B' and 'Turb_C' turbines which are members of the turbines family. The turbines data source status information shown in the figure is for the date and time: 03/18/2008 13:25:04. And finally, the Figure 4.22 shows the status information from RO for the date and time: 03/28/2008 10:10:00.



Date: 03/18/2008 13:25:04

Turb.T	NGP	True Pwr	Kvars	PF
Turb_A	99.93	3043.00	1443.(	-0.90
Turb_B	99.96	2861.00	1695.(	-0.86
Turb_C	0.00	0.00	0.00	1.00

Figure 4.21 Handheld Device Module - Turbines Data Source Status



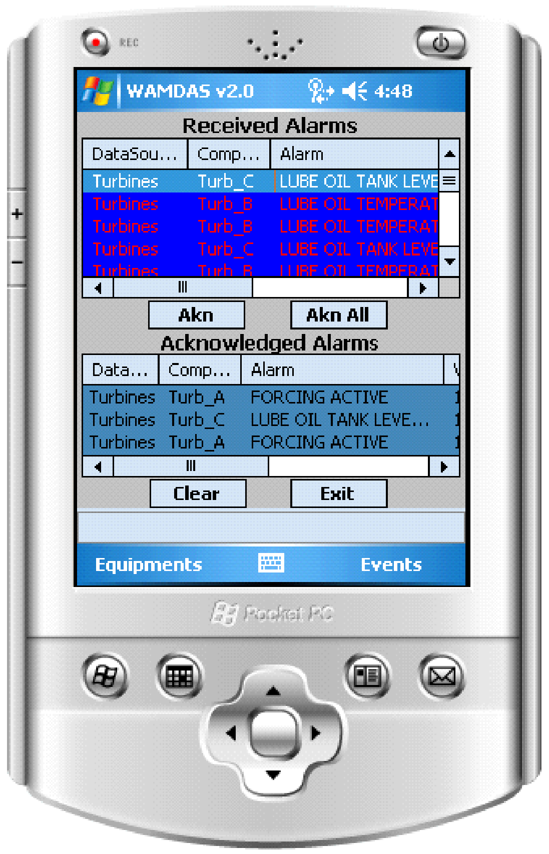
Date: 03/28/2008 10:10:00

Feed Flow	0.00	GPM
Train 1: Status	H ☐ O ☐ A ✓	
Pump Dis. Press.	-0.70	PSI
Feed Conductivity	407.30	uS/cm
Train 2: Status	H ☐ O ☐ A ✓	
Pump Dis. Press.	-3.10	PSI
Feed Conductivity	498.30	uS/cm
Conductivity Out	5.60	uS/cm
Product Out	0.00	GPM

Figure 4.22 Handheld Device Module - RO Status Information

The Figure 4.23 shows the main window for the alarms handling, the top part of the window shows the information of the received alarms, waiting to be acknowledged. If the user wants to acknowledge an alarm, he/she must select the alarm and then push the “Akn” button. And, if the user wants to acknowledge all the received alarms, he/she only must push the “Akn All” button. The low part of the window shows the information of the acknowledged alarms, if the user wants to clear this part of the windows, he/she must push the “Clear” button.

The Figure 4.24 shows an alarm being notified to the user, this notification displays the following information: the data source, the component, the alarm name, the alarm value, the date and time when the alarm occurred, and the alarm severity.

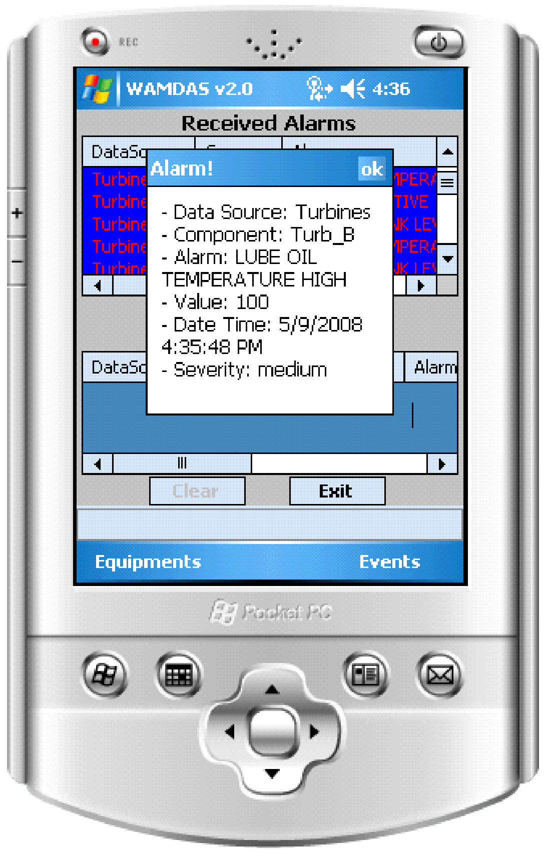


Received Alarms

DataSou...	Comp...	Alarm
Turbines	Turb_C	LUBE OIL TANK LEVE
Turbines	Turb_B	LUBE OIL TEMPERAT
Turbines	Turb_B	LUBE OIL TEMPERAT
Turbines	Turb_C	LUBE OIL TANK LEVE
Turbines	Turb_B	LUBE OIL TEMPERAT

Acknowledged Alarms

Data...	Comp...	Alarm
Turbines	Turb_A	FORCING ACTIVE
Turbines	Turb_C	LUBE OIL TANK LEVE...
Turbines	Turb_A	FORCING ACTIVE



- Data Source: Turbines
- Component: Turb_B
- Alarm: LUBE OIL TEMPERATURE HIGH
- Value: 100
- Date Time: 5/9/2008 4:35:48 PM
- Severity: medium

Figure 4.23 Handheld Device Module – Alarms Handling Main Window

Figure 4.24 Handheld Device Module - Alarm Notification

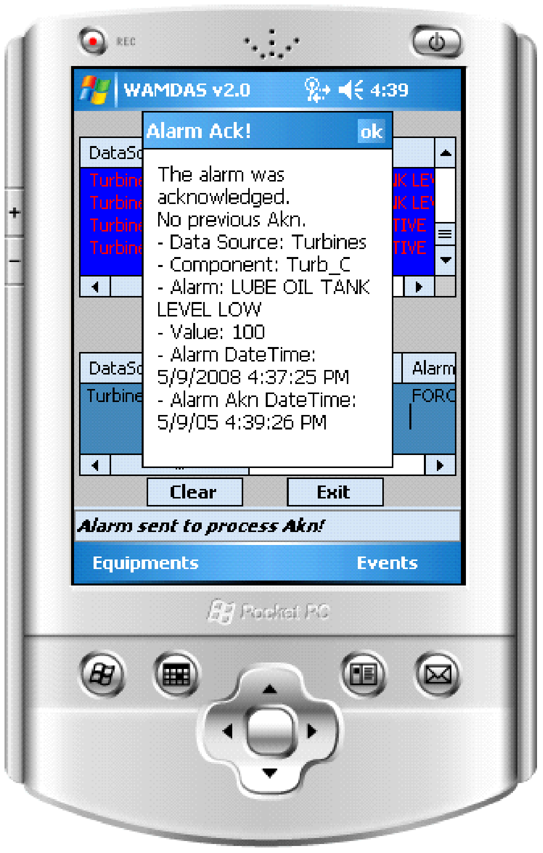
After a user acknowledges an alarm and the system processes the acknowledgement, he/she receives an alarm acknowledgment result notification; there may be three different cases as described as follows:

1. The alarm was acknowledged and there are no previous acknowledgements (see Figure 4.25).

2. The alarm was acknowledged and there are previous acknowledgements (see Figure 4.26).

3. The alarm was already acknowledged by the configured number of users (see Figure 4.27).

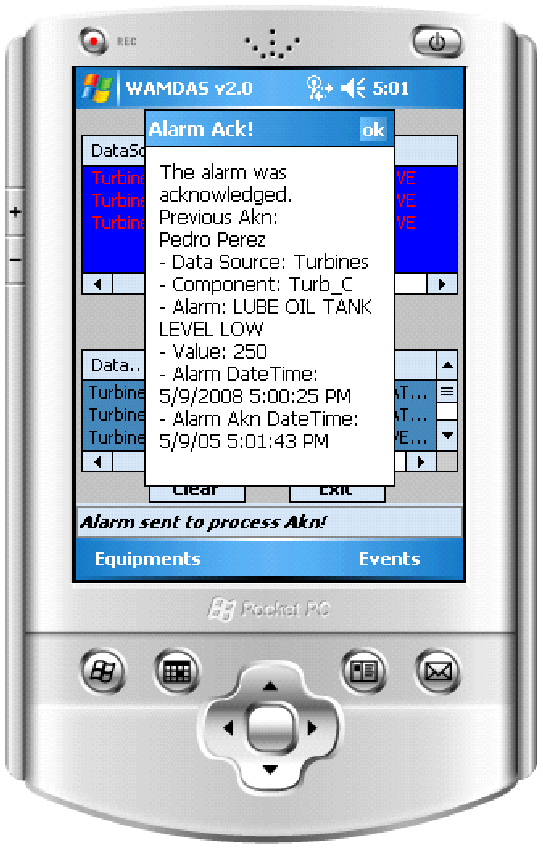
In the cases 2 and 3, the notification displays the information of the users who have already acknowledged the alarm.



The alarm was acknowledged.
No previous Akn.
- Data Source: Turbines
- Component: Turb_C
- Alarm: LUBE OIL TANK LEVEL LOW
- Value: 100
- Alarm DateTime: 5/9/2008 4:37:25 PM
- Alarm Akn DateTime: 5/9/05 4:39:26 PM

Alarm sent to process Akn!

Figure 4.25 Alarm Acknowledgement Result Notification - Case 1

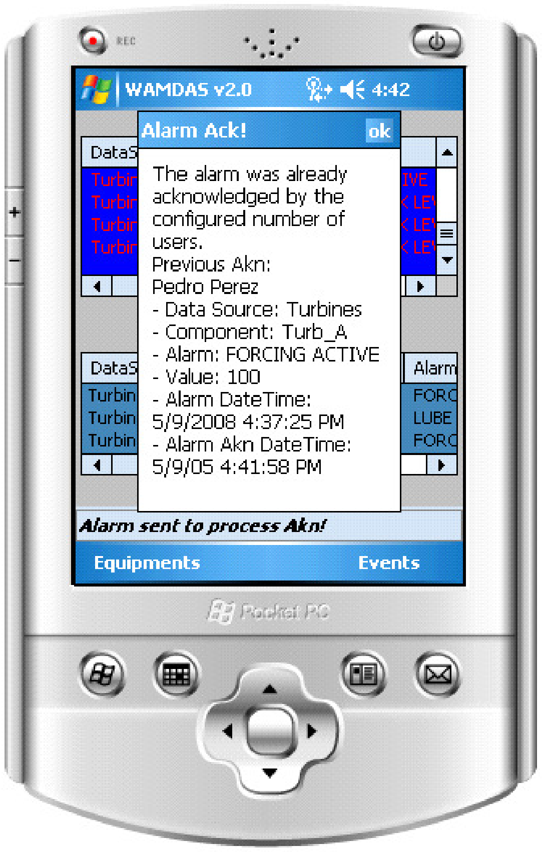


The alarm was acknowledged.
Previous Akn:
Pedro Perez
- Data Source: Turbines
- Component: Turb_C
- Alarm: LUBE OIL TANK LEVEL LOW
- Value: 250
- Alarm DateTime: 5/9/2008 5:00:25 PM
- Alarm Akn DateTime: 5/9/05 5:01:43 PM

Alarm sent to process Akn!

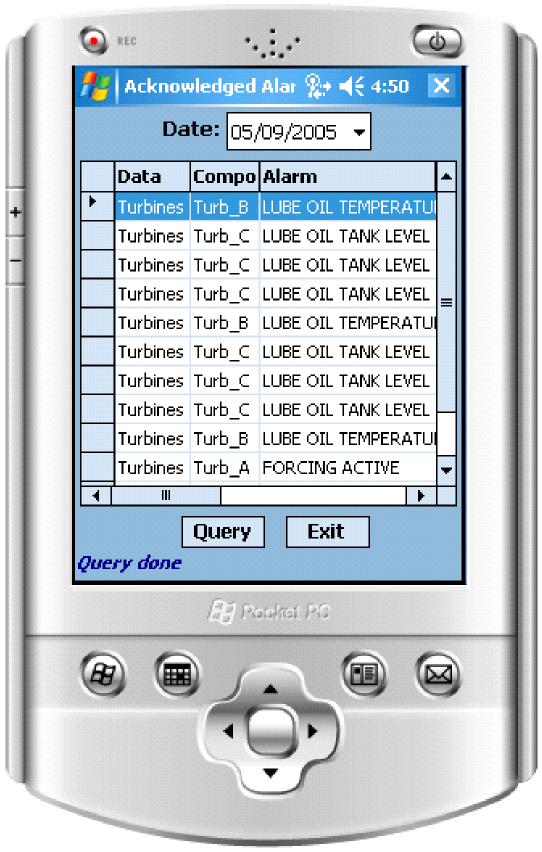
Figure 4.26 Alarm Acknowledgement Result Notification - Case 2

The user to query for historic alarms acknowledgement information (see figure 4.28) must first select a date and then must push the button ‘Query’. The information retrieved is displayed ordered by acknowledgement time.



The alarm was already acknowledged by the configured number of users.
Previous Akn:
Pedro Perez
- Data Source: Turbines
- Component: Turb_A
- Alarm: FORCING ACTIVE
- Value: 100
- Alarm DateTime:
5/9/2008 4:37:25 PM
- Alarm Akn DateTime:
5/9/05 4:41:58 PM

Alarm sent to process Akn!



Date: 05/09/2005

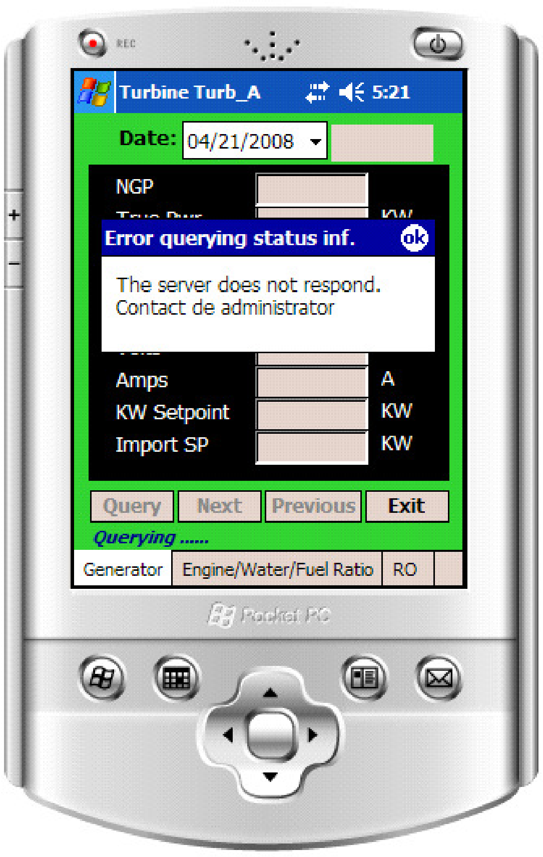
Data	Compo	Alarm
Turbines	Turb_B	LUBE OIL TEMPERATU
Turbines	Turb_C	LUBE OIL TANK LEVEL
Turbines	Turb_C	LUBE OIL TANK LEVEL
Turbines	Turb_C	LUBE OIL TANK LEVEL
Turbines	Turb_B	LUBE OIL TEMPERATU
Turbines	Turb_C	LUBE OIL TANK LEVEL
Turbines	Turb_C	LUBE OIL TANK LEVEL
Turbines	Turb_C	LUBE OIL TANK LEVEL
Turbines	Turb_B	LUBE OIL TEMPERATU
Turbines	Turb_A	FORCING ACTIVE

Query done

Figure 4.27 Alarm Acknowledgement Result Notification - Case 3

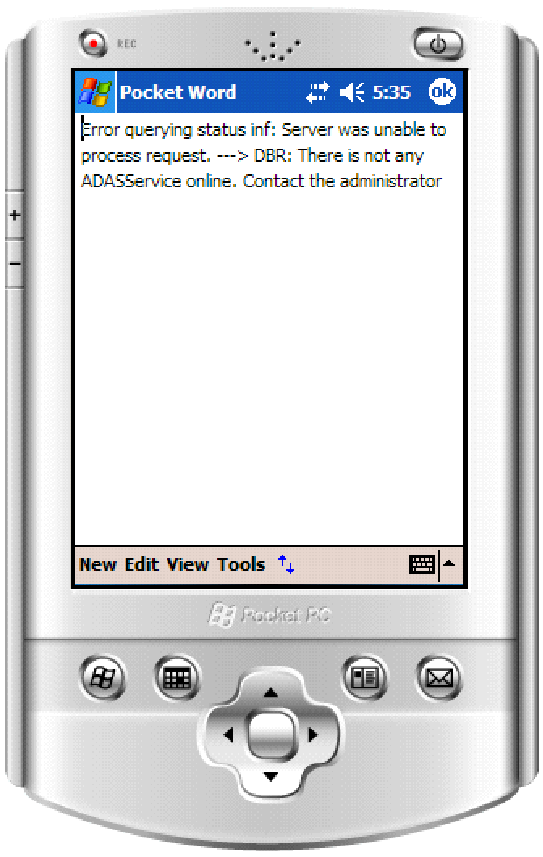
Figure 4.28 Handheld Device Module - Historic Acknowledged Alarms

When an unexpected error occurs in the application, a friendly message is shown to the user notifying the error (see Figure 4.29); in addition, a very specific error message is logged in a log file for troubleshooting purposes (see Figure 4.30).



Error querying status inf.

The server does not respond.
Contact de administrator



Error querying status inf: Server was unable to process request. ---> DBR: There is not any ADASService online. Contact the administrator

Figure 4.29 Handheld Device Module - Friendly Error Message

Figure 4.30 Handheld Device Module - Specific Error Message

The handheld device module can be deployed in handheld devices that support the following Windows platforms: Pocket PC 2003, Windows Mobile 5.0, and later versions. The module can be rewritten in other programming language of the Microsoft family such as Visual Basic and Visual C++, using the .NET Compact Framework. However for rewriting the module in Java or other programming languages, COM bridges or third party bridges must be used to access to MSMQ.

,

5 EXPERIMENTS AND RESULTS

We conducted a series of experiments on our new version of WAMDAS to test performance and fault tolerance of the system protocols, and also we conducted a usability evaluation at a pharmaceutical plant.

5.1 Evaluation Scenario

The evaluation scenario used for the experiments is as follows (see Figure 5.1):

- We used four workstations as web servers in which we deployed instances of the system web services (DBR, ADAS, ANS and RS), the databases, and the Status Query Simulator and Alarms Simulator programs. These workstations were Dell machines with 2.8 GHz, Pentium IV CPU, 1 GB of RAM and 80 GB HD; all of them were networked by a 100 Mbps Ethernet. And all of these machines were running Windows XP Professional version 2002, and they also had configured the Internet Information Services (IIS) version 5.1 and the .NET Framework version 2.0. We used Microsoft SQL Server 2000 as the DBMS to manage the WAMDASdb system database and the database where the status and alarms information were kept.
- We used one laptop in which we deployed the WAMDAS Administrator module. This laptop was a Dell machine with 1.73 GHz, Pentium IV CPU, 1 GB of RAM and 80 GB HD; and it was connected to a Wi-Fi 802.11g network.

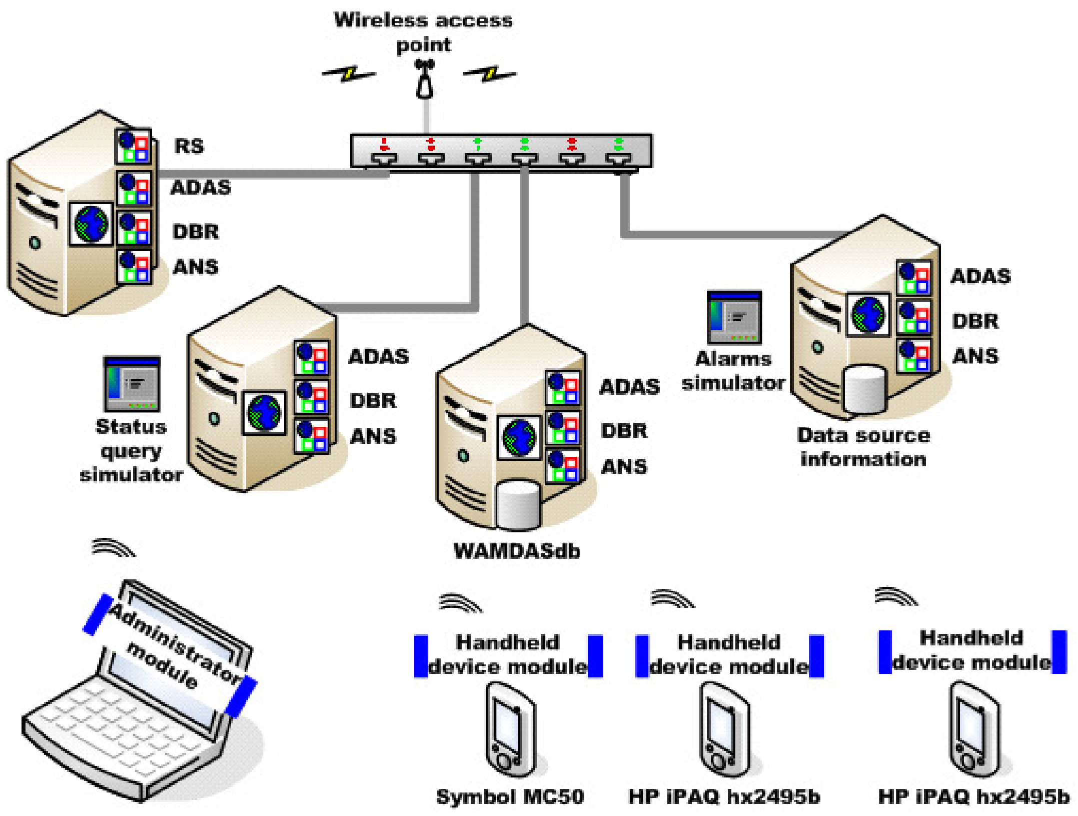


Figure 5.1 Experiments Scenario

- Finally, we used three PDAs, two of them were HP iPAQ hx2495b running Window Mobile 5.0 and the other was a Symbol MC50 running Pocket PC 2003. In these PDAs we deployed our handheld device application. All PDAs were connected to a Wi-Fi 802.11g network.

5.2 Status Services Protocol Evaluation

To evaluate the status services protocol, we used a database of 278.63 MB size with 1,590,376 records which was a backup from the pharmaceutical plant database, and also the

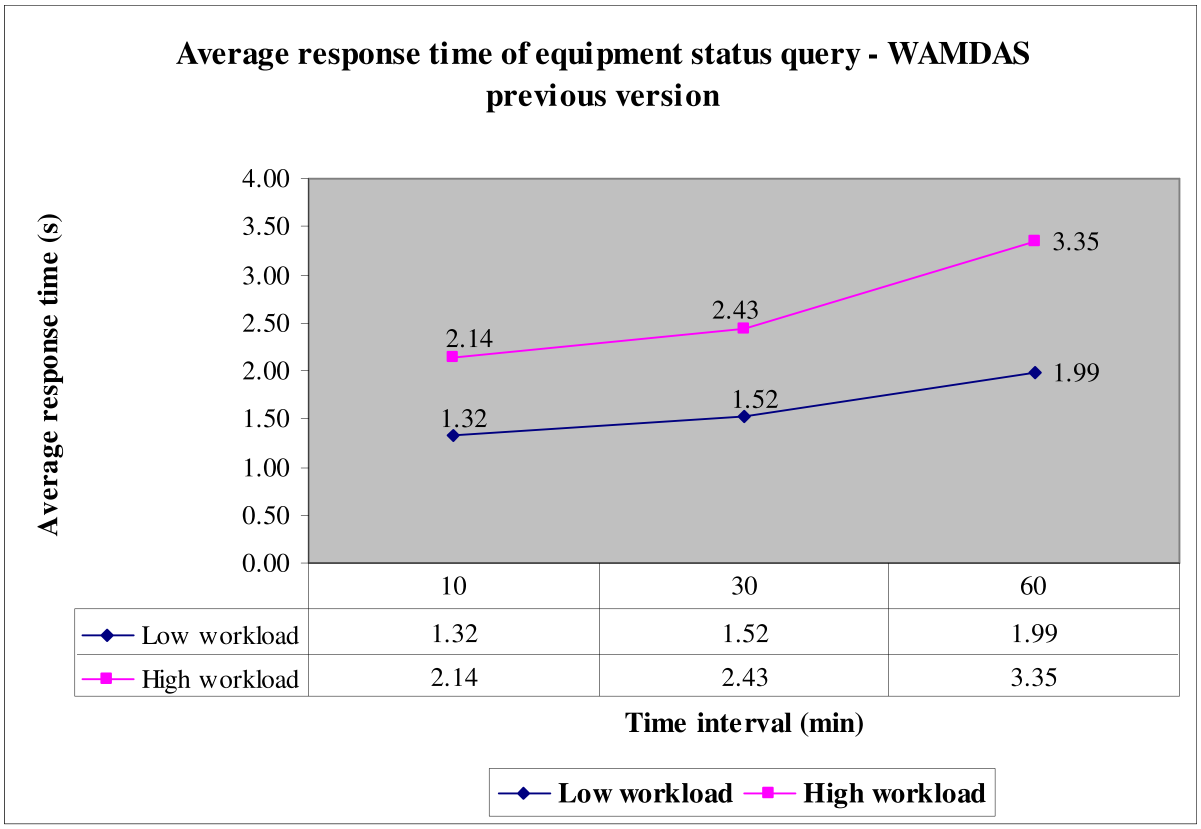
Status Query Simulator program was used to generate queries randomly during different fixed intervals of time, at low and high query workloads. The specific time intervals and the number of queries that the program generates are approximately as follows:

- 10 minutes and at least from 10 to 100 queries,
- 30 minutes and at least from 30 to 300 queries, and
- 60 minutes and at least from 65 to 660 queries.

The metric that we measured was the response time of equipment status query, which is the elapsed time since a query of equipment status information is requested until the whole query result is obtained. We measured this metric using the previous version of WAMDAS implemented with in-Line SQL statements and the new version implemented with stored procedures. For the previous version of WAMDAS, the Table 5.1 shows the average number of queries generated by the simulator program and the Figure 5.2 shows the average response time of status query, both of them for low and high query workloads during the three time intervals cited above.

Table 5.1 Average number of queries to measure the Response Time of Equipment Status Query - WAMDAS previous version

Average number of queries generated by the Status Query Simulator program		
Time interval (min)	**Low workload**	**High workload**
10	13	58
30	68	134
60	205	372



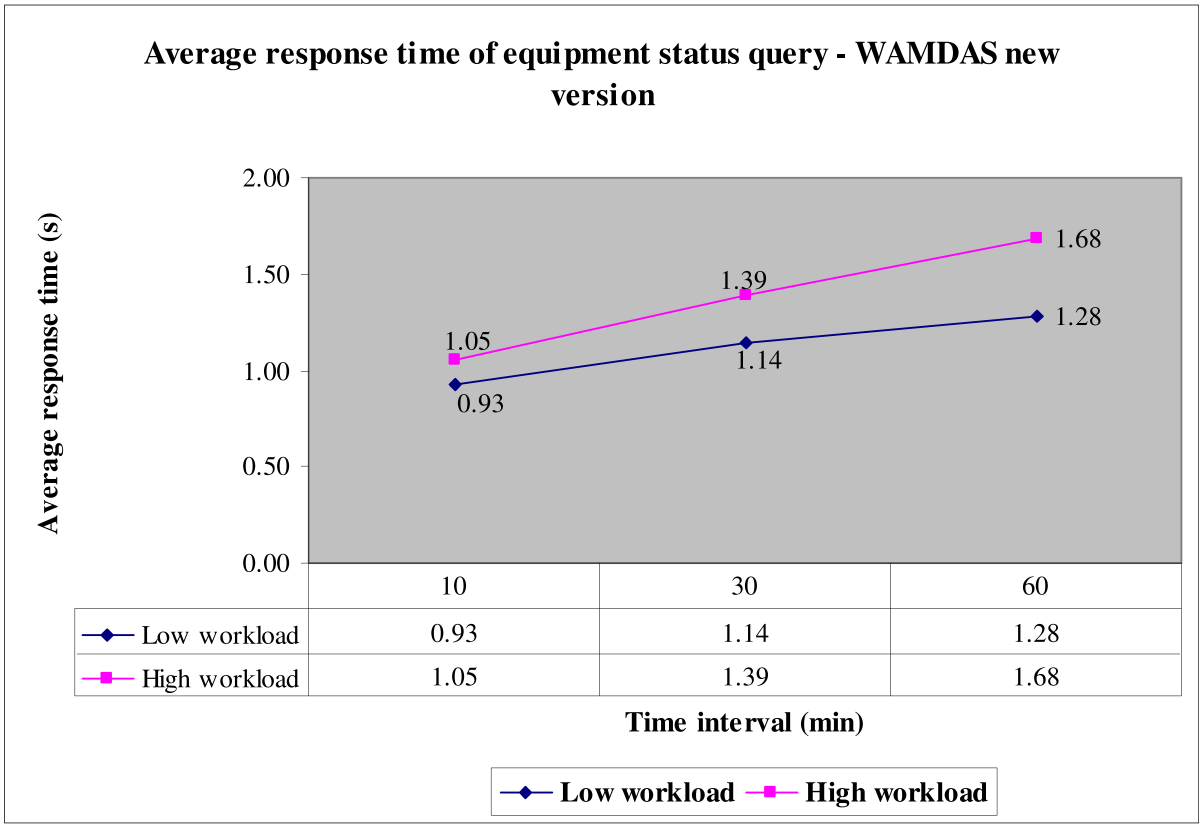
	10	30	60
◆ Low workload	1.32	1.52	1.99
■ High workload	2.14	2.43	3.35

Figure 5.2 Average response time of equipment status query - WAMDAS previous version

And for the new version of WAMDAS, the Table 5.2 shows the average number of queries generated by the simulator program and the Figure 5.3 shows the average response time of status query, both of them for low and high query workloads also during the three time intervals.

Table 5.2 Average number of queries to measure the Response Time of Equipment Status Query - WAMDAS new version

Average number of queries generated by the Status Query Simulator program		
Time interval (min)	Low workload	High workload
10	10	39
30	70	213
60	139	317



	10	30	60
Low workload	0.93	1.14	1.28
High workload	1.05	1.39	1.68

Figure 5.3 Average response time of equipment status query - WAMDAS new version

As we can see in Figure 5.4, the average response times of equipment status query with the new version of WAMDAS are much lower than the average response times of equipment status query with the previous version. The response time of equipment status query was reduced by 41%. Thus, we can state that the response time of equipment status query was optimized by our new implementation of the status services protocol using stored procedures.

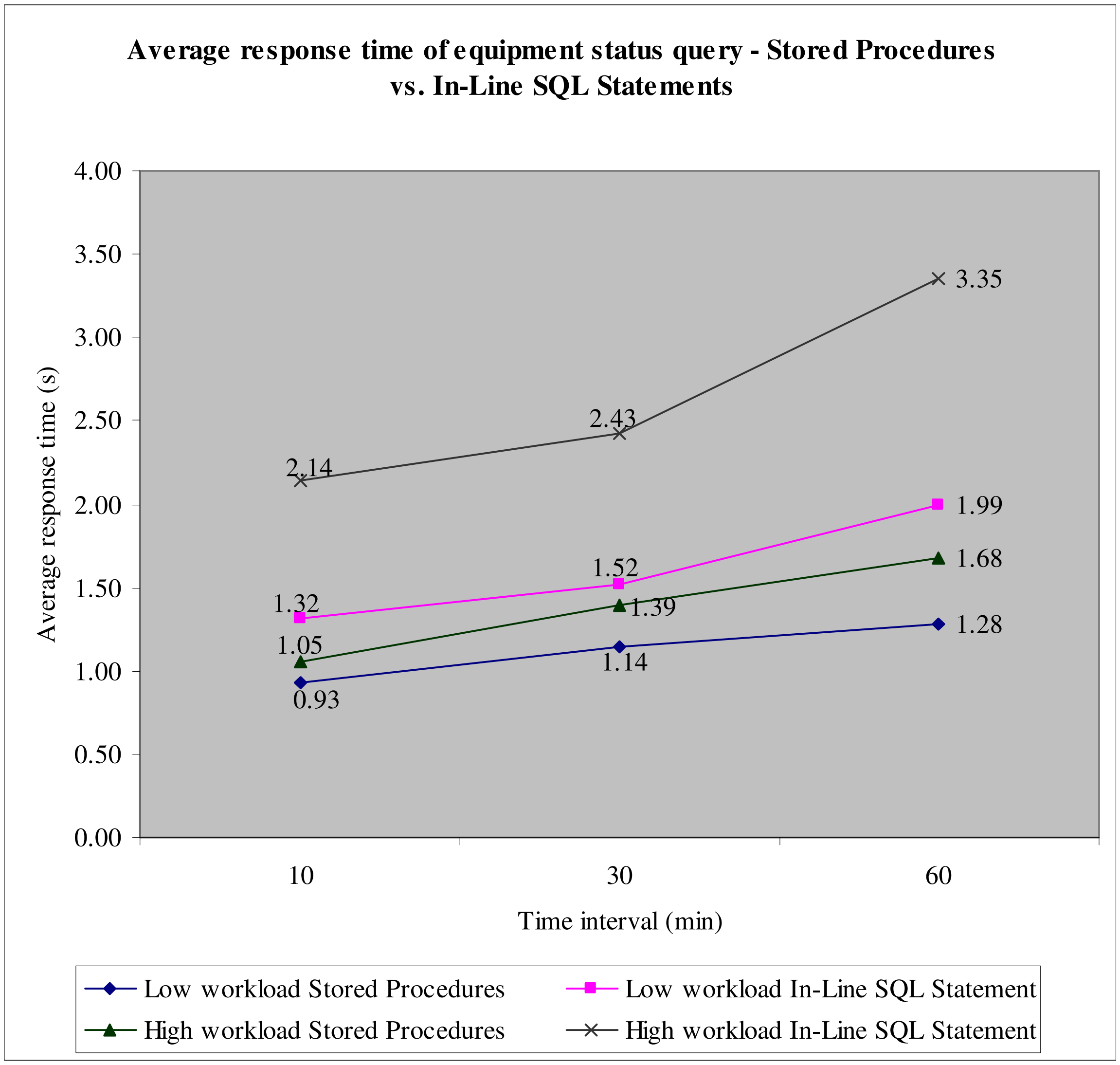


Figure 5.4 Average response time of equipment status query - Stored procedures vs. In-Line SQL statements

5.3 Alarm Services Protocol Evaluation

To evaluate the alarm services protocol, we used an Alarms Simulator program that generates alarms randomly during different fixed intervals of time, at low and high alarm workloads. The specific time intervals and the number of alarms that the program generates are approximately as follows:

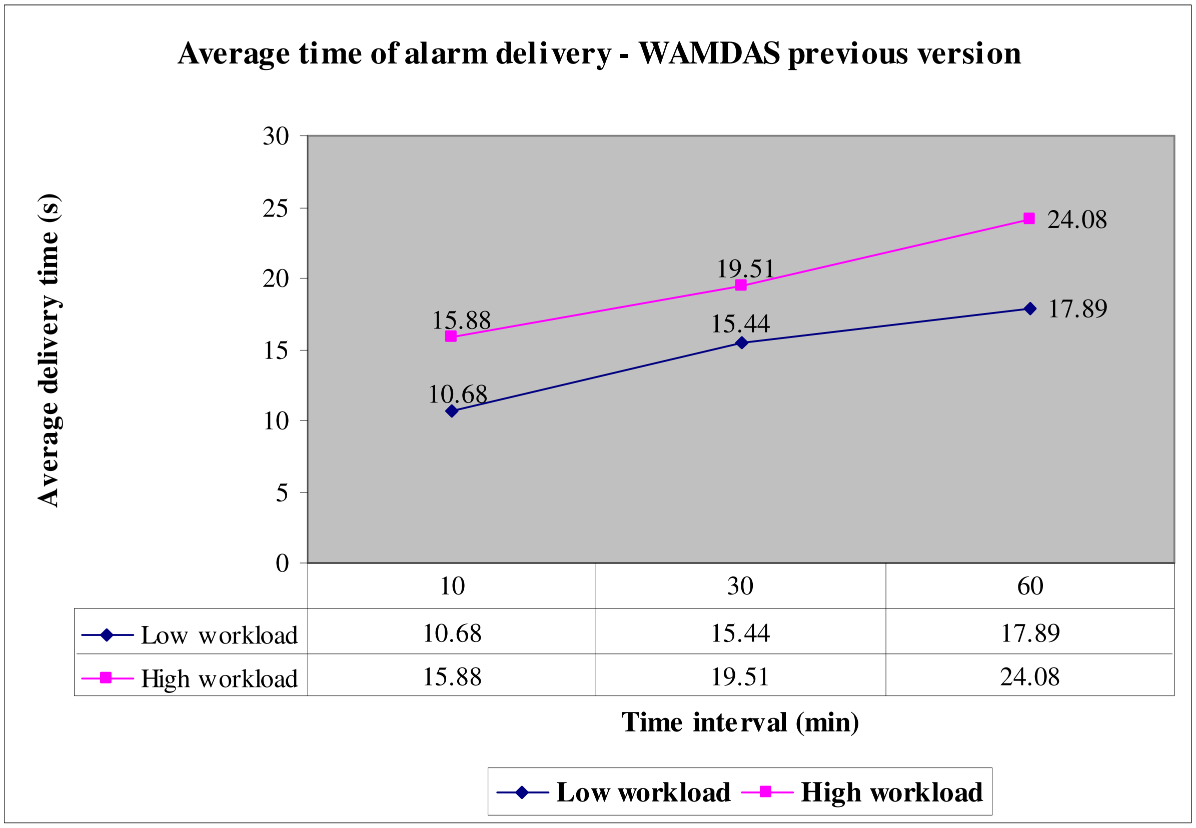
- 10 minutes and at least from 10 to 100 alarms,
- 30 minutes and at least from 30 to 300 alarms, and
- 60 minutes and at least from 65 to 660 alarms.

The first metric we measured was the alarm delivery time, which is the elapsed time since a new alarm event is registered in the system until the alarm message is received by the handheld devices. We measured this metric using the previous version of WAMDAS implemented with TCP/IP sockets and the new version implemented with MSMQ.

For the previous version of WAMDAS, the Table 5.3 shows the average number of alarms generated by the simulator program and the Figure 5.5 shows the average time of alarm delivery, both of them for low and high alarm workloads during the three time intervals cited above.

Table 5.3 Average number of alarms to measure the Alarm Delivery Time - WAMDAS previous version

Average number of alarms generated by the Alarms Simulator program		
Time interval (min)	**Low workload**	**High workload**
10	24	95
30	75	254
60	115	366



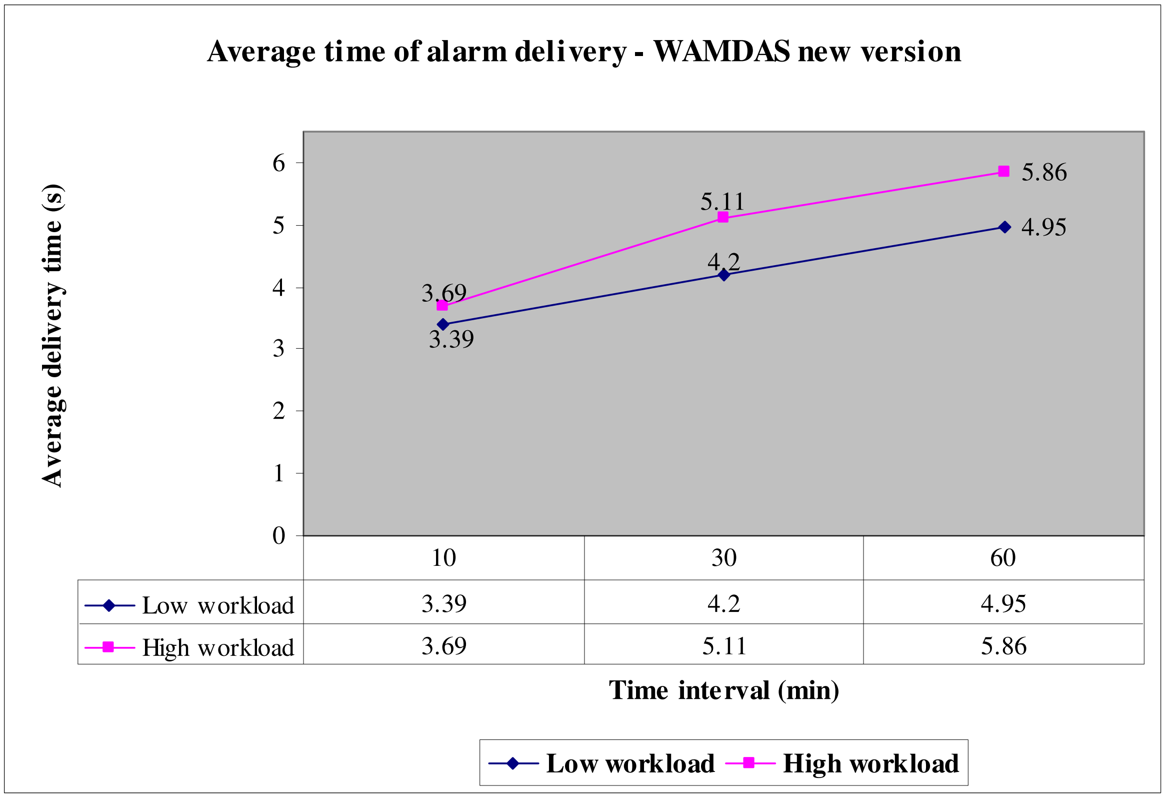
	10	30	60
Low workload	10.68	15.44	17.89
High workload	15.88	19.51	24.08

Figure 5.5 Average time of alarm delivery - WAMDAS previous version

And for the new version of WAMDAS, the Table 5.4 shows the average number of alarms generated by the simulator program and the Figure 5.6 shows the average time of alarm delivery, both of them for low and high alarm workloads also during the three time intervals.

Table 5.4 Average number of alarms to measure the Alarm Delivery Time - WAMDAS new version

Average number of alarms generated by the Alarms Simulator program		
Time interval (min)	**Low workload**	**High workload**
10	38	82
30	78	248
60	176	477



	10	30	60
Low workload	3.39	4.2	4.95
High workload	3.69	5.11	5.86

Figure 5.6 Average time of alarm delivery - WAMDAS new version

As we can see in Figure 5.7, the average times of alarm delivery with the new version of WAMDAS are much lower than the average times of alarm delivery with the previous version. The alarm delivery time was reduced by 73%. Thus, we can state that the alarm delivery time was optimized by our new implementation of the alarm services protocol using MSMQ.

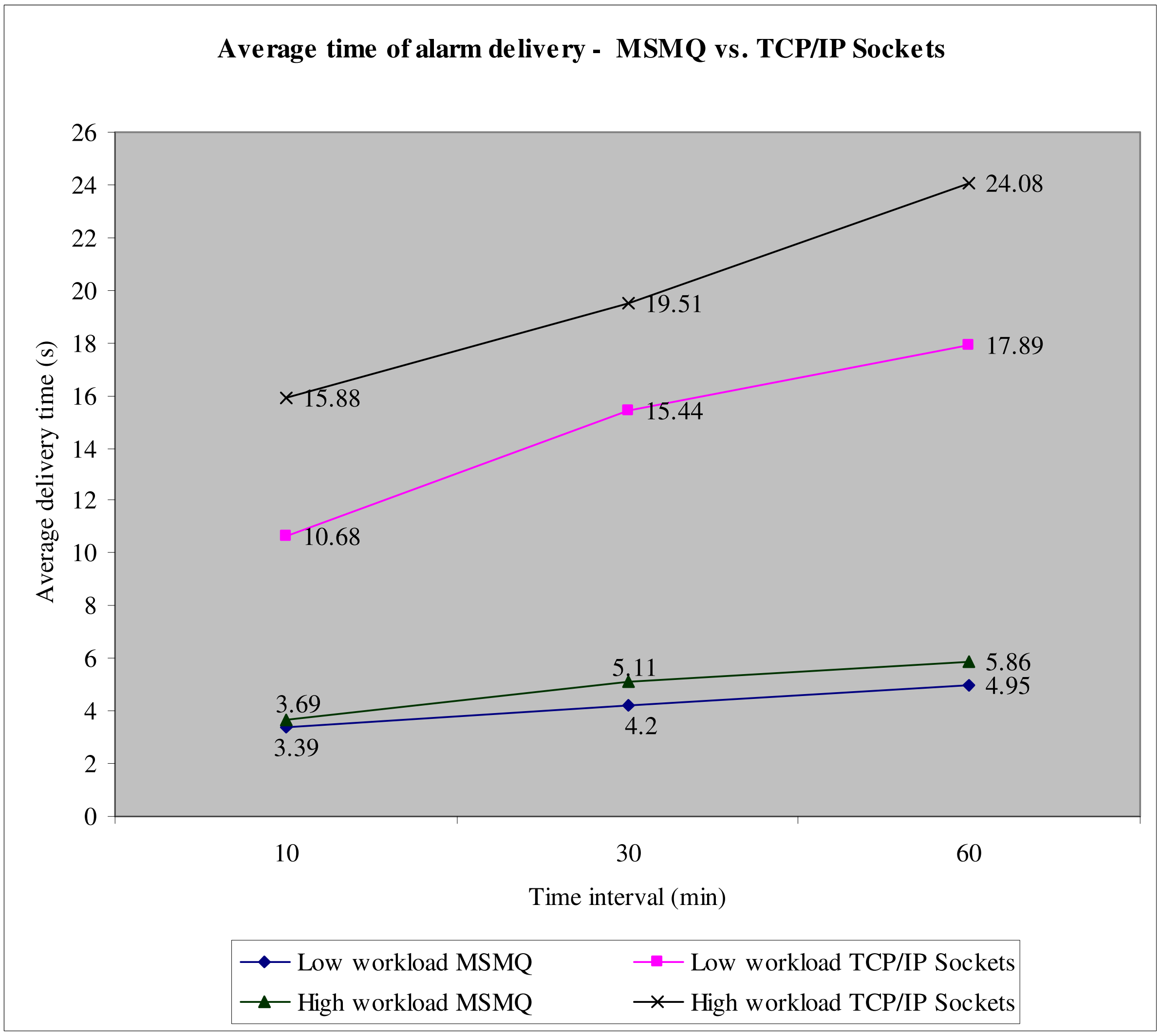


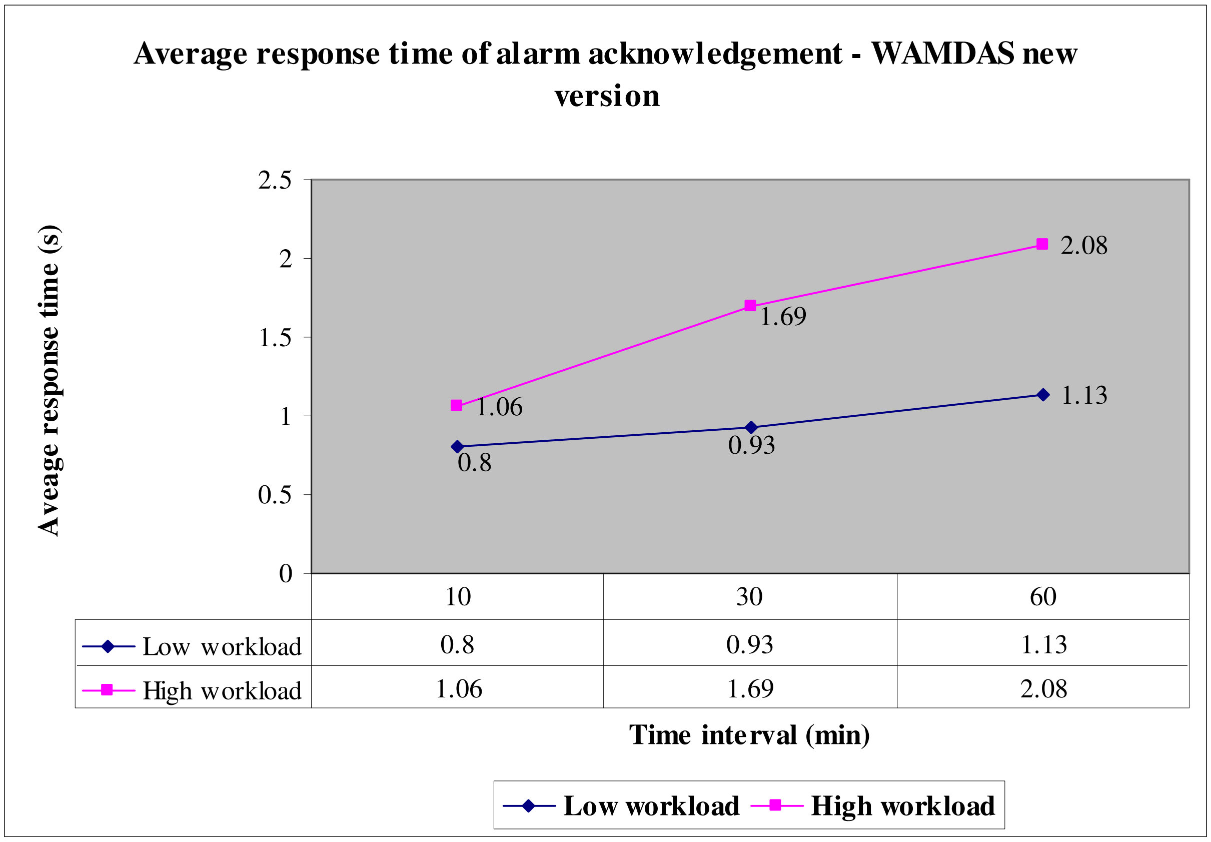
Figure 5.7 Average time of alarm delivery - MSMQ vs. TCP/IP Sockets

The second metric that we measured was the response time of alarm acknowledgement, which is the elapsed time since an alarm is acknowledged until the handheld device receives the alarm acknowledgement result message. This metric was measured using only the new version of WAMDAS implemented with MSMQ, because the previous version does not have this functionality.

The Table 5.5 shows the average number of alarms generated by the simulator program and the Figure 5.8 shows the average response time of alarm acknowledgement, both of them for low and high alarm workloads during the three time intervals (10, 30 and 60 min). As we can see, our new version of WAMDAS performs well with average response time values less than or equal to 2.08 seconds.

Table 5.5 Average number of alarms to measure the Response Time of Alarm Acknowledgement - WAMDAS new version

Average number of alarms generated by the Alarms Simulator program		
Time interval (min)	**Low workload**	**High workload**
10	15	54
30	49	185
60	104	520



	10	30	60
Low workload	0.8	0.93	1.13
High workload	1.06	1.69	2.08

Figure 5.8 Average response time of alarm acknowledgement - WAMDAS new version

The third metric that we measured was the fault tolerance of the alarm services protocol, for this evaluation we simulated fault circumstances like network failures, PDAs going out of wireless network range, and PDAs having their battery discharged. We measured this metric using the previous version of WAMDAS implemented with TCP/IP sockets and the new version implemented with MSMQ.

For the previous version of WAMDAS, the Table 5.6 and the Figure 5.9 show the number of triggered alarms and the number of received alarms, both of them for low and high alarm workloads during the three time intervals (10, 30 and 60 min).

Table 5.6 Fault tolerance of the Alarm Services Protocol – WAMDAS previous version

Fault tolerance of the alarm services protocol – WAMDAS previous version				
Time interval (min)	**Low workload**		**High workload**	
	Number of triggered alarms	Number of received alarms	Number of triggered alarms	Number of received alarms
10	15	4	99	31
30	77	15	252	106
60	149	41	576	217

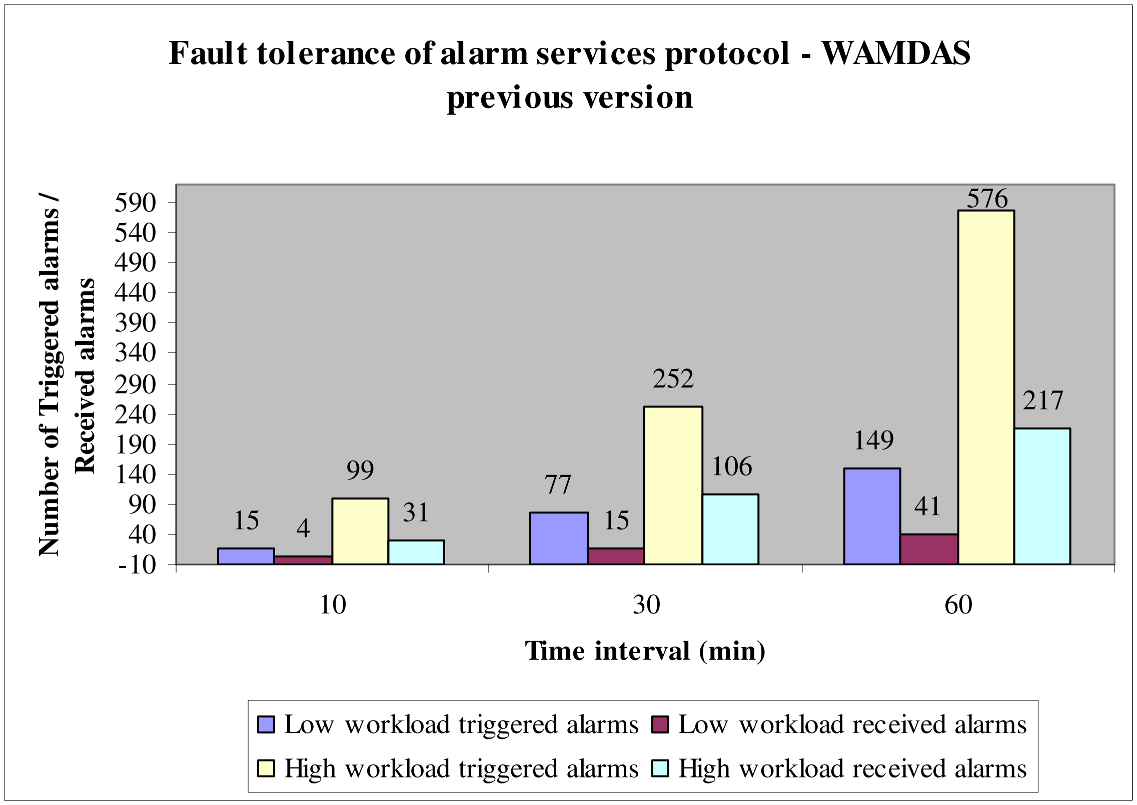


Figure 5.9 Fault tolerance of alarm services protocol - WAMDAS previous version

And for the new version of WAMDAS, the Table 5.7 and the Figure 5.10 show the number of triggered alarms and the number of received alarms, both of them for low and high alarm workloads also during the three time intervals.

Table 5.7 Fault tolerance of the Alarm Services Protocol – WAMDAS new version

Fault tolerance of the alarm services protocol – WAMDAS new version				
Time interval (min)	**Low workload**		**High workload**	
	Number of triggered alarms	Number of received alarms	Number of triggered alarms	Number of received alarms
10	13	13	115	115
30	90	90	222	222
60	138	138	604	604

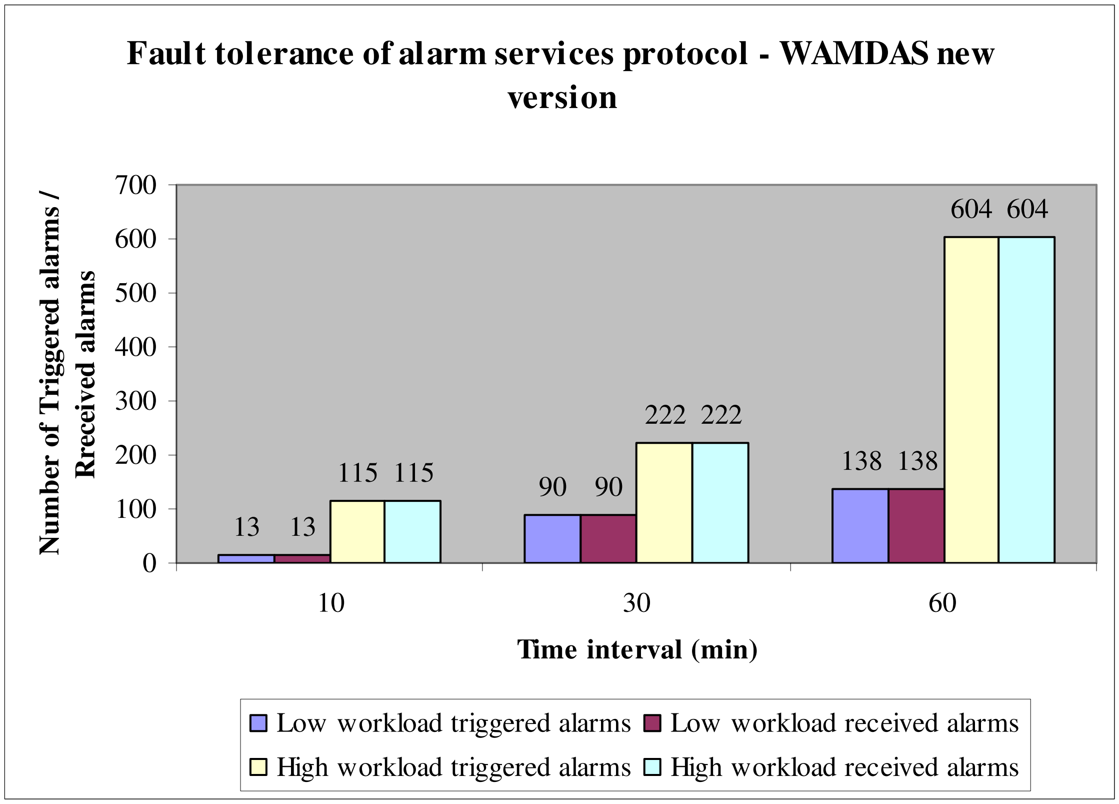


Figure 5.10 Fault tolerance of alarm services protocol - WAMDAS new version

As we can see when we were running the previous version of WAMDAS and the fault circumstances occurred, the alarms were lost; but when running the new version of WAMDAS, even though the fault circumstances occurred, all the alarms were received successfully. Thus, we can state that the alarms services protocol is more reliable and fault tolerant using MSMQ.

5.4 Usability Evaluation

We conducted a usability evaluation to measure the usability of the WAMDAS system in terms of effectiveness and satisfaction from the end-users' point of view. The end-users were real operators working on regular shifts at a pharmaceutical plant. We prepared a task list and a user evaluation questionnaire for both the handheld device module and administrator module of WAMDAS (see Appendix B). First, the users were instructed on how to use the handheld device and administrator modules. And then, the task lists were provided to the users to be completed. After completing the task lists, the users were required to fill out the user evaluation questionnaires.

In the evaluation of the handheld device module, three users participated, one Senior Instrumentation technician, one Instrumentalist and one Operator. On average, they have been working on their positions during 16.7 years. Similarly, they have around 11.7 years of experience using computers. Only one of them has experience using a PDA for about 5 years. And only one of them needs glasses to read. We show in Table 5.8, the average of the punctuation (1 to 7 scale: 1 = not satisfied and 7= very satisfied) that the users gave to the different aspects of the application during the evaluation.

Table 5.8 Evaluation of the Handheld Device Module

Evaluation of the Handheld Device Module	**Punctuation average**
General characteristics of the application	
Size of the letters	7.00
Organization of the information on the screens	7.00
Clarity of the command buttons on the screens	6.00
Handling of the equipment alarms	
Effectiveness of the alarm notification	6.00
Quality of the information received about the alarm	6.67
Handling of multiple alarms	6.67
Effectiveness in the alarm acknowledgement	6.67
Effectiveness in the acknowledgement of multiple alarms	6.33
Quality of the information received about the acknowledgement result	6.67
Handling of the equipment status information	
Effectiveness in receiving the equipment status information	6.33
Effectiveness in receiving the status information of an equipment family	7.00
Quality of the information received about equipment status	6.67
General satisfaction	
With the Handheld Device Module of WAMDAS	6.00
With the actual system (HMIs and SCADAs)	3.33

As we can see in Table 5.8, the punctuation average for all aspects of the handheld device module is greater than or equal to 6 points. And in Figure 5.11, we can see that users preferred WAMDAS by a margin of almost 2 to 1 compared with the actual system. In addition, in the comments section of the questionnaire, the users stated that they would like to use the application because it is a great tool, it would be of big help when they are out of the control room doing other tasks, the application would help them to keep control and monitor the equipment all the time. These results demonstrate a high level of effectiveness and user satisfaction of our handheld device module at the pharmaceutical plant.

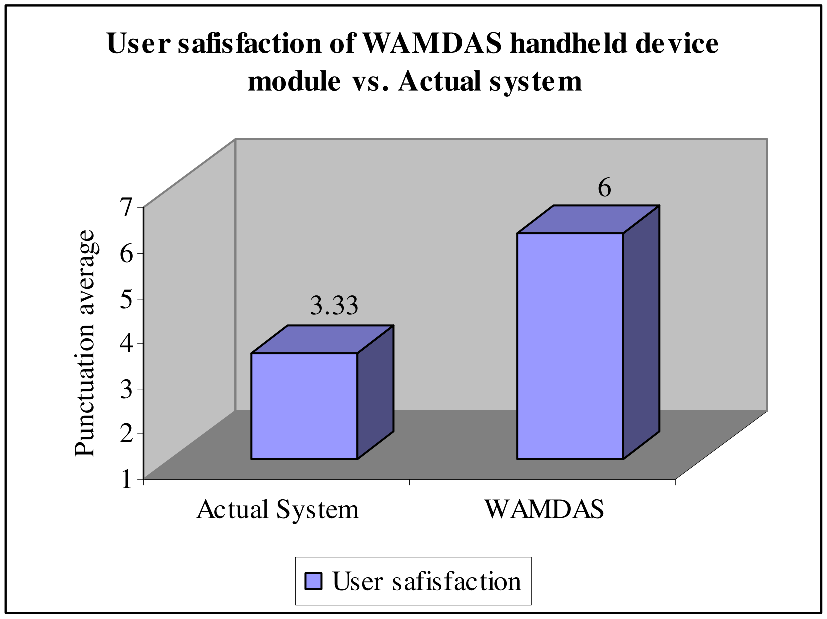


Figure 5.11 User satisfaction of WAMDAS handheld device module vs. Actual system

In the evaluation of the administrator module, two users participated; both of them Process Control Engineers. On average, they have been working on their positions during 7 years. Similarly, they have around 12 years of experience using computers. We show in Table 5.9, the punctuation (1 to 7 scale: 1 = not satisfied and 7= very satisfied) that the users gave to the different aspects of the application during the evaluation.

Table 5.9 Evaluation of the Administrator Module

Evaluation of the Administrator Module	**Punctuation average**
General characteristics of the application	
Organization of the main menu of options	6.00
Organization of the information on the screens	6.00
Clarity of the command buttons on the screens	5.00
Usage of the main menu options	
Effectiveness in searching existing information	7.00
Effectiveness in entering new information	7.00
Effectiveness in updating existing information	7.00
Effectiveness in deleting existing information	5.00
General satisfaction	
With the Administrator Module of WAMDAS	6.00

As we can see in Table 5.9, the punctuation for all aspects of the administrator module is greater than or equal to 5 points, and the user satisfaction punctuation is 6. In addition, in the comments section of the questionnaire, the users stated that the application facilitates very much the administration task of the WAMDAS system. These results demonstrate a high level of effectiveness and user satisfaction of our administrator module at the pharmaceutical plant.

6 CONCLUSIONS AND FUTURE WORK

6.1 Summary and Conclusions

In this thesis, we have presented a novel infrastructure which allows the WAMDAS system to be a more portable, reliable, secure and robust wireless web service-based middleware system. This infrastructure provides a reliable alarm management mechanism to guarantee that alarms are sent to and received by the handheld devices used by operators. Moreover this alarm delivery process obeys a defined configuration scheme for alarms. Likewise our new approach guarantees that alarm acknowledgements are sent to and received by the alarm notification server. And, it also guarantees that acknowledgement results are sent to and received by the handheld devices.

WAMDAS now provides a functionality to have a pool of available web services to respond to the requests on time, even under unexpected circumstances such as extreme workloads and system or network failures.

We presented our Windows-based application module, provided to dynamically configure all system information and parameters, as well as to define the configuration scheme for alarm management.

We discussed the improvements to the handheld device application, including the strong security mechanism to access the system, the mechanism to control system usage and the redesign of the user interfaces.

We conducted a series of experiments on our new version of WAMDAS to test performance and fault tolerance of the system protocols, and also we conducted a usability evaluation at a pharmaceutical plant. The status services protocol evaluation shows that the response time of equipment status query was reduced by 41% when using stored procedures instead of in-Line SQL statements. And the alarm services protocol evaluation shows that when using MSMQ instead of TCP/IP sockets, the alarm delivery time was reduced by 73%, the response time of alarm acknowledgement performs well with average response time values less than or equal to 2.08 seconds, likewise, this protocol is more reliable and fault tolerant. All of these results show that the system protocols were optimized. Moreover, the usability evaluation shows a high level of effectiveness and user satisfaction of our WAMDAS system at the pharmaceutical plant.

Finally, WAMDAS is a widely applicable system; it can be used in pharmaceutical plants and in any other type of manufacturing plants such as electric energy industry, metalworking, telecommunications, textile, construction, transportation, and others. In general, WAMDAS can be used in any environment where is necessary to monitor equipment status information and to manage equipment alarms.

6.2 Future Work

This section presents ideas to improve and extend WAMDAS:

- Implement a web-based version of the administrator module to make it more accessible; allowing system administrators to use the module wherever they have access to a web browser.
- Provide to the status services protocol with fault-tolerance and recovery capabilities against communication loss, which may occur when the network fails, the handheld device is out of wireless network range, or the battery of the handheld device is discharged.
- Extend the alarms services protocol to consider that if an alarm A has not been fully attended during a maximum waiting time; another alarm should be triggered notifying that the alarm A has not been attended yet. Also consider the options of automatic mails and automatic phone calls in order to guarantee that someone hears the alarm.
- Extend the handheld device module in order to be deployed not only in PDAs, but also in Smart Phones and Cell Phones.

REFERENCES

[1] E. García-Rodríguez. "WAMDAS: A Web Service-Based Wireless Alarm Monitoring and Data Acquisition System for Pharmaceutical Plants". Master Thesis, University of Puerto Rico - Mayaguez, 2005.

[2] E. García-Rodríguez and M. Rodriguez-Martínez. WAMDAS: A Web Service-Based Wireless Alarm Monitoring and Data Acquisition System for Pharmaceutical Plants. Advanced International Conference on Telecommunications / International Conference on Internet and Web Applications and Services 2006 (AICT-ICIW). IEEE. 19-25 Feb. 2006.

[3] K. Ballinger. .NET Web Services: Architecture and Implementation First edition. Addison Wesley Professional. 2003.

[4] S. Short. Building XML Web Services for the Microsoft .NET Platform. First edition. Microsoft Press. 2002.

[5] F. Cabrera and C. Kurt. Web Services Architecture and Its Specifications: Essentials for Understanding WS. First edition. Microsoft Press. 2005.

[6] XML Protocol Working Group, Simple Object Access Protocol (SOAP) Version 1.2, W3C Recommendation. www.w3.org/TR/soap/ . Jun 2003

[7] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web Services Description Language (WSDL) Version 1.1. W3C Note. www.w3.org/TR/wsdl.html. Mar 2001

[8] UDDI Spec Technical Committee, UDDI Version 3.0.2.

http://uddi.org/pubs/uddi-v3.0.2-20041019.htm. Oct 2004.

[9] R. Ramakrishnan and J. Gehrke. Database Management Systems. Third edition. McGraw-Hill. 2003.

[10] Codd, E.F. A Relational Model of Data for Large Shared Data Banks. Communications of the ACM. Jun 1970.

[11] A. Silberschatz, H. Korth, and S. Sudarshan. Database System Concepts. Fifth edition. McGraw-Hill. 2006.

[12] JDBC Technology. www.java.sun.com/products/jdbc/ . Dec 2006.

[13] ODCB Basics. www.datadirect.com/developer/odbc/basics/index.ssp. Dec 2006.

[14] S. Sesay, Z. Yang, J. Chen, and D. Xu. A Secure Database Encryption Scheme. Second Consumer Communications and Networking Conference 2005 (CCNC). IEEE. 3-6 Jan 2005.

[15] B. Hartman. Pocket PC: MSMQ for Windows CE Brings Advanced Windows Messaging to Embedded Devices. The Microsoft Journal for Developers. MSDN Magazine. Dec 2001.

[16] MSDN Library. Queues in Windows Communication Foundation. Retrieved February 4, 2008, from Microsoft Corporation Web site: http://msdn2.microsoft.com/en-us/library/ms731089.aspx

[17] C. Carpentiere. An Evaluation of Stored Procedures for the .NET Developer. Microsoft .NET Development Technical Articles. MSDN Library. Mar 2004.

[18] D. Rothaus and M. Pizzo. ADO.NET for the ADO Programmer. .NET Development Technical Articles. MSDN Library. Dec 2001.

[19] MSDN Library. ADO.NET. .NET Framework Developer's Guide. Retrieved May 5, 2008, from Microsoft Corporation Web site:
http://msdn.microsoft.com/en-us/library/e80y5yhx(vs.80).aspx

[20] Wikipedia, the free encyclopedia. Database trigger. Retrieved May 7, 2008, from Wikimedia Foundation, Inc web site:
http://en.wikipedia.org/wiki/Database_trigger

[21] A. Redkar, C. Walzer, S. Boyd, R. Costall, K. Rabold, and T. Redkar. Pro MSMQ: Microsoft Message Queue Programming. First edition. Apress. 2004.

[22] D. Makofse, M. Donahoo, and K. Calvert. TCP/IP Sockets in C#: Practical Guide for programmers. First edition. Morgan Kaufmann. 2004.

[23] R. Steele. A Web Services-based system for ad-hoc mobile application integration. International Conference on Information Technology: Computers and Communications (ITCC). IEEE. 28-30 Apr 2003.

[24] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom. The TSIMMIS Project: Integration of Heterogeneous Information Sources. IPSJ Conference.1994.

[25] M. Rodriguez-Martinez, and N. Roussopoulos. MOCHA: A Self-Extensible Middleware System for Distributed Data Sources. 2000 ACM SIGMOD International Conference on Management of Data. ACM. May 2000.

[26] M. Rodriguez-Martinez. Automatic Deployment of Application-Specific Functionality in Database Middleware Systems. Department of Computer Science, University of Maryland, College Park. 2001.

[27] M. Rodríguez-Martínez, JF. Enseñat, E. Pagan, JG. Arzola, M. Sotomayor, and E. Vargas. Registration and Discovery of Services in the NetTraveler Integration System for Mobile Devices. International Conference on Information Technology: Coding and Computing (ITCC). IEEE. 2004.

APPENDIX A. MSMQ IMPLEMENTATION

APPENDIX A1. SERVER SIDE IMPLEMENTATION

```
namespace DBRService
{
    public class DBRService : System.Web.Services.WebService
    {
        .....
        //Queue name to receive the acknowledgement messages from the handheld devices
        private const string QueueName = @".\private$\mqServWamdasAkn";
        //Message queue to receive the acknowledgment messages from the handheld devices
        private MessageQueue mqAlarmsAkn;

        public DBRService()
        {
            .....
            //Create the message queue where acknowledgement messages will be stored
            if (!MessageQueue.Exists(QueueName))
            {
                mqAlarmsAkn = MessageQueue.Create(QueueName);
            }
            else
            {
                mqAlarmsAkn = new MessageQueue(QueueName);
            }
            //Set the formatter to indicate that message body will be of String type
            mqAlarmsAkn.Formatter = new XmlMessageFormatter(new Type[] { typeof(String) });
            //Initiate the asynchronous reception operation by telling the message queue to begin
            //receiving messages and notifies the event handler when finished
            mqAlarmsAkn.ReceiveCompleted += new
                ReceiveCompletedEventHandler(QueueReceiveCompleted);
            mqAlarmsAkn.BeginReceive();
        }


        //Web method that processes and sends to the handheld devices the alarm messages
        //forwarded by the ANS web service
        public string ProcessAlarmMessage(int AlarmEventID)
        {
            …..
            //Build the alarm message body ‘alarmMessageBody’
            …..
            //Retrieve the IP addresses of handheld devices to which the alarm will be sent
            …..
            //Send the alarm to the handheld devices
            int i = 0;
            while (i < ClientsNumberToSendAlarm)
            {
                .....
                //Create an alarms queue instance of the handheld device with IP = ClientIP
                MessageQueue deviceQueue = new MessageQueue (String.Format
                    (System.Globalization.CultureInfo.InvariantCulture,@"FORMATNAME:
                    DIRECT=TCP:{0}\private$\mqDevWamdasAlarm", ClientIP));
                //Create the alarm message to be sent
                Message alarmMessage = new Message (alarmMessageBody);
                alarmMessage.Recoverable = true;
                //Set the alarm message response queue to the server queue
                alarmMessage.ResponseQueue = new MessageQueue (String.Format
                    (System.Globalization.CultureInfo.InvariantCulture,@"FORMATNAME:
                    DIRECT=TCP:{0}\private$\mqServWamdasAkn", ServerIP);
                // Send the Message to the handheld device queue
                deviceQueue.Send(alarmMessage);
                i++;
            }
            .....
        }

        //Method that processes the completion of the ReceiveCompletedEventHandler
        private void QueueReceiveCompleted(Object source, ReceiveCompletedEventArgs
            asyncResult)
        {
            .....
            //Pick up the received acknowledgement message from the server queue
            Message alarmAknMessage =this.mqAlarmsAkn.EndReceive(asyncResult.AsyncResult);
            string alarmAknMessageBody = (string) alarmAknMessage.Body;
            //Process the alarm acknowledgement message body and build the body of the alarm
            //acknowledgement result message 'alarmAknResultMessageBody'
            .....
            //Create a new message for the alarm acknowledgement result
            Message alarmAknResultMessage = new Message (alarmAknResultMessageBody);
            //Send the alarm acknowledgement result message to the hand held device queue
            alarmAknMessage.ResponseQueue.Send(alarmAknResultMessage);
            //Begin the next asynchronous reception operation
            this.mqAlarmsAkn.BeginReceive();
            .....
        }
        .....
    }
}
```

APPENDIX A2. CLIENT SIDE IMPLEMENTATION

```
namespace WAMDAS_Cogen_Client
{
    public partial class Form1 : Form
    {
        …..
        //Variable for the instance of the DBR web service
        DBRService wsDBR;
        //Queue name to receive the alarm messages from the server
        private const string AlarmQueueName = @".\private$\mqDevWamdasAlarm";
        //Queue name to receive the alarm acknowledgement result messages from the server
        private const string AlarmAknQueueName = @".\private$\mqDevWamdasAkn";
        //Message queue to receive the alarm messages from the server
        private MessageQueue mqAlarms;
        //Message queue to receive the alarm acknowledgement result messages from the server
        private MessageQueue mqAlarmsAkn;
        //Thread to peek at the alarm messages
        private Thread peekAlarmsThread;
        //Thread to peek at the alarm acknowledgement result messages
        private Thread peekAlarmsAknThread;
        …..

        //Method that starts the device queues
        private void StartDeviceQueues()
        {
            …..
            //Create the message queue where alarm messages will be stored
            if (!MessageQueue.Exists(AlarmQueueName))
            {
                mqAlarms = MessageQueue.Create(AlarmQueueName);
            }
            else
            {
                mqAlarms = new MessageQueue(AlarmQueueName);
            }
            //Create message queue where alarm acknowledgement result messages will be stored
            if (!MessageQueue.Exists(AlarmAknQueueName))
            {
                mqAlarmsAkn = MessageQueue.Create(AlarmAknQueueName);
            }
            else
            {
                mqAlarmsAkn = new MessageQueue(AlarmAknQueueName);
            }
            //Set the formatter to indicate that alarm message body will be of String type
            mqAlarms.Formatter = new XmlMessageFormatter(new Type[] { typeof(String) });
    //Set the formatter to indicate that alarm acknowledgement result message body will be of
    //String type
    mqAlarmsAkn.Formatter = new XmlMessageFormatter(new Type[] { typeof(String) });
    …..
}

//Method that starts the threads which peek the messages from the queues
private void StartThreads()
{
    …..
    //Create and start the thread which peeks at messages from the alarm messages queue
    peekAlarmsThread = new Thread(new ThreadStart(PeekAlarms));
    peekAlarmsThread.IsBackground = true;
    peekAlarmsThread.Start();
    //Create and start the thread which peeks at messages from the alarm acknowledgement
    //result messages queue
    peekAlarmsAknThread = new Thread(new ThreadStart(PeekAlarmsAkn));
    peekAlarmsAknThread.IsBackground = true;
    peekAlarmsAknThread.Start();
    …..
}

/Load event of the main window for alarm handling
private void Form1_Load(object sender, EventArgs e)
{
    …..
    //Create a new instance of the DBR web service
    wsDBR = new DBRService();
    //Start the device queues
    StartDeviceQueues();
    //Start the threads to peek at the messages from the queues
    StartThreads();
    …..
}

//Method that peeks at messages from the queue of alarm messages
protected void PeekAlarms()
{
    //Create a dynamic list of all the messages in the queue
    MessageEnumerator messageEnum = mqAlarms.GetMessageEnumerator();
    //Advance the enumerator to the next message in the queue. If the enumerator is
    //positioned at the end of the queue, MoveNext waits until a message is available or the
    //given timeout expires.
    while (messageEnum.MoveNext(new TimeSpan(168, 0, 0)))
    {
      //Show the alarm message to the user
      ShowAlarmMessage((String)messageEnum.Current.Body, messageEnum.Current.Id);
    }
    …..
}

//Method that peeks at messages from the queue of alarm acknowledgement result
//messages
protected void PeekAlarmsAkn()
{
    //Create a dynamic list of all the messages in the queue
    MessageEnumerator aknMessageEnum = mqAlarmsAkn.GetMessageEnumerator();
    //Advance the enumerator to the next message in the queue. If the enumerator is
    //positioned at the end of the queue, MoveNext waits until a message is available or the
    //given timeout expires.
    while (aknMessageEnum.MoveNext(new TimeSpan(168, 0, 0)))
    {
        //Show the alarm message  to the user
        ShowAlarmAknMessage((String)aknMessageEnum.Current.Body,
            aknMessageEnum.Current.Id);
    }
    …..
}

//Method that shows the alarm message body to the user, and then it removes the alarm
// message from the queue
protected void ShowAlarmMessage(String AlarmMessageBody, String QueueMessageId)
{
    …..
    //Verify if the number of acknowledgements for the alarm is completed
    if (wsDBR.AlarmAcknowledgementCompleted(AlarmEventID))
    {
        //Remove from the queue the message with ID equal to the received parameter
        mqAlarms.ReceiveById(QueueMessageId);
    }
    else
    {
         …..
        //Register in the system that the operator device received the alarm message
        wsDBR.RegisterReceivedAlarm(OperatorDeviceID, AlarmEventID,
            ReceivedAlarmDateTime);
        //Show the alarm message body to the user, which is the AlarmMessageBody value
        //received as parameter
        …..
    }
    …..
}
```

```
        //Method that shows the alarm acknowledgement result message body to the user and then
        //it removes the alarm acknowledgement result message from the queue
        protected void ShowAlarmAknMessage(String AlarmAknResultMessageBody, String
            QueueMessageId)
        {
            …..
            //Show the alarm acknowledgement result message body to the user, which is the
            //AlarmAknResultMessageBody value received as parameter
            …..
            //Remove from the queue the message with ID equal to the received parameter
            mqAlarmsAkn.ReceiveById(QueueMessageId);
            …..
        }

        //Method that processes the alarm acknowledgement after the user clicked the "Akn" or
        //"Akn All" button in the Alarms Handling Main Window (See Figure 4.23)
        public void AcknowledgeAlarm (String QueueAlarmMessageId, int AlarmEventID,
            DateTime AlarmAknTime)
        {
            …..
            //Pick up and remove the acknowledged alarm message from the alarm messages queue
            Message alarmMessage = mqAlarms.ReceiveById(QueueAlarmMessageId);
            //Build the alarm acknowledgement message body 'alarmAknMessageBody'
            …..
            //Create the acknowledgement message to be sent to the server queue
            Message alarmAknMessage = new Message (alarmAknMessageBody);
            alarmAknMessage.Recoverable = true;
            //Set the alarm acknowledgement message response queue to the device queue for alarm
            //acknowledgement result messages
            alarmAknMessage.ResponseQueue = new MessageQueue (String.Format
                (System.Globalization.CultureInfo.InvariantCulture, @"FORMATNAME:DIRECT=
                TCP:{0}\private$\mqDevWamdasAkn", HandheldDeviceIP));
            //Send the alarm acknowledgement message to the server queue
            alarmMessage.ResponseQueue.Send(alarmAknMessage);
            …..
        }
        …..
    }
}
```

APPENDIX B. USABILITY EVALUATION

APPENDIX B1. TASK LISTS

Lista de Tareas - WAMDAS Handheld Device Module

1. Hacer una consulta de información de status de Generator del equipo Turb_A para la fecha 05-15-2008.

2. Hacer una consulta de información de status de RO del equipo Turb_A para la fecha 05-15-2008.

3. Hacer una consulta de información de status de Engine/Water/FuelRatio del equipo Turb_B para la fecha 05-15-2008.

4. Hacer una consulta de información de status de Generator del equipo Turb_C para la fecha 05-15-2008.

5. Hacer una consulta de información de status de RO para la fecha 05-15-2008.

6. Hacer una consulta de información de status de todas las tubinas para la fecha 05-15-2008

7. Hacer Acknowledgement a una alarma recibida

8. Hacer Acknowledgement a todas las alarmas recibidas

9. Consultar la informacion histórica de alarmas que se hicieron Acknowledgements la fecha 05-15-2008.

<u>Lista de Tareas - WAMDAS Administrator Module</u>

1. Elegir la opción “Alarm Management” del menu principal, e ingresar un nuevo data source, components y alarms. Y probar las opciones de modificar y borrar.

2. Elegir la opción “Alarm's Mapping” del menu principal, y registrar una nueva alarma real definiendo su mapeo a una alarm creada en el paso anterior (hacer uso de la funcionalidad de busqueda para encontrar la alarma de WAMDAS).

3. Elegir la opción “Severity” del menu principal, y definir una nueva severidad, probar las opciones de modificar y borrar.

4. Elegir la opción “User - Alarm” del menu principal, y asignar a un usuario un data source existente (hacer uso de la funcionalidad de busqueda), para que reciba las alarms de ese data source.

5. Elegir la opción “Handheld Devices” del menu principal, y registrar en el sistema un nuevo handheld device, probar las opciones de modificar y borrar.

6. Elegir la opción “Parameters” del menu principal, y registrar en el sistema un nuevo parámetro, probar las opciones de modificar y borrar.

7. Elegir la opción “Web Services” del menu principal, y registrar en el sistema un nuevo web service, probar las opciones de modificar y borrar.

APPENDIX B2. USER EVALUATION QUESTIONNAIRES

<u>Evaluación de WAMDAS Handheld Device Module</u>

Por favor conteste las siguientes preguntas:

1. Puesto que ocupa:_____________________________________

2. Tiempo que lleva en el puesto indicado: _____años.

3. Tiene experiencia utilizando computadoras: __ Si ___ No

4. Si contestó SI a la pregunta anterior indique cuantos años de experiencia tiene utilizando computadoras: ____ años.

5. ¿Tiene experiencia o ha interactuado con computadoras portátiles inalámbricas (PDAs)? ___Si ___No

6. Si contestó SI a la pregunta anterior indique cuantos años de experiencia tiene utilizando computadoras portátiles inalámbricas (PDAs): ____ años.

7. Necesita lentes para leer? ___ Si ___No

Seleccione con un círculo el número que represente su nivel de satisfacción con los siguientes aspectos (1 significa no satisfactorio, 7 significa muy satisfactorio):

Características generales de la aplicación	
Tamaños de letras	1 2 3 4 5 6 7
Organización de la información en las pantallas	1 2 3 4 5 6 7
Claridad de los botones de comando provistos en las pantallas	1 2 3 4 5 6 7
Manejo de alarmas de equipos	
Efectividad del aviso al recibir una alarma	1 2 3 4 5 6 7
Calidad de la información sobre la alarma	1 2 3 4 5 6 7
Manejo de múltiples alarmas	1 2 3 4 5 6 7
Efectividad en el “Acknowledgement” de una alarma	1 2 3 4 5 6 7

Efectividad en el "Acknowledgement" de múltiples alarmas	1 2 3 4 5 6 7
Calidad de la información sobre el resultado del "Acknowledgement"	1 2 3 4 5 6 7
Manejo de información de estatus de equipos	
Efectividad en recibir información de estatus de un equipo	1 2 3 4 5 6 7
Efectividad en recibir información de estatus de varios equipos	1 2 3 4 5 6 7
Calidad de la información recibida sobre el estatus del(los) equipo(s).	1 2 3 4 5 6 7
Satisfacción general	
Con el Handheld Device Module de WAMDAS	1 2 3 4 5 6 7
Con el sistema existente	1 2 3 4 5 6 7

Si tiene algún comentario u observación sobre la aplicación, por favor expréselo en el siguiente espacio:

____

____

____

Evaluación de WAMDAS Administrator Module

Por favor conteste las siguientes preguntas:

1. Puesto que ocupa:____

2. Tiempo que lleva en el puesto indicado: _____años.

3. Tiene experiencia utilizando computadoras: __ Si ___ No

4. Si contestó SI a la pregunta anterior indique cuantos años de experiencia tiene utilizando computadoras: ____ años.

Seleccione con un círculo el número que represente su nivel de satisfacción con los siguientes aspectos (1 significa no satisfactorio, 7 significa muy satisfactorio):

Características generales de la aplicación	
Organización del menu principal de opciones	1 2 3 4 5 6 7
Organización de la información en las pantallas	1 2 3 4 5 6 7
Claridad de los botones de comando provistos en las pantallas	1 2 3 4 5 6 7
Uso de las opciones del menu	
Efectividad al buscar información existente	1 2 3 4 5 6 7
Efectividad al ingresar nueva información	1 2 3 4 5 6 7
Efectividad al actualizar información existente	1 2 3 4 5 6 7
Efectividad al borrar información existente	1 2 3 4 5 6 7
Satisfacción general	
Con el Administrator Module de WAMDAS	1 2 3 4 5 6 7

Si tiene algún comentario u observación sobre la aplicación, por favor expréselo en el siguiente espacio:

____

____

____