AN ICT PLATFORM FOR THE VERTICAL AND HORIZONTAL INTEGRATION OF INFORMATION IN LARGE UTILITIES PLANTS

Salvatore Cavalieri

University of Catania, Faculty of Engineering, Dept.Computer and Telecommunications Engineering
Viale A.Doria 6, 95125, Catania, Italy

Abstract: One of the main limits of the current technology for the management of large plant is the lack of Vertical and/or Horizontal Integration of Information. Vertical integration refers to the need to integrate systems implemented at different administrative level of an organisation. Horizontal integration allows the user of large plant to execute very complex transformations on data coming from several real applications, also of different types. The Mobicossum project, IST 1999-57455, is a CRAFT project which aims to realise both Vertical and Horizontal Integrations, providing for a set of high level services useful for the management of large plant. This goal has been realised through the definition of a Middleware between user (including mobile workers using mobile devices) and lower level applications.

Keywords: Plants, Management Systems, Information Integration, Enterprise Integration, Man/Machine Interfaces, Networking

1. INTRODUCTION

One of the main limits in the current technology for the management of large plant is the lack of Integration of Information, both Vertical and Horizontal one (Wangler and Paheerathan, 2000).

Vertical integration refers to the need to integrate systems implemented at different administrative level of an organisation. As an example, in manufacturing industry there is the presence, at the lowest level, of process control systems and computerised Numerical Control machinery adopting proprietary formats of data and messages. Frequently they are based on different operating systems and use different networking technologies. These systems need to be fed with control data stemming from higher level planning and scheduling systems while the lower level applications need to collect data and pass them upwards (Wangler and Paheerathan, 2000). In the management of plants, the lower level applications are Supervisory Control and Data Acquisition System - SCADA, Decision Support System - DSS, and Geographic Information System - GIS. These systems collect and maintain information coming from the field devices present in the plant (in a plant for water distribution, examples of field devices are pumps and valves). The higher level of the monitoring and control organisation relevant to the management of a plant is made up by applications performing useful management functions basing all the activities on the information maintained by a SCADA, GIS or DSS. A software tool managing alarms (e.g. alerting the proper team of technicians when an alarm occurs), is an example of high level application which interacts with a SCADA system from which it receives information about the kind of alarm and the features of the malfunctioning devices. Vertical integration implies integration of information transmitted along the vertical direction from high level applications towards low level applications (e.g. SCADA, GIS and DSS) and vice versa; the integration must be platform-, data representation- and data mining- independent.

Horizontal integration improves the concept of integration, allowing the user of large plant execution of very complex transformations on data coming not only from one application, but relevant to several real applications, also of different types. A typical example of horizontal integration is Supply Chain Management, in which an organisation tries to

optimise the complete set of activities of order entry, purchasing, production, shipment, etc., in order to minimise the lead-time and costs for production, at the same time maximising value for the customer (Wangler and Paheerathan, 2000). In the plant management, a typical example of horizontal integration is the realisation of a SCADA, which acquires information coming from other SCADAs placed inside the plant. In this case, Horizontal integration is required as the different SCADAs run on different platform, are featured by different set of services and commands and use different data representation and mining.

Vertical and Horizontal integration is often realised in automatic fashion by the definition of particular algorithms, known as Business Logics. They are made up by a set of more or less complex functions, concerning transformation of data produced and/or maintained by different applications inside the plant.

Vertical and Horizontal integration generally features the definition of the same (graphical) interface to the final user, nevertheless which lower level applications are present. Definition of this interface has the advantage to avoid to the user the need to be trained to use different tools and interfaces. So, coupling Vertical/Horizontal integration with the definition of a common interface to the lower level applications, has the direct advantages to reduce the training period for the workers, to optimise their performances and to minimise their response time to critical events (e.g. faults).

The Mobicossum project, IST 1999-57455, is a CRAFT project (V European Framework Programme) involving medium and small European enterprises working in the field of gas and water distribution and wastewater treatment. It started on September 2002 and has been concluded in December 2004. One of the RTD performers inside the project is the Department of Computer and Telecommunications Engineering, to which the author of the paper belongs. The main goal of the Mobicossum project was realization of both Vertical and Horizontal Integrations in large utilities plant in the area of interest of the partners, mainly water and gas distribution and wastewater systems. Both integrations have been realised through the definition of a set of high level services useful for the management of the plant. All these services have been grouped inside a Middleware between (mobile) worker and the lower level applications.

Literature presents many other approaches aiming to define middleware to integrate information in industrial environment. Among them, the most recent ones propose technologies like Java and/or CORBA; see for example (Martì et al., 1999) (Martì et al., 2000). The main innovation introduced by this paper to the existing approaches, is the use of a technology based on Web Services (Newcomer, 2002). This technology is platform-independent, like Java and CORBA; but unlike Java and CORBA, many applications like SCADAs, DSSs currently offer interfaces based on it. For this reason, the author believes that an easier integration may be achieved adopting Web Service technology.

The paper will focus on the description of the Middleware, highlighting the software technologies adopted. The description will allow to point out how the Vertical and Horizontal integration has been realised, taking into account the management of large plant.

2. IST 1999-57455 "MOBICOSSUM" PROJECT

Mobicossum aims to define a Middleware conceived to offer specialised services for the management of large plant, like water/gas distribution and wastewater treatment systems. Middleware is placed between users and the applications providing for information related to the plant (e.g. SCADA, GIS and DSS). One of the typical user profiles considered in the project is a mobile worker inside the large plant, equipped with PalmPC, connected to the Control Station by GSM/GPRS/Wi-Fi; the tasks carried on by the mobile worker may range from the recovery of a failure in some device or part of the plant, to the installation/removal of devices. Another typical Mobicossum user profile is a supervisor who is in charge to schedule the activities of the crews of mobile workers.

Mobicossum is made up by different subsystems: Presentation Manager (PM), Central Services (CS), Logical View (LV), Data Management (DM) and the Generalised Interface (GI). Figure 1 shows the Mobicossum internal architecture.

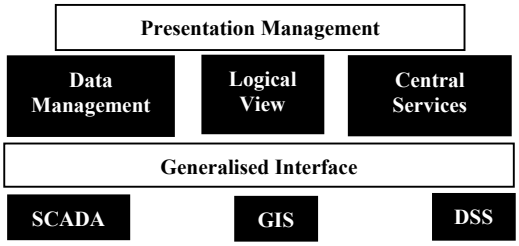

Fig. 1. Mobicossum Middleware Architecture.

The PM generates the pages presented to the users through his mobile device, as a function of the user name, his location, the used presentation device (Pocket PC, WAP phone, etc.), the application status, etc. These pages are needed by the user to pass and/or to receive information to/from the Control Room.

The CS offers several services, among which secure user access logging. Furthermore, when a particular user accesses the system, it is required that the middleware identifies its profile (e.g. technician, manager, and so on). In this way, the Mobicossum

system may be automatically aware of the data needed by the user, avoiding the need to explicitly request the data desired. Other Central Services are those concerning localisation of each mobile worker, which is a very important requirement in large plant.

The DM is in charge of providing data brokerage. It collects data from the different application connected to Mobicossum. The data can be retrieved according logical names and/or geo-references.

The LV realises both the Horizontal and the Vertical Integrations described in the introduction.

The GI is the core of the Middleware. This subsystem directly interfaces to the SCADA, GIS and DSS applications. The main aim of the GI is that to offer a unique set of services to access data maintained by SCADA, GIS and DSS applications and unique way to access these data. This allows decoupling the high level applications with the real SCADA, GIS and DSS ones.

The paper will focus on the description of the LV and GI as they are the two modules mainly involved in the realisation of the Vertical and Horizontal Integration. In fact, the GI provides for a common interface to every low level applications integrated into Mobicossum environment. The LV uses this common interface to allow the user to run specific Business Logic realising the Vertical and/or Horizontal Integration. It's very important to point out that the Business Logics are not strictly linked to a specific application (i.e. vendor) as they are written using the common interface offered by the GI.

3. GENERALISED INTERFACE

Low level applications in a plant (e.g. SCADA, GIS and DSS) offer specialised services, whose meaning and scope is common to the most part of the products available in the market. If the meaning and scope of these services is the same, their implementation is strictly linked to the particular application. Considering, for example, a SCADA application, it's always be possible to find groups of services conceived to retrieve and update information from/to field devices, but the names, the relevant syntax, the number of parameters, the error codes returned by the services are totally different changing from a SCADA application to another one.

Presence of different set of services for the different low level applications makes impossible (vertical and/or horizontal) integration of the information flow from/to each of these applications. For this reason, the main aim in the definition of the Generalised Interface was that to define a unique set of services for each SCADA, GIS and DSS application. Each service defined in the GI is (one-to-one or one-to-many) mapped into the real corresponding service(s) offered by the particular SCADA, GIS and DSS application.

The definition of the common services for the GI has been based on a study carried on during the first months of the Mobicossum project. This study pointed out the user requirements, in the sense of services real needed by users in large plant for its management, focusing on the field of water/gas distribution and wastewater treatment systems, which are the three areas of interest of the Mobicossum partners.

A great problem strongly felt when accessing to different SCADA, GIS and DSS applications, is the different mechanisms used to achieve the access. Different solutions, like COM, DCOM, OPC, Web Services, currently exist and development of a client application often means including different access schema according to the SCADA, GIS, DSS application the client application needs to access. Accessing to a new SCADA, GIS, DSS application implies to add new accessing schema for the specific application. Again it's clear that (Vertical and/or Horizontal) Integration could be achieved only if a common access schema has been defined.

Current literature presents several approaches for definition of communications between distributed applications, but that based on Web Services technology (Newcomer, 2002) features a lot of advantages, as pointed out in the introduction.

For this reason, development of the access schema offered by the GI was mainly based on Web Service technology. In particular, it was assumed that the GI exports the services defined trying to generalise the services generally offered by SCADA, GIS and DSS applications available on the market (as said before), using the Web Service technologies. Web Service technology has been used also for the exchange of information between GI and the SCADA, GIS and DSS applications.

XML/SOAP based communications (Newcomer, 2002) has been assumed to send/receive request/response to/from the Generalised Interface and to realise the exchange of information between the Generalised Interface and the Web Services related to each application.

As said before, integration of existing applications may occur if each SCADA, GIS and DSS application provides for Web Services-based access. Further, Mobicossum Middleware, and in particular the GI, has to acquire a complete know-how about the applications below it and their available Web Services, including all the Web Methods offered. This is realised through the definition of a *Web Service Specification* for each application. A *Web Service specification* is a XML-Schema (Walmsley, 2001) file describing both Web Methods offered by

Web Services related to each SCADA, GIS and DSS application, and the format of the XML/SOAP messages to be exchanged with these Web Services. According to the Web Service-based philosophy, also the Generalised Interface has to provide for a XML-Schema *Generalised Interface Specification*, describing the generalised services offered by the GI.

On the basis of the Generalised Interface Specification and the Web Service Specification related to each application registered inside Mobicossum, two mapping files must be defined for each application; they will be used by a particular component of the GI (the Mapper) to map the request/response written according to Generalised Interface Specification, into request/response written according to Web Service specification of the specific application. It was assumed to realise the mapping of XML/SOAP document using XSLT language (Tennison, 2002).

A key point in the definition of a Middleware like that relevant to Mobicossum, is that to reduce as much as possible the effort to integrate existing SCADA/GIS/DSS applications, otherwise real use of such a Middleware is very improbable. On the basis of what said until now, the main requirement for the integration of existing application into Mobicossum is that these applications export their own functionalities through Web Services. This seems a constraint not so hard, as Web Services technology is currently adopted by many SCADA/GIS and DSS vendors, as said in the introduction. In any case, realisation of wrappers translating existing interfaces into Web Services-based Methods is feasible with a very low effort.

3.1. Internal Architecture of the GI

The GI is made up by the following internal components: Parser, Mapper and Dispatcher. The Parser receives the XML\SOAP request coming from the client of the GI (i.e. the upper modules in Mobicossum Middleware, according to Figure 1) and checks if the XML document is valid. For the document validation, the Parser verifies that the XML/SOAP message is well formed and has been prepared according to the *Generalised Interface Specification* describing the correct structure of the incoming requests, as said before. If the request doesn't match, an error message will be returned; on the other hand, if the document is valid, the request will be sent to the Mapper module. For each response coming from the application, the Parser will receive a document from the Mapper component (described in the following). In this case the Parser will check that the format of the document has been prepared according to the XML-Schema for the response, passing the valid document to the client.

The Mapper performs the translation of XML/SOAP request, sent to GI, into the format recognised by destination application. As said before, this mapping has been realised on the basis of suitable Mapping files written in XSLT language (Tennison, 2002). When the Mapper receives a request, written according to *Generalised Interface Specification*, it maps the request in the XML/SOAP request format of the destination application, using a Request Map file, that is the map file (in XSLT language) related to the specific application. The document resulting by mapping represents the format of the request that must be delivered to the specified destination application. Once the Mapper has processed a request, the request is passed to the Dispatcher. The same process is made for the response, coming from the application. To map the response, coming from application, the Mapper uses another map file (Response Map file) that allows translating the message wrote according to the format of the specific application in the format defined by GI specification. For each request forwarded by the Mapper, the Dispatcher establishes the HTTP connection with web service relevant to the requested application.

A particular problem concerning the communication between Dispatcher and the Web Services of each application is that the HTTP is a stateless protocol. Each request to a Web Service is independent, and the application retains no memory of a client's past requests. To overcome this limitation, the sessions state handle has been introduced, allowing the GI to use HTTP cookies for maintenance of the state.

GI is able to handle concurrent multiple connections concerning different requests by the same client or by different clients. For each request, an instance of the GI modules (Parser, Mapper, Dispatcher) is created. The instance is deleted when the request has been satisfied.

3.2 GI Implementation

The architecture of the GI described in the previous section, has been implemented trying to avoid dependence on particular commercial products; so it was based on the use of free libraries and wasn't linked to a specific platform.

Implementation has been realised on the basis of an analysis of the state of the art about free software/libraries useful to manipulate XML files for the parser, mapper and dispatcher. The analysis performed has highlighted the advantages in using DOM libraries to realise parsing of XML documents and Dispatcher functionalities, and XSLT engine to realise the Mapper. DOM presents an easy processed standardised interpretation of an XML document to applications and scripts. Different free implementations of the DOM exist in different languages and platforms, see (Marini, 2002) for example.

In order to implement the Mapper, use of the XSLT engine has been considered. The XSLT engine can be an external component, a library or a class. Different free implementations of this engine exist, again in different platforms.

Dispatcher has been implemented as class. It is instanced after the Mapper has performed the SOAP request transformation. The Dispatcher exports one method that implements the routine to handle the communication with the Web Service of each specific SCADA, GIS and DSS application.

Implementation based on the use of DOM and XSLT Engine is featured by very few constraints on the software requirement. DOM and XSLT Engine are not linked to a specific platform, as it's possible to find libraries for Windows and for Unix platform. Management of Web Services can be realised by IIS or by Apache for example, but there is no constraints on this item. No need to use a specific software run-time or development environment (like .NET run-time and Visual Studio .NET) is present.

4 LOGICAL VIEW: VERTICAL AND HORIZONTAL INTEGRATION

As said in the introduction, vertical and horizontal integration is a very felt need in the management of large plant, due to the daily execution of Business Logics retrieving and transforming data maintained by distributed applications, like SCADA, GIS and DSS.

An example of Business Logic is related to the management of anomalies in a utility plant for the water distribution. A technician, using a mobile device equipped with GSM/GPRS and GPS cards, discovers an anomaly in the plant, while moving inside it. Through his mobile device (connected to the control system by the GSM/GPRS card), he invokes the Business Logic relevant to the management of anomalies. This Business Logic, first of all, allows the mobile worker to be localised (through GPS card) by the control system; then information maintained in GIS and concerning the map of the plant near to the worker (with the list of devices there present) is sent to him. On the basis of the map received, the worker can enquire the system about detailed information concerning the devices he assumes involved in the anomaly encountered. The system retrieves information from a SCADA to give the worker what he has requested. In order to take decisions about what to do (for example, closing a valve or starting a pump), the technician may ask the system to be supported by a DSS, running a simulation in order to understand, for example, what happens closing a valve or starting a pump.

This example of Business Logic clearly clarifies what Vertical and Horizontal integration means. As it can be seen, Vertical integration is realised for each single application taken into account (i.e. SCADA, DSS and GIS), as information needed to the user are given to him in the exact format he requests; this includes data transformation according to a requested data representation and visualisation according to the specific mobile device used by the worker. Horizontal integration is realised as the Business Logic seen before involves data flow and data transformations relevant to several applications.

Management of the Business Logics is realised by the Logical View (LV) module in Mobicossum Middleware. As can be seen from Figure 1, this module is placed between the Presentation Management (PM) and the GI, because it has to receive, through the PM, the identification of the Mobicossum user (e.g. the mobile worker), the kind of business logic he wants to run, all the information needed to execute the business logic; finally, the LV has to provide for the relevant results to the user. Data exchange with the real applications occurs during execution of the business logic and is based on the Generalised Interface (GI) services, so making the Vertical and Horizontal Integration of Information independent from the particular application, vendor and platform. This points out the important role played by the GI inside Mobicossum platform.

In order to perform its role, the LV has also to interact with the Data Management (DM) and the Central Services (CS). Mainly the LV uses the DM to have information about the real applications and data available; as an example, it may happen that the LV has to know the path (e.g. internet URL address) of the real application (e.g. SCADA) maintaining data relevant to the devices placed near to the worker, and has to know the list of the variable tags, inside each SCADA, relevant to these devices (to read their values, for instance). Central Service is mainly used by LV to authorise user access and for his localisation.

The realisation of the Horizontal and Vertical integration performed by the LV has been based, inside Mobicossum project, on the following hypotheses:

1. For each user profile (linked to its role inside the plant), a set of business logics is defined and maintained inside Mobicossum middleware (in the Central Service).
2. Each business logic is implemented as a script able to perform simple/complex manipulations on data coming (i.e. maintained) by SCADA/GIS/DSS applications.
3. A Directory Service inside the Central Service module is in charge to maintain all the information needed to the LV to retrieve the Business Logic script requested by the (mobile) user (e.g. its path inside a repository).

4. Each script corresponding to a business logic contains invocations to methods offered by the LV. Each LV methods may invoke in turn, methods offered by the other modules inside Mobicossum (i.e. GI, DM, CS).

According to the hypotheses seen before, the main activities performed by the LV can be summarised:

- The LV receives, from the PM, the identification of the (mobile) user accessing the middleware and the information about the Business Logic he has chosen.
- The LV invokes particular services offered by the Central Service module to have back from the Directory Service the path needed to retrieve the script requested by the (mobile) user.
- The LV executes the Business Logic Script chosen by the (mobile) user, giving him the related results through the PM.

The LV has been realised by a layered architecture, made up by the Business Logic Level (BLL) and the Basic Logical View (BLV). The BLL offers to the PM the methods allowing the user to interact with Mobicossum Middleware. A set of web service-based methods have been defined at the interface with PM, allowing the (mobile) user to retrieve value, image, time series and to dynamically interact with Mobicossum environment trough a dialog mechanism. Further, the BLL is in charge to search and retrieve (using the CS services) the Business Logic selected by the user through the PM, and to execute it. In order to achieve this goal, a script manager module inside the BLL has been defined.

The BLV mainly offers the services useful to access the other modules of Mobicossum (GI, DM and CS), according to the Business Logic to be executed. Following the philosophy of the project, the communication between the BLL and the BLV has been realised through Web Services technology. The Web Services offered by the BLV to the BLL concern the monitor and control of a plant, the management of anomalies and alarms of devices, the access to simulation models, and the warehouse management.

4.1. LV Implementation

Implementation of both the BLL and BLV has been based on open source software, without any constraints to use a specific operating system. As an example, the implementation of the Script Manager module and the Web Services offered by the BLL has been based on PHP (Schlossnagle, 2004).

5. FINAL REMARKS

The paper has presented an overview on the main features of the CRAFT IST 1999-57455 project, highlighting the internal architecture and implementation of the Middleware, conceived to provide for services useful in the management of large plants (generally utilities plant, like that for water distribution). The main feature of this middleware is its capability to provide for vertical and horizontal integration, which are very needed in the management of plants, as clearly explained in the paper.

Mobicossum project has been concluded in 2004. Before its conclusion, three pilot installations have been realised in order to highlights the real advantages of the projects. They were realised in a Spain, in a gas distribution utility plant, in Italy, in a water distribution utility plant, and in Germany, in a wastewater treatment system. The partners and the RTD performers of the project made a lot of effort to develop the Mobicossum project, mainly because they trust on the real advantages that can be introduced by Mobicossum ICT Platform in a real management of a large plant.

REFERENCES

Martí, P., Aguado, J.C., Rolando, F.,Velasco, M., Colomar, J., and Fuertes, J.M., (1999), *A Java-Based Framework for distributed supervision and Control of industrial processes*, Proceedings of the 7th IEEE International Conference on Emerging Technologies and Factory Automation. Barcelona, Spain.

Martí, P., Aguado, J.C., Rolando, F.,Velasco, M., Colomar, J., and Fuertes, J.M., (2000), *Distributed Supervision and Control of Fieldbus-Based Industrial Processes*, Proceedings of the 3th IEEE Internacional Workshop on Factory Communication Systems. Porto, Portugal.

Marini, J. (2002), *Document Object Model (Developer's Guide)*, Osborne McGraw-Hill.

Newcomer, E. (2002), *Understanding Web Services: XML, WSDL, SOAP, and UDDI*, Addison-Wesley Professional.

Schlossnagle, G. (2004), *Advanced PHP Programming,* Sams

Tennison, J. (2002), *Beginning XSLT*, Wrox Press Inc.

Walmsley, P. (2001), *Definitive XML Schema*, Prentice Hall.

Wangler, B. and S.J. Paheerathan (2000), Horizontal and Vertical Integration of Organizational IT Systems, in *Information Systems Engineering: State of the Art and Research Themes*, (S. Brinkkemper, E. Lindencrona and A. Solvberg (eds.)), Springer.