Hindawi Publishing Corporation
EURASIP Journal on Embedded Systems
Volume 2008, Article ID 219807, 10 pages
doi:10.1155/2008/219807

Research Article

Industrial TCP/IP Services Monitoring through Embedded Web Services

Francisco Maciá-Pérez, Diego Marcos-Jorquera, and Virgilio Gilart-Iglesias

Computer Science Department, University of Alicante, P.O. Box 99, 03690 Alicante, Spain

Correspondence should be addressed to Diego Marcos-Jorquera, dmarcos@dtic.ua.es

Received 1 February 2007; Revised 27 August 2007; Accepted 5 November 2007

Recommended by Luca Ferrarini

The amount of IT devices and services incorporated in the industrial environment has led to the need to design mechanisms that will ensure its correct operation and minimise stoppage times. This paper proposes a system based on service-oriented architectures that allows the correct operation and monitoring of the applications and services running in this type of production elements. The main component of the system is a reduced size network device—that we have named eNSM device—in which the monitoring function proposed has been embedded as a web service. The whole system is based on a distributed application whose components are software agents. In addition, an application protocol named NSMP has been defined for communication between these agents.

1. INTRODUCTION

Internet capacity for providing clients with a choice of the latest consumer goods available at competitive prices is currently requiring the industrial sector to progress from traditional *mass production* manufacturing paradigms towards models which will facilitate *mass customisation* [1].

With these new production models, the customer is no longer considered as a mere entity, quite separate from the manufacturing process itself, and instead becomes part of it as an active component, actually determining the specific characteristics of the desired product.

In order to ensure that these manufacturing models are viable, the manufacturing processes must be fully integrated in the organisation global process map [2]. Although new technologies may help considerably in this aim, it is also true that they require a change of scenario for which not all organisations are completely ready [3].

One of the main problems faced by these organisations is the emergence of new management tasks deriving from the introduction of complex and innovative technologies in manufacturing levels [4–6] with new production machinery incorporating IT elements [7, 8], ranging from the simplest to the most complex. This situation is further aggravated by the fact that organisations encounter problems in training their employees or finding professionals with the requisite profile and skills.

Service-oriented architectures (SOAs) are a style of IT architectures that support the transformation of a business in a set of chained services. When an SOA implementation is guided by strategic business objectives, alignment between information technologies and business models is possible, taking maximum advantages of these technologies. The features that provide this type of approach (interoperability, composition, reusability, etc.) have been considered suitable in order to undertake open problems in the industrial environment [7].

Our proposal consists of providing embedded IT management services in physical network devices—generally, small sized devices with simple services—so that, in order to deploy those services, it is enough to select the specific device providing the service, and connect it to the communications network. The device itself will obtain the minimum information required to activate the initial setup and, once this has been completed, execute the management tasks with minimal human intervention. In addition, since the service is provided from a physical device, it does not set in motion

too many additional maintenance tasks relating to the infrastructures providing support for the new IT services.

Obviously, from a functional point of view, the services offered by these devices are totally compatible with the traditional network services, and therefore, their integration and interoperability are ensured. More precisely, the services implemented are compatible with standard web services and with other more traditional client-server protocols within the scope of systems management such as telnet, TFTP, HTTP, or SNMP.

By way of illustration and with the aim of explaining the motivation behind the proposal, in this work we suggest a specific management service: network service monitoring (NSM) built into the current manufacturing equipment as well as the physical device associated with that service (eNSM device). The goal of this service is to reduce stoppage times in the event of failures and discontinuity in the production process.

The main task of the system consists of monitoring the correct functioning of the applications and services of the manufacturing components by means of an eNSM device. These monitoring processes must be previously scheduled (generally by using an IP address, an IP port, and the expected result for the service analysed). If an incident is detected, it will be registered and reported.

In the following sections we provide a review of the current state of the art of the technologies involved, a description of the NSM service, hardware and software structure of the device in which it is embedded, the specification of the application protocol and its implementation as web service embedded in a specific network device, and, finally, the conclusions on the research and the current lines of work.

2. BACKGROUND

The integration of manufacturing processes in the general organisation process map in order to achieve continuity is one of the main goals of the industrial sector [9]. Due to physical and technological limitations, manufacturing processes have not reached the desired level of integration and automation, and in most cases they have to be considered as legacy systems. Until quite recently, integration proposals were centred on traditional automation models based on proprietary protocols situated at a resource level of the eBusiness model as systems external to business processes (Modbus, Profibus, AS-I, FIPIO, DeviceNET, Interbus, or Ethernet industrial). These proposals were the first attempts to facilitate their integration with business components using ad-hoc adaptors [10].

With the development of internet and electronic advances, solutions have been proposed based on service paradigms, distributed systems and embedded devices with the computational and communications capacity appropriate for environments with hostile physical conditions, such as those occurring in industrial atmospheres.

Schneider was one of the first manufacturers of automation and industrial control devices to have incorporated these ideas in its automation apparatus in order to communicate with management applications. This trend is reflected in concepts such as *transparent factory* [11]. Other manufacturers such as ABB go somewhat further by raising communication to higher levels of organisation using the simple object access protocol (SOAP), and incorporating intelligence, self-management, and proactivity into its embedded devices [12]. Along the same lines, in [5] the use of web services is proposed as a normalised means of accessing functionalities of the devices so that they can be integrated with enterprise resource planning (ERP) systems. Reference [13] proposes to use these same techniques to raise the level of abstraction of production elements to a business level so that the integration of resources, processes, and, in general, business logic are produced naturally and in a transparent manner within the current business models.

Within the framework of European research projects there are some important initiatives which bear out our interest in this line of research, and which have produced significant results and are progressing towards acceptance of service-oriented architectures (SOAs) and embedded devices in industrial machinery as valid technologies [7, 14].

The scientific community is clearly interested in the use of IT paradigms which have established the present web bases as a technological framework supporting the execution of processes associated with production elements.

However, as information technologies continue to inundate production plants, increasingly further new associated tasks arise, namely, the management of the new services and infrastructures used. These tasks are gaining greater importance as the organisation becomes increasingly reliant on IT, requiring the same levels of robustness and security as in the industrial sector.

The first open standards which attempted to address problems of IT management in a global manner were the simple network management protocol (SNMP) and the common management information protocol (CMIP) [15], proposed by the Internet engineering task force (IETF); both protocols being principally oriented towards network monitoring and control. The main inconvenience of these administration models was their dependence on the platform.

Based on these, and seeking an integration with heterogeneous systems, two principal lines of work arose: an attempt to achieve integration of the systems using the same network management protocol, as is the case of [16, 17] with the use of common object request broker architecture (CORBA), and more ambitiously, to propose a network management protocol which would be independent of the infrastructures. Some of the more extensive proposals include CORBA/JIDM, specification of the work group joint inter-domain management (JIDM) [18] of the object management group (OMG) [19], CIM/WBEM, proposal of the distributed management task force (DMTF) [20] using techniques oriented towards computer integrated manufacturing (CIM) objects and interoperation using HTTP and XML with web-based enterprise management (WBEM), JMX specification defined by the Java Community Process (JCP) [21] which defines a series of application programming interfaces (API) oriented to Java for network management, and WS-management specification carried out by various companies in the sector (Sun, Intel, MS, AMD) for

the integration of service management systems and resources based on web services.

The use of multiagent systems for computer network management provides a series of characteristics which favour automation and self-reliance in maintenance processes [22, 23]. The creation of projects such as AgentLink III, the first coordinated action on based on agents financed by the 6th European Commission Framework Programme [24], is a clear indicator of the considerable degree of interest in research into software agents.

In [25], a proposal is made for a group of basic operations for a web service to be standardised within the management networks as a counterpart to standardisation of the SNMP information model under XML development in other works [26].

As a result of the considerable number of tasks associated with network management, as well as its diversity and complexity, the work of maintaining these systems involves a high cost for organisations, both in terms of resources and also in terms of time and personnel; added to this are the difficulties inherent in engaging staff with the required skills for addressing this new scenario.

In order to relieve these problems, the current trend in IT management is to use outsourcing as a strategy to recoup investments, ensure the continuous availability of infrastructures and services, and to achieve sufficient levels of quality to enable organisations to keep abreast of a changing environment. However, outsourcing is not usually a valid strategy for handling production environments due to problems raised by security, privacy and immediacy.

In areas where automated handling of information and those where several devices are involved, such as industrial processes or domotics, there has been a trend in the development of autonomous management towards architectures designed for services for embedded systems [12, 14]. This final framework includes monitoring systems developed by third parties but residing with the client, who is responsible for their control and management. Along these lines we find proposals such as NAGIOS [27], MON [28], MUNIN/MONIT [29, 30], or nPULSE [31] generic monitoring systems for network services for Linux, with web interface, highly configurable and based on open code which monitors the availability of network services and applications. The disadvantage of these proposals lies in the complexity of their installation and configuration in environments without qualified system administrators, in addition to the complex systems and infrastructures required for their implementation.

Reference [32] presents an approach based on the use of embedded network devices for the deployment of small network services suitable for IT management in industrial and manufacturing environments. The proposal is innovative in that it allows IT services to be implemented which can be deployed without the need for specialised IT staff, since these services in question have a zero maintenance design philosophy. Other interesting aspects of these devices are the fact they are very small, in that every device specialises in a specific IT service, and that they are specially designed to operate with minimum maintenance, and also they are presented under both conventional (client-server) and more open (SOA) standards, more specifically, as web services. In addition, these embedded services can work individually or in collaboration with other IT enterprise services, either through conventional systems or by means of others embedded devices.

In conclusion, it is possible to say that the incorporation of IT in industrial production environments is as necessary as it is inevitable; due to the volume and complexity of these new technologies, it is important to look at them from the perspective of their own conception self-management features. Our approach focuses on this environment: by proposing a device which will help the existing IT management system, designed to operate with IT technologies in an integrated way and without any need for attention from system administrators.

3. NETWORK MONITORING SERVICE

The main goal of the network monitoring service is to check the correct operation of the TCP/IP network applications and services running in manufacturing components.

The embedded network service monitoring (eNSM) is the version of the monitoring service that has been implemented in both web service and client-server version, and it has been embedded in a network device (known as eNSM Device) designed for this purpose (see Figure 1). This device is small in size, robust, and transparent to existing IT infrastructures and with minimum maintenance required from the system administrators.

The system administrator informs the eNSM device, by means of its *interface agents*, which of the applications or services of the manufacturing components require verification. The eNSM device has sufficient knowledge of each service to carry out this task. This knowledge is included in *monitoring agents* displaced to the device for this purpose. In this way, if the device receives a request for monitoring a new service, it will request the adequate *monitoring agent* in a self sufficient manner in order to carry out its work.

The monitoring procedure consists of establishing connections with the services and applications to be monitored by means of its own protocols based on TCP/IP, analysing the responses to standard requests in search of differences with regard to a previously established pattern, either in the operation of the protocol itself or in the data received.

Thus, the eNSM device represents the core of the system. Figure 1 shows a diagram of the main elements and actors involved in the service, together with the existing relation between them. We may synthesise these as eNSM Device, manufacturing components, discovery service, NSM center, NSM clients, a set of software agents and the NSM application protocol (NSMP). These elements shall subsequently be described in greater detail.

The *eNSM device*, as has been seen, is the cornerstone of the monitoring service. It is designed in order to act as a proxy between the wide area network (WAN) and local area network (LAN) to which it provides support. This device provides a container in which different agents and applications ensure that the service can be executed.

In the proposal implementation, the device interface with the system administrators and with other management

devices or management equipments is provided by agents acting as embedded web services (see *SOA interface agent* in Figure 1). From a functional point of view, this is the reason why an eNSM device can be to taken into account, simply, as if it were a web service. Of course, as well as providing the web service interface, a more classical client-server interface is also proposed in order to enlarge the device compatibility range (see *CS interface agent* in Figure 1). In this way, an eNSM device is responsible for collecting the monitoring request from the WAN. These requests are based on NSMP protocol and encapsulated in SOAP messages when they are sent to the web service interface, or as plain text request-response messages if they are sent to the client-server interface. Each monitoring request will cause a specific agent (*monitoring agent*) to ensure that the requested service testing is carried out by means of the suitable protocols (based on TCP/IP).

The *manufacturing components* are the goal of the network monitoring service and comprise all those industrial devices connected to the TCP/IP network acting as network services containers, which will be monitored.

The *discovery service* comprises a standard UDDI registration service. It is responsible for maintaining the pages describing the NSM services in WSDL format, as well as facilitating that information to the clients wishing to access the service.

NSM centers usually act as automated control panels for the eNSM devices distributed through Internet. This control is implemented through the *planning agents* who carry out, execute, and verify all the previously established tasks on the eNSM devices. NSM Centres are also responsible for managing the repository of *monitoring agents* with the know-how of each monitoring service. Although in large installations it is recommended that management and scheduling services are included, the existence of an NSM centre is not essential. Likewise, although each NSM centre can manage around a thousand eNSM devices, it is possible to use the number of NSM centres considered appropriate, and it is possible to create one hierarchy with these elements.

NSM clients, through the *NSM agents*, provide the user with access to the NSM centre (in order to manage work plans or query log files) and to the eNSM devices (in order to manage particular services of specific manufacturing components directly). These clients are not necessary for the normal operating system; however, they avoid physical movements of the system administration staff.

System functionality has been defined as a distributed application based on *software agents*, because this approach intrinsically includes aspects such as communications, synchronisation, updates. Among the agents that have been defined in the system, the most important are the agents placed in the eNSM device, and as a result, they comprise the system core. Of these last agents, the *interface agents* are of prime importance as they allow the device to provide its functionality to external elements. Two types of interfaces have been implemented: *the SOA interface Agents* which provide a compatible interface with web services and *the CS interface agents* which provide a compatible front-end with classical client-server technology. Section 4 provides a detailed description of system agents.

The *NSM protocol* (NSMP) is a request-response application level protocol, based on plain-text and designed to work alone or with other protocols used as transport protocols, for example, HTTP, SMTP, or in the case of SOA agents, using SOAP messages. This protocol is used by the different system components in order to communicate between each other. In fact, as the application has been designed as a set of software agents, the protocol will be used by the software agents to communicate with each other. In Section 5, the main NSMP protocol commands are analysed.

4. SOFTWARE AGENTS

The software agents do not constitute a conventional multiagent system because a generic context has not been defined for them, they do not use standard agent communication languages and they do not work collaborating to achieve a general target which is used by the agents to take its decisions. In fact, the set of software agents implement part of the functionality of a distributed application which has been designed to provide a network service; in this case, the monitoring service. The reason why agent approach is used lies in its simplicity to design distributed applications and to take into account aspects such as communication, mobility or software updates.

Of all the agents defined in the system, the most important are those located in the eNSM device. In this way, each eNSM device comprises a set of agents that implement its interface with the system administrators or with others system elements (NSM clients or NSM centres). In order to guarantee the system's compatibility with a large range of technologies, several interface agents have been implemented. In this way, the *SOA interface agent* provides a matching interface with web services-based applications. At the same time, another interface agent, known as *CS interface agent*, has been defined in order to guarantee the service compatibility with classical client-server systems (e.g., with HTTP or Telnet compatible applications). Every interface agent can identify commands based on *NSMP protocol* and, from these commands, schedule the eNSM device work plan. *NSM agents* and *monitoring agents* are another type of agent placed in the eNSM device and designed to perform the monitoring service. The first type of agents ensure execution of the scheduling, delegating the specific monitoring task to a *monitoring agent*. Each service type to be monitored has a specific *monitoring agent* provided with the know-how to perform its task. In addition to these core agents, other agents are included in each eNSM device in order to perform auxiliary tasks. Thus, the *register agents* undertake to check the monitoring service in a discovery service, and the *employer agents* are responsible for locating the *monitoring agents* required by the eNSM device to carry out its task. *Monitoring agents* are mobile agents that, initially, can reside in an agent farm located in an NSM Centre.

As has been mentioned, besides the agents located in each eNSM device, the distributed application is completed by other auxiliary agents located outside the device which, while

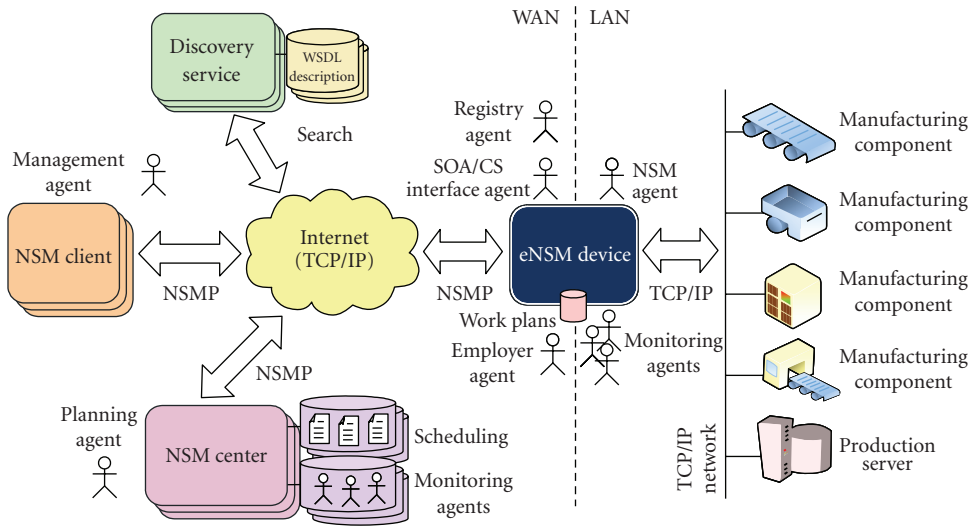

FIGURE 1: Organisation of functional elements of the NSM service.

not being crucial to the service, serve to make it more functional. As a result, the *management agents* reside in an NSM client and are responsible for providing an appropriate interface for the administrators so that they can access the NSM Centre or an eNSM device from any node connected to Internet. The *planning agents* reside in the NSM centres and undertake the planning management of eNSM devices.

5. NSM PROTOCOL

The system agents, implemented in our prototype both as web services and client-server, communicate with each other by means of messages containing instructions capable of interpreting and executing. These instructions, together with their syntax and its pertinent response, come defined by the NSM protocol or NSMP. When the agents specifically behave as web services, these commands will be incrusted inside the request and response SOAP messages.

The NSM protocol (NSMP) is a text-based request-response application level protocol which gathers all monitoring service functionality through a set of instructions. The protocol has been defined as a request-response text-based application protocol. This enables it be easily adapted to different models, such as client-server (over basic protocols like HTTP, SMTP, or telnet) and SOA (over protocols like SOAP).

The syntactic elements for an instruction are the following:

(i) *CMD* defines the service actions and corresponds to the name of the remote procedure which implements the functionality of the NSMP command;

(ii) *ACTION* is a special parameter which discriminates the functionality of the request;

(iii) *ARGS* represents the necessary information for executing the <action>.

The main instructions (shown in Table 1) that can be embedded in an NSM request are SET, GET, PUT, MONITOR, and ALERT. *SET* command manages the configuration of the internal system variables which determines their function mode. *PUT* and *GET* commands, combined with the *SCHDL* action, programme and obtain, respectively, the agents work planning. *GET* command, with *STATUS* action, allows specific information to be obtained from the device status. *MONITOR* command provides access to the principal service of the device. The instruction syntax for activating a monitoring service is as follows:

MONITOR ON <host> <port> <time> <service> [args]*.

In order to disable the monitoring of a service, the instruction format will be

MONITOR OFF <host><port><service>.

In both cases, the labels have the following meaning:

(i) <host> is the IP address or the name of the device to monitoring;

(ii) <port> identifies the service or application whose status requires verification;

(iii) <service> identifies the monitoring agent which analyses the service.

ALERT Command is designed so that the eNSM device is able to notify the NSM centre of the error.

As this is a case of a request-response protocol, for each NSMP request there will be a corresponding NSMP response. Basically, this request will be OK if the order is correctly executed or conversely ERROR if it is not. In some cases, the answer may contain more specific information about the operation carried out such as the *GET SCHDL* command, which returns the list of programmed tasks.

These instructions should be embedded inside the specific request-response protocol which will be used as a transport layer. In the event of choosing a telnet service, the instructions and result of this execution will be literal. In the case of HTTP, the instructions will need to be inserted in an

Table 1: Main instructions of the NSM protocol.

CMD	Action	ARG	Function
SET	MODE		Reports the current operation mode.
		PASSIVE [port]	Sets the passive mode and optionally, the listening port number.
		ACTIVE <ip> [:port]	Sets the active mode, specifing the NSM center's IP address and port number.
	RUN		Reports the current NSM service state.
		< STARTS | STOP >	Starts or stops the NSM service.
GET	SCHDL		Returns the list of scheduled tasks in the device.
	STATUS	[<host> [:port] [<service>]]	Returns the status of a specific service or a set of services.
PUT	SCHDL	<schdl-table>	Adds a task or a set of tasks to the scheduling.
MONITOR	ON	<host>:<port><time> <service>[arguments]*	Establishes a monitoring rule for the address <host>:<port>, establishing the poliing time in seconds and the monitor that will be utilized as well as the arguments that this require.
	OFF	<host>:<port><service>	Cancels a monitoring rule.
ALERT			Send an error alert.

HTTP request body. In strategies to support service-oriented architecture, in particular, Web Services, it uses SOAP as mechanism for message interchange. This case is a little more complex and it requires further attention.

5.1. NSMP and web services

In a SOAP request, *document style* and *RPC style* are supported. In the case of *document style*, each NSM instruction is embedded in the body of a SOAP message which contains the NSMP document, which implements the functionality of the command and the arguments required for its execution. The syntax of NMSP is defined in its corresponding *document type definition* (DTD). Figure 2 shows a DTD fragment which defines the syntax for the MONITOR instruction—request message in Figure 2(a) and response message in Figure 2(b). The DTD document allows new messages to be created with the correct syntax and validation if the sent messages accomplish this syntax.

In the case of *RPC style* SOAP messages, there is a single operation (named *EXECUTE*) whose single argument is a text string with the NSMP instruction. This case is very similar to the client-server approach.

Figure 3 displays an example of SOAP request message for the MONITOR instruction in document style, using the DTD described in Figure 2. This message creates a new monitoring rule for the HTTP service (Port 80) located in the manufacturing component identified by its IP address (192.168.7.58).

The sequence diagram in Figure 4 shows the basic operation of the service and the communication between the system software agents. The diagram comprises two blocks and is executed constantly in parallel mode.

In the first block, the device interface agents (*SOA agent* and *CS agent*) are on standby for requests either from a *planning agent*, or directly from a *management agent*. When the interface agents receive a monitoring request (MONITOR command), they add the task to the work plan database of the eNSM device.

The second diagram block corresponds to the execution of the programmed tasks. In this case, the *NSM agent* is constantly checking the work plan data base and selecting the suitable *monitoring agent* to carry out the tasks requested.

Although it is not shown in this diagram, there is also a third block which concerns the contracting of the *monitoring agents*. When there is not a *monitoring agent* able to deal with the monitoring service requested, the *NSM agent* and the *SOA* or the *CS agents* who have detected this lack may make a request to the *employer agent* programming it into its work plan. The *employer agent* then undertakes to obtain the *monitoring agents* required by the device. This agent is responsible for negotiating and validating the whole process. The *monitoring agents* are mobile agents located in the agent repository in the NSM Centres.

6. eNSM ARCHITECTURE

As has been previously described, the monitoring service is part of a wider system the cornerstone of which is the eNSM device. The eNSM device combines hardware and software in order to deploy all its functionality, thus achieving the different, previously established requirements.

For this reason, a general device architecture has been defined (Figure 5). This device architecture is organised in a layer-based manner and has a widely accepted structure for embedded devices. In the lower layer, the device hardware has been defined, based on a computational system including microprocessor, volatile memory (for the execution), nonvolatile memory (for the stored system), and communication module for its connection to the network.

In addition to these basic components, it is important to highlight the absence of mechanical elements (such as hard disks) and the inclusion of auxiliary elements (such as watchdog mechanism or Power over Ethernet supply).

The embedded operative system is placed on the hardware layer. In order to satisfy industrial environment specific requirements—where the device will be included and will provide support—real time operative systems have been

(a) DTD Request

```
<!DOCTYPE command[
<!ELEMENT command (set | get | put| monitor | alert)>
<!ELEMENT monitor (action)>
<!ATTLIST monitor host CDATA #REQUIRED>
<!ATTLIST monitor port CDATA #REQUIRED>
<!ELEMENT action (on | off)>
<!ELEMENT on (arguments)>
<!ATTLIST on time CDATA #REQUIRED>
<!ATTLIST on service CDATA #REQUIRED>
<!ELEMENT arguments (arg*)>
<!ELEMENT arg (EMPTY)>
<!ELEMENT arg name CDATA #REQUIRED>
<!ELEMENT arg value CDATA #REQUIRED>
<!ELEMENT off EMPTY>
<!ATTLIST off service CDATA #IMPLIED>
...
]>
```

(b) DTD Response

```
<!DOCTYPE response[
<!ELEMENT response (ok | error)>
<!ELEMENT ok (ok_description)>
<!ELEMENT ok_description (#PCDATA)>
<!ELEMENT error (error_description)>
<!ATTLIST error code CDATA #REQUIRED>
<!ELEMENT error_description (#PCDATA)>
]>
```

Figure 2: Fragment of NSM application protocol DTD document (document style).

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <soap:Envelope
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
   Soap:encodingStyle="http://schemas.xmlsoap.org/soap/">
3    <soap:Header>
       ...
     </soap:Header>
4    <soap:Body>
5      <nsmp:command
         xmlns:nsmp="http://www.dtic.ua.es/grupM/nsmp">
6        <nsmp:monitor host="192.168.7.58" port="80">
7          <nsmp:args>
             <nsmp:on time="60" service="http">
8              <nsmp:arguments>
                 <nsmp:arg name="page" value="index.htm" />
               </nsmp:arguments>
             </nsmp:on>
           <nsmp:args>
         <nsmp:monitor>
       </nsmp:command>
     </soap:Body>
   </soap:Envelope>
```

Figure 3: Example of an NSMP SOAP request message.

chosen. The subsequent layer contains the middleware platform to provide services to the applications.

The first two important elements in this layer comprise different network services, commonly used in the management field, under the client-server (CS) model and service-oriented architectures (SOAs). These modules give basic services so the device can communicate, at application level, with other external components. In order to adapt this communication to the syntax of monitoring service instructions, the NSMP protocol is implemented on these modules (in both SOA and CS versions). One of the other important services placed in this layer is the core, which contains all the procedures offered by the NSM services. The middleware platform is also located in the same layer in order to provide support to the service software agents. These agents are placed in the last layer (application layer), and in fact, they undertake to provide the service, acting as interface with other services and applications (*SOA/CS agents*), registering the service in a discovery service (*register agents*), coordinating the monitoring service (*NSM agents*), or, simply, monitoring selected services (*monitoring agents*).

7. eNSM DEVICE IMPLEMENTATION

In this section, the implementation of an eNSM prototype device is presented, taking into account the general architecture described in the previous section and specifying the different structural blocks according to the available technologies. In Figure 6, the resulting architecture has been given graphic shape.

The hardware platform chosen for the prototype development is a *Lantronix Xport AR* device which has a 16-bit *DSTni-EX* processor with 120 MHz frequency reaching 30 MIPS, respectively (Figure 7 shows an image of an eNSM device prototype connected to the service network). The various memory modulates provided by this device undertake specific tasks according to their intrinsic features: the execution programmes and the dates handled by the device SRAM memory reside in the "1.25 MB," the ROM memory (16 KB) holds the system startup application, and, finally, the flash memory, with 4 MB, stores information which is though nonvolatile and is susceptible to change, such as the setup of the eNSM device or the system applications which may be updated. These capacities are sufficient for the memory requirements of the software developed for implementing the protocol.

Among other I/O interfaces, the device has a Fast Ethernet network interface which allows suitable external communications ratios. In addition, in order to ensure the correct system operation, there are several auxiliary elements such as a watchdog which monitors the CPU and prevents it from blocking and a PLL frequency divider required to set up the

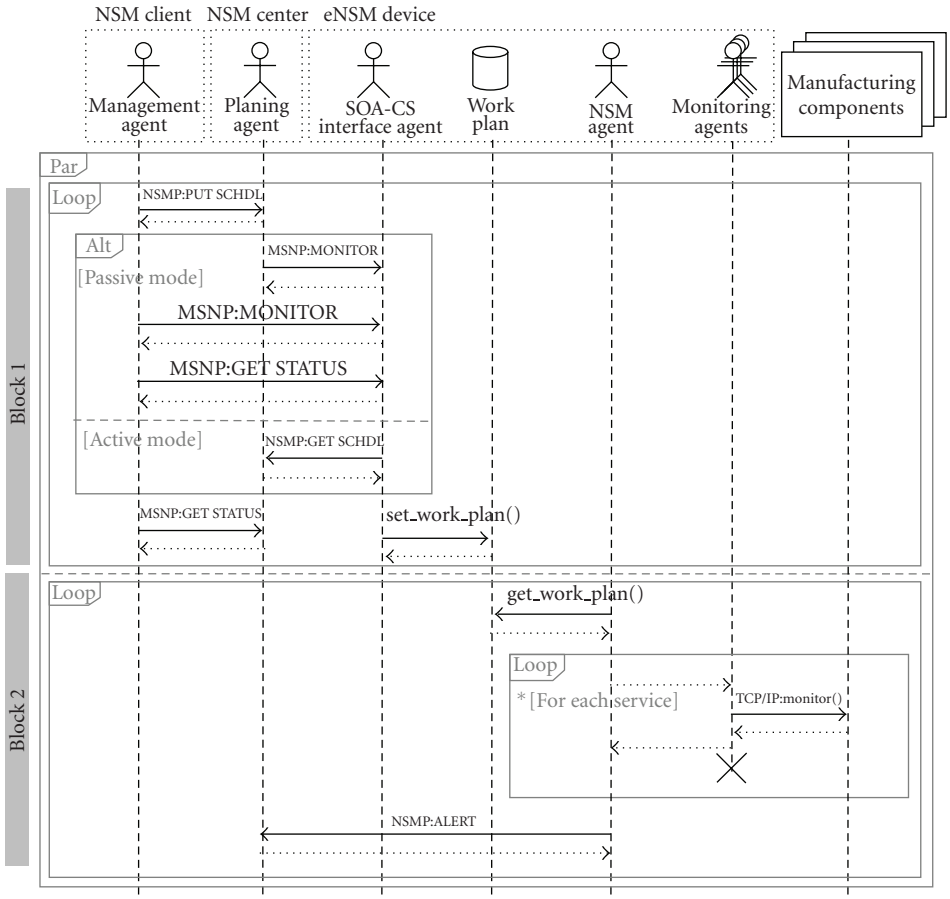

Figure 4: Sequence diagram of the NSM main functionality.

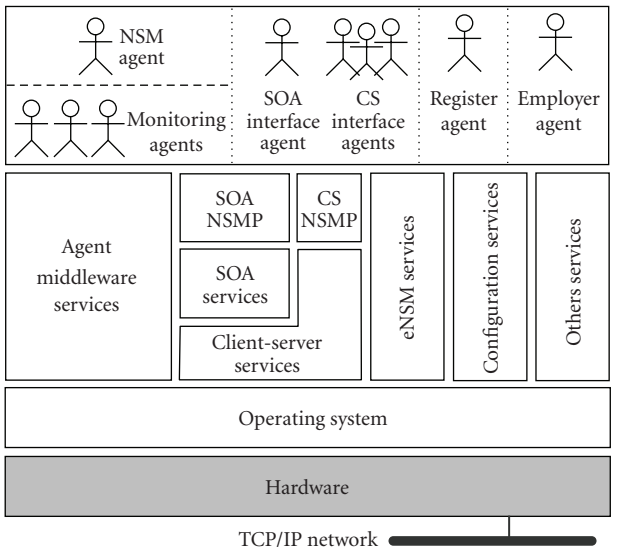

Figure 5: eNSM device architecture.

frequency of the system clock, with an adjustable clock signal (CLK) to optimise consumption or performance according to needs.

As a real time operating system, the device incorporates version 3 of the *Lantronix* OS, *Evolution OS*. Through a confidentiality agreement with *Lantronix*, we have had access to the different modules of the system. Given the space restrictions, this has been crucial to develop a made-to-measure version of this OS. Salient elements of this version include a TCP/IP stack together with several client-server application protocols (HTTP, TFTP, SNMP, and Telnet).

In the service layer, the implementation process has been conditioned by the characteristics of *XPort* device. Although, with the current hardware miniaturisation level, their computational capacities have been increased, the devices continue to present considerable limitations in their resources.

In this layer, three service blocks are implemented: the middleware that provides the communication mechanisms of the monitoring service, the NSM service kernel with the implementation of NSM instructions, and the middleware platform that provides the execution of software agents.

The communication service middleware is upheld by standard protocols and technologies included in the *Evolution OS*. In the client-server-based NSMP implementation, a *C* module has been developed to provide a protocol syntactic analyser. In the SOA-based NSMP implementation (i.e., the web service interface), the cSOAP library was used for development, which is appropriate for these devices [33].

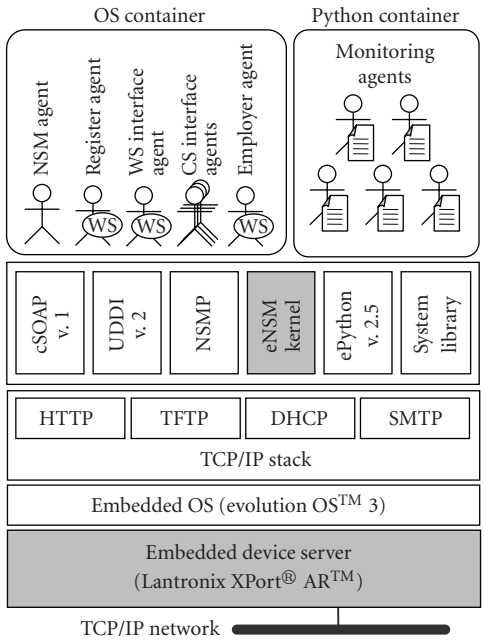

FIGURE 6: Architecture of the eNSM device prototype.

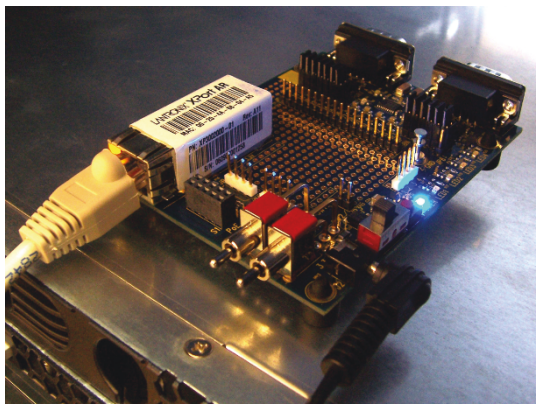

FIGURE 7: The eNSM device prototype.

However, some changes have been made to the original cSOAP library due to device limitations (restriction of memory use, proprietary libraries, etc.). These limitations have forced us to replace cSOAP XML parser, LibXML2 (over 1 MB in size), by another adapted XML parser with limited but sufficient functionalities to achieve our objective. Due to cSOAP limitations, only *RPC style* which uses the same protocol analyser used in the client-server version has been developed.

In addition, in order to register and to publish the services, a UDDI embedded version has been implemented based on UDDI version 2.0 which simply permits publishing the WSDL document associated with the monitoring service. Figure 8 shows a fragment of the WSDL page with the definition of the RPC procedure for the EXECUTE command.

```
<operation name="monitor">
  <SOAP:operation style="rpc" soapAction=""/>
  <input>
<SOAP:body use="encoded" namespace="urn:EXECUTE" \
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
  </input>
  <output>
<SOAP:body use="encoded" namespace="urn:EXECUTE" \
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
  </output>
</operation>
```

FIGURE 8: WSDL document fragment for MONITOR command (RPC style).

The NSM service kernel has been implemented as a functions library written in C language and offered as API for the others eNSM device modules. By means of this library, the monitoring service intrinsic functionalities are achieved.

In order to implement service agents, a division has been made in the implementation process between static and mobile agents. In the first case, an ad hoc implementation for the *XPort* device has been developed in C language, using an operative system such as the agents' container. In the second case, in order to establish an execution framework for the mobile agents (the *monitoring agents*), a Python embedded engine (*ePython* version 2.5) has been adapted to the *XPort* features. These *monitoring agents* are implemented as *Python* text scripts.

8. CONCLUSIONS

In this paper, we have presented a system for the provision of IT services designed to manage applications and embedded services in machinery and production components in industry. The aim of the proposal is to provide a reference framework in order to design and implement specifics-embedded services in a systematic way. One of the most relevant aspects of this system consists of providing these embedded management services in network devices. The devices must be adapted to the characteristics of the production and IT environments: small size, simple, low-power consumption, adjusted costs, autonomous, designed with safety criteria and robustness, and compatible with the traditional network services through the standard protocols such as SOAP, SMTP, or HTTP. In order to validate the proposal, a specific service has been designed and implemented to monitor the IT embedded services in the current components of manufacture, together with a functional prototype of the network device.

We are currently working with other embedded network services and integrating them all in a model based on semantic web services, so that in future they will be compatible not only with existing services, but also with new services or setups which were not considered in the initial design.

The final objective of our research is focussed on assuring continuity in the manufacturing processes, through technologies that should be as transparent as possible to the users.

ACKNOWLEDGMENTS

This work was supported by the Spanish Ministry of Education and Science with Grant TIN2006-04081.

REFERENCES

[1] Y. Choi, K. Kim, and C. Kim, "A design chain collaboration framework using reference models," *International Journal of Advanced Manufacturing Technology*, vol. 26, no. 1-2, pp. 183–190, 2005.

[2] L. Avella and D. Vázquez, "Is agile manufacturing a new production paradigm?" *Universia Business Review*, no. 6, pp. 94–107, 2005.

[3] P. Harmon, M. Rosen, and M. Guttman, *Developing E-business Systems and Architectures: A Manager's Guide*, Morgan Kaufmann, San Francisco, Calif, USA, 2001.

[4] D. C. McFarlane and S. Bussmann, "Developments in holonic production planning and control," *Production Planning & Control*, vol. 11, no. 6, pp. 522–536, 2000.

[5] A. P. Kalogeras, J. V. Gialelis, C. E. Alexakos, M. J. Georgoudakis, and S. A. Koubias, "Vertical integration of enterprise industrial systems utilizing web services," *IEEE Transactions on Industrial Informatics*, vol. 2, no. 2, pp. 120–128, 2006.

[6] J. L. M. Lastra and M. Delamer, "Semantic web services in factory automation: fundamental insights and research roadmap," *IEEE Transactions on Industrial Informatics*, vol. 2, no. 1, pp. 1–11, 2006.

[7] F. Jammes and H. Smit, "Service-oriented paradigms in industrial automation," *IEEE Transactions on Industrial Informatics*, vol. 1, no. 1, pp. 62–70, 2005.

[8] S.-M. Lee, R. Harrison, and A. A. West, "A component-based distributed control system for assembly automation," in *Proceedings of the 2nd IEEE International Conference on Industrial Informatics (INDIN '04)*, pp. 33–38, Berlin, Germany, June 2004.

[9] J. V. Gialelis, A. P. Kalogeras, C. E. Alexakos, M. J. Georgoudakis, and G. Papadopoulos, "Manufacturing collaborative process integration utilizing state of the art technologies," in *Proceedings of IEEE International Symposium on Industrial Electronics (ISIE '05)*, vol. 4, pp. 1429–1434, Stockholm, Sweden, June 2005.

[10] R. P. Moreno, "Ingeniería de la automatización industrial," RaMa, Madrid, Spain, 2004.

[11] Transparent Factory. Manual de usuario y planificación. http://www.modicon.com, 2001.

[12] U. Topp, P. Müller, J. Konnertz, and A. Pick, "Web based service for embedded devices," in *Web-Services, and Database Systems*, vol. 2593 of *Lecture Notes in Computer Science*, pp. 141–153, Erfurt, Germany, October 2003.

[13] V. Gilart-Iglesias, F. Maciá-Pérez, J. A. Gil-Martínez-Abarca, and A. Capella-D'alton, "Industrial machines as a service: a model based on embedded devices and web services," in *Proceedings of 4th International IEEE Conference on Industrial Informatics (INDIN '06)*, Singapore, August 2006.

[14] F. Jammes, H. Smit, J. L. Martinez Lastra, and I. M. Delamer, "Orchestration of service-oriented manufacturing processes," in *Proceedings of the 10th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA '05)*, vol. 12, pp. 617–624, Catania, Italy, September 2005.

[15] RFC Project: http://www.rfc.net.

[16] M. S. Jeong, K. H. Kim, J. H. Kwon, and J. T. Park, "CORBS/CMIP: gateway service scheme for CORBA/TMN integration," *Knom Review*, vol. 2, no. 1, pp. 55–62, 1999.

[17] G. Aschemann, T. Mohr, and M. Ruppert, "Integration of SNMP into a CORBA—and web-based management environment," in *Proceedings of Kommunikation in Verteilten Systemen*, pp. 210–221, Darmstadt, Germany, March 1999.

[18] Work Group JIDM: http://www.opengroup.org.

[19] OMG: http://www.omg.org.

[20] DMTF:http://www.dmtf.org.

[21] JCP: http://www.jcp.org.

[22] T. C. Du, E. Y. Li, and A.-P. Chang, "Mobile agents in distributed network management," *Communications at the ACM*, vol. 46, no. 7, pp. 127–132, 2003.

[23] J. Guo, Y. Liao, and B. Parviz, "An agent-based network management system," in *Proceedings of the IASTED International Conference on Internet and Multimedia Systems and Applications (IMSA '05)*, pp. 7–12, Honolulu, Hawaii, USA, August 2005.

[24] European co-ordination action for agent-based computing: http://www.rfc.net.

[25] J. Sloten, A. Pras, and M. Van Sinderen, "On the standardisation of web service management operations," in *Proceedings of the 10th Open European Summer School and IFIP WG 6.3 Workshop (EUNICE '04)*, pp. 143–150, Tampere, Finland, June 2004.

[26] T. Klie and F. Strauss, "Integrating SNMP agents with XML-based management systems," *IEEE Communications Magazine*, vol. 42, no. 7, pp. 76–83, 2004.

[27] NAGIOS: http://nagios.org.

[28] MON: http://www.kernel.org/software/mon/.

[29] MUNIN: http://munin.projects.linpro.no/.

[30] MONIT: http://www.tildeslash.com/monit/.

[31] nPULSE: http://www.horsburgh.com/h_npulse.html.

[32] J. A. Gil Martínez-Abarca, F. Maciá Pérez, D. Marcos Jorquera, and V. Gilart Iglesias, "Wake on LAN over Internet as web service," in *Proceedings of the 11th IEEE International Conference on Emerging Technologies and Factory Automation*, Prague, Czech Republic, September 2006.

[33] V. Miori, L. Tarrini, and R. Bianchi, "LIGHT: XML-innovative generation for home networking technologies," *Ercim News*, no. 62, 2005.