The 4th International Conference
"Advanced Composite Materials Engineering "
COMAT 2012
18- 20 October 2012, Brasov, Romania

SERVICE ORIENTATION IN DISTRIBUTED AUTOMATION AND CONTROL SERVICE

C. Voican
University Politechnic of Bucharest,voicancristiana@yahoo.com

Abstract: *An experimental study shows the feasibility ofservice-oriented architectures for industrial automation and control systems even with respect to lower, real-time dependent control functions. For that purpose, general SOA-guidelines were refined in order to cover the distribution of control functions between services and the lay-out and management of devicebased sensor, actor and control services. Particular emphasis was placed on the dynamic lease-based binding of services which on the one hand provides flexible and loose coupling of system components but on the other hand has to ensure reliable communication and cooperation. The guidelines were applied to the experimental implementation of a manufacturing cell control system using a real-time version of the Java Runtime Environment.*
The Device Profile for Web Services (DPWS) was used as basic infrastructure technology. Test and evaluation were performed under distributed simulation of technical processes and devices.

1. INTRODUCTION

Today, many modern business applications adhere to the paradigms of service orientation and service oriented architectures in order to create loosely coupled, modular software systems, easy to maintain and to extend. In the field of automation and control systems, SOA-based flexibility is of even more interest, because it contributes to substantial reductions of installation and setup costs .
These costs are of particular importance since manufacturing plants again and again have to be adapted to new products resulting in changes of the technical equipment and the process flows performed.
Additional reconfigurations are applied occasionally in the course of repair measures in order to bypass defect equipment and to avoid expensive production downtimes.
Despite the desired flexibility, however, there is a needs for stable and reliable operation phases since the efficiency of the production equipment usually depends on steady operational conditions.
For a certain manufacturing operation usually an ensemble of suitable devices, machines and transport equipment is necessary.
The members of the ensemble must initially be configured in harmony with each other and thereafter be available for a certain minimal period of operation time, which may only be aborted due to exceptional circumstances.
The members of the ensemble have to be allocated before configuration, some of them because they can only be used exclusively, others may be sharable but have to allow for the additional load.
In the service-oriented setting this means, that a client – which may be either a control application or a compound service – must be able to search, find and allocate a suitable ensemble of used services.
Since a used service may already have other obligations, it may not be disposable and deny a current allocation request.
Then, one member of the planned ensemble fails, and the ensemble as a whole is currently not useful.
Therefore, the client shall be able to withdraw the other allocation requests and look for alternative ensembles.
In order to fulfill these functional requirements of temporary and atomic ensemble allocation we extended the approach of lease-based allocation by introducing an explicit reservation phase in a way that reservation and allocation perform a two-phasem commitment.
Moreover we transposed the architecture of hierarchical control systems to the field of service systems using the platform the Device Profile for Web Services (DPWS) as basic infrastructure technology supporting the communication between devices via service interfaces as well as the exploration and binding of services.

The application of the resulting architecture guidelines and the usage of the lease-based allocation were exemplified by means of a production cell scenario using a real-time Java Runtime Environment.
In the sequel, we outline DPWS and its application to service-oriented industrial applications.

2. SERVICE ORIENTED ARCHIECTURES

In SOA, interoperability of different platforms is established through the definition of common communication protocol and message exchange standards.
But not only in enterprise domain software service-orientation is a feasible way of creating flexible software systems, as through the growth of computing power of embedded devices these paradigms are also applicable to embedded software solutions.
Universal Plug'n'Play (UpnP was the first specification of a service oriented infrastructure to be used in embedded application scenarios, using SOAP and HTTP as a basic communication layer and providing mechanisms for service discovery, action invocation and event based communication schemes.
Its successor, the Devices Profile for Web Services (DPWS) , is completely based on standardized Web service specifications and defines a profile (a subset) for the use of Web service technology in the embedded domain.

2.1. Devices profile for web services

The Devices Profile for Web Services defines a common subset of web service based communication patterns for use in embedded devices.
The protocol stack utilizes standardized internet protocols, namely TCP/IP and UDP (Single- and Multicast). For basic messaging HTTP and SOAP respectively SOAP-over- UDP are employed.
On top Web service protocols are arranged that deal with service and device description, discovery, eventing and security.
A DPWS device may host several services, which can be discovered and used by DPWS clients. The DPWS protocol stack is depicted in Figure 1.

2.2 SOA in industrial automation

The emergence of powerful but less power consuming, affordable, and embedded computing components facilitates the employment of SOA paradigms even in the world of industrial automation
Currently a lot of proprietary standards in device control and communication protocols often prevent the vendors.
Thus upgrades or extensions of the manufacturing automation system tend to be costly and time consuming.
The usage of SOA in industrial automation provides a common ground for interoperability of all devices in a device network.
Moreover an integration of low-level devices and highlevel enterprise applications (e.g. an ERP system) is possible.
In the European ITEA SIRENA project the applicability of DPWS in an industrial automation scenario was demonstrated for the first time.

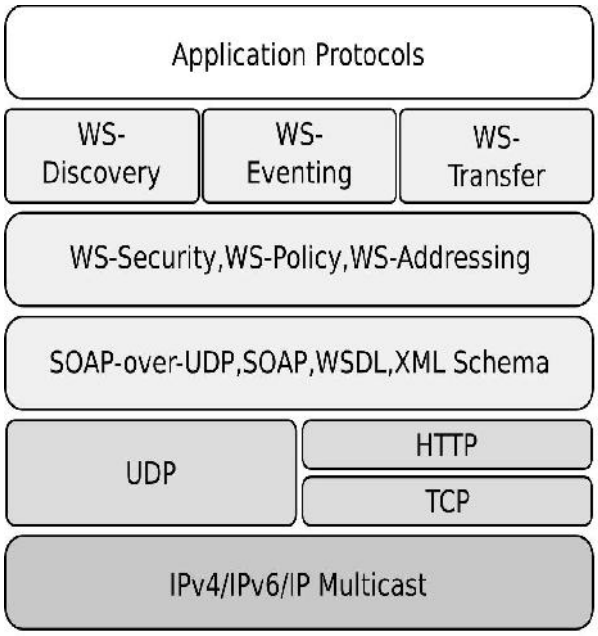

Figure 1. DPWS protocol stack

3. AUTOMATION AND CONTROL

An industrial control system commonly has a structure as depicted in Figure 2. This architecture could be divided into three main layers: sensors and actors, control and management.
The actual technical process is located at the bottom of the control hierarchy and subsumes all technical lowlevel components involved in the production process like motors, pushers or drilling machines.
The process is monitored by sensors, collecting data from the involved resources including e.g. temperature, rpm or the position of work pieces (indicated by a light-barrier state change).
This information is send via a specialized communication infrastructure to the process control level and is repeatedly evaluated by the control algorithm.
Based on the sensor information the control algorithm computes control signals which are in turn send to the actuators connected to the technical process.
Moreover status information from the process control level is sent to the process management level.
This may include forwarded sensor values, proinformation and fault messages.
At process management level a human operator monitors the overall process behavior, adjusts particular parameters and sends configuration commands to the process control system.
Besides the remote high-level controlling and monitoring of the technical process, in some occasions (e.g. a severe fault that requires local intervention and repair) the operator may be forced to directly intervene with the low-level hardware components via the attached control pan.

4. SERVICE CONTROL ARCHITECTURE

The process control architecture shown in the last paragraph is the structural basis for the service-oriented architecture presented in this paper.
The serviceorientation of the devices involved in the technical process and the attached sensors suggests the use of service-orientation also on the control and management levels.
The sensors and actuators export their functionality through defined interfaces which can be used by higher level control services.
Control services may also be layered and arranged in a service hierarchy.
Figure 3 illustrates this architecture: the application process interacts with the technical process using the supplied control services.
The control services themselves are acting both as a service consumer (client role) and service provider (server role) and thus enable control service layering.
For example, a rotary disk consists of a rotation motor and a motor for moving the conveyer belt on top of the disk.

Additionally the disk is supplied with sensors, detecting the location of the work piece currently transported on the conveyer belt and a sensor to measure the position of the rotary disk itself.
Both, the rotary part and the transportation part are each controlled by their own control service.
For the control of the overall process of moving a work piece on the disk, stopping the conveyer, turning the disk to its new position and finally transporting the work piece away from the rotary table, an additional control service is provided that uses the control services of the particular parts of the rotary disk.
Therefore the control services themselves offer service functionality to higher level control or management services.
However, the stacking of control services is constrained by the real-time requirements of the process, as each new layer of control implies additional, time consuming
communication between the services
.

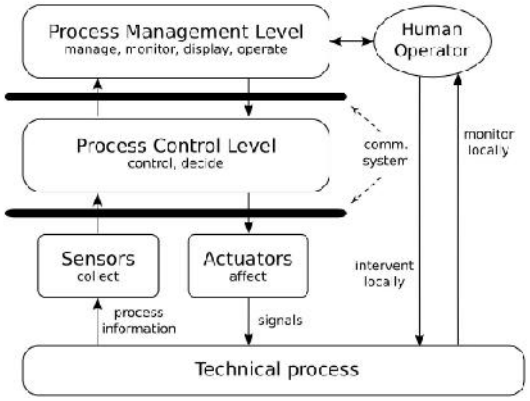

Figure 2 Control system

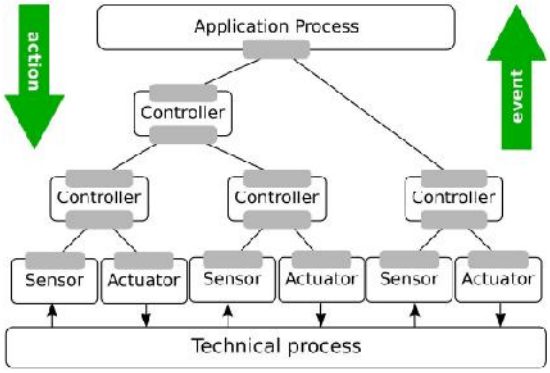

Figure 3. Service hierarchy

5. CONCLUSIONS

The services (e.g. sensor or controller) offer different interfaces which can be categorized using the follow three classes:

- functional purpose
- discovery and description
- service binding

The functional interface offers the functionality of the service, e.g. a getVariable method for sensor or a setVariable method for actuator services.
The functional service interface of control services offers high-level methods like drillHole.
The control services comply with the notion of so called function building blocks (IEC 61499).
Each building block comprises input and output variables plus local status variables.
The functionality of a particular function block is defined by the algorithm that is used to compute the outputs by using the inputs and the local variables.
The discovery interface contains the necessary methods for services to be able to answer to search requests and to provide data concerning device type, location and binding address.
Finally, the binding interface subsumes the features for lease based service binding and reservation.

6. REFERENCES

[1] H. Smit, F. Jammes, “Service-Oriented Paradigms in Industrial Automation”, IEEE Transactions on Industrial Informatics, Vol. 1, No. 1, pp. 62-70, 2005.
[2] C. Gray, D. Cheriton, “Leases: an efficient fault-tolerant mechanism for distributed file cache consistency”, ACM SIGOPS Operating Systems Review, Vol. 23, Issue 5, pp. 202-210, Dec. 1989.
[3] Universal Plug and Play (UPnP), http://www.upnp.org, 1999.
[4] Devices Profile for Web Services (DPWS), http://schemas.xmlsoap.org/ws/2006/02/devprof/, 2006.
[5] Service Infrastructure for Real-time Embedded Networked Applications (SIRENA), http://www.sirena-itea.org, 2006.
[6] Sun Microsystems, Jini, Network Technology, http://www.sun.com/software/jini, 1999.

[7] Kapsers, Küfner, “Messen – Steuern – Regeln: Elemente der Automatisierungstechnik”, Vieweg Verlag, 6th Edition, p. 253, 2006.
[8] WS4D.org Java Multi Edition DPWS Stack, http://www.ws4d.org, 2007.
[9] Sun Java Real-time System 2.0 (Java RTS), http://java.sun.com/javase/technologies/realtime, 2007.
[10] PROFINET, http://www.profibus.com/pn/, 2007.