Building a Robotic Cyber-Physical Production Component

Paulo Leitão[1,2], José Barbosa[1]

[1] Polytechnic Institute of Bragança, Campus Sta Apolónia, 5301-857 Bragança, Portugal
[2] LIACC - Artificial Intelligence and Computer Science Laboratory, Rua Dr. Roberto Frias, 4200-465 Porto, Portugal
{pleitao,jbarbosa}@ipb.pt

Abstract. Cyber-physical systems are a network of integrated computational decisional components and physical elements. The integration of computational decisional components with the heterogeneous physical automation systems and devices is not transparent and constitutes a critical challenge for the success of this approach. The objective of the paper is to describe an approach to establish standard interfaces based on the use of the ISO 9506 Manufacturing Message Specification international standard. The proposed approach is exemplified by the construction of a robotic cyber-physical production component that is plug-in in a cyber-physical system for a small-scale production system based on Fischertechnik systems.

Keywords: Cyber-physical systems, industrial agents, integration.

1 Introduction

The manufacturing world is being subject to a paradigm shift, both at the organizational and control levels, facing the current demands for more robust, flexible, modular, adaptive and responsive systems. Several opportunities arise for the introduction of new and innovative approaches, such as the Cyber-Physical System (CPS) approach. This CPS approach is being supported by strong financing measures, such as the European Horizon 2020 framework or the German Industrie 4.0 initiative, leveraging a new industrial revolution and capturing the attention of academia or industry.

CPS constitutes a network of interacting cyber and physical elements aiming a common goal [1]. A major challenge is to integrate the computational decisional components (i.e. cyber part) with the physical automation systems and devices (i.e. physical part) to create such network of smart cyber-physical components. However, this integration is not transparent and constitutes a critical challenge for the success of this approach. In fact, it is not easy and transparent the integration of heterogeneous automation devices, such as sensors, robots, numerical control machines or automation solutions based on Programmable logic Controllers (PLCs), which usually requires a complex and time consuming activity. To face this problem, the challenge is to define standard industrial interfaces that allow a completely transparent development of the computational decisional components without knowing the particularities of the automation device; in such process, these interfaces may be developed by automation providers or system integrators and (re-)used by the system developers.

The objective of the paper is to describe an approach to establish such standard interfaces based on the use of the ISO 9506 Manufacturing Message Specification (MMS) international standard [2], initially introduced by ADACOR holonic control architecture [3]. This approach is exemplified by deploying a cyber-physical production component for an industrial manipulator robot, which is part of a small-scale production system based on Fischertechnik systems.

The rest of the paper is organized as follows: Section 2 overviews the concept of cyber-physical systems and identifies the integration of computational components with automation devices as a critical challenge for its industrial implementation. Section 3 presents an approach to engineer cyber-physical production components and Section 4 illustrates its applicability by developing a robotic cyber-physical production component. Finally, Section 5 rounds up the paper with the conclusions.

2 Overview of Cyber-Physical Systems

Embedded systems have been in use for many years. They can be characterized by the conjunction of computational, electrical and mechanical capabilities, being often executed in real-time and providing some sort of intelligence to the system. Embedded systems are present everywhere and in different sectors, such as civil infrastructure, aerospace, energy, healthcare, manufacturing, transportation. Examples are vendor machines, cars' Automatic Breaking System (ABS) or even elevators.

With the widely improvement and spread of communication technologies, namely wireless communication and optical fiber, used currently in internet infrastructures, these embedded systems have gain a mean to share information, cooperate and collaborate each other. This missing communication capability and the collaboration inability of embedded systems gave rise to CPS. In this way, CPS can be defined as a triad of computation capabilities with a control component, interconnected over a communication channel, as depicted in Fig. 1.

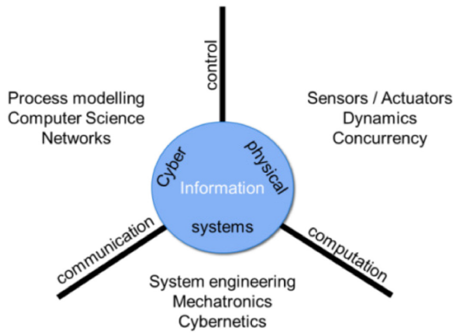


Fig. 1. Cyber-physical system triad [4]

The cyber, or logic world, can be found at the upper level of this triad, being responsible for bringing the logic and intelligence features found in CPS. At the lower level resides the physical process, being often composed by the combination of electromechanical components. This triad is complete by the communication capability, tying every Cyber Physical Component (CPC) from which the CPS is built upon.

The application of CPS will impose tremendous changes at several levels, particularly in the way the systems are designed and how they interact. Referring to Fig. 1, at the cyber level, the challenges are related to process modelling, computer science and communication networks, while at the system level, new system engineering methodologies, developments in mechatronics and a new cybernetics discipline approach will be mandatory. At the physical level, the challenges are related to sensors and actuators, dynamics and concurrency. In manufacturing, CPS will imply the migration from the typical ISA-95 organizational structure, where all levels are vertically interconnected, into a decentralization of these levels, meaning that the components/applications placed at different levels can access to data provided by others.

An important feature is the need to integrate the cyber part, i.e. computational decisional components, with the physical world, which are responsible to sense, process and act on the environment. This fundamental feature, particularly in manufacturing, implies several challenges where the definition of standardized interfaces assumes a crucial importance to handle the usually heterogeneous automation device presented at shop floor.

3 Engineering Cyber-Physical Components

The engineering of cyber-physical production components requires the integration of the computational and physical automation counterparts. The computational components may use the agent technology [5] to implement the intelligence and adaptation layer that will control the automation hardware (HW) device. Intelligent software agents developed in this context are known as industrial agents, which are faced with industrial requirements, namely HW integration, reliability, fault tolerance, scalability, industrial standards compliance, resilience, manageability, and maintainability [6]. Additionally, the integration of computational and physical automation counterparts recalls the holon concept, which is composed by an informational component (the agent) and the physical component (the HW device if exists) [7].

The integration of cyber and physical components can be performed in two different manners, namely embedding the agent within the physical control device or connecting the agent with the existing control device in a coupled manner [8-9]. Independently of the use of these two approaches, it is necessary to create a standard approach that allow the transparent and independent development of the computational entities from the heterogeneity and particularities of the HW automation device and communication infra-structure, which can expose their functionalities in terms of services. This imposes a crucial challenge for the engineering of these components regarding the establishment of standard interfaces, focusing the semantics and the protocols, and industrial middleware.

Having this in mind, ADACOR holonic manufacturing control architecture [3] proposed an approach based on the use of standard interfaces, using the service-oriented architectures (SoA) principles, where the physical automation resource is abstracted in form of standard services, as illustrated in Fig. 2. These services were defined based on the ISO 9506 Manufacturing Message Specification (MMS) standard [2], which defines the syntax and semantics for a set of clusters of services for automation domain, which are invoked by the agent (which plays the role of client) independently from the device particularities. MMS was designed as a control and monitoring specification for the OSI application layer, enabling the cooperation between applications and/or devices at the shop-floor. This specification is used in this approach to provide the necessary interface specification guidelines, namely allowing to define standard services that expose the functionalities of the HW automation device, namely in terms of variable handling, program handling and events [2].

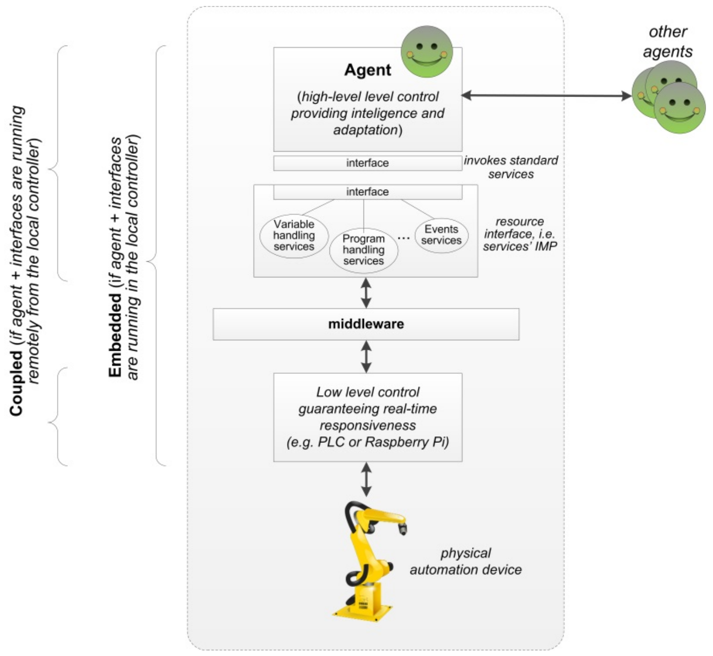

Agent
(high-level level control providing inteligence and adaptation)
other agents
interface
invokes standard services
interface
Variable handling services
Program handling services
...
Events services
resource interface, i.e. services' IMP
middleware
Low level control guaranteeing real-time responsiveness (e.g. PLC or Raspberry Pi)
physical automation device
Coupled (if agent + interfaces are running remotely from the local controller)
Embedded (if agent + interfaces are running in the local controller)

Fig. 2. Integration of agents with low level automation functions to form a cyber-physical production component

These services are implemented, usually by system integrators, in the server component according to the particularities of the device available at the shop-floor (from different types, e.g. robots and numerical control machines, and from different automation providers) and the communication infra-structure (e.g. serial communication, Modbus or OPC-UA). These services, after being developed, can be re-used and offered as drivers or wrappers, to be used in a pluggable and modular manner by other control applications for similar resources.

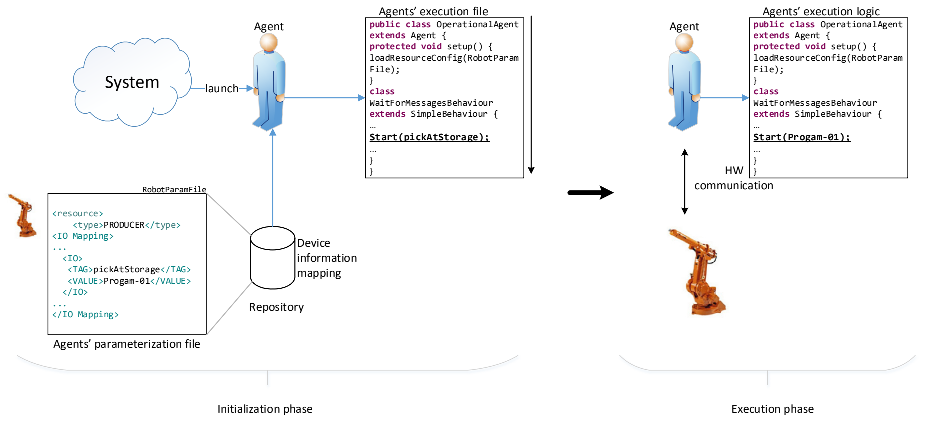
The transparent and standard invocation of these services by the computational entity requires the specification of the syntax of each service, i.e. the definition of input and output parameters. As example, the services available in the *Program Invocation*

Service package, which are invoked in a unique way by the client side, i.e. the agents, are the follow.

```
public interface ProgramInvocationService {
   boolean CreateProgramInvocation(String program);
   boolean DeleteProgramInvocation(String program);
   …
   boolean Start(String program);
   boolean Stop(String program);
   boolean Resume(String program);
   boolean Reset(String program);
   boolean Kill(String program);
   Attributes GetProgramInvocationAttributes(String program);
   boolean Select(String program);
   Attributes AlterProgramInvocationAttributes String program, At-
tributes att);
   …
   boolean ReconfigureProgramInvocation(String program);
}
```

Note that, as an example, whenever an agent needs to start the execution of a robotic a program, it uses the service *Start (String program)*, where *program* represents the program to be executed.

At this point, agents invoke standardized services based on an abstraction layer, which enables the transparent design and development of agents. The challenge here focuses in the abstraction level that the standard interfaces impose to the generic development of agents. In fact, this abstraction culminates with the generalization of the parameters of the methods to be executed, where two instantiated agents from the same type have the same method invocation but in reality the physical access differs in each case. For instance, the parameter writing in a memory space differs from one agent to the other, accordingly with the HW to be accessed (e.g. the physical address where the temperature sensor is connected to the PLC).



```
public class OperationalAgent
extends Agent {
protected void setup() {
loadResourceConfig(RobotParam
File);
}
class
WaitForMessagesBehaviour
extends SimpleBehaviour {
…
Start(pickAtStorage);
…
}
}
```

```
<resource>
    <type>PRODUCER</type>
<IO Mapping>
...
  <IO>
   <TAG>pickAtStorage</TAG>
   <VALUE>Progam-01</VALUE>
  </IO>
...
</IO Mapping>
```

```
public class OperationalAgent
extends Agent {
protected void setup() {
loadResourceConfig(RobotParam
File);
}
class
WaitForMessagesBehaviour
extends SimpleBehaviour {
…
Start(Progam-01);
…
}
}
```

Fig. 3. Agent-HW interface: initialization and execution phases

As illustrated in Fig. 3, initially when launched, each agent parameterizes the generic parameters used in the MMS-based interface layer in a *xml* configuration file. This file contains pairs of *{tag, value}*, where *tag* represents the service parameter name used in the agent development, while *value* points to the physical/logical connection that the agent need to access. It is worthy to be noted also that *value* can represent more complex structures aside direct/simple type. On example of such pair is *{part_input, IRB1400.signaldi.DI10_1}*, where *part_input* represents the sensor that detects that a part is at the beginning of a conveyor, and *IRB1400.signaldi.DI10_1* is the physical label used in the OPC server. This is crucial to guarantee that two similar automation devices have the same processing logic from the agent's point of view but due to their HW differences they must be parameterized differently.

4 Deploying a Robotic Cyber-Physical Component

This section describes the application of the described engineering approach to deploy a robotic cyber-physical component to be used in an agent-based control system for a small scale production system.

4.1 Description of the Case Study

The robotic device is an IRB 1400 ABB robot that is part of a real small-scale production system, which also comprises two punching machines and two indexed lines supplied by Fischertechnik™, as illustrated in Fig. 4.

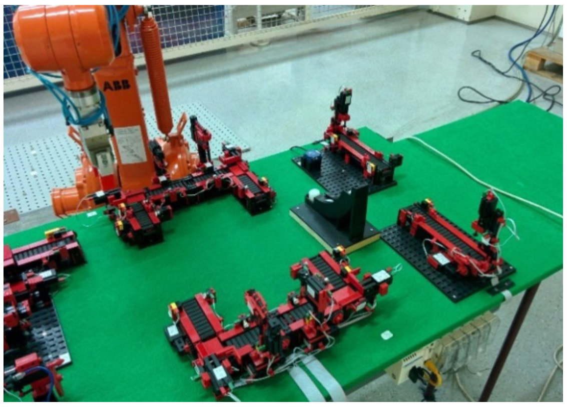


Fig. 4. Layout of the small-scale production system

The punching and indexed machines are controlled by IEC 61131-3 programs running in a Modicon M340 PLC. Two different parts circulate in the system, each one having a particular process plan. The circulation of parts within the flexible production system is tracked by a radio-frequency identification (RFiD) reader. An industrial manipulator robot executes the transfer of the parts between the machines using proper RAPID programs and is accessible through the ABB S4 DDE Server (that can be accessed by OPC).

The idea in this work is to describe the way the cyber-physical production component for the industrial robot was engineered, and particularly how the software agent, which is providing intelligence and adaptation to the robot, was interfaced with the physical controller of the automation device.

4.2 Development of the Software Agent

An ADACOR-based system [3] was developed to control, in a distributed manner, this small-scale production system, using agent technology to implement the control logic, i.e. the cyber part. For this purpose, product, task, operational and supervisor holons were developed to represent the system components, each one contributing with their knowledge and skills to achieve the system's goals, by interacting through several cooperation patterns. Particularly, two product holons were created, one for each product type defined in the catalogue of available parts. Task holons are launched by the associated product holons according to the order demand. The ecosystem of heterogeneous resources has associated an operational holon (OH) for each one, namely for the two punching machines, for the two indexed lines, for the RFID reader, for the manipulator robot and for the human inspector. Finally, aiming to introduce production optimization into the system, a supervisor holon is also considered. The computational decisional component of these holons is implemented as software agents. The dynamic behaviour of these agents was modelled using the Petri nets formalism [10], which is a suitable approach to support the formal analysis and simulation of the desired agent functionality, allowing to detect undesired behaviour or possible execution deadlocks at the design stage.

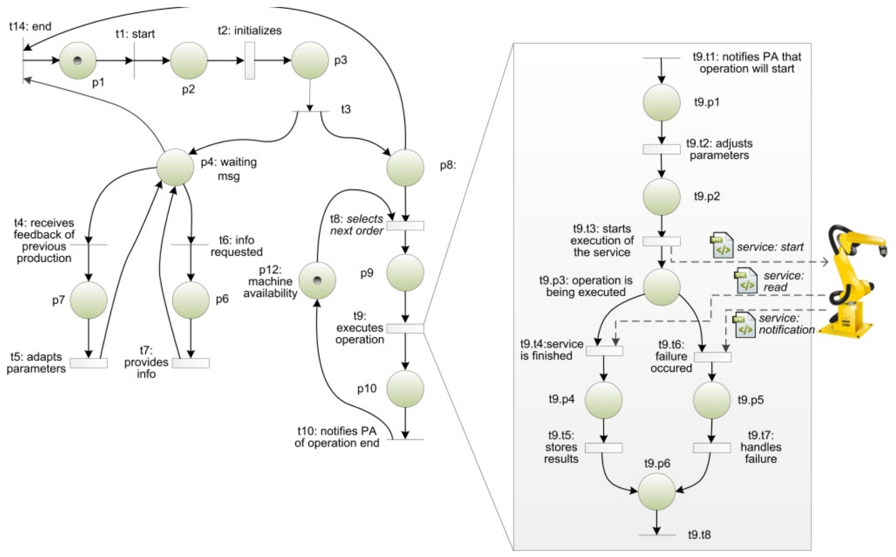


Fig. 5. Petri net model for the behaviour of the Operational Holon

In particular, a software agent is managing the activities of the robotic device, introducing intelligence and adaptation do this automation device. The Petri net model representing the dynamic behaviour of this software agent is illustrated in Fig. 5.

The agents' behaviours were implemented using the well- known JADE framework [11], enabling the development of an intelligent and distributed architecture in a transparent manner. Despite this, and since the underground technology used in JADE is the Java™ programming language, a Java Virtual Machine (JVM) container is mandatory as the support to the developed agents. The need to have this JVM limits the number of devices that have the HW resources necessary to accommodate JADE agents. JADE agents follow the object oriented paradigm and consequently they use objects, methods and threads as the core components.

As illustrated in Petri nets model, the agent invokes several services defined in the resource interface, namely the *start* service to start the execution of a pick-and-place program, the *read* to detect the end of the robotic program and *notification* to warm about the occurrence of a failure during the program execution. The invocation of these services is made in an undistinguished manner, and without knowing the particularies of the automation device.

4.3 Integrating the Automation HW Device

Having the interface defined, particularized instantiations of those services are performed according to the existing HW in the system (and particularly their controllers). Illustratively, two different examples, implementing the read of a bit, one using the OPC connection to a server and the other using a Modbus command, are described in the following.

```
public Boolean read (String var ) {
   ...
   JIVariant intaux = null;
   String straux = null;
   try {
     final Item item = group.addItem (var);
     item.setActive (true);
     intaux = item.read (false).getValue();
     straux=intaux.toString();
     String straux1 = straux.substring(2, 3);
   }
   catch ( final JIException e ){e.printStackTrace();}
   return(extractValue(straux1));
}
```

where the *var* parameter contains the specification of the PLC type and address extracted from the *xml* configuration file. The same read interface, using a Modbus communication protocol is now recoded using the following code excerpt.

```
public Boolean read (String var) {
     ...
     try {
        int regReference = Integer.parseInt(var);
        rcreq=new ReadCoilsRequest(regReference, 1);
        trans=new ModbusTCPTransaction(con);
        trans.setRequest(rcreq);
        trans.execute();
        rcres=(ReadCoilsResponse)trans.getResponse();
    }
    Catch(Exception e){e.printStackTrace();}
     return(extractValue(rcres.getCoils().toString().trim());
}
```

More examples could be given using the same approach, namely interfacing different robot controllers or using different communication infra-structures, e.g. a serial communication channel.

It is worthy to mention that all the aforementioned examples use a decoupled approach, where the agent control layer, due to the JVM needs, is not directly deployable into the controlled HW, i.e. into the robot controller. Despite this, the standard interfaces approach is also used when a direct HW control can be performed, using an agent coupled approach, as in the case of a Raspberry Pi [9].

The experimental tests shows that this approach simplifies the development and deployment of agent based systems in the control of physical devices. On one side agents developers can only focus on developing the desired agents' functionalities, while, on the other side, automation integrators can focus on developing these interfaces, parameterized according to the particularities of the physical HW devices.

5 Conclusions

This paper presents a simple and effective approach to develop standardized interfaces that can be used to access physical automation components by the cyber layer in CPS. The proposed methodology uses the MMS standard as the ground base for the specification of the interface layer. This abstraction layer allows a fast integration of the agent with the physical world, being only necessary the implementation of the service interfaces and the parameterization of an *xml* like file that maps the used parameters, or tags, in the agent development to the real physical/logical address.

In ADACOR holonic architecture, at this stage, only the OHs use the proposed methodology. Despite this, all the other holons can benefit from this approach, where, e.g., the SH has access to different mathematical solvers, and the THs and PHs have access to their different data sources, through the use of a common interface to access different legacy systems.

In this work, this approach was exemplified to build a robotic cyber-physical production component that is deployed in a cyber-physical system for a small-scale production system. Although the use of the simplicity of the proposed approach, it turn out to be a very effective solution allowing the fast development and deployment of industrial agent-based systems.

The experimental development, parameterization and deployment of the agent based system were successfully achieved. In fact, after having the underlying OHs agents' logic developed, the tasks needed to fully complete the integration was to instantiate the libraries of the different communication protocols present at the system, namely an OPC server, a Modbus communication and a serial protocol, and to create the parameterization files. Although every communication protocol could be wrapped around the OPC server, it was opted to develop this amount of libraries in order to further push this methodology.

As future work, this approach will further forester the CPS where higher level applications could use the developed services to compose more complex services, e.g., a SCADA system could use a service that is also used by the agents. Additionally, this approach must be deployed in more test beds. Only at this point one can truly take advantage of this, since, expectedly, the deployment efforts would be greatly reduced.

References

1. P. Leitão, A.W. Colombo and S. Karnouskos. Industrial Automation based on Cyber-Physical Systems Technologies: Prototype Implementations and Challenges. Accepted for publication in Computers in Industry, Elsevier, 2015.
2. ISO/IEC 9506-1: Industrial Automation Systems - Manufacturing Message Specification, Part 1 - Service Definition, 1992.
3. P. Leitão and F. Restivo. ADACOR: A Holonic Architecture for Agile and Adaptive Manufacturing Control. Computers in Industry, vol. 57, nº 2, pp. 121-130, Elsevier, 2006.
4. M. Schmid. Cyber-Physical Systems ganz konkret. Technical Article, ELEKTRONIKPRAXIS, n. 7, 2014.
5. M. Wooldridge. An Introduction to Multi-Agent Systems. John Wiley & Sons, 2002.
6. P. Leitão and S. Karnouskos. A Survey on Factors that Impact Industrial Agent Acceptance, Industrial Agents: Emerging Applications of Software Agents in Industry, P. Leitão and S. Karnouskos (eds.). Elsevier, pp. 401-429, 2015.
7. M. Winkler and M. Mey. Holonic Manufacturing Systems. European Production Engineering, 1994.
8. L. Ribeiro. The Design, Deployment, and Assessment of Industrial Agent Systems. Industrial Agents: Emerging Applications of Software Agents in Industry, P. Leitão and S. Karnouskos (eds.). Elsevier, pp. 45-63, 2015.
9. J. Dias, J. Barbosa and P. Leitão. Deployment of Industrial Agents in Heterogeneous Automation Environments. Proceedings of the 13th IEEE International Conference on Industrial Informatics (INDIN'15), 22-25 July, Cambridge, UK, pp. 1330-1335, 2015.
10. T. Murata, "Petri Nets: Properties, Analysis and Applications," IEEE, vol. 77, no. 4, pp. 541-580, 1989.
11. F. Bellifemine, G. Caire, D. Greenwood. Developing Multi-Agent Systems with JADE. Wiley, 2007.