A field level architecture for reconfigurable real-time automation systems

Lars Dürkop[1], Henning Trsek[1], Jens Otto[2], and Jürgen Jasperneite[1,2]

[1]inIT – Institute Industrial IT, Ostwestfalen-Lippe University of Applied Sciences, D-32657 Lemgo, Germany {lars.duerkop, henning.trsek, juergen.jasperneite}@hs-owl.de
[2]Fraunhofer IOSB-INA Application Center Industrial Automation, D-32657 Lemgo, Germany {jens.otto, juergen.jasperneite}@iosb-ina.fraunhofer.de

Abstract

Rapidly changing customer demands lead to a paradigm shift from mass production to mass customization within the manufacturing industry. However, todays production systems are of a very static nature. Changing the manufacturing process requires a high amount of expensive human resources and is quite error prone. Hence, reconfigurability will become a key factor in the manufacturing industry and industrial automation systems must provide suitable solutions to support this new paradigm.

Service-oriented architectures (SOAs) are a potential technology which can provide the requested capability of automatic reconfiguration. Originating from the IT world, the adaptation of SOAs to industrial automation systems has to face several difficulties – especially real-time requirements must be met. This paper proposes an innovative solution approach for the integration of a SOA into real-time systems for industrial automation.

1 Introduction

Today, the manufacturing industry faces challenges due to the changing demands of its customers. Formerly, the cost-effective mass production of standardized products was a key factor for the competitiveness of a company. Nowadays and in the future the customization of products will become more and more important. Furthermore, product life cycles are about to become much shorter. This requires new manufacturing system paradigms which enable the manufacturer to react fast and cost-effective on market changes and individual customer demands [1] [2]. Two of these paradigms are flexible and reconfigurable manufacturing.

In flexible manufacturing systems (FMS) the variability is built-in a priori in the production process. An FMS consists of manufacturing cells with different capabilities and a connecting transportation system which allows various work flows. Variants of a product can be produced by choosing another subset of the available production tools. Since no physical changes of the manufacturing system itself are possible, the flexibility of a FMS is limited to the pre-defined boundaries of the system [2].

On the contrary a reconfigurable manufacturing system (RMS) allows physical modifications. For example, modules and machines can be added or removed from the production process to achieve new functionalities enabling a response to unforeseen requests [2]. Therefore, reconfigurable manufacturing is considered as the main production paradigm of the future [3]. The design of reconfigurable systems is an ongoing research topic. It covers different research fields, starting from the level of reconfigurable process planning down to the level of modular machine component design. The key challenge for industrial automation are control systems which must handle the newly introduced high degree of complexity. Since the control logic of today's production processes is rather static, it must be enabled to react with respect to the dynamic nature of an RMS.

One approach to control an RMS is the introduction of service-oriented architectures (SOAs) in industrial automation [4] [5]. A SOA consists of independent, but interoperable services. Each service exposes only its functionality to other services, the implementation is not visible from outside the service. Furthermore, all services are loosely coupled: They operate independently from each other, their interactions are stateless, asynchronous and not context-related [4]. In a SOA, a process (i. e., the production of a certain good) is composed as an aggregation of these services. An orchestration engine maps the process logic to the service level by connecting the services and schedules in their execution order. Finally, the SOA should offer a Plug-and-Produce (PnP) functionality, i. e., physical changes (e. g., removing or adding a device) lead to an automatic reconfiguration of the automation process on the basis of an abstract process definition.

The SOA paradigm originates from the information technology domain, where it is mainly used to implement

business processes on distributed systems. However, the requirements of industrial automation are contrary to the SOA approach in many cases – especially in the field of real-time communication. The remainder of this paper is organized as follows. In section 2 an overview of SOA approaches for industrial automation is given and existing problems are highlighted. An architecture to solve these issues is presented in section 3. To proof the feasibility a prototypical implementation of the architecture is described in section 4. The paper is concluded in section 5 followed by a brief outlook towards future work.

2 SOA in industrial automation systems – an overview

The SIRENA project [6] was one of the first approaches of porting SOAs to the industrial automation domain. SIRENA introduced a device-level SOA based on Web Services, the Device Profile for Web Services (DPWS), where each device offers its functionality as a service. For example a "Smart Motor" is given which exposes services like *RunMotor(duration)*. Web Services are a technology which supports the design of distributed systems. They consist of service providers which offer a functionality and service consumers which utilize that functionality. The interface of a web service is described in a machine-readable format like the XML-based Web Service Description Language (WSDL) [7]. The description contains information about the parameters the services expects and what data it returns. Furthermore, it determines the format in which the messages between services are exchanged. The most common protocol used for Web Services communication is SOAP over HTTP [8].

One advantage of web services is that basic services can be combined to generate new higher order services. So the production process of a product can be composed by using the services of different productions cells and transportation systems. This composition of services is called orchestration. An orchestration engine must offer possibilities to connect services, to schedule their execution order and to offer an interface of the newly composed service to higher layers. The most common specification to orchestrate web services is the Business Process Execution Language (BPEL) [9].

The device-level SOA approach can simplify the reconfiguration of production systems. Although the system developer has to change the orchestration logic, there is no need for going into details of data exchange and network communication, for example. The SOCRADES project introduced a framework offering the opportunity to orchestrate services flexible on the basis of petri nets [10]. However, every reconfiguration step results in manual effort. In order to achieve the objective of PnP, the production system must react autonomously to changes. Languages like WSDL and BPEL offer only syntactic description methods. They do not describe the meaning of the functionality, also called the semantics of a service, which is necessary to automate the orchestration process. This issue is addressed by semantic Web Services, whose best-known examples are SAWSDL, WSMO and OWL-S [11]. The semantics describe the meaning of a service definition and the relationships between production components and their services. To combine services to a fully functional production process the orchestration manager further requires a knowledge base in form of an ontology. The ontology could contain information about the available devices, their location and their dependencies [11] [12]. For example, in [13] a systematic procedure to define device profiles is shown. A proof-of-concept of a SOA-based production process is presented in [11] and [14].

Even though the SOA in automation approach has been discussed in the scientific community for more than eight years now and despite of its advantages, it is not established in industrial practice yet. In [15], technology- and human-related adoption barriers are identified. From a technology point of view, the resource constraint devices used in industrial automation have insufficient computing power for SOA communication protocols. Although this can be resolved in the future by the continuously increasing performance of hardware components, other problems regarding robustness, engineering tool support, safety and standardization must be addressed. On the human side, a conservative attitude towards new technologies in the industrial automation is stated. To convince engineers and managers of production sites, examples of SOA-based automation systems in comprehensive real scenarios must be successfully implemented.

From our perspective, there is another fundamental architectural problem when trying to realize SOA on a device level in industrial automation. To visualize this issue a SOA-based automation system, as defined by the SOCRADES project, is outlined in Figure 1 [10].

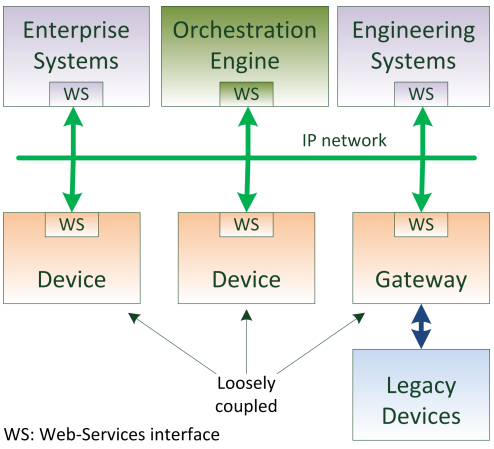

Figure 1. SOA-based automation system

As already mentioned a main premise of SOAs is the loose coupling of services. However, the resulting loose coupling of devices does not necessarily reflect the reality in the industrial automation. On the contrary, there are of-

ten close links between automation functions and devices like the need for real-time communication. For example, in a motion control application process data values must be sent every 100 µs with a latency jitter of less than 1 µs [16]. These demands cannot be met by current SOA implementations. Indeed, the SOA concept is not intended to fulfill such critical temporal constraints.

However, there are several approaches to introduce real-time capabilities to SOAs. The SOAP4IPC engine [17] analyzes the timing behavior of Web Services and determines the maximum execution time of a service. The Time-Constrained Services (TiCS) framework [18] offers tools to model the time dependencies of composed services. Based on the SOAP4IPC results, the framework checks if the dependencies can be met. Methods for improving the performance of SOA communication stacks are described in [19]. The authors suggest to replace the SOAP communication protocol by Efficient XML Interchange (EXI) [20], which is a binary XML representation.

All these solutions use SOA-inherent communication approaches based on the Internet Protocol (IP), in most cases based on standard Ethernet networks. Both technologies are not able to provide deterministic communication with low latency and jitter which is an essential requirement for many automation applications. Therefore real-time capable networks such as Real-time Ethernet (RTE) are used in the industrial automation. RTE is a general term for different communication network standards like Profinet IO, EtherNet/IP or EtherCAT. They all are based on standard Ethernet and use some modifications to provide real-time guarantees. One feature of RTEs is the backward compatibility to standard Ethernet (in most cases) which allows an easy integration into existing networks. The advantage of real-time communication is at the expense of an increased manual configuration effort.

In the iLAND project [21] methods for supporting time-bounded service operations have been developed. A middleware manages the communication between services considering temporal constraints. The middleware contains a placeholder for custom protocol stacks which could be replaced – in principle – by an RTE. However, no examples are provided on how to implement an RTE practically. Especially, the specific requirements concerning the complex configuration of RTEs are not considered.

In the next sections an architecture for reconfigurable automation systems is presented which integrates RTE into existing SOA concepts.

3 A reconfigurable architecture for the field level

In this section, the challenging implementation of real-time communication in SOAs is again addressed. After a short description of current industrial automation systems, a promising approach to extend these systems is proposed in order to enable SOAs to provide real-time guarantees.

3.1 Basic idea: Encapsulation of real-time communication

As stated in section 2 the SOA paradigm is limited whenever real-time communication is needed in industrial automation systems. However, real-time relationships exist only among a small group of devices in a clearly delimited part of the whole automation process. To distinguish between parts with and without real-time requirements we propose to introduce a module level in the automation process. Here, a module is characterized as a mechatronic unit which provides a certain functionality to the outside world. To realize this functionality it internally consists of several devices like sensors, actuators and typically one control device. Within a module there can be real-time communication between the devices, whereas there is none between the modules. This concept is depicted in Figure 2.

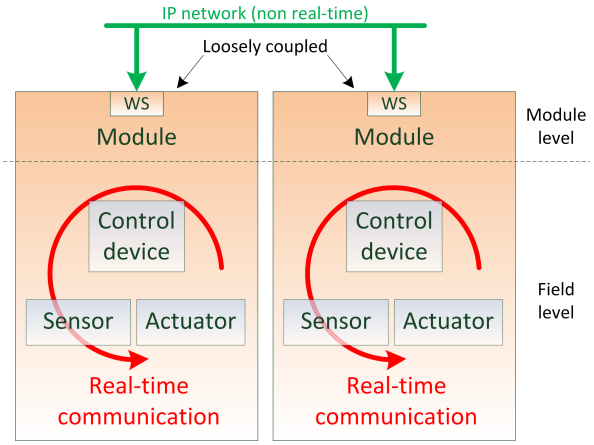

Figure 2. Segmentation of a automation process into modules

Instead of a device-level SOA, this approach introduces a module-level SOA where the services are not exposed by devices but by modules. All SOA principles like loose coupling, service composition and orchestration are still applicable to the module level. The critical real-time communication is shifted to the field level where it is encapsulated by the modules. The devices on the field level generally form the interface to the mechanic components of the automation process. In our opinion, this segmentation reflects the reality of the industrial automation in a better way. A similar approach is presented in [22] where a procedure for the dynamic orchestration of module-based services is presented. The author assumes that real-time communication exists only inside a module. The reconfiguration of module-internal communication is explicitly excluded in that work.

In [23] a SOA-based architecture for reconfigurable manufacturing processes is presented. Although the focus of that paper is on the computation of optimized production schedules, it also covers the integration of field

devices. The proposed architecture turns away from the device-level SOA concept, as well. Instead, on the field level the production process is divided into cells controlled by standard automation devices. OPC UA servers collect data from their allocated cells and form the interface to the service level by exposing basic services.

However, modules or cells in RMS cannot be considered as static units. Therefore, subsection 3.3 shows a field level architecture allowing the dynamic reconfiguration of a module.

3.2 Current industrial automation control systems

The architecture of a reconfigurable module will be oriented towards state-of-the-art industrial control systems. In this section an introduction to such a system is provided. A typical control system structure is shown in Figure 3.

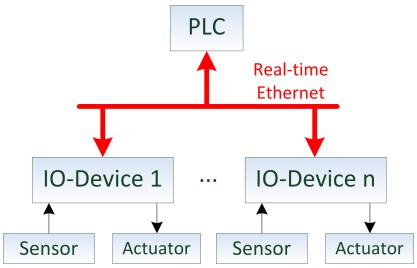

Figure 3. Current industrial automation control system

The sensors and actuators are connected to IO-Devices, which offer an electrical interface for them (i. e., supply voltage, digital/analog conversion and vice versa). The IO-Devices send process data from sensors to the programmable logic controller (PLC) or receive process data for actuators from the PLC. The software with the control logic is executed on the PLC. The process data between PLC and IO-Devices is transferred over an RTE. Setting-up such a control system requires several configuration steps, which are listed in the following.

1. The control logic has to be defined. This is usually done in a programming language according to IEC 61131-3. This program contains variables which represent the process data.
2. The RTE must be configured. Usually, the user has to define which IO-Devices are present in the network and an RTE-dependent addresses must be assigned.
3. The binding between a variable of the control logic and a concrete sensor/actuator signal must be defined.

The result of these steps is a very static automation process. After every reconfiguration the steps two and three have to be at least repeated. The aim of the architecture described in section 3.3 is to automate the reconfiguration procedure.

3.3 Module architecture

The starting point for the suggested module architecture is the standard automation control system described in subsection 3.2. At the end, that system and the module will execute the same functionality. The difference is that the module architecture supports automatic reconfiguration and has an interface to the upper level SOA.

Before the reconfiguration process of the module can start its control logic must be defined. Thereby the logic can be formulated independent of the module's hardware design. This allows the definition of abstract and reusable logic modules. The selection of the concrete devices, the used communication network, etc. is up to the user – as long as the hardware complies with the specifications within the logic module. A possible work flow could look like this:

1. The overall process logic is formulated or generated in a process description language like BPEL.
2. An orchestration engine analyzes the BPEL definition and derives the module-level services needed to execute the process.
3. The identified services including interface and logic definitions are chosen from a service directory.

In [24] an app-based approach for automation devices is shown. In that concept a function specific control software can be loaded onto generalized field devices - depending on the requested automation functionality. On this basis, the user could select the appropriate "control apps" matching to identified services. Another possibility to define the control logic is a model-driven development process. In [25] the automatic generation of IEC 61131-3 code from graphical UML and SysML models is evaluated. A method for control code generation by integration of knowledge from existing engineered artifacts like piping and instrumentation diagrams is presented in [26].

Besides choosing or defining the appropriate control logic, the engineer should be relieved from all other tasks. Especially, all tasks related to configuration which mainly includes the configuration of the RTE and the mapping between logic variables and device signals. As a result, there is no knowledge required about the used network technology during control logic definition. This abstraction between logic and technology facilitates the reconfiguration process – the control logic does not define which concrete physical devices must be used. Instead, it only specifies which information must be provided or consumed by the devices. The proposed architecture and its function blocks are depicted in Figure 4.

Like in a current automation control system, the architecture consists of one PLC where most of the new function blocks reside. The individual blocks are described in the next subsection.

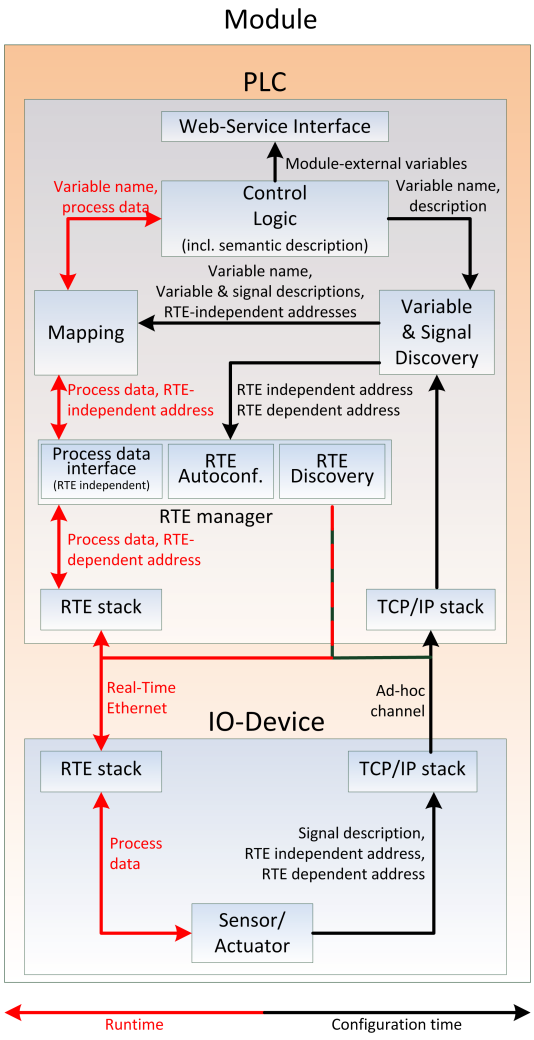

Figure 4. Architecture of a reconfigurable module

3.3.1 Control Logic

As mentioned before the control logic must be still defined manually. A part of this logic are the external variables, which represent the process data of the sensors and actuators. Normally, the engineer must map the variables to the corresponding devices manually. Here, the logic is independent from the underlying physical devices, and the mapping is done by another function block of the module architecture. The user only has to define the semantics of the variables, which describe the content and meaning of them. Therefore it is necessary to define an information model and a description language, which formalizes the model in a machine-readable way. Semantic description is also used in SOAs to describe the functionality of services [27].

In comparison, the challenge of defining an information model for variables respectively signals existing within a module is less complex. In this approach, only the content of a variable or signal must be described, whereas a service description also includes the functionality. With a simple information model the physical quantity (e. g., temperature, pressure, torque), which is expressed by a signal, could be assigned as semantic information, for example.

However, this model reaches its limits when one physical entity exists multiple times within a module. As fall-back strategy in such a case the user could be asked to manually choose the correct mapping. The information model which is included in the description language mINA-DL [28] includes additional attributes like neighbourhood information to solve the signal identification issue. In [29] the sorting of typical signals existing in an automation system in specific signal classes is proposed. The analysis of existing information models and the definition of a specific information model for the proposed architecture is part of future work.

At the configuration phase, the control logic sends its variable names and their semantic descriptions to the ”Variable & Signal Discovery” block. At runtime, the variables with their corresponding data are exchanged with the ”Mapping” block.

3.3.2 Web-Services interface

This function block forms the interface between the control logic and the external SOA. Therefore, it describes the functionality of the module as well as its input and output parameters. The interface could be formalized by using WSDL, for example. The nature of the data exchanged over the Web-Services interface is normally status-oriented since time-critical data resides inside the module.

3.3.3 Mapping

This architectural component has to perform two functions. At the configuration phase it gets the semantic variable descriptions from the discovery block. When using the information model mentioned above, an example description could look like this: *”Variable X is of type temperature.”* Furthermore, the discovery block provides the semantic descriptions from the IO-Devices and a corresponding RTE-independent address for each signal. On this basis, the mapping block merges the variables from the control logic and the signals from the IO-Devices and inserts the corresponding mappings to a mapping table. After the mapping process, this information is available: *”Variable X of type temperature is connected to the temperature-type signal at the RTE-independent address 0x0001”*. As long as an ambiguous assignment between variables and signals exists, the user has to be informed.

At runtime the mapping block routes the process data between the control logic and the RTE interface. For example, the mapping block could receive this data from the control logic: *”Variable X has the value 0x1a”*. It will

route the data to the RTE-independent process data interface of the RTE manager as follows: *"Send data 0x1a to RTE-independent address 0x0001"*.

3.3.4 Variable and signal discovery

The function of this block is the discovery of all IO-Devices available in the network and the retrieval of the semantic information of their signals and the variables of the control logic. For each signal also an RTE-independent and an RTE-dependent address is provided by the IO-Devices. Therefore, corresponding function blocks at the IO-Device are needed and it must be considered that these devices are generally very constrained in their resources. A possible solution to fulfil these tasks are current SOA implementations like DPWS and the Nano Embedded Device Server Profile of the OPC Unified Architecture (OPC UA) [30] which are designed for low-resource embedded devices. In this context SOAs are not used to define services of an IO-Device. Instead, the integrated mechanisms of both SOAs for device discovery and description are a suitable technical solution and therefore they are utilized to realize this function block. DPWS offers the WS-Discovery standard [31] to discover devices in a network without any pre-configuration by using multicast addresses. The signals of the IO-Devices can be described by WS-MetadataExchange [32] which is also part of DPWS. In the OPC UA discovery process the devices have to register themselves at dedicated discovery servers. The disadvantage of this procedure is that these servers have to be pre-configured at the IO-Devices. In OPC UA the used information model can be integrated directly into the address space of an OPC UA server. A comparison of OPC UA and DPWS is given in [33] and [34]. The latter also describes a lightweight OPC UA server implementation, the same is done in [35] for DPWS.

Both OPC UA and DPWS are using standard TCP/IP communication. For this purpose, the architecture consists of two separated logical channels: an ad-hoc channel for device and signal discovery and a real-time channel for process data exchange. The ad-hoc channel transports TCP/IP data and must offer its functionality without any manual pre-configuration needs. Therefore it is necessary that the used RTE allows zero-configuration TCP/IP traffic in parallel to the real-time process data. This is the case for Profinet IO (but not for the isochronous variant Profinet IRT), Ethernet/IP and Ethernet Powerlink, for example. Here, the RTE forms a common physical channel for both logical channels.

3.3.5 RTE manager

The RTE manager is one of the core components of the architecture. It offers an RTE-independent process data interface to the upper layer mapping block for process data exchange. The RTE manager converts the data and the addressing scheme according to the underlying RTE layer. The RTE stack of the PLC is then used to send the process data over the RTE. Traditionally, the RTE must configured manually before the process data exchange can start. For example, this step includes address allocation, configuring the format and length of the process data and several other important RTE parameters. To automate these steps the RTE manager provides an RTE autoconfiguration and discovery service. This service detects all available IO-Devices, explores their RTE dependent properties and finally configures the RTE stack of the PLC device. This procedure repeats every time when a device is connected to or disconnected from the network. The functioning of the RTE autoconfiguration is explained in detail in [36] and [37]. A very similar concept has been presented in [38] afterwards.

3.3.6 IO-Device

In this architecture the sensor/actuator is integrated into the IO-Device. Today, this is not always the case due to economical reasons, especially for very cheap sensors with a high quantity in the system, such as proximity sensors. Often both elements are separated, so that the cost-intensive interface to the RTE does not have to be integrated into each sensor/actuator. The decreasing costs of computing power could make it possible to generally include the RTE interface on the sensor level.

In addition, the combination of IO-Device and sensor in the described architecture contains a TCP/IP stack and provides the semantic description of the offered/requested signals and the corresponding addresses. This can be realized by DPWS or OPC UA, as stated in section 3.3.4. Adding more intelligence to this kind of devices is also a central point in the vision of the Internet of Things [39].

4 Implementation

In the following, a prototypical implementation of the architecture described in section 3.3 is given. To illustrate the implementation more clearly, its description is divided into two parts. First, the components and operations involved during the configuration phase are shown in Figure 5. Afterwards the behaviour during the process data exchange is explained.

4.1 Configuration phase

In this proof-of-concept the control logic is statically implemented according to IEC 61131-3. The control logic contains all variables which should be mapped to the sensor/actuators signals at a later stage. First of all, the semantic description has to be added to the variables. Therefore the program is exported from the IEC 61131-3 development environment to the PLCopen [40] format (step 1 in Figure 5). The resulting file contains all external variables and their data types in a human-readable XML-format. In step 2, the semantic description of each variable is inserted directly to the XML-file by the user. This is possible here, since in the used simple information model

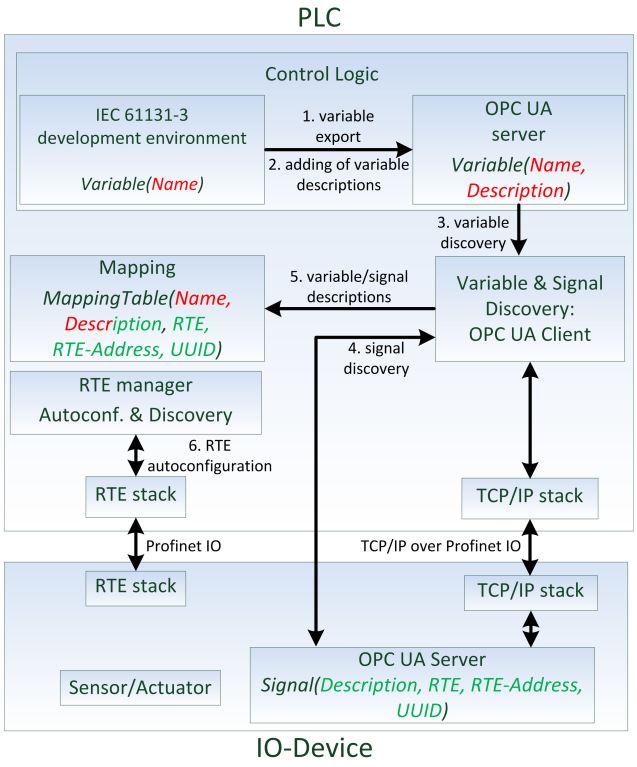

Figure 5. Implementation at configuration phase

every variable is described by a string only. When a more complex model is used, tool support for adding the semantic descriptions would be necessary. After the descriptions have been added, the XML-file is converted to an OPC UA information model description file.

When the automation system goes online after a reconfiguration, the OPC UA server of the PLC reads its model description file which defines the address space enriched with the variable names and their semantic descriptions. At the same time the OPC UA server of all IO-Devices start. In each server instance the description of the sensor/actuator signal has been included at manufacturing time. Also the supported RTE type of the IO-Device, the RTE-specific address and a unique RTE-independent identifier (UUID) of each sensor/actuator is pre-configured. Figure 6 shows a screenshot of the OPC UA information model of an IO-Device. The basis for this model is formed by the OPC UA For Devices (DI) specification [41] which is expanded by the signal descriptions. It is also possible that an IO-Device contains more than one sensor/actuator signal. In this case, the OPC UA server contains the corresponding number of signal descriptions.

When the OPC UA servers are running, they are discovered by the central OPC UA client (see [34] for the details of the discovery process). The client browses through

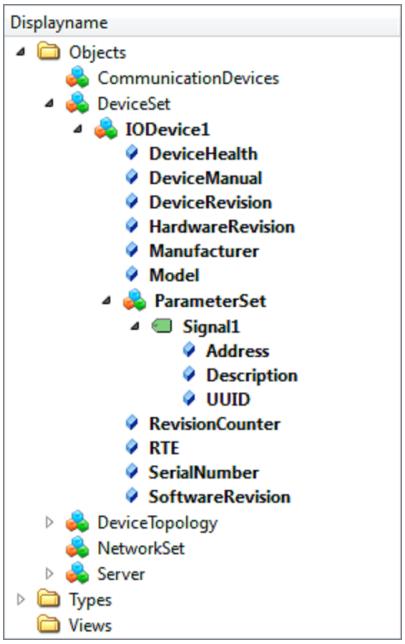

Figure 6. OPC UA information model

the address space of the PLC's server and collects information about the available variable names and their descriptions (step 3). From the servers at the IO-Devices the client gets the description, the supported RTE, the RTE-independent address and the UUID of each signal (step 4). In step 5 the information of all OPC UA servers are sent to a mapping service which maps the variables to the corresponding signals. In this implementation, a very simple matching algorithm is used: the strings which form the variable and the signal descriptions are compared. If they match, a new entry in a mapping table is generated. An user has to define the mapping for all variables and signals which have not been assigned to a counterpart. The complexity of the matching procedure grows with the complexity of the information model used for describing the variables and signals. One possible extension could be the integration of expert systems into the mapping algorithm.

Parallel to the semantic configuration process, the RTE manager starts in step 6 the RTE autoconfiguration procedure, so that after the configuration phase the process data exchange can start.

4.2 Operational phase

The function blocks involved in the process data exchange are depicted in Figure 7. The control logic is executed in a standard industrial IEC 61131-3 runtime environment. In this implementation sockets are used to enable data exchange between the environment and the mapping function block. This block replaces the variable names by the used RTE and the UUID retrieved from the mapping table and forwards the data to the process data interface of the RTE manager. The latter translate the upper layer write and read-calls to RTE-dependent functions.

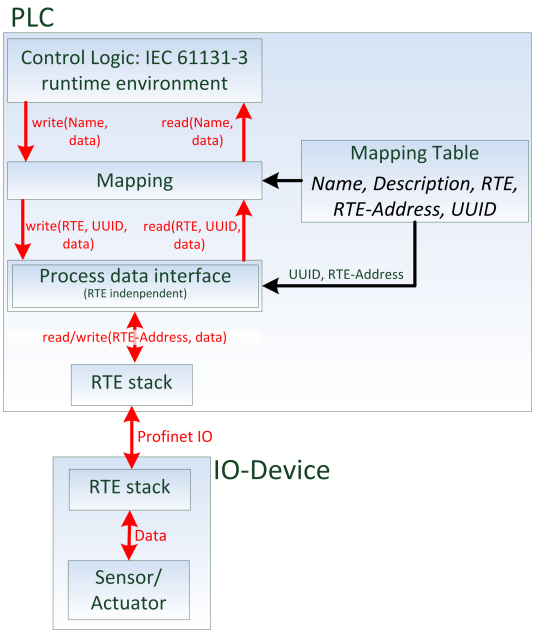

Figure 7. Implementation at runtime

In the presented implementation the data is exchanged via Named Pipes between the RTE manager and a commercial RTE stack.

4.3 Physical setup

The setup of the evaluation system for the implementation is shown in Figure 8. All PLC components are realized on a Windows PC. The RTE functionality is provided by a Profinet IO software stack from KW software [42]. On the IO-Device side, two alternative approaches were chosen. On the one hand, the needed OPC UA server functionality was implemented directly in the firmware of a Profinet IO-Device. Therefore an OPC UA server has been developed which has extreme small requirements in terms of memory (see also [34]). This is enclose to future solutions, where sensor, RTE interface and OPC UA form one single device.

On the other side, also standard IO-Devices are included in the evaluation system where no additional features can be added. Here, the OPC UA server is implemented on a small additional device like the Linux-based Raspberry Pi [43]. From an outside point of view, the Raspberry Pi and the standard IO-Device are considered as one logical device.

5 Conclusion and future work

This paper proposes an approach towards enabling reconfigurable production systems. Today, reconfigurability is not supported by industrial automation systems. Service-oriented architectures (SOAs) are generally considered as an enabling technology for reconfiguration in automation systems. Thus, this paper starts with an introduction to SOAs. Subsequently, technical difficulties are

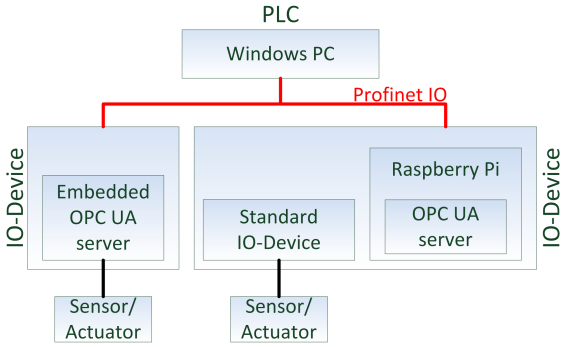

Figure 8. Physical setup of the evaluation system

discussed which arise from transferring the SOA approach to the industrial automation domain – especially the realization of real-time process data exchange is a challenging task. Therefore, this paper proposes to divide an automation system into a module and a field level. Here, the real-time traffic takes place at the field level. All devices with real-time communication relations are encapsulated into a module. On the module level existing SOA solutions can remain unchanged. To enable the automatic reconfiguration on field level, an appropriate architecture is proposed. The main components of this architecture are a mapping service based on the semantic description of variables and sensors/actuators as well as an autoconfiguration service for the underlying real-time Ethernet (RTE) network. The proposed solution facilitates a configuration-less startup of an automation module. The real-time data exchange is configured automatically according to the identified relations between the control logic and the field devices.

Since a few parts of the proposed architecture are still in the conceptional phase, future work will further investigate information models which are able to model signals appearing in an automation system. As a step towards such a model the communication relations in existing automation systems should be thoroughly examined. Based on these analyses a structured model could be designed. The implementation of this architecture in application scenarios with high temporal requirements on the communication system shall prove the feasibility of the presented approach.

Acknowledgement

This work was partly funded by the German Federal Ministry of Education and Research (BMBF) within the Leading-Edge Cluster ”Intelligent Technical Systems OstWestfalenLippe” (it’s OWL).

References

[1] G. Chryssolouris, *Manufacturing Systems: Theory and Practice*, Springer, 2006.

[2] H. ElMaraghy and H.-P. Wiendahl, "Changeability - An Introduction", in *Changeable and Reconfigurable Manufacturing Systems*, pp. 1–24. Springer, 2009.

[3] Y. Koren, "General RMS Characteristics. Comparison with Dedicated and Flexible Systems", in *Reconfigurable Manufacturing Systems and Transformable Factories*, pp. 27–45. Springer, 2006.

[4] F. Jammes, H. Smit, J. L. M. Lastra, and I. M. Delamer, "Orchestration of Service-Oriented Manufacturing Processes", in *10th IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, sep 2005.

[5] J. M. Mendes, P. Leitao, A. W. Colombo, and F. Restivo, "Service-oriented control architecture for reconfigurable production systems", in *6th IEEE International Conference on Industrial Informations (INDIN)*, jul 2008.

[6] F. Jammes and H. Smit, "Service-Oriented Paradigms in Industrial Automation", *IEEE Transactions on Industrial Informatics*, vol. 1, pp. 62–70, feb 2005.

[7] Web Services Description Working Group, (2007), Web Services Description Language Recommendation [Online]. Available: http://www.w3.org/2002/ws/desc/

[8] World Wide Web Consortium, (2007), SOAP Version 1.2 specification [Online]. Available: http://www.w3.org/TR/soap12-part1/

[9] OASIS, (2007), Web Service Business Process Execution Language [Online]. Available: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel

[10] A.-W. Colombo, S. Karnouskos, and J.-M. Mendes, "Factory of the Future: A Service-oriented System of Modular, Dynamic Reconfigurable and Collaborative Systems", in *Artificial Intelligence Techniques for Networked Manufacturing Enterprises Management*, pp. 459–481. Springer, 2010.

[11] A. Dengel, "Semantische Webservices zur Steuerung von Produktionsprozessen", in *Semantische Technologien*, pp. 285–314. Springer, 2012.

[12] M. Loskyll, J. Schlick, S. Hodek, L. Ollinger, T. Gerberg, and B. Pîrvu, "Semantic Service Discovery and Orchestration for Manufacturing Processes", in *16th IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, sep 2011.

[13] S. Hodek and J. Schlick, "Ad hoc field device integration using device profiles, concepts for automated configuration and web service technologies: Plug&Play field device integration concepts for industrial production processes", in *9th International Multi-Conference on Systems, Signals and Devices (SSD)*, mar 2012.

[14] S. Feldmann, M. Loskyll, S. Rosch, J. Schlick, D. Zuhlke, and B. Vogel-Heuser, "Increasing Agility in Engineering and Runtime of Automated Manufacturing Systems", in *IEEE International Conference on Industrial Technology (ICIT)*, feb 2013.

[15] A. Cannata, S. Karnouskos, and M. Taisch, "Evaluating the potential of a service oriented infrastructure for the factory of the future", in *8th IEEE International Conference on Industrial Informatics (INDIN)*, jul 2010.

[16] S. Vitturi, L. Peretti, L. Seno, M. Zigliotto, and C. Zunino, "Real-time Ethernet networks for motion control", *Computer Standards & Interfaces*, vol. 33, pp. 465–476, sep 2011.

[17] M. Mathes, J. Gärtner, H. Dohmann, and B. Freisleben, "SOAP4IPC: A Real-Time SOAP Engine for Industrial Automation", in *17th Euromicro International Conference on Parallel, Distributed and Network-based Processing*, feb 2009.

[18] M. Mathes, C. Stoidner, R. Schwarzkopf, S. Heinzl, T. Dörnemann, H. Dohmann, and B. Freisleben, "Time-constrained services: a framework for using real-time web services in industrial automation", *Service Oriented Computing and Applications*, vol. 3, pp. 239–262, dec 2009.

[19] F. Jammes, "Real time device level Service-Oriented Architectures", in *IEEE International Symposium on Industrial Electronics (ISIE)*, jun 2011.

[20] World Wide Web Consortium, (2011), Efficient XML Interchange (EXI) Format 1.0 [Online]. Available: http://www.w3.org/TR/exi/

[21] M. G. Valls, I. R. López, and L. F. Villar, "iLAND: An Enhanced Middleware for Real-Time Reconfiguration of Serive Oriented Distributed Real-Time Systems", *IEEE Transactions on Industrial Informatics*, vol. 9, pp. 228–236, feb 2013.

[22] M. Loskyll, *Entwicklung einer Methodik zur dynamischen kontextbasierten Orchestrierung semantischer Feldgerätefunktionalitäten*, Technische Universität Kaiserslautern, 2013, Dissertation.

[23] A. Girbea, C. Suciu, S. Nechifor, and F. Sisak, "Design and Implementation of a Service-Oriented Architecture for the Optimization of Industrial Applications", *IEEE Transactions on Industrial Informatics*, vol. 10, pp. 185–196, feb 2014.

[24] T. Tack and J. Jasperneite, "Application specific Automation Devices - the App-based approach", in *4th Annual Colloquium Communication in Automation (KommA)*, nov 2013.

[25] K. Thramboulidis and G. Frey, "Towards a Model-Driven IEC 61131-Based Development Process in Industrial Automation", *Journal of Software Engineering and Applications*, vol. 4, pp. 217–226, apr 2011.

[26] M. Steinegger and A. Zoitl, "Automated Code Generation for Programmable Logic Controllers based on Knowledge Acquisition from Engineering Artifacts: Concept and Case Study", in *17th IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, sep 2012.

[27] J. Puttonen, A. Lobov, and J. L. M. Lastra, "Semantics-Based Composition of Factory Automation Processes Encapsulated by Web Services", *IEEE Transactions on Industrial Informatics*, vol. 9, pp. 2349–2359, nov 2013.

[28] M. Wienke, S. Faltinski, O. Niggemann, and J. Jasperneite, "mINA-DL: A Novel Description Language Enabling Dynamic Reconfiguration in Industrial Automation", in *16th IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, sep 2011.

[29] J. Otto, B. Böttcher, and O. Niggemann, "Plug-and-Produce: Semantic module profile", in *Dagstuhl Workshop on Model-Based Development of Embedded Systems (MBEES 2013-9)*, apr 2013.

[30] OPC Foundation, (dec 2013), OPC Unified Architecture [Online]. Available: http://www.opcfoundation.org/ua/

[31] OASIS, (jul 2009), Web Services Dynamic Discovery (WS-Discovery) Version 1.1 [Online]. Available: http://docs.oasis-open.org/ws-dd/discovery/1.1/wsdd-discovery-1.1-spec.html

[32] W3C, (dec 2011), Web Services Metadata Exchange (WS-MetadataExchange) [Online]. Available: http://www.w3.org/TR/ws-metadata-exchange/

[33] G. Cândido, F. Jammes, J. Barata de Oliveira, and A. Colombo, "SOA at device level in the industrial domain: Assessment of OPC UA and DPWS specifications", in *8th IEEE International Conference on Industrial Informatics (INDIN)*, jul 2010.

[34] L. Dürkop, J. Imtiaz, H. Trsek, and J. Jasperneite, "Service-Oriented Architecture for the Autoconfiguration of Real-Time Ethernet Systems", in *3rd Annual Colloquium Communication in Automation (KommA)*, nov 2012.

[35] G. Moritz, F. Golatowski, C. Lerche, and D. Timmermann, "Beyond 6LoWPAN: Web Services in Wireless Sensor Networks", *IEEE Transactions on Industrial Informatics*, vol. 9, pp. 1795–1805, nov 2013.

[36] L. Dürkop, H. Trsek, J. Jasperneite, and L. Wisniewski, "Towards Autoconfiguration of Industrial Automation Systems: A Case Study Using Profinet IO", in *17th IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, sep 2012.

[37] L. Dürkop, J. Imtiaz, H. Trsek, L. Wisniewski, and J. Jasperneite, "Using OPC-UA for the Autoconfiguration of Real-time Ethernet Systems", in *11th IEEE International Conference on Industrial Informatics (INDIN)*, jul 2013.

[38] K. Krüning and U. Epple, "Plug-and-produce von Feldbuskomponenten - Allgemeines Framework dargestellt am Beispiel Profinet IO", *atp edition*, vol. 11, pp. 50–56, nov 2013.

[39] G. Kortuem, F. Kawsar, D. Fitton, and Y. Sundramoorthy, "Smart Objects as Building Blocks for the Internet of Things", *IEEE Internet Computing*, vol. 14, pp. 44–51, feb 2010.

[40] PLCopen Association, (2008), PLCopen XML [Online]. Available: http://www.plcopen.org/pages/tc6_xml/xml_intro/

[41] OPC Foundation, "OPC UA For Devices (DI) Companion Specification", 2013.

[42] "KW Software Profinet Communication Stacks", https://www.kw-software.com/en/profinet-industrial-ethernet/communication-stacks.

[43] "Raspberry Pi website", http://www.raspberrypi.org.