Poster: Industrie 4.0 Virtual Automation Bus

Thomas Kuhn, Pablo Oliveira Antonino, Markus Damm, Andreas Morgenstern
Fraunhofer Institute IESE
Kaiserslautern, Germany
{thomas.kuhn, pablo.antonino, markus.damm, andreas.morgenstern}@iese.fraunhofer.de

Dirk Schulz
Networks & Devices
ABB AG Research Center
Ladenburg, Germany
dirk.schulz@de.abb.com

Constantin Ziesche
Bosch Connected Industry
Robert Bosch GmbH
Stuttgart, Germany
constantin.ziesche@de.bosch.com

Thorsten Müller
Basic Technologies
SMS group GmbH
Hilchenbach, Germany
thorsten.mueller2@sms-group.com

ABSTRACT

A main goal of the fourth industrial revolution is changeability of production processes, which is the ability to react efficiently to unplanned production changes. Existing automation system architectures limit this changeability. PLC programs used for automation include low-level behavior of actuators, strategies, management functions without information hiding. This yields unmaintainable, and therefore hard to change systems. In this paper, we document our Virtual Automation Bus that enables changeable production.

KEYWORDS

Industrie 4.0, Industrial IoT, software platform, changeability

1 SCOPE

Industrie 4.0 propagates peer-to-peer communication between production assets (devices, controllers, etc.) on both shop floor and office floor. This requires on one hand technical end-to-end communication abilities that connect different types of networks and middleware technologies such as OPC-UA and OneM2M. On the other hand, end-to-end communication requires one language with a defined set of communication primitives.

The Industrie 4.0 middleware BaSys 4.0 defines the Virtual Automation Bus (VAB), a communication layer that provides end-to-end communication between production assets. It presents all data and services through a common, reflective model-based API. The VAB represents every asset with an asset administration shell (AAS). The asset administration shell is an object that consists of one or multiple sub models. A sub model provides information regarding one aspect of the asset and conforms to a defined meta model. The AAS of a sensor for example defines sub models that describe the technical specifications of the sensor, that provide access to live sensor data, and that provide calibration services. Sub-models of the same AAS may physically reside on different devices.

2 VIRTUAL AUTOMATION BUS

Industrie 4.0 functions interact with the VAB: Controller RTE execute real-time functions that control parts of the production process. These functions execute for example on PLC controllers. Application RTE are runtime environments for regular software functions, e.g. implemented in Java or C#. These implement for example controller applications or data aggregators for condition monitoring or dashboard applications. Model providers provide information through a defined model API. Administration shell providers provide administration shell objects as reflective models. Directory servers support discovery and lookup of AAS objects.

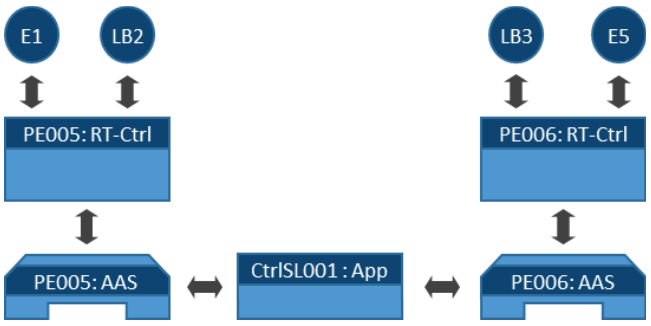

Figure 1: Functional components of plant automation

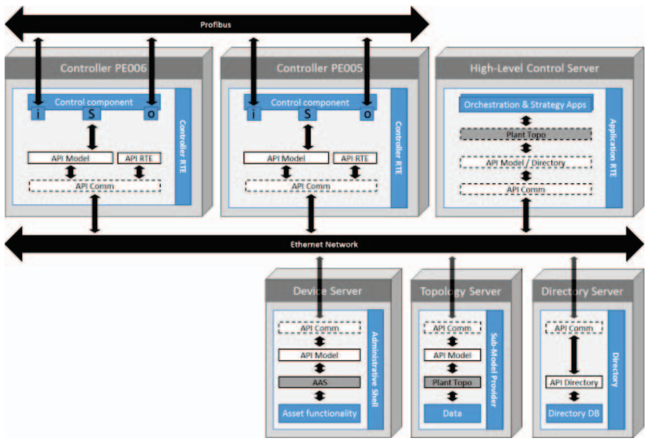

Figure 2: Ethernet technology mapping example

Figure 2. illustrates an example deployment of functions into an Ethernet domain. Different controllers and runtime environments in the shop floor and office floor execute Industrie 4.0 functions. Sensors and actuators connect via PROFIBUS to PLC real-time controllers. The Ethernet network that connects all devices realizes the VAB communication. The VAB defines five semantic primitives for inter-device interaction (cf. Table 1.)

Table 1: Virtual Automation Bus Primitives

BaSys Virtual Automation Bus Communication primitives	
get(objID, submodel, property)	Reads a value from submodel property of object objID
set(objID, submodel, property, value)	Set value of submodel property
create(objID, submodel, property, type)	Create a new property in submodel
delete(objID, submodel, property)	Delete submodel property
invoke(objID, submodel, name, par)	Invoke service name provided by submodel of object with given parameter

The VAB primitives provide a model-based façade to data and services of production devices. It therefore implements one common language for all devices. The VAB primitives enable access to objects, which are asset administration shells, and their sub models. Sub models are either standardized sub models, in this case the meta model is defined by BaSys, or plant specific sub models that are used e.g. for production planning or optimization. In this case, the sub model structure is plant specific.

Sub models contain hierarchical lists of elements and typed properties. All sub-models are reflective, and therefore provide common interfaces to query their structure at runtime. The basic meta model of every sub model is illustrated in Figure 3. Elements are the base class of all model elements, e.g. nodes and links, and may contain other elements. Properties describe elements in detail. All properties are typed; the type is available through the type property of every Property element.

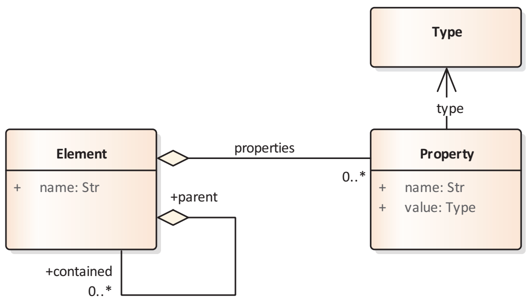

Figure 3: Reflexive presentation of data as hierarchical model

Figure 4. illustrates the technology mapping of the primitives from Table 1 to HTTP/REST web services with JSON parameter serialization. It illustrates the querying of an *isFree* property from an AAS sub model that is provided by the real time execution environment *PE005Ctrl* by the AAS *PE005*. The HTTP/REST web service mapping maps the five semantic primitives of the VAB to specific web service calls. Object ids are resolved to URLs, sub models, properties, and values, are web service call parameter.

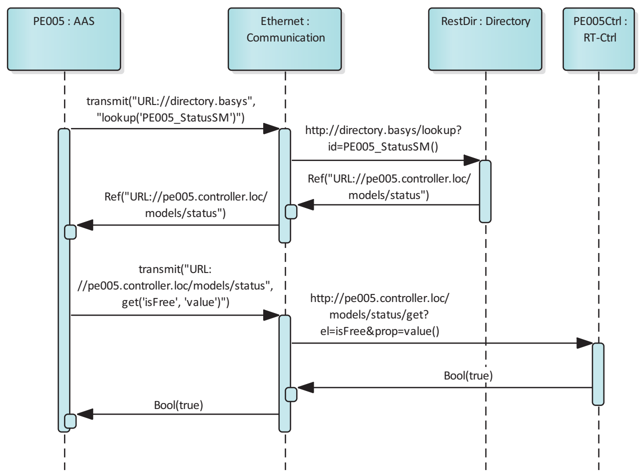
PE005 : AAS
Ethernet : Communication
RestDir : Directory
PE005Ctrl : RT-Ctrl
transmit("URL://directory.basys", "lookup('PE005_StatusSM')")
http://directory.basys/lookup?id=PE005_StatusSM()
Ref("URL://pe005.controller.loc/models/status")
Ref("URL://pe005.controller.loc/models/status")
transmit("URL://pe005.controller.loc/models/status", get('isFree', 'value')")
http://pe005.controller.loc/models/status/get?el=isFree&prop=value()
Bool(true)
Bool(true)

Figure 4: Example call sequence for pallet transportation

The five mentioned primitives are sufficient to execute a large number of communications scenarios. Moreover, these primitives map to different communication middleware standards. This enables the realization of the VAB on top of different middleware protocols. In context of the BaSys project, we develop technology mappings for the VAB for HTTP/REST, OneM2M, OPC-UA, and other types of bus systems.

ACKNOWLEDGEMENT

The BaSys 4.0 project is funded by the German Ministry for Education and Research (BMBF).