Cross-Industry State of the Art Analysis of Modular Automation

Dr.-Ing. **Sten Grüner**, ABB Corporate Research Center, Ladenburg
Mario Hoernicke, ABB Corporate Research Center, Ladenburg
Gerrit Fachinger, Chair of Process Control Engineering, RWTH Aachen, Aachen
Julian Grothoff, Chair of Process Control Engineering, RWTH Aachen, Aachen
Sophia Cordes, Professur für Automatisierungstechnik, Helmut-Schmidt-Universität, Hamburg
Prof. Dr.-Ing. **Alexander Fay**, Professur für Automatisierungstechnik, Helmut-Schmidt-Universität, Hamburg

Kurzfassung

Modulare Produktion ist ein anhaltender Trend sowohl in diskreter Fertigung als auch in der Prozessindustrie. In der Prozessindustrie schreitet die Einführung von NAMUR Module Type Package (MTP) rapide voran mit ersten Pilotanlagen und kommerziellen Produkten. In der diskreten Fertigung existiert mit PackML ein bekannter Standard für Verpackungsmaschinen. Trotz ähnlicher Konzepte und Geschäftsmodelle variieren die Anforderungen und die technische Realisierung beider Standards. Dieser Beitrag stellt eine Analyse der existierenden Standards für modulare Produktion vor. Neben MTP und PackML berücksichtigen wir ein Prozessführungskonzept, welches im Rahmen des öffentlich-geförderten Projekts BaSys entwickelt und anhand eines Demonstrators validiert wurde. Unsere Ergebnisse bringen Vorteile für Hersteller, Integratoren und Betreiber von modularen Anlagen insbesondere in gemischten Anwendungen, in denen diskrete und prozesstechnische Module vorkommen. Wir zeigen eine Perspektive für Langzeit-Konvergenz der Standards und helfen mittelfristige Koexistenz-Strategien für gemischte Anwendungen zu definieren.

Abstract

Modular production is a persistent trend in both manufacturing and process industry. For process industry, the adoption of NAMUR Module Type Package (MTP) progresses rapidly including first commercial products. In the manufacturing industry, one prominent representative of modular production in factory automation is the PackML standard for packaging machines. Despite similar concepts and business advantages, requirements and current technical realization of module description and process orchestration differ across the industries. This paper is a state-of-the-art analysis of modular automation concepts. Along with MTP and PackML we also discuss the process control approach developed in scope of publicly funded BaSys 4.2 project and validated in a demonstrator. Our results bring advantages for manufacturers, integrators and operators of modular automation plants, especially in mixed discrete/process applications. We open a perspective for long-term convergence of the standards' aspects and help to define mid-term coexistence strategies especially for mixed applications.

1. Introduction

Modular production is a persistent trend in both manufacturing and process industry. Advantages include a shorter time to market and a higher flexibility of the production facilities. As example for the process industry, the adoption of NAMUR Module Type Package (MTP) concepts is to be mentioned especially in domains of special and fine chemicals as well as the pharmaceutical industry. In the manufacturing industry, modular production resources play an even higher role since it is applicable to a larger fraction of factories. One prominent representative of modular production in factory automation is the PackML standard for packaging machines.

Despite similar concepts and business advantages of modular automation, requirements and current technical realization of module description and process orchestration differ across the industries. This fact increases the needed implementation efforts for hybrid, i.e. cross-industry applications. One typical hybrid application is a beverage production process with subsequent bottling including a process part (e.g. actual beverage production) and a discrete part (e.g., filling, labeling, transporting).

In this paper we work on identification of similarities and differences between existing approaches for modular automation. This paper focuses on process control interfaces that are typically deployed on the module itself and are reachable via OPC UA.

Along with existing standards like MTP and PackML we also discuss the AAS-enabled process control approach that was developed in scope of BaSys 4.0 and BaSys 4.2 publicly funded projects and validated in a demonstrator. For engineering information of modules, BaSys utilizes features of AAS like different file or API-based serialization (e.g. XML or RESTful interfaces with JSON payload) or AAS-suited infrastructure like discovery servers for component and meta-data discovery.

The remainder of this paper is structured as follows: Section 2 characterizes different requirements on modular automation from discrete and process domains, in Section 3 we briefly introduce existing standards for modular automation from both domains, Section 4 outlines a modular control approach that was developed in scope of BaSys projects, Section 5 structurally compares the presented three approaches before in Section 6 we provide a summary and conclusions.

2. Characterization and Motivation for Modular Production

Manufacturing and process industry have developed separately in the past for several reasons. The difference in product characteristics is one of the main reasons for that. Whereas in the process industry products are shapeless fluids or bulk materials, in manufacturing products are solid objects or assemblies of such. Consequently, there are significant differences in processing/machining of those products. In processing, products are changed by chemical or physical transformation, which are often sensitive to timings and sequential order. In machining, products are changed by modifying the shape, surface or material properties or by joining or separating them [1].

All those differences have led to a process-focused development in process industries and a product-focused development in manufacturing. In the process industry atomic units like valves, pumps and pipes are assembled to a plant that fits to the process. On the other hand, in manufacturing, several machines can execute atomic processes with which the product can be build up.

As a result, there are different information models and types of description in manufacturing and process industry. However today, both face similar challenges that ask for a modular approach. Increasing volatility on the market, a bigger product portfolio and shorter development cycles demand modular and adaptable plants.

In addition to the advantages of quicker plant construction and greater flexibility in the case of changes during the life cycle of the plant, the plant capacity can be gradually expanded at a later stage by numbering up the modules to the required production volume. Unnecessary investments in early phases of the product lifecycle are avoided. Further cost reductions and the increase of flexibility in individual cases are apparent in the reusability of modules for different applications or the temporary provision of equipment based on rental models or by means of leasing [10].

Volume fluctuations and increasing product diversification (e.g. due to a growing diversity of container types and sizes) have an intensive effect on the production-related logistics processes. The changed requirements on production thus also affect the design and operation of production-related logistics processes, systems and structures. This applies to the diverse, in many cases extensively automated production-related logistics processes in the process industry. Ideally, the associated logistics plant units should in future be interconnected flexibly and with minimum effort according to the plug & operate principle [14].

3. Existing Standards for Modular Production

3.1 Module Type Package

In context of process automation, a modular plant consists of different modules, encapsulating devices and apparatuses that likewise consist of process controllers and instruments. Every module has a built-in controller that takes care of automating the functions provided by the module, as well as fault handling and other automation tasks. Each module (also known as “Process Equipment Assembly” – PEA) fulfills at least one process function, which is encapsulated and offered as a service. All modules have process connections to other modules and an information connection to a so-called process orchestration layer (POL). The communication between PEA and POL is realized using OPC UA communication protocol.

A key element within the area of modular production plants is the standardization of the automation system interfaces between the module automation system and the POL. This has been standardized by a module description called Module Type Package (MTP), which allows the seamless integration of modules into the POL [2]. An MTP file typically includes P&ID description of a module, its services, a list of built-in equipment. The MTP is a vendor independent description, which defines the interfaces to a module in AutomationML format. Therefore, it can be used to integrate modules into the POL, independent of the vendor and type of the controllers used for the module.

3.2 PackML

The Packaging Machine Language (PackML) is an industrial automation standard for the control of packaging machines. The Organization for Machine Automation and Control (OMAC) developed this standard from 2003-2009 in collaboration with the International Society of Automation (ISA) [12]. PackML suggests a standard program architecture and programming methodologies with the aim of providing a consistent appearance and handling of packaging machines for operators and technicians. Key arguments for using the standard are improvements in robustness, cost, development cycles, cross vendor interoperability, vertical and horizontal integration and reconfiguration [13].

The three main elements of the standard are a state-machine model, tags (PackTags) for a consistent terminology and the modular program architecture Make2Pack [12].

The state-machine model can be adopted in almost any industrial machine, since states and transitions are not exclusive for packaging machines. PackTags should be used for control, status administration purposes. Furthermore, modularized machine code should match the lower 3 levels of ISA 88 / IEC 61512 [8] physical hierarchy: Unit, Equipment Module (EM) and Control Module (CM) [12].

The recommended way for inter-module and line-to-module communication is to use OPC UA. A PackML OPC UA companion specification is available [6].

4. Modular Process Control in BaSys 4.0 and BaSys 4.2

The German research project “Basis System Industrie 4.0” (BaSys 4.0 and the successor project BaSys 4.2, jointly denoted as BaSys) aims to create a highly adaptable production system. Hence a component-based architecture was designed, which intrinsically introduces modularity to the production system. The architecture mainly comprises Control Components and Asset Administration Shells.

Control Components (CCs) enable the process control and supervision by unifying the interfaces and behavior of all equipment units that are involved in the production process. They represent the production equipment within the BaSys environment. Their structure follows the separation of preferably orthogonal state machines, that cover different aspects like occupation, execution states or operation modes. The state machines are based on reference state machines like the PackML states and modes.

The concept of the Asset Administration Shell (AAS) focusses on the asset centered access to and grouping of information. BaSys is developing SDKs in different programming languages to create and host AASs[1]. The production equipment can be considered as multiple assets. Consequently, AASs related to CCs can be used to support the production process. In return a CC submodel was developed.

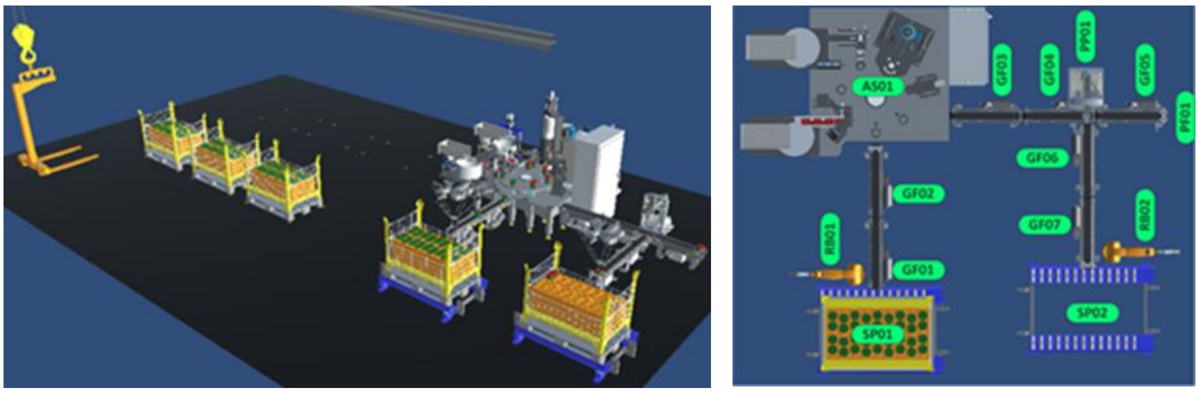

Figure 1: BaSys 4.0 virtual demonstrator

[1] https://www.eclipse.org/basyx/

To evaluate the concepts a common demonstrator was created. The demonstrator was build up on a real-time 3D simulation (Figure 1) of a simple production process of screwing a lid onto a can with some logistics and a correction workstation. 17 CCs were created for 7 different types of equipment modules by at least 8 different project partners. The process was orchestrated via a BPMN workflow engine. The CC specification ensured the interoperability of the independent developed components and the orchestrator. AASs were used as a proxy to translate from workflow engine HTTP/REST to OPC UA calls for demonstration purposes, direct CC calls from orchestrator are also possible and might be required depending on operational constrains. The AAS registry was used to discover OPC UA endpoints of CCs. Moreover, AASs also contained prototypical semantic capability descriptions via ontologies [11] and auxiliary data such as documentation.

5. Comparison

In this section we provide a detailed comparison of the introduced standards. For MTP we refer to the current state of MTP standardization [1] [3] [4]. For PackML we refer to ANSI/ISA specification [5], OPC UA companion specification [6] and implementation guide [12]. BaSys CC description state is based on the publicly available specification [7] and refers to the state of the virtual production line demonstrator presented in July 2019.

5.1 Standards’ Scope and Feature Comparison

In the comparison we differentiate between the module layer (called Process Equipment Assembly (PEA), Unit/Machine or Component in MTP, PackML and BaSys, respectively) and the orchestration layer (called Process Orchestration Layer (POL), Supervisory Control or Orchestrator in MTP, PackML and BaSys, respectively).

Module Layer

The comparison summary can be found in **Error! Reference source not found.**. Despite different naming conventions, the concept for the module can be traced across the three approaches. MTP standard defines a set of criteria for physical module units for process plant modules [9] and narrows the scope down to integrated, i.e. not multi-level, modules with integrated intelligence that are modeled as a gray box, i.e. exposing some internal details. PackML assumes a physical production machine to correspond to the unit equipment hierarchy level of IEC 61512 [8] and focuses on its black box model, i.e. internal equipment or implementation details are hidden. Finally, BaSys 4.0 component has a broader scope and can, in theory, range from hardware to software. Nevertheless, the application of the BaSys 4.0 components with the scope of the virtual production line demonstrator, narrows

BaSys 4.0 components down to (simulated) physical equipment unit such as robot, crane or a conveyor belt. Equipment units are modeled as black boxes.

Table 1: Standards' scope overview on module level

	MTP	PackML	BaSys
Module term	PEA (VDI 2776), Unit	Unit/Machine	Control Component (CC)
Module model	gray box	black box	black box
Module type/instance focus	Type	Instance (unified interface)	Instance (with instance variance possibility)
Module description features:			
- HMI	vendor-independent	-	-
- Diagnostics/Maintenance	planned	pack tags	-
- Alarms	planned	pack tags	-
- Documentation	-	-	using AAS submodels
- Services	multiple per unit, including procedures (eq. to modes)	one, but including multiple modes	one, but including multiple modes
- Service semantics	no	no	capability submodel, semantic identifiers possible via AAS
- Data of included devices	instance list	-	-
- Included data block library	yes	-	-
Nested modules (also called modular modules [9][10])	FEA, VDI 2776	-	yes
Module runtime interfaces	OPC UA variable node set	OPC UA companion specification, field buses	different OPC UA profiles

Standards treat modules and module types differently: MTP has a clear focus on describing module types in standardization documents. PackML defines a unified look and feel for various machine instances, hence it can be considered to describe a unified module instance interface. BaSys 4.0 components are mostly described in an instance-specific way including all the variations without a focus on components' typing.
Let us consider the "features" of the module descriptions that are defined by the respective approach. MTP includes a vendor-neutral HMI / P&ID of the module type as a unique feature. This HMI is represented in terms of an object model with the MTP and can be rendered by the POL to create a unified look and feel for the user. Alarm and diagnostic information are on MTP's roadmap while PackML has a set standardized alarm and diagnostic variables included in pack tags.
BaSys orchestration concept has its unique feature in the ability to use existing AAS's submodels that are developed by Industrie 4.0 community to represent auxiliary module information. In the virtual production line demonstrator, documentation according to VDI 2770 standard was included into the administration shell to provide access to module type-specific (e.g., manuals) and instance-specific (e.g., commissioning protocols) documentation.
MTP allows having multiple services per module (e.g. stirring, filling), PackML and BaSys have only one service per module. BaSys approach additionally allows adding meta-information to service description such as capability definition from a predefined ontology [11] that is realized via Asset Administration Shell mechanisms.
As a single standard, MTP allows to include an instance list of included devices into module type description. Furthermore, the standard defined a list of pre-defined data block libraries.
MTP and BaSys allows to describe nested submodels.
Regarding interfaces of the modules, all three approaches support OPC UA with a ranging variety of information modeling. More runtime details can be found in the next section.

Orchestration Layer

Overview of the orchestration layer is presented in Table 2. All concepts require a manual configuration of module topology, i.e. physical connections. In MTP POL takes over the generation of a consistent HMI for the defined module configuration and the runtime orchestration and monitoring. The orchestration recipe consisting of service calls and access to integrated equipment are vendor dependent.

Table 2: Standards' overview regarding orchestration layer

	MTP	PackML	BaSys
Orchestration layer term	Process Orchestration Layer (POL)	Supervisory Control	Orchestrator
Module topology definition	manual	manual	manual
Runtime bootstrapping	not specified, OPC UA mechanisms can be used	not specified, OPC UA mechanisms can be used	dynamic component registration and discovery via AAS
Runtime module topology change	possible	possible	possible
Serialization for engineering and meta information	AML	self-contained OPC UA model	AAS-available serialization (RESTful HTTP in demonstrator)
Cross-Module Communication	possible, via POL	possible, direct	-
Orchestrator or line orientation	orchestrator	both	orchestrator

The runtime bootstrapping process, e.g. finding currently available modules, is not specified in both MTP and PackML. Since both standards rely on OPC UA communication, typical OPC UA mechanisms like Local Discovery Server (LDS) can be used for that matter. BaSys uses discovery mechanisms for AAS to locate OPC UA endpoints for orchestration.

All three approaches support different runtime topology changes, e.g. module redundancy scenarios and reaction to module faults.

A large difference lies in the serialization of engineering artifacts that describe the module. MTP rely on AML files, BaSys uses AAS with various serialization possibilities incl. AML and RESTful APIs. PackML provides a rich OPC UA information model that contains most of machines meta-data.

PackML is also the only standard that provisions direct module-to-module communication. This can be particularly used for a line-operation use cases where no explicit orchestrator is needed. Still PackML can also be used in orchestrated mode called Supervisory Control mode. MTP and BaSys currently focus on use cases with a dedicated orchestrator.

5.2 Service Interface Comparison

In this section we focus on various facets of the three approaches regarding service interfaces. The service interface detail overview is presented below in Table 3.

Table 3: Service interface comparison overview

	MTP	PackML	BaSys
Mode of operation term: "how service / unit / component reacts to commands"	Operation Mode offline, online, manual, automatic incl. internal and external sub states (Ger. "Betriebsart")	Unit/Machine Mode automatic/production, maintenance, manual and custom defined	Execution Mode complies to standard PackML + simulation mode
- Changeable at state	in every state but may be constrained by module logic. Switch between Offline and another mode of operation is only possible in IDLE state	in any wait states	in Stopped state
State model and state model variability (cf. **Error! Reference source not found.**)	fixed state set, modified PackML, S88	removable states based on unit mode and addable sub-state machine	removable states and addable sub-states, PackML based
- Different state model per operation mode	fixed	yes, automatic mode requires all states	yes
- Active transition indication	command enabled vector	available transition variable	-
Detailed execution semantics "defines what service / unit / component is doing in detail"	Procedure per service (prev. called control strategy) (Ger. "Fahrweise")	no (can be simulated by unit modes)	Operation mode per component (Ger. "Fahrweise")
Parameter interface	per procedure and	per unit mode	per operation

	service		mode
- Parameters for procedures/ modes of operation	yes	yes	yes
- Parameters for module	yes	yes (e.g. status and admin tags)	yes
Occupation information	yes, per service/tag/parameter (OSLevel)	-	yes, per module, including prioritized occupation
Auxiliary information	health state diagnostic (planed) alarms (planed)	admin tags	error status
OPC UA representation	yes, but standardized only at variable value level	OPC UA companion spec [6]	multiple profiles: variables only, variables and methods
- OPC UA server requirements	Variables (with standardized variable values), no custom types	Variables, method calls, custom types, events	Variables and/or method calls dep. on profile, no custom types

The first aspect of a module's service interface is its "mode of operation" describing how the module reacts to different commands physically and via its service interfaces. This concept is called Operation Mode, Unit Mode or Execution mode by MTP, PackML and BaSys orchestration, respectively. There are detailed differences when the mode of operation can be changed in each standard's domain. For example, MTP allows the change in every state (Except of Offline mode switches) in principle, while BaSys limits the mode change to the "Stopped" state.

When it comes to the state model, all three approaches follow a similar state model that is based on PackML. BaSys inherits the state model, while MTP uses slight modifications, e.g. no "Cleaning" state. The state models are shown in **Error! Reference source not found.**.

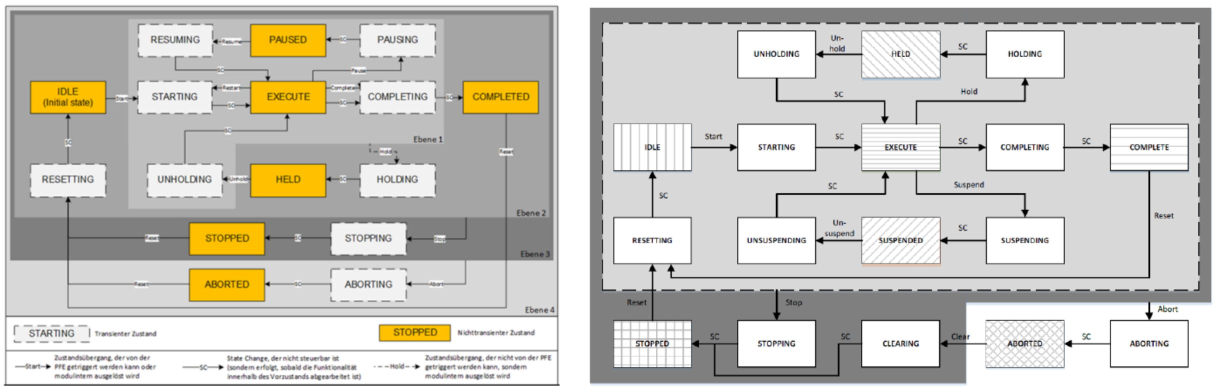

Figure 2: State space of MTP (left) [2] und PackML (right) [11]

MTP assumes an always fixed state set. PackML allows to remove states in different modes (however automatic mode requires all states implemented). It is also possible to add sub-states, e.g. vendor-specific states within the "Execute" state. BaSys follows PackML in this regard. MTP has a unique feature, its service interface can indicate enabled transitions from the current state, other standards require a transition request that might fail.

The second aspect of the module service interface is the so-called "Detailed execution semantics" that defines the details of module's or service's operation like "heat up" or "cool down" for a heating service. Here we see a huge naming and structuring discrepancy between the three approaches. In MTP modules have services which have procedures (called "Fahrweise" in German) including a default procedure per service. PackML has only one service per module, however execution semantics can be simulated by custom unit modes. In similarity to PackML, BaSys approach has only one service per module, however this one has dedicated operation modes (also called "Fahrweise" in German).

These differences can also be seen in the parameter interface for services, modes or procedures. MTP supports parameters per procedure and service, PackML has paramters per unit mode, and BaSys has parameters for operation modes. Furthermore, modules have additional configuration parameters that can be set from the outside.

Occupation information indicates the allocation and exclusive usage of a module or a service. MTP allows this allocation per service, tag or parameter. BaSys unit of allocation is the whole module. To our best knowledge, PackML does not support allocation features.

All approaches support auxiliary information such as health state, diagnostics or alarms. MTP has diagnostic an alarm states on its roadmap. PackML has a dedicated set of tags for this purpose. BaSys CC provides a dedicated error status per module.

The last comparison category is the OPC UA runtime representation of all approaches. MTP uses OPC UA but does not put any requirements on information space structure. For a successful orchestration, POL requires specific values of variables which OPC UA node IDs

are defined in the MTP file of the module. PackML uses OPC UA as one possible runtime interfaces, here a fully specified companion specification is available [6] which is pretty self-describing and contains, for example, a full object-oriented representation of unit's state machine including sub-states and transitions. BaSys CC defines multiple profiles for OPC UA representation: one profile allows to communicate to the module by using OPC UA variables only e.g. by writing a command variable. Another profile allows orther interaction patterns, for example triggering module's services by parameterized method calls.
Depending on the used OPC UA representation, different requirements arose for the OPC UA server which implements module's runtime interface. Here, PackML imposes the highest requirements as it prescribes method calls, custom types and events to be implemented by the OPC UA server.

6. Summary and Conclusions

In this paper we compared existing standards and approaches for modular automation in discrete manufacturing and process automation domains. We conclude that despite similar motivation and application scenarios, all approaches differ both on service interface level and on meta/engineering information level of module and its type:

- Differences in runtime interfaces include many fine-grained varieties like hierarchy of operation modes, execution semantics, service invocation specifics and the information model.
- Differences in the meta- and engineering-information are even larger ranging from different available module information (e. g. HMI information) to different information representation like (AAS or file-based type information) or mechanisms of differentiation between module types and instances.
- Conceptually, PackML and BaSys CC miss an abstraction hierarchy level of MTP-services allowing to embed multiple concurrent services into a module. Nevertheless, there are several workarounds to remedy this limitation.

The presented comparison should be used to guide the user while selecting the appropriate standard for a specific modular application. Furthermore, it provides input for standard-homogenization activities and stimulates re-use of aspects between standards. One of the first examples is the ongoing work on synergies of MTP and AAS including augmenting MTP files with AAS documentation submodels. Another example are ongoing efforts in BaSys 4.2 project to implement a PackML compatible BaSys CC profile to drive harmonization on runtime level.

Acknowledgement

This work is partially supported by German Federal Ministry of Education and Research in the scope of the BaSys 4.2 project (01IS19022).

References

[1] Mersch, Behnen, Schmitz, Dominik, Epple, Brecher, Jarke, Matthias: Gemeinsamkeiten und Unterschiede der Prozess- und Fertigungstechnik - Commonalities and Differences of Process and Production Technology, at – Automatisierungstechnik, January 2011

[2] VDI/VDE/NAMUR 2658-3: Automation engineering of modular systems in the process industry – Interfaces and Libraries for Basic Object Types, Draft, March 2019 (publicly available)

[3] VDI/VDE/NAMUR 2658-4: Automation engineering of modular systems in the process industry – Services for Process Equipment Assemblies, Draft, April 2020

[4] VDI/VDE/NAMUR 2658-5: Automation engineering of modular systems in the process industry – Runtime and Communication Aspects, Draft, March 2020

[5] ANSI/ISA-TR 88.00.02-2015: Machine and Unit States: An implementation example of ANSI/ISA-88.00.01, 2015

[6] OPC Foundation OPC 30050: UA for PackML companion specification, 2018

[7] Epple et al.: BaSys 4.0: Metamodell der Komponenten und ihres Aufbaus. Available online: http://doi.org/10.18154/RWTH-2018-225880

[8] IEC 61512: Batch control – Part 1: Models and terminology, IEC, 2000

[9] VDI/VDE/NAMUR 2658-1: Automation engineering of modular systems in the process industry – Automation engineering of modular systems in the process industry - General concept and interfaces, October 2019

[10] NAMUR NE 148: Automation Requirements relating to Modularisation of Process Plants. NAMUR Recommendation 148, 2013

[11] Alexander Perzylo et al.: Capability-based semantic interoperability of manufacturing resources: A BaSys 4.0 perspective, Manufacturing Modelling, Management and Control - 9th MIM 2019

[12] Carsten Nøkleby: PackML Unit/Machine Implementation Guide, v. 1.00. Available online: http://omac.org/wp-content/uploads/2016/11/PackML_Unit_Machine_Implementation_Guide-V1-00.pdf

[13] OMAC: Implementing ISA-TR88.00.02 (PackML), Whitepaper, Available online: http://omac.org/wp-content/uploads/2015/12/OMAC-OEM-PackML-Whitepaper-Final-9-26.pdf

[14] NAMUR NE 171: Application of a modular system concept to the requirements of production-related logistics. NAMUR Recommendation 171 [in print], 2020.