Towards Smart Factory: Multi-Agent integration on industrial standards for service-oriented communication and semantic data exchange

Ronald Rosendahl*, Ambra Calà*, Konstantin Kirchheim*, Arndt Lueder* and Nikolai D'Agostino†
*Otto-v.-Guericke University, Magdeburg, Germany
Email: [ronald.rosendahl, ambra.cala. konstantin.kirchheim, arndt.lueder]@ovgu.de
†CENIT AG Digital Factory Solutions, Stuttgart, Germany
Email: n.dagostino@cenit.de

***Abstract*—The concept of smart factories in the industry 4.0 (I40) paradigm is based on the concept of cyberphysical systems which is more specifically elaborated within the concept of the smart manufacturing component. In that concept an arbitrary production asset is assigned to an administration shell, that covers partial models and services for arbitrary functions. The desired attributes of smart factories to be self-organized, self-optimized, self-learning, etc. is based on the idea of an intelligent co-operation between the smart manufacturing components. However, the actual performing of this co-operation is not appropriately elaborated yet. In this paper the approach of a Multi-Agent-System to be applied as a well elaborated mean for intelligent co-operation of smart manufacturing components shall overcome this lack. Furthermore the application of existing standards for service-oriented industrial communication infrastructure by OPC-UA and for semantic data exchange by AutomationML provides existing means for realizing the required connectivity and interoperability for intelligent co-operation that shall realize self-X capabilities of smart factories.**

***Index Terms*—Agents, Flexible manufacturing systems, Knowledge representation, Service**

I. Introduction

Driven by increased demands for shorter life cycles and a higher degree of individualization of products the manufacturing industry is looking to raise the dynamics in their systems. With the evolving technological capabilities strong social and organizational changes in the management of all life cycles of those artificial systems are being expected. This issue was addressed by a funding of the German government to enforce a 4th industrial revolution and introduced the term "Industrie 4.0" (I40) [1]. In the domain of production systems engineering the requirements for I40 are in a first step being addressed by an increase in modularization of system components. One essential building block in this discussion is the Smart Manufacturing Component (former: I40 component) [2] which is being developed on the idea of Cyber Physical Production Systems (CPPS) and constitutes the concept of an administration shell which combines the presentation of the physical components of a production system along with its digital representation [3]. This modularity renders the creation and adjustments of systems a semi-automatable process. The necessary information and data models within such design processes form an ideal basement for the acquisition, structuring and management of system related data at the runtime of a facility. In the area of discrete manufacturing the data model and metamodeling capabilities of the file format AutomationML evolved which supports the modular thinking in terms of mechatronical engineering, high level of model dynamics and concepts for building common models and libraries for different engineering domains and disciplines. In several research activities and industrial applications AutomationML proved a suitable data model for the multi disciplinary modular engineering of production systems as well as the transition of data from the design and virtual commissioning phases of a system to its runtime. In [4] a concept of an AutomationML and OPC UA based implementation of I40 components is being discussed. It is shown how AutomationML projects are automatically translated into namespaces of OPC UA in a standardized way following the specification given in DIN Spec 16592 [5]. Combining this auto-generated component description in a standardized communication architecture with agent technology seems a promising candidate for the creation of dynamically adaptable system modules thus easily bringing smart applications to the production domain. An actual research question is to evaluate how the model and workflows out of the engineering processes at design time associated with AutomationML fit the requirements of the runtime management of a system specifically in an automated way performed by Multi-Agent-Systems (MAS). Therefor it has to be proven which methodological and technological adjustments or extensions are needed to support or enable the AutomationML ecosystem to be a valuable part of the development and operation of an MAS.

Within this paper it will be presented how the requirements for the acquisition of system information by Multi-Agent-Systems will be met by a service based online model of the system represented in AutomationML. After a brief introduction to the underlying technologies for the architectural proposal given in this paper the requirements are collected that need to be matched to apply an AutomationML knowledgebase for MASs in the domain of production systems. Special attention is spent to the modeling capabilities for semantical

concepts of the system. In the following section it is being explained how this requirements are met by the proposed technologies and which auxiliary methodologies or technologies have to be applied to the architecture. The final section presents a prototypic implementation which is exploited to prove the solution in a lab size production system.

II. Industrial Standards and architectures for the proposed smart manufacturing component

A. Industrial Agents and Architectures

Agent technology is a suitable technology to decentralize control automation architectures in production systems in order to dynamically react to changing requirements according to the I40 paradigm. Especially, MAS [6] have been used in the past years to develop modular, flexible, robust, adaptive, reconfigurable and responsive complex production systems, based on the decentralization of control functions over a community of distributed, autonomous and cooperative agents. Agents applied to the industrial domain are called Industrial Agents [7] and share the same characteristics of software agents, such as intelligence, autonomy and cooperation. Depending on the application scenario and the degrees of importance, agents can fulfill different requirements in production control systems. Agents have been used historically in industry to connect the cyber to the physical part. In fact, several distributed control automation architectures based on MAS technology have been developed in the past years within research projects, highlighting their benefits in production systems in terms of flexibility and reconfigurability [8]. These research projects have derived MAS-based reference architectures focusing on control, planning and scheduling, and supervision solutions, positioning agents at the bottom levels of the ISA-95 and predominantly at MES level. Some of the realized MAS architectures need to be mentioned, such as GRACE project [9], which developed a multi-agent architecture that operates at all stages of a production line integrating process control with quality control. The MAS architecture was applied on a washing machine production assembly line and tested against planned and unforeseen variation of variables. IDEAS project [10] developed a fully distributed and pluggable mechatronic environment, based on agent technology, capable to self-organize itself. The assembly system operated with a totally distributed multi-agent control system. PRIME project [11] developed a multi-agent architecture using plug-and-produce principles for semi-automatically configuring production systems through innovative human-machine interaction mechanisms. The multi-agent architecture here is used for module integration including legacy systems. Finally, ARUM project [12] also proposed an agent-based architecture aiming at minimizing the response time to unexpected events during the ramp up phase of plant. As seen in the past projects, main requirements of MAS comprise the specific hardware integration, reliability, fault-tolerance, scalability, industrial standard compliance, quality assurance, resilience, manageability and maintainability. For industries that aim at mass production of individual customized products, MAS can support everyday operation like variable customization requirements, changes in plans and schedules, changes in technology, and equipment failures [7]. Other common application scenarios appear in the transportation and material-handling systems, production management of frequently disrupted operations, coordination of organization with conflicting goals and frequently reconfigured environments [13]. A more comprehensive set of MAS applications in industry can be found in [14]. Different requirements in the application domains can be derived also from other emergent reference control architectures developed within the ongoing projects like IMPROVE [15], PERFoRM [16], BaSys4.0 [17] and INTER-IoT [18], [19] and [20]. Besides the general requirements of flexibility, scalability, modularity and standardized interface as well as common information model, other requirements can be included depending on the application use case, such as: data curation; inter-enterprise data exchange; privacy, integrity and security; service detection and orchestration; and real-time communication between architecture participants.

B. AutomationML

In the year 2006 a leading automotive OEM initiated several cooperations of developer teams of their engineering departments, service contractors and suppliers. The goal was to overcome identified waste of resources in the engineering chain of production systems. One major problem was that at some transition points between the engineering phases there was no digital transmission of the engineering data at all. The manual submission by printed documents lead to avoidable and error-prone reinterpretations of information to digital representations. The solution proposal was to develop a data format which addresses the special needs of a gapless transmission of digital representations of engineering information along the whole development cycle of discrete manufacturing systems. To evolve and promote this format in the year 2009 the ”AutomationML” society was founded. At the time of development object orientation and model driven development where state of the art paradigms and prototypic object models gained some popularity with the ascent of web technologies in languages like JavaScript. In consequence the main requirements to the data model of AutomationML that where derived together with production system engineers were:

- **object orientation** to support the modeling of systems in terms of mechatronical engineering and to represent a system component with several distinct (mechanical, electrical, information-technological) aspects as one unique self-contained entity,
- **prototypic inheritance** to support the reuse of model elements with the capability to be easily evolved, avoiding the complexity of managing consistent class hierarchies at the same time,
- **weak inheritance** rules affecting instance data regarding the fact that engineering information is assembled at design time and its associated data will be incomplete and inconsistent meanwhile,

- **object relations** for the modeling of dependencies beyond the hierarchical structures as well as
- **object roles** for semantical richness of the model.

Regarding this requirements AutomationML was defined as an open, XML-based format in a series of standards (IEC 62714) for the model based description of production systems and components. Given the mentioned capabilities AutomationML has proven a valuable data model for the exchange of engineering information of different disciplines in the design stage of a system [21], [22], [23]. It integrates sets of very heterogeneous information of the system being modeled and forms a sound repository of machine readable descriptions of the system. In several applications it has shown that systems represented in AutomationML exploiting the full modeling capabilities form a directory of component descriptions in a systemic way. Consequently the idea came up to use this data objects across further life cycle phases of the underlying system. Several research projects are currently concerned with the integration of this data at the runtime of a system [24]. Since the majority of agent based approaches rely on an information model of the system they are being applied to, AutomationML seems to be a reasonable candidate for the acquisition and management of system information enabling a repository of multi purpose system descriptions [25].

C. OPC-UA Information Models

OPC UA (Open Platform Communications Unified Architecture) is a platform independent architecture for communication of devices and systems in industrial communication networks defined in the series of standards (IEC 62541). It focuses on providing a unified service oriented infrastructure for the transport of data and the modeling of information in order to facilitate vendor independent interoperability between devices. OPC UA models information as a full meshed network, consisting of nodes that represent for example types or objects, and references that describe relation between these nodes. Following this approach, object-oriented concepts like inheritance and type hierarchies can be described in a way that enables clients to query and manipulate instance- and type-related information in a uniform manner, while also allowing the extension of existing type hierarchies. OPC UA imposes no further limitations on how to model information in order to allow the representation of arbitrary existing rich information models [26].

This integration of vendor-specific information models into OPC UA is specified in so called companion specifications. DIN Spec 16592 specifies the translation of AutomationML projects into OPC UA information models, enabling the online exchange of AutomationML models along with all of OPC UAs features concerning data management, multi user support, access methods, security etc. [27].

III. Requirements on Life-Cycle-Spanning Information Models for the shop floor

Multi-Agent-Systems performing tasks on industrial appliances require suitable descriptions of the underlying hard- and software. In this section the requirements to a knowledgebase for such an integration are discussed. After introducing some basic principles that are being expected as general requirements for data models applicable with MAS, more specific concepts will be pointed out. Those address the requirements derived from the aforementioned use-cases and reference architectures for MAS. These requirements are evaluated here against the capabilities of the AutomationML concepts of the systems engineering phases. On the basis of these facts implementation gaps are identified and enhancements suggested to probe the limitations of the proposed solution of an AutomationML based distributed system model for the use in MASs.

A. Structure of information

Not just to the application of MAS one major principle of information modeling is the Divide and Conquer paradigm. It helps to manage complexity by breaking huge systems down to simpler ones and have the responsibility for the subsystems assigned to a squad of professionals (algorithms, experts, 3rd party companies) who not just share the load but are specialized to a certain task. To integrate this partially managed subsystems and make them work as a whole it is necessary to conceptualize elements that represent the part and aggregate its related information. These elements are required to be ***coherent compositions of data*** based on a common data model and to have an ***identity***. These elements are subordinate to other elements which represent the (sub-)system they constitute and recursively form a ***part of.. hierarchy***. AutomationML meets these requirements with the concept of an *InternalElement* (IE) which may contain an unlimited number of Attributes. The standard specifies a mandatory identifier for each of this elements. The identifier is defined to be unique and shall be implemented using Globally Unique Identifier (GUID) that per definition persists along the life-cycle of the distributed set of elements. Each *InternalElement* may contain an unlimited number of other *InternalElements* constituting a tree structure which is contained within a so called *InstanceHierarchy*. This way an MAS may browse and access data of certain assets within the system in a structured way.

B. Types of information

A basic principle in all MAS architectures is the specialization of agents to purpose and asset. If agents with different purpose access the same asset they may require different kinds of contained data with distinct semantical meaning not satisfied by the definitions of the underlying data model and a static structure. These use cases require the attributes to be recognized by a ***name*** and to have a ***type*** to qualify the value as well as a ***unit*** to quantify the value. In AutomationML there is a mechanism for dynamic typing and only spatial, spatio-temporal and kinematic information are statically defined. Therefore as part of the basic concepts each *InternalElement* may declare a *Frame* attribute which specifies its location by transformation parameters in the three-dimensional space

relative to its parent. Spatio-temporal and kinematic data is embedded by a specific linking mechanism to instances of the open standard data model Collada (ISO/PAS 17506) originally for the purpose of the virtual commissioning. This mechanism may be useful for autonomous navigation of movable system parts but is a very special use case for agents and therefore out of scope of this document. To dynamically define the meaning of data each Attribute contains an entry *Unit* to explicitly quantify a measure and an entry *RefSemantic* to add a string based hint for the semantical interpretation to qualify the value. Additionally AutomationML declares the extended concept of *PropertySets*. The definition of *PropertySets* allows to map proprietary attributes of AutomationML objects with semantically predefined attributes. These attributes are managed in shared libraries of roles. The *RoleClassLibrary* concept will be explained in detail in the paragraph about semantical depth. This way the data models of the platform the agent and the production system have been defined in are being decoupled a semantically correct interpretation of the data by the agents is ensured.

C. Views on information

As already mentioned different types of agents access different information derived from different ***subsets of data*** of an asset. Depending on the purpose an agent was designed for there are different decomposition structures of a system. Therefore it is required to overcome the limitation of a single hierarchy and represent the same system elements in optional ***alternative hierarchies***. To avoid the expensive processing of search and filter operations there are several mechanisms in AutomationML that may be used to explicitly define aggregates or subsets of data. To organize the elements in different decomposition structures a methodology was introduced in [28] to implement three distinct views on the same system elements related to the focus on Product, Process and Resource (PPR) demands. To represent the same element in more then one hierarchy the standardized *MirrorObject* was used which basically is a pointer to the original element. To aggregate sets of data the same principle of pointers is used with the extended concept of a so called *GroupObject* which containerizes a set of *MirrorObjects*. In comparison to alternative hierarchies, groups may be associated with *Facets*. A *Facet* is also one of the extended concepts of AutomationML and supports the specification of a sub-view of a parent element. It specifies a subset of *Attributes* and *ExternalInterfaces* of an *InternalElement*. Different agents may access the different hierarchies and signatures of an element to avoid intense search and preprocessing operations.

D. Relations and networks of information objects

In several use cases like self-organization, plug-and produce, scheduling and logistics the agents have to rely on more complex ***relations*** between elements. While chains of elements (processes, routes, dependencies) may easily be represented in ordered sets hierarchical structures are not useful for the distinction between ***uni- and bidirectional*** relations as well as slopes and cycles up to full meshed ***networks*** of elements. AutomationML has built-in mechanisms for the definition of connection points of elements called *ExternalInterface* and *InternalLink* objects that establish connections between them. In [29] a method is described to exploit this linking mechanism to define directed graphs in AutomationML. By defining an *InternalElement* being an edge object and inserting it in between two links to the related elements a connection with rich semantical capabilities may be specified. That way agents may discern routes through the system, dependencies as well as logical or physical connections from the model enabling a variety of applications.

E. Semantical depth

To express ***commonalities of elements*** for algorithmic manipulations class based inheritance is a very popular concept. But for the application of MAS in dynamic system scenarios statically typed systems of data structures like inheritance of classes are not practicable. In contrast to applications with a fixed purpose it is not possible to anticipate all occurrences of variety and each necessary class structure in advance. Therefor the information model is required to be capable of ***runtime changes***. But modifying the model structure dynamically inhibits the power of type safety. To avoid the necessity of runtime analysis a concept of an ***alternate access pattern*** to identified structures is necessary. This concept requires the possibility to indicate the occurrences of the structural pattern by some kind of ***annotation*** to the model. Patterns used in the application are documented in application specific ***pattern catalogs***. A possible implementation of this requirements with prototypical objects was discussed in [30]. AutomationML is based on the CAEX standard which implements its infomration model on prototypical objects. But in AutomationML the identification of instances was changed to an explicit addressing scheme by IDs without adjusting the inheritance concept of CAEX which uses path to allocate base classes. Additionally the inheritance rules were loosened to enable the exchange of draft design data which is by definition incomplete and possibly inconsistent. But at the same time it also enables more flexible ways to evolve the data model with changing demands. This freedom requires carefully defined processes and semantical enhancements. The semantical expressiveness and extensibility of AutomationML has been discussed in several publications [31], [32], [33]. The main concepts defined in the standard are the class libraries for *RoleClasses*, *InterfaceClasses* and *SystemUnitClasses*. The concept of prototypic inheritance holds for class hierarchies and allows the definition of commonalities. *Interface- and RoleClasses* allow the semantical enrichment of *ExternalInterfaces* and *InternalElements*. *SystemUnitClasses* are used to define prototypes for *InternalElements*. But the classes are not meant to be inherited but used as a master copy. For implementation of typesafe access to the model additional concepts have to be applied to link content to a standardized feature system or taxonomy [34]. Such classification schemes may be represented in RoleClassHierarchies where each RoleClass

represents an entry and defines the required attributes and interfaces of an element having this role assigned. Annotating instance data with standardized assignment of *RoleClasses* realizes the alternate access pattern while the weak inheritance brings the freedom of agility to the information model. This way the requirements of the application of MAS in dynamic system setups are met.

F. Consistency

To ensure or at least asses the ***correctness*** of information and the agents' decision based upon it, it is very important to express ***constraints*** on the content in the information model. Beyond the formal description of the data model by related XML-Schema definitions which ensures syntactical and basic structural correctness it is required to define logical and functional dependencies within the system. In [31] it has been shown that the Object Constraint Language (OCL) is capable of describing logical dependencies within AutomationML models to proove their semantical correctness. In a different context MathML was used within AutomationML to describe functional dependencies in mathematical formulas. Adding MathML and OCL definitions directly to the AutomationML model facilitates the dynamic enhancement of the model.

G. Interactivity

The ideas of the I40 require production systems to evolve continuously throughout every stage of their life-cycle. As a production system changes, agents have to be able to manipulate its model in order to provide up-to-date information to other entities. In a full fledged Self-X application on a dynamically changing production system the MAS has to rely on a powerful knowlegdebase to aquire and ***evolve the system descriptions***. This knowlegdebase is meant to contain all information that may be collected along the life-cycles of the system. It ranges from static, centralized and well structured models out of the systems engineering, along still structured but frequently changing runtime-data to dynamically evolving and partially unstructured or even inconsistent support data. This tangle of data raises high demands on the agility of the applied information models as well as the information infrastructure for the aggregation and exploitation of knowledge [35]. As we have shown before AutomationML adresses the requirements on ***agility of the information models*** with several concepts. The introduced changes in structural allocation, granularity, existence and availability of elements in the system have additionally to be supported by the services representing the model in an online application and form an ***agile information infrastructure***. To enable a consistent and vivid runtime solution the characteristics of object orientation have to be fully implemented serving the need to distribute the information objects across the infrastructure. Identity and state have already been applied but the encapsulation and behavior are partially described in the standard documents. Also the server applications that were developed together with the companion specification for AutomationML and OPC-UA didn't address this characteristics. Therefor a customized OPC UA AML server has to be utilized that ensures the encapsulation and realizes the behavior of a component to change its internal structure and state.

There are (at least) 4 different events that entail agents to update the production systems model in a modular configuration:

- Integration of a new resource
- Change of resource
- Removal of a resource
- Exchange of a resource

These modifications implicate different changes to an AutomationML model. When integrating a new or changing an existing resource, it may occur that *InternalElements*, *ExternalInterfaces* or *SupportedRoles* have to be added. In each of these cases, dependencies (*SystemUnitClasses*, *InterfaceClasses*, *RoleClasses*) have to be resolved by importing missing libraries into the model. In order to actually integrate the added objects into the model, relations to other objects in the system have to be defined. In AutomationML, this is accomplished by connecting the objects interfaces with so called *InternalLinks*. On the other hand, when removing elements from the model by either removing or changing an existing resource, these *InternalLinks* have to be removed prior to the execution of the remove operation in order to keep the model in a consistent state. When removing elements, it is reasonable to also remove unused dependencies from the model in order to keep it as slim and performant as possible. These considerations introduce the need for some kind of dependency management. In order to encapsulate the model from the agents, this maintenance should not be performed by the agents themselves, but rather by the server that holds the model. This design adds up all the key concepts of object orientation (identity, state, behavior, encapsulation). Each object may be represented as an OPC UA node and may be located on any resource hosting an OPC UA - AML Service. Given this services agents may manipulate structure, views, annotations and relations of elements forming knowledgebase systems.

IV. Integration of Production system models in shop floor for the application of MASs

As introduced in the previous section there are distinct sets of requirements to a possible knowledgebase for MASs in a production system scenario. In the following three subsections a testbed will be described that was used to evaluate how the proposed solution meets those requirements. In the first subsection the general architecture for an integrated system of AutomationML, OPC UA and MASs will be described. In the second subsection an example implementation of the proposed architecture will be presented. In subsection three some use-cases are described that were run on the presented lab size production system.

A. Architecture Proposal

The parts of the proposed architecture may be distinguished into three major conceptual structures that are the execution

and control of the behavior of system parts, the management of the integral system and the management of knowledge about the system. The execution structure is located within or close to the control system on the field or edge level of the production system. Each appliance for the manufacturing is self-contained with its control in industrial quality. Therefor it is connected to the hardware by field bus technologies and meets constraints on real-time behavior and safety of the governed system part. For higher level applications each control interfaces with a service platform e.g. a service API within the control hardware. As a migration strategy a realization on state-of-the-art PLC runtime with an integrated OPC UA server per appliance is suggested. Due to the decoupling of components by vendor- and platform independent OPC UA architecture future technologies may transparently replace this legacy systems. Each of these clusters is associated with field level agents along with a model based description of their logical cooperation interfaces in AutomationML, rendering them intelligent system modules. The closer the agents and the model are integrated with the control hardware the more modularity and adaptability is gained.

The second structure in the architecture is composed by hard- and software which enables the system to be run as an integrated whole. Its functions are on a higher level of abstraction of the production process and therefor called higher level applications even though they are not necessarily superordinate to the field level. They contain the planning, operation, supervision, service and maintenance of the production as well as the production system. In this sense it heavily overlaps with traditional functions of manufacturing execution system (MES), supervisory control and data acquisition (SCADA) as well as enterprise resource planning (ERP) technologies. Here functions like “smartness” and “self-X” of the production and production system are realized by MAS. While the interfaces to the control system are realized using OPC UA the implementation of cooperative behavior has been based on the middleware of the MAS. This design enables the separation of concerns between different levels of operation and especially the loose coupling between data describing the structural conditions of the system and the operational data of the application executed on the system. This avoids complex integration dependencies between system modules and their application and supports platform independence and maintainability of system parts.

The third and most novel structure of this proposal is the information infrastructure. Exploiting the power of the applied middleware solutions it is well tailored to the intended purpose. While the information out of the semantical context of cooperation stays within the implementation of the MAS the communication with the information repository and the executive parts of the system is performed using the open industrial OPC UA technology. Special attention should be spent to the fact that OPC UA is not only used as transmission protocol of the data. The strong mechanism of self-descriptiveness using the OPC UA information models enables the establishment of the distributed machine readable knowledgebase which is the cardinal gain of the proposed architecture. By the instantiation of the AutomationML model within the information model of OPC UA all the knowledge collected in the design process of the system may get published to the servers of the proposed architecture and forms the distributed knowledgebase which is continuously evolved. For this purpose an OPC UA server software is used within the architecture which serves AutomationML files according to the companion specification for AutomationML extended with the consistency preservation explained in section III G. A key feature here is that within this model pointers to remote OPC UA servers and nodes are represented. This way it realizes an arbitrary distributable model of the system with the expressiveness of AutomationML. Within this ecosystem the agents are able to read AutomationML, browse the description of the system, learn and reason upon the information and in a mature implementation enhance or improve the model. Thereby the model dynamically evolves and satisfies the requirement of the agility of information model and information infrastructure.

B. Application Example

The application presented here implements an examplary version of MES agents in the sense of [36] and [14] integrated with the mentioned technologies. It consists of a modular lab size production system of eight modules. The modules may be arranged in the laboratory arbitrarily. Each module represents a manufacturing resource and is self-contained with a state-of-the-art control system and an integrated OPC UA Server which serves the control parameters of the PLC to the network (see figure 1).

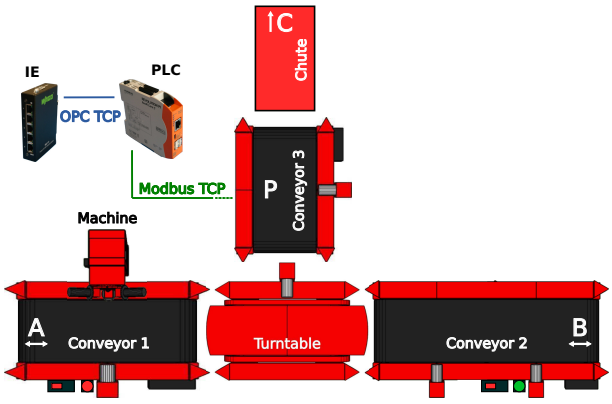

Fig. 1. architecture of a single component in lab size production system

The modules are connected to an Industrial Ethernet infrastructure. In the scope of the whole system the components are integrated with an aggregating OPC UA server that holds an AutomationML model (see listing 1). It contains an object tree with an overall description of the system and points to the modules’ servers (see figure 2). Changes in the physical setup of the system have to be observed and reported to this server. The MAS is built on JADE technology. The agents are equipped with the feature to surf OPC UA networks and interpret discovered AutomationML descriptions.

Listing 1. AutomationML segments of production systems' resource topology

```
<InstanceHierarchy Name="ResourceTopology">
[..]
 <InternalElement Name="PPLab" ID="GUID_01">
  <InternalElement Name="Conveyor 1" ID="
      GUID_02">
   <ExternalInterface Name="WPPort_Conv1C"
       RefBaseClassPath="TransportClassLib/
       WPPort" ID="GUID_11" />
   <RoleRequirements RefBaseRoleClassPath="
       PPLabRoleClassLib/TransportNode" />
  </InternalElement>
  <InternalElement Name="Turntable" ID="GUID_03
      ">
   <ExternalInterface Name="WPPort_TurnA"
       RefBaseClassPath="TransportClassLib/
       WPPort" ID="GUID_12" />
   <RoleRequirements RefBaseRoleClassPath="
       PPLabRoleClassLib/TransportNode" />
  </InternalElement>
  <InternalLink Name="Conv1_Turn"
      RefPartnerSideA="GUID_02:WPPort_Conv1C"
      RefPartnerSideB="GUID_03:WPPort_TurnA" />
  <RoleRequirements RefBaseRoleClassPath="
      AutomationMLBaseRoleClassLib/
      AutomationMLBaseRole" />
 </InternalElement>
</InstanceHierarchy>
<InterfaceClassLib Name="TransportClassLib">
 <InterfaceClass Name="WPPort" />
</InterfaceClassLib>
[..]
<RoleClassLib Name="PPLabRoleClassLib">
 <RoleClass Name="TransportNode">
  <ExternalInterface Name="WPPort_NodeX"
      RefBaseClassPath="TransportClassLib/
      WPPort" ID="GUID_22" />
 </RoleClass>
</RoleClassLib>
[..]
```

1) Field Level Components: Each of the 8 modules contains a single-board computer which operates an industrial PLC runtime with a built-in OPC UA Server. The PLC program is written in the PLC programming language *structured text (ST)* defined in the 3rd part of the open international standard IEC 61131 [37] and stored in the related PLCOpenXML exchange format. For each module there is an AutomationML object tree containing pointers to the control code and references to its variables as well as a description of the module behaviors controlled by the code linking all together (see listing 2). This behaviors are whitelisted to the OPC UA interfaces of the PLC and thereby exposed to the outside as a service of the module. Deploying this AutomationML project to the PLC programming environment and an aggregating OPC UA Server capable of AutomationML hierarchies constitutes a service based interactive module with a description of its skills. For the prototype presented here the control code implemented only indecomposable behaviors of the module while sequences of behaviors were realized outside of the field control components. At the moment this is realized by an agent not further introduced in this paper that knows to execute a static sequence on a module. It forms the base of future work of the authors about skill based discovery and commission of agents on the job shop.

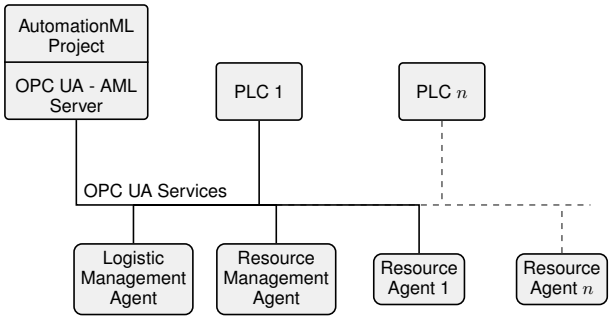

Fig. 2. MAS architecture for the application example

2) OPC UA Servers with AutomationML backend: To gain the full potential of an OPC UA cluster serving AutomationML models the interactivity had to be realized for the AutomationML information model in a server application. Starting from an implementation of a Java based AutomationML engine joined with the OPC UA Stack implementation of the MILO project an OPC UA server implementation was created that allows the object-oriented interaction with AutomationML object hierarchies. Thereby the agents may not just acquire information from the model but rather reason upon the structures based on inference rules to gain new knowledge and communicate this knowledge through the models.

Listing 2. AutomationML code segments of OPC-UA declaration

```
<InternalElement Name="OPCUA Server" ID="5
    b196fe6-83a5-4b6e-b011-f21e3436ce1b">
 <Attribute Name="DiscoveryURL"
     AttributeDataType="xs:anyURI">
  <Value>opc.tcp://pplab_master:48010/</Value>
 </Attribute>
[..]
 <RoleRequirements RefBaseRoleClassPath="
     DataVariableRoleClassLib/DataSource/OPCUA-
     Server" />
</InternalElement>
 <InternalElement Name="PLCProgram" ID="
     GUID_33" RefBaseSystemUnitPath="
     DataVariableExampleSUCLib/OPCUA-Example">
  [..]
  <Attribute Name="OPCUA-Variable_productType"
      AttributeDataType="xs:string">
   <RefSemantic CorrespondingAttributePath="aml
       -dataSourceType:OPCUA" />
   <Attribute Name="NodeId">
    <Value>ns=4;s=PROGRAM1.productType</Value>
   </Attribute>
   <Attribute Name="RefDataSource">
    <Value>5b196fe6-83a5-4b6e-b011-
        f21e3436ce1b</Value>
   </Attribute>
  </Attribute>
 </InternalElement>
```

3) Applied agent types: Within this infrastructure of modular self describing modules an MAS was set up that was taught to interpret and execute a production process description given in AutomationML. The agents automatically acquired the information of the process order, the capabilities of the modules and the path the product may travel through the system. Therefor the following agent types were designed.

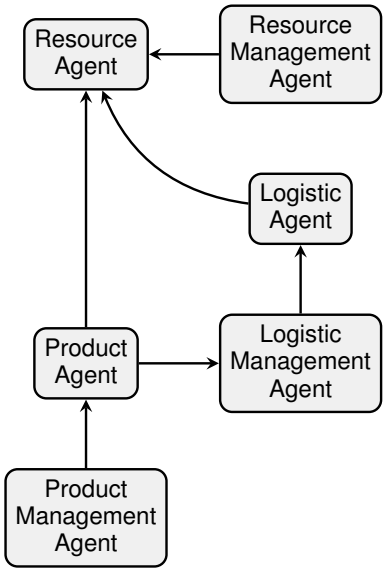

Fig. 3. Applied agent types

- **Resource Agent (RA)**: RAs manage resources. They are responsible for the initialization of resources, the execution of processes and the prevention of multiple resource allocation. Resource agents serve as gateways between the field level and the MAS.
- **Resource Management Agent (RMA)**: The RMA manages the Resource Agents. Whenever a resource is added to or removed from the (model of the) factory,the RMA will adjust the number RAs accordingly.
- **Logistic Agent (LA)**: A LA offers *some* simple transport service to the LMA. (simple in the sense of its abstract representation within the model)
- **Logistic Management Agent (LMA)**: The LMA offers a complex transport service to the PA. If a good has to be moved from one place to another, the LMA will (use sophisticated optimization methods in order to) find the shortest path from target to destination. Afterwards it will search for LAs that offer the simple transport services necessary in order to execute the complex transport and call for proposals. The cheapest transport offer will be accepted and scheduled.
- **Product Agent (PA)**: The PA is responsible for the production of a single product. Upon creation, it will determine the production steps that have to be performed in order to create the product (with respect to certain constraints, e.g deadlines, costs or quality). It will then sequentially find RAs that offer the execution of the next production step, call for proposals, commission the transport (to the cheapest available resource), configure the machine(s) and execute the appropriate process.
- **Product Management Agent (PMA)**: The PMA manages production orders. When an order is placed, the PMA spawns new Pas.
- **Directory Facilitator**: The DF provides a central discovery service for services offered by other agents in the MAS architecture.

C. Application Scenarios

The presented testbed primarily serves experiments for (optimization) methods of the industrial engineering. Different setups and production program plans are executed on the system by the agents. According to the physical arrangement the setups are published and updated to the system in form of AutomationML models. Additionally an agent monitors process variables and calculates several KPIs to monitor the performance of a solution. This way the production system may be easily reconfigured to simulate different layouts and programs for a production process. A second scenario that was successfully implemented for a doctoral thesis is a fault detection and classification system [38]. The applied MAS tracked KPIs and component state indicators. This data was evaluated against constraints to identify errors. Based on the fault register recovery strategies where performed by other agents. A third scenario was implemented which realized automatic reconfiguration of the system. The modules and products were equipped with fiducials and a computer vision system based on openCV was implemented that tracks the presence and positional relation of products and resources. To simulate a fault the fiducial of a resource was covered and got invisible for the agents this way. A specific agent recognizes the change and updates the AutomationML knowledgebase. Supervisory agents then successfully adjust the production and transportation routes in the system based on the dynamic model. The mentioned scenarios have proven applicable and give the opportunity to be scaled to industrial implementation.

V. Conclusion

In this paper it has been shown how the open industrial standards OPC UA for service-oriented communication as well as AutomationML for semantic data exchange meet the requirements of the establishment of a Smart Manufacturing Component (I40 component). Providing a single service interface to the semantical description as well as to the behavior within a self-contained component the infrastructure of Self-X is realized upon already available standardized technologies. The presented solution scales from legacy to modern hardware as well as from small facilities to huge geographically spread value networks. The applied use cases have shown the applicability of the approach and the ease of integration with innovative applications. This proposal enables a lot of opportunities for the implementation of a knowledge base system for production and will be the basement for further work of the authors on third party application integration and dynamically evolving skillsets of agents.

References

[1] H. Kagermann, J. Helbig, A. Hellinger, and W. Wahlster, *Recommendations for Implementing the Strategic Initiative INDUSTRIE 4.0: Securing the Future of German Manufacturing Industry ; Final Report of the Industrie 4.0 Working Group*. Forschungsunion, 2013.

[2] P. I. 4.0, "Structure of the administration shell - continuation of the development of the reference model for the industrie 4.0 component." ebook (PDF) available at https://www.plattform-i40.de/I40/Redaktion/EN/Downloads/Publikation/structure-of-the-administration-shell.html, 01.04.2016.

[3] DIN Spec, "91345: Reference architecture model industry 4.0 (rami 4.0)," *DIN Spec*, 2016.

[4] A. Lüder, M. Schleipen, N. Schmidt, J. Pfrommer, and R. Henßen, "One step towards an industry 4.0 component," in *2017 13th IEEE Conference on Automation Science and Engineering (CASE)*, pp. 1268–1273, Aug 2017.

[5] DIN Spec, "16592: Combining opc unified architecture and automation markup language," *DIN Spec*, 2016.

[6] M. Wooldridge, *An Introduction to MultiAgent Systems*. Wiley Publishing, 2nd ed., 2009.

[7] R. Unland, "Chapter 2 - Industrial Agents," in *Industrial Agents* (P. Leitão and S. Karnouskos, eds.), pp. 23–44, Boston: Morgan Kaufmann, 2015.

[8] P. Leitão, "Agent-based distributed manufacturing control: A state-of-the-art survey," *Engineering Applications of Artificial Intelligence*, vol. 22, no. 7, pp. 979 – 991, 2009. Distributed Control of Production Systems.

[9] P. Castellini, C. Cristalli, M. Foehr, P. Leitao, N. Paone, I. Schjolberg, J. Tjønnås, C. Turrin, and T. Wagner, "Towards the integration of process and quality control using multi-agent technology," in *IECON 2011 - 37th Annual Conference of the IEEE Industrial Electronics Society*, pp. 421–426, Nov 2011.

[10] M. Onori, A. Maffei, and F. Durand, "The ideas plug and produce system," p. 339, 01 2013.

[11] N. Antzoulatos, E. Castro, D. Scrimieri, and S. Ratchev, "A multi-agent architecture for plug and produce on an industrial assembly platform," *Production Engineering*, vol. 8, pp. 773–781, Dec 2014.

[12] P. Leitão, J. Barbosa, P. Vrba, P. Skobelev, A. Tsarev, and D. Kazanskaia, "Multi-agent system approach for the strategic planning in ramp-up production of small lots," in *2013 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 4743–4748, Oct 2013.

[13] F. L. Almeida, B. M. Terra, P. A. Dias, and G. M. Goncalves, "Adoption issues of multi-agent systems in manufacturing industry," in *2010 Fifth International Multi-conference on Computing in the Global Information Technology*, pp. 238–244, Sept 2010.

[14] A. Lüder, A. Calá, J. Zawisza, and R. Rosendahl, "Design pattern for agent based production system control – A survey," in *2017 13th IEEE Conference on Automation Science and Engineering (CASE)*, pp. 717–722, Aug 2017.

[15] "Innovative modelling approaches for production systems to raise validatable efficiency (improve).." http://www.improve-vfof.eu/. Accessed: 2017-12-19.

[16] "Perform – production harmonized reconfiguration of flexible robots and machinery.." http://www.horizon2020-perform.eu. Accessed: 2017-12-19.

[17] "Basissystem industrie 4.0.." https://www.basys40.de/. Accessed: 2017-12-19.

[18] Y. Luo, Y. Duan, W. Li, P. Pace, and G. Fortino, "Workshop networks integration using mobile intelligence in smart factories," *IEEE Communications Magazine*, vol. 56, pp. 68–75, Feb 2018.

[19] G. Fortino, C. Savaglio, and M. Zhou, "Toward opportunistic services for the industrial internet of things," in *2017 13th IEEE Conference on Automation Science and Engineering (CASE)*, pp. 825–830, Aug 2017.

[20] G. Fortino, W. Russo, C. Savaglio, W. Shen, and M. Zhou, "Agent-oriented cooperative smart objects: From iot system design to implementation," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1–18, 2017.

[21] A. Lüder, N. Schmidt, and R. Drath, *Standardized Information Exchange Within Production System Engineering*, pp. 235–257. Cham: Springer International Publishing, 2017.

[22] A. Lüder, N. Schmidt, and R. Rosendahl, "Data exchange toward plc programming and virtual commissioning: Is automationml an appropriate data exchange format?," in *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*, pp. 492–498, July 2015.

[23] L. Berardinelli, S. Biffl, A. Lüder, E. Mätzler, T. Mayerhofer, M. Wimmer, and S. Wolny, "Cross-disciplinary engineering with automationml and sysml," vol. 64, 01 2016.

[24] AutomationML e.V., "Automationml - current projects." https://www.automationml.org/o.red.c/projects.html, 2018. Accessed: 2018-05-10.

[25] DIN, DKE, VDE, "German standardization roadmap on industrie 4.0." Available at https://www.din.de/de/forschung-und-innovation/themen/industrie4-0/roadmap-industrie40-62178, 18.05.2018.

[26] W. Mahnke, S.-H. Leitner, and M. Damm, *OPC unified architecture*. Springer Science & Business Media, 2009.

[27] M. Schleipen, R. Henßen, O. Sauer, N. D'Agostino, M. Damm, A. Dogan, J. Ladiges, C. Ewertz, A. Gössling, T. Holm, S. Hoppe, A. Lüder, N. Schmidt, and R. Wilmes, "Harmonisierung im Kontext Industrie 4.0 AutomationML und OPC UA," 06 2017.

[28] M. Schleipen and R. Drath, "Three-view-concept for modeling process or manufacturing plants with AutomationML," in *2009 IEEE Conference on Emerging Technologies Factory Automation*, pp. 1–4, Sept 2009.

[29] A. Lüder, N. Schmidt, and S. Helgermann, "Lossless exchange of graph based structure information of production systems by AutomationML," in *2013 IEEE 18th Conference on Emerging Technologies Factory Automation (ETFA)*, pp. 1–4, Sept 2013.

[30] H. Lieberman, "Using Prototypical Objects to Implement Shared Behavior in Object Oriented Systems," pp. 214–223, 1986.

[31] M. Schleipen, "A concept for conformance testing of automationml models by means of formal proof using ocl," in *2010 IEEE 15th Conference on Emerging Technologies Factory Automation (ETFA 2010)*, pp. 1–5, Sept 2010.

[32] M. Sabou, F. Ekaputra, O. Kovalenko, and S. Biffl, "Supporting the engineering of cyber-physical production systems with the automationml analyzer," in *2016 1st International Workshop on Cyber-Physical Production Systems (CPPS)*, pp. 1–8, April 2016.

[33] I. Grangel-González, D. Collarana, L. Halilaj, S. Lohmann, C. Lange, M.-E. Vidal, and S. Auer, "Alligator: A Deductive Approach for the Integration of Industry 4.0 Standards," in *Knowledge Engineering and Knowledge Management* (E. Blomqvist, P. Ciancarini, F. Poggi, and F. Vitali, eds.), (Cham), pp. 272–287, Springer International Publishing, 2016.

[34] M. Obst, T. Holm, L. Urbas, A. Fay, S. Kreft, U. Hempen, and T. Albers, "Semantic description of process modules," in *2015 IEEE 20th Conference on Emerging Technologies Factory Automation (ETFA)*, pp. 1–8, Sept 2015.

[35] J. Scott, "A practical guide to microservices and containers." eBook (pdf) available at https://mapr.com/ebooks/microservices-and-containers/, 2017.

[36] V. G. F. . Spec, "2653 part3: Multi-agent systems in industrial automation application," *VDI/VDE Spec*, 2012.

[37] I. E. Commission, "Programmable controllers - part 3: Programming languages," *IEC Spec*, 2013.

[38] H. Bayanifar, *Agent-based mechanism for smart distributed dependability and security supervision and control of Cyber-Physical Production Systems*. 2017. xvi, 159 Seiten, Illustrationen.