Presented as late breaking results contribution
21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Customization of agent-based manufacturing applications based on domain modelling

O. Casquero*, A. Armentia*, E. Estevez, A. López*, M. Marcos***

**Systems Engineering and Automatic Control Department, University of the Basque Country (UPV/EHU), Bilbao, Spain (e-mail: {oskar.casquero, aintzane.armentia, alejandro.lopez, marga.marcos}@ehu.eus).*
*** Electronics and Automation Engineering Department, University of Jaén, Jaén, Spain, (e-mail: eestevez@ujaen.es)*

Abstract: Agent-based architectures have become a mainstream technological concept that may allow factories to adopt distributed intelligence patterns that enable the advanced manufacturing model of Industry 4.0. However, there is a lack of methodologies and tools that support the specification, deployment and execution of agent-based manufacturing applications. This article describes the first steps to build an agent-based platform that provides a reusable software core that can be customized to offer the services required for factory-specific manufacturing systems. In this sense, the contribution of this article is two-fold: on the one hand, the proposal of a model-based definition of manufacturing applications based on factory-specific concepts that are represented in three XML schemas; on the other hand, a proposal for distributing the complexity of product intelligence in a set of agents that allow achieving separation of concerns regarding customer interaction and traceability of the production.

Keywords: flexible manufacturing, product-oriented manufacturing, model-driven design, multi-agent systems, XML schema

1. INTRODUCTION

Holonic multi-agent architectures (HMAA) (Leitão, 2009) and the notion of product intelligence (McFarlane et al., 2013) have become a great influence to adopt distributed intelligence patterns for enabling the advanced manufacturing model of Industry 4.0 (Cruz Salazar et al., 2019). The holonic approach can be implemented using multi-agent technology. The majority of agent-based HMAA are based on two agents: the Product Agent (PA) and the Resource Agent (RA) (Kovalenko et al., 2019b). The PA is where the product intelligence is introduced: the PA is tasked with seeking the manufacturing resources needed by its physical part and triggering the different manufacturing services offered by RAs. The RA is responsible for interfacing the equipment in response to service requests from the PAs.

Most of the research efforts in this area have been focused on the extension of PA and RA-based agent architectures (Cruz Salazar et al., 2019) or the proposal of different methods for product planning, scheduling and execution control (Kovalenko et al., 2019a). In order to assess the feasibility of their proposals, most of these works simulate the manufacturing systems used in their test benches, while only a few of them address the issue of implementing them. As far as authors know, the design of digital platforms that enable flexible orchestration of manufacturing applications remains a challenge that requires further research. The modelling and formal validation of manufacturing applications plays a key role to deal with this challenge (Leitao et al., 2006). The modelling of manufacturing applications involves identifying the entities and the structure that underlie those applications, whereas the formal validation aims to verify the correctness of factory-specific instances of those applications. Traditionally, the modelling of manufacturing applications has been focused in UML, whereas Petri nets have been proposed for the formalization of the behavior and interaction of the collaborative entities in a manufacturing system (Leitao et al., 2006). However, to the knowledge of the authors, little attention has been given to the specification of the manufacturing applications from which the corresponding agents can be automatically generated. In order to generate a correct agent set, it is mandatory to analyze the correctness of the application specification.

In this context, the contribution of this work is twofold. On the one hand, it presents a generic registration process for the definition of custom manufacturing applications. It is based on a model-driven design (MDD) approach (i.e., in terms of concepts, terminology and syntax of the application), making it possible to customize an agent-based platform to a concrete manufacturing domain. On the other hand, the feasibility of the MDD approach is illustrated with a proposal for distributing the complexity of the product-oriented manufacturing (POM) in a set of agents that allow achieving separation of concerns regarding customer interaction and production scheduling and supervision.

The rest of the paper is organized as follows: Section 2 details the modelling approach and the registration process that allows the definition of factory-specific manufacturing applications; in Section 3, the modelling approach is illustrated with a proposal for the separation of concerns of the POM; finally, Section 4 collects the conclusions.

Copyright lies with the authors

2. DOMAIN MODELLING AND VALIDATION

The model-based approach for the complete and correct definition of manufacturing applications must cope with variations in the manufacturing application definition (new characterization of a concept, new concepts or new relationships among them) which can be derived from the flexibility demands of current evolving manufacturing systems. This paper proposes defining these applications through the definition of the domain model in three steps.

Initially, it is necessary to identify the set of application concepts relevant to the manufacturing domain. These refer to application entities that will play a relevant role within the multi-agent manufacturing architecture. For example, Kovalenko et al. (2019b) consider manufacturing applications as a set of PAs, each one in charge of scheduling and supervising the manufacturing processes related to its associated physical part. Similarly, the approach described in Vrba et al. (2011) also identifies PAs as key elements of their holonic architecture, in charge of scheduling the production plan to achieve the final product. In addition, Vrba et al. (2011) also define Order Agents (OAs) for receiving orders from higher levels of the control system and creating the PAs. Thus, Kovalenko et al. (2019b) identify a unique application concept: The Product; whereas Vrba et al. (2011) distinguish two concepts: The Order and the Product.

Once the relevant concepts are identified, the first step is to characterize such concepts with the information that the corresponding agents need to ensure their correct execution. For instance, the Product concept of Vrba et al. (2011) can be characterized in terms of the type of the product, the quantity of items to be produced and the production steps that conform its production plan. As the management of a manufacturing system may evolve, it could happen that an agent would need to manage new information, which implies extending or modifying the characterization of its corresponding concept. To that end, we propose collecting the set of properties that characterize every concept in a separate meta-model, the *Properties* meta-model. A possible XML schema for this meta-model is presented in Fig. 1. It is possible to define simple properties as attributes (e.g., the type of the product as the *type* attribute) and complex properties through groups of elements (e.g., the *productionPlan* property as a sequence of production steps).

The second step is to declare the concepts in the so-called *Concepts* meta-model. As illustrated in Fig. 2, the *Concepts* XML schema includes the *Properties* XML schema, linking concepts with their characterizations. Besides, Fig. 2 exemplifies the approach described in Vrba et al. (2011), declaring the Order and Product concepts as root elements.

The third step is to define the hierarchy, if any, among the different concepts. This is the case of the approach proposed by Vrba et al. (2011), where a hierarchy exists between the Order concept and the Product concept, since the OAs create as many PAs as the number of products characterizing the order. Therefore, the agent related to a concept of the hierarchy is responsible for creating the agents belonging to the immediate lower level. The hierarchy is defined in the *Hierarchy* meta-model, which redefines the concepts to reflect the hierarchical structure among them. Fig. 3 illustrates the case of Vrba et al. (2011), where the Order concept is redefined to reflect the hierarchical relation with the Product concept.

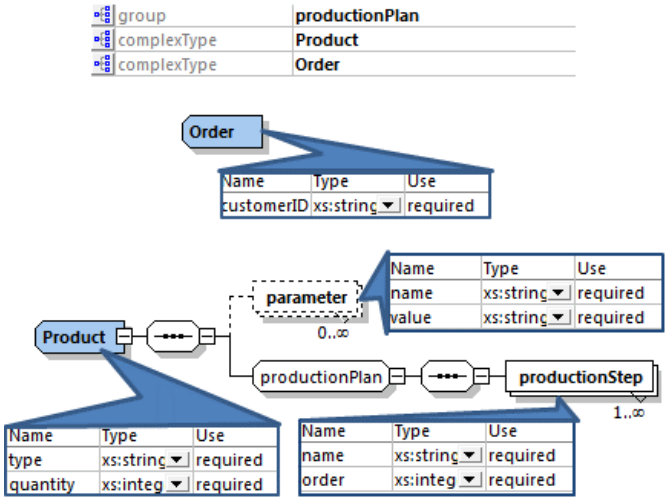

Fig. 1. Implementation of the *Properties* meta-models in an XML schema.

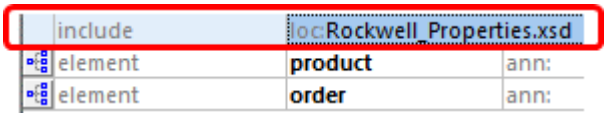

Fig. 2. Implementation of the *Concepts* meta-models in an XML schema

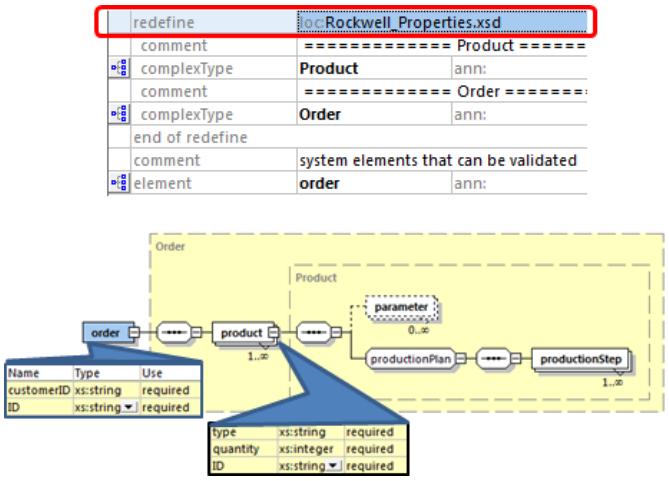

Fig. 3. Implementation of the *Hierarchy* meta-model in an XML schema.

Regarding the validation of manufacturing applications, a generic registration process is proposed to ensure that only complete and correct factory-specific manufacturing applications are defined. The registration of a manufacturing application consists of the registration of all its composing entities as follows (see Algorithm 1):

- Phase 1. An initial iterative entity registration which includes unitary validation. In this phase, the manufacturing platform assigns a unique identifier to each new correct entity. In the example, the order and its products must be registered to complete the whole application registration. Additionally, in the case of a hierarchical application structure, the iterative registration of application entities must be performed in a top-down way. That is, product registration does not follow any concrete order for Kovalenko et al. (2019b), whereas the orders of Vrba et al. (2011) have to be registered before their products.

- Phase 2. The validation of the application hierarchy.

The approach proposed by Vrba et al. (2011) will be used to illustrate the registration of an order with two products. Every entity registration starts with a request from an external actor, which must provide the manufacturing platform with the following information (Algorithm 1, Phase 1): a) the concept related to the new application entity, e.g., a Product concept; b) main characteristics of the new entity, namely: the type of the product (P1), the quantity of items to be produced (100) and the production steps that conform its production plan (Drill, Fill, Weld…); and c) the parent entity at the hierarchy, identified by its unique identifier and the related concept. In the example, the parent of the P1 product must be a previously registered entity of the Order concept.

Then, the new entity is validated (Algorithm 1, line 6) as it is detailed in Algorithm 2. Initially, it is verified that the parent entity has been previously registered. Then, the new entity is validated (Algorithm 12, line 10) in terms of its main properties by means of the *Concepts* meta-model (*Concepts.xsd*). If correct (Algorithm 1, line 10), up-to-now registered entities are stored at a temporary model (*temp_app.xml*). Finally, when all the application entities have been registered, the whole manufacturing application is checked (Algorithm 1, Phase 2). For that, the temporary model is validated against the *Hierarchy* meta-model (*Hierarchy.xsd*) to ensure that the application follows the previously defined hierarchical and dependency relations. If the application structure is correct, the new manufacturing application is stored in the platform, ready for its start-up.

3. A PROPOSAL FOR ACHIEVING POM

The second contribution of the paper is the proposal of a platform for POM, named FLEXMANSYS (FLEXible MANufacturing SYStem) which supports a) model-based definition of manufacturing applications based on factory-specific concepts, and b) the execution and management of those applications through a set of distributed agents that are created from those factory-specific concepts. The concepts, terminology and syntax of the applications handled by FLEXMANSYS are defined by the structure of the three XML schemas described in Section 2. Thus, FLEXMANSYS can be customized to any factory or manufacturing domain.

FLEXMANSYS lies on an agent-based middleware whose core consists of the System Agent (SA). The SA provides an application programming interface (API) that allows a) registering, starting and stopping manufacturing applications; and b) querying and updating the status of the whole manufacturing system (stored in the so-called System Model, SM) throughout the whole execution cycle of the application.

Algorithm 1. Generic registration process of manufacturing applications.

```
Input: manufacturing_app as an entity_set
Output: temp_app.xml
 1: // Phase 1: iterative entity registration and validation
 2: for each entity in entity_set ordered by top_down in hierarchy do
 3:   set concept of entity
 4:   set properties of entity
 5:   set parent_id and parent_concept of the parent of entity
 6:   entity_result ← call entity_validation with entity parameter
 7:   if entity_result is valid then
 8:     assign entity_id to entity
 9:     get hierarchical_position of entity from Hierarchy.xsd
10:     append entity to temp_app.xml
11:   else
12:     break registration_process
13:   end if
14: end for
15: // Phase 2: manufacturing_app validation
16: result ← validate temp_app.xml against Hierarchy.xsd
17: if result is valid then
18:   return temp_app.xml
19: else
20:   break registration_process
21: end if
```

Algorithm 2. Unitary validation of a manufacturing application entity.

```
Input: entity
Output: result
 1: // entity_validation method
 2: result ← not_valid
 3: get parent_id and parent_concept of entity
 4: // check if parent_concept exists
 5: if parent_concept in Concepts.xsd then
 6:   // check if the parent of entity was previously registered
 7:   if parent_id in temp_app.xml then
 8:     // check if entity is correct
 9:     create entity.xml from entity
10:     result ← validate entity.xml against Concepts.xsd
11:   end if
12: end if
13: return result
```

The SA offers a registration API that allows the Manufacturing Execution System (MES), or other type of external actors, to register manufacturing applications through the process described in Algorithm 1. Every time an application is registered and its correctness is ensured, its data is stored in the SM. The registration API also offers another endpoint that is used by the Resource Agents (representing the equipment needed to perform manufacturing operations) to register their services in the SM. When both the manufacturing application and the Resource Agents are registered, the application is ready to be started, what leads to the creation of the set of Application Agents corresponding to each of the entities defined in the manufacturing application.

In FLEXMANSYS applications are defined by a *Hierarchy* of *Concepts*: Manufacturing Plan, Order and Batch (Fig. 4). Through proper separation of concerns, FLEXMANSYS handles the complexity of POM by means of a set of instances of agents handling those concepts. Thus, the start-

up of a manufacturing application leads to the creation of the Manufacturing Plan Agent (MPA), Order Agent (OA) and Batch Agent (BA). The agents related to a concept of the hierarchy are responsible for creating the agents belonging to the immediate lower level, i.e.: the MPA, creates a set of OAs, whereas each OA creates a set of BAs.

The OA provides a customer-oriented approach to product intelligence. Specifically, currently the OA implements Level-1 of product intelligence by offering an interface to the customer, so that they can monitor the status of their orders (McFarlane et al., 2013). On the other hand, the BA accounts for product intelligence in terms of its aggregation level (Meyer eta al., 2009), since in many manufacturing contexts an order is a collection of products that a) are manufactured in sets (commonly called lots) that must be treated as one; and b) can be obtained by operating on different sub-products (McFarlane et al., 2013). Thus, the BA is responsible for the scheduling of a production lot by defining the operation sequence that should be performed to different sub-products in parallel in order to manufacture a product. In addition, the BA is also responsible for interacting with RAs (machines, robots) to detect faults, delays, or anomalous situations, and initiate the subsequent recovery process. Thus, the BA oversees the traceability of the products, for which it reports about the relevant manufacturing events to its superior in the hierarchy, i.e., the OA.

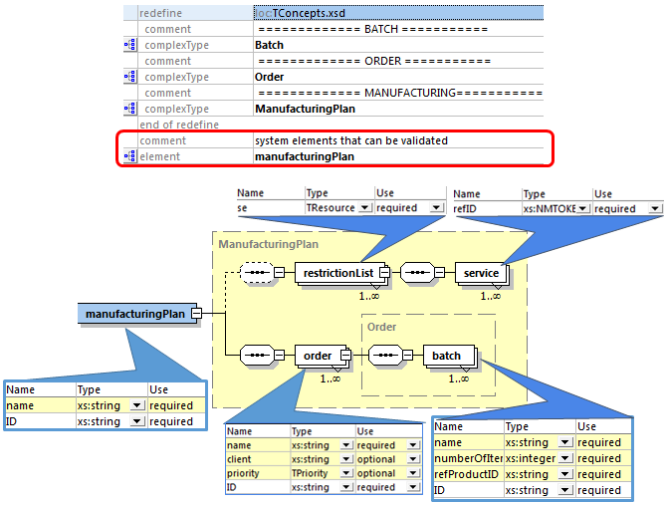

Fig. 4. *Hierarchy* meta-model of FLEMANSYS made up of Manufacturing Plan, Order and Batch *Concepts*.

4. CONCLUSIONS

On the one hand, this article presents a model-based approach for the definition of manufacturing applications by means of three XML schemas that provide flexibility in terms of modification and extension of a) the characterization of manufacturing concepts, b) the identification of those concepts, and c) the relations among them. The feasibility of this approach has been exemplified by modelling an holonic manufacturing architecture taken from the literature. In addition, taking advantage of the validation mechanisms of XML schemas, a registration process is proposed for ensuring the correctness of the definition of any factory-specific manufacturing application. This registration process could be potentially implemented in any agent platform.

On the other hand, this article introduces FLEXMANSYS, a platform for POM, that currently provides a reusable software core with an API that a) implements the model-based approach for the registration of manufacturing applications, and b) deploys those applications creating the agents identified by the manufacturing concepts. In order to achieve POM, in FLEXMANSYS product intelligence is distributed in two agents, the OA and the BA, that account for customer-oriented and lot-based product intelligence, respectively.

ACKNOWLEDGMENT

This work was financed by MCIU/AEI/FEDER, UE (grant number RTI2018-096116-B-I00) and by GV/EJ (grant number IT1324-19).

REFERENCES

Cruz Salazar, L. A., Ryashentseva, D., Lüder, A., & Vogel-Heuser, B. (2019). Cyber-physical production systems architecture based on multi-agent's design pattern—comparison of selected approaches mapping four agent patterns. *The International Journal of Advanced Manufacturing Technology*.

Kovalenko, I., Ryashentseva, D., Vogel-Heuser, B., Tilbury, D., & Barton, K. (2019). Dynamic Resource Task Negotiation to Enable Product Agent Exploration in Multi-Agent Manufacturing Systems. *IEEE Robotics and Automation Letters*, *4*(3), 2854–2861.

Kovalenko, I., Tilbury, D., & Barton, K. (2019). The model-based product agent: A control oriented architecture for intelligent products in multi-agent manufacturing systems. *Control Engineering Practice*, *86*, 105–117.

Leitão, P. (2009). Agent-based distributed manufacturing control: A state-of-the-art survey. *Engineering Applications of Artificial Intelligence*, *22*(7), 979–991. https://doi.org/10.1016/j.engappai.2008.09.005

Leitao, P., & Colombo, A. W. (2006). Petri net based Methodology for the Development of Collaborative Production Systems. *2006 IEEE Conference on Emerging Technologies and Factory Automation*, 819–826.

Li, K., Zhou, T., Liu, B., & Li, H. (2018). A multi-agent system for sharing distributed manufacturing resources. *Expert Systems with Applications*, *99*, 32–43.

McFarlane, D., Giannikas, V., Wong, A. C. Y., & Harrison, M. (2013). Product intelligence in industrial control: Theory and practice. *Annual Reviews in Control*, *37*(1), 69–88.

Meyer, G. G., Främling, K., & Holmström, J. (2009). Intelligent Products: A survey. *Computers in Industry*, *60*(3), 137–148.

Vrba, P., Tichý, P., Mařík, V., Hall, K. H., Staron, R. J., Maturana, F. P., & Kadera, P. (2011). Rockwell Automation's Holonic and Multiagent Control Systems Compendium. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *41*(1), 14–30.