A Framework for Automatic Initialization of Multi-Agent Production Systems Using Semantic Web Technologies

Felix Ocker, Ilya Kovalenko, Kira Barton, Dawn Tilbury, and Birgit Vogel-Heuser

***Abstract*—Mass customization and global competition require Cyber-Physical Systems of Systems (CPSoS) to become increasingly flexible. Modern CPSoS have to be able to create a wide and versatile variety of products, which takes centralized approaches to their limits. In addition, they have to produce these products as quickly as possible. Hence, they must be able to react promptly if problems arise, such as the failure of a single machine. Modern agent-based production systems provide the flexibility required to cope with these challenges. While resource agents (RAs) represent the available resources, i.e., machines, such as robots, individual customer orders can be represented by so-called product agents (PAs). However, a challenge in the design of agent-based production systems is still the amount of communication and computation that is necessary online. The PAs have to communicate their requests and the RAs their capabilities and capacities. On this basis, PAs must compute the appropriate production sequence. We propose to automatically initialize every agent with a knowledge base (KB) created a priori using semantic web technologies (SWT). On the one hand, the KBs of RAs describe the RAs' capabilities in terms of product features and production processes. Every KB of a PA, on the other hand, expresses all possible production sequences based on the customer specification and the CPSoS in question. This allows consistency checks regarding the specification as well as more purposeful communication that focuses on aspects that actually need to be determined at runtime, such as the resources' current capacities or failures. The framework presented aims to reduce both the communication and computational load necessary at runtime for agent-based CPSoS.**

***Index Terms*—Agent-Based Systems, Intelligent and Flexible Manufacturing, Semantic Technology.**

Manuscript received February 15, 2019; accepted July 12, 2019. Date of publication July 29, 2019; date of current version August 15, 2019. This letter was recommended for publication by Associate Editor Y. Pan and Editor D. Popa upon evaluation of the reviewers' comments. This work was supported in part by an NSF Graduate Fellowship, a gift from Rockwell Automation, in part by a Bayerische Forschungsstiftung Scholarship, and in part by the German Research Foundation via the Collaborative Research Center 768. *(Corresponding author: Felix Ocker.)*

F. Ocker and B. Vogel-Heuser are with the Institute of Automation and Information Systems, Technical University of Munich, 85748 Munich, Germany (e-mail: felix.ocker@tum.de; vogel-heuser@tum.de).

I. Kovalenko, K. Barton, and D. Tilbury are with the University of Michigan, Ann Arbor, MI 48109 USA (e-mail: ikoval@umich.edu; bartonkl@umich.edu; tilbury@umich.edu).

This letter has supplementary downloadable multimedia material available at http://ieeexplore.ieee.org provided by the authors. This includes a video file, which demonstrates the exemplary application of the framework in simulation. This material is 58.8 MB in size.

Digital Object Identifier 10.1109/LRA.2019.2931825

I. Introduction

CURRENT and future CPSoS will face a wide variety of challenges, including effective integration of new manufacturing technologies and efficient production of highly customized products [1]. To address these challenges, manufacturing systems will have to incorporate flexible and adaptable control strategies that can quickly and efficiently respond to changes on the plant floor [2]. A distributed strategy that has been used to control various types of complex systems is Multi Agent System (MAS) control. Game-theoretical [3], [4] and event-based feedback [5] MAS approaches have been used to effectively manage the performance of CPSoS. For control of manufacturing systems, the MAS control strategy has been proposed to improve the system response to unplanned disturbances in the dynamic manufacturing environment [6], [7]. In this control strategy, a distributed system of heterogeneous agents is responsible for a wide range of tasks, from controlling system resources and fulfilling customer orders to scheduling maintenance activities. While each agent has individual goals, the coordination and cooperation of these agents is crucial to maintain the productivity of the manufacturing system.

Various MAS architectures for control of manufacturing systems have been proposed [8]. Most of these architectures have interacting PAs (representing physical parts) and RAs (representing machines with various capabilities, e.g., manufacturing or handling), among a number of other agents [9]. The goal of PAs is to request actions from available RAs to accomplish a given process plan. The process plan is a representation of the manufacturing processes required to accomplish a customer request. To effectively guide the physical product through the manufacturing system, a PA's process plan should consider both the capabilities of the manufacturing system and the customer specification.

Despite the benefits of increased flexibility and adaptability, there are still challenges to using the MAS paradigm in the production context. First, extensive communication is necessary for agents to build their environment models on the fly *(C1a)*. This challenge goes along with the computational effort to make decisions during runtime *(C1b)*. Secondly, the various agents require a common language, i.e. semantics, and their environment models should be consistent *(C2)*.

The contribution of this letter is a framework based on SWT to enable the automatic processing and manufacturing of customer

orders through the use of intelligent agents. The letter presents a framework that automatically synthesizes customer requests with the capabilities of the production system to initialize intelligent PAs and RAs. The main benefit of this framework over existing approaches lies in the combination of both "offline" and "online" components to improve the performance of intelligent products in manufacturing systems. During the "offline" phase, i.e. prior to initialization of a PA, engineering knowledge is used to create an appropriate KB for the PA and pass the information regarding possible PA requests to the RAs. After the PA is initialized and a physical product is placed into the system, i.e. the PA goes "online", the PA can explore the system and react to changes in the dynamic environment in real-time.

The rest of the letter is organized as follows. Section II provides background regarding MASs in production and SWT. Section III describes the framework developed to create the agents' initial knowledge bases. Section IV presents a feasibility study and a discussion. Lastly, Section V summarizes the letter and discusses future work.

II. Background

Cyber-Physical Systems of Systems (CPSoS) are systems of systems, with each system being a Cyber-Physical System (CPS). CPSs are characterized by collaboration and an intensive connection with the physical environment [10]. These are also key aspects for MASs in the production domain. Due to the relevance of CPSs to smart factories [11], various standards have already been established [12], including SWT.

A. Agent Development for Cyber-Physical Systems of Systems (CPSoS)

CPSoSs can take advantage of an agent-based control architecture to fulfill the necessary flexibility and adaptability requirements [13], [14]. Agents represent individual elements and have to be autonomous, reactive, and adaptable to effectively control a dynamic system [15]. The agents developed to control industrial systems need to make decisions based on their inherent goals and information obtained from the plant floor. Over the past few decades, various types of MAS architectures have been presented in detail [7], [8], [14]. For this letter, some properties of individual, goal-driven agents, specifically PAs and RAs, are discussed.

The notion of an intelligent product that makes decisions and actively affects operations on the plant floor is discussed in detail in [16]. Some recent MAS architectures that use the idea of product intelligence to design PAs include [17]–[19]. Each of these PAs contains a formal, discrete event model to encode its process plan, i.e. the goals of the PA [20]. However, automatic initialization of these process plans for PAs is not provided. In [21], Vrba *et al.* propose another agent to automatically supply the PAs with their process plans. However, a formal representation for a process plan is not presented. Several other letters focus on the communication and decision making of PAs in the system, e.g., [22], [23].

Similarly, the development of intelligent RAs has been the focus of various MAS architectures. Farid *et al.* [17] provide architecture and communication requirements for RAs in the system. An architecture for implementation in the real-world is provided in [24] and a case study with an intelligent RA is provided in [25]. Note that there are many other examples of RA development in other works.

In summary, various MAS approaches have been proposed to increase flexibility and adaptability of production systems and the need for a mediator between RAs and PAs has been discussed. However, the communication load *(C1a)* and the computational effort *(C1b)* during runtime still pose major challenges [26]. To the best of our knowledge, a framework to address these challenges towards PA and RA development and to initialize this type of MAS has not yet been proposed.

B. Knowledge Bases in the Context of Agent-Based Systems

Ontologies, i.e. KBs, provide three major benefits. They enable engineers to share knowledge, reuse knowledge across domains, and make assumptions explicit [27]. Reusing formalized knowledge from previous design phases, e.g., the capabilities of available resources, saves time and enables feasibility feedback along the design process [28]. SWT provide the means to leverage the knowledge stored, using queries and inference engines. Thus, SWT provide the means to define a common language for agents and manage inconsistencies among the agents' KBs *(C2)*.

To represent the production context of agents, engineers can rely on the Product Process Resource (PPR) model [29], which revolves around resources that execute processes to create products. Products are typically defined by their features; a feature being something that provides engineering meaning or application-dependent semantics to a product [29]. This letter focuses on form features, e.g., holes and chamfers [30], and functional features, e.g., holes for assembly [30]. Various feature taxonomies are available in the literature, e.g., one for semiconductors [31].

The application of SWT to the design of intelligent production systems has long been suggested. Lastra *et al.* [32] stressed the benefits of using SWT to increase the reconfigurability of factories already in 2006. They infer process classifications and autonomous manufacturing orchestrations but do not consider PAs. The applicability of SWT to design agents in the production domain is analyzed by Ribeiro *et al.* [33], who also introduce a lightweight ontology for agent-based manufacturing systems. They match required processes and skills for finding appropriate resources for processes. The project ADACOR [34] aims to model distributed manufacturing systems including modules and processes to support scheduling and monitoring. However, features are not explicitly modeled. Borgo *et al.* [35] also present an approach to use ontologies for online decision making of RAs. Vrba *et al.* [21] focus on production planning and material handling, but their PAs and RAs are managed via a central RDF database instead of communicating directly. Similarly, Lin *et al.* [36] suggest to develop manufacturing ontologies for knowledge reuse in a distributed manufacturing environment. More specific applications can be found, e.g., in the semiconductor domain [31], [37]. To optimize resource utilization via the production-as-a-service paradigm, Balta *et al.* [38] specify a minimalist ontology to compare a customer order to manufacturer

TABLE I
OVERVIEW OF COMMON NOTIONS IN PRODUCTION KBS FOR AGENTS

	Vrba [21]	Ribeiro [33]	Borgo [34]	Lin [36]	Mönch [37]	Balta [38]
resource	workstation	equipment	resource	equipment	resource	-
process	operation	process	operation	process	activity	process
specification	specification	product design	order	-	order	service request
feature	parameter	technical requirement	-	feature	-	feature

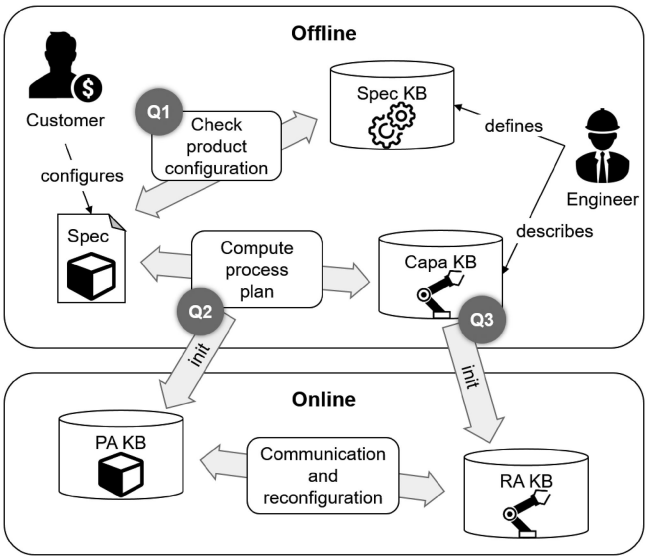

Fig. 1. Framework for initializing intelligent agents for CPSoS.

capabilities. They do not explicitly model resources because the approach is based on a description of available services. Also, the creation of a knowledge base for an integrated smart factory is not the intention of their work. Finally, SWT can also be applied to product configuration [39], allowing engineers to integrate the product configuration and the initialization of agents. Table I gives an overview of selected notions used in these approaches.

In summary, none of the approaches aims to initialize PAs and RAs with consistent KBs *(C2)* to reduce the effort for communication *(C1a)* and decision making *(C1b)* during runtime while ensuring flexibility and adaptability. The value of these other approaches lies in different goals, which we implicitly support by including the same underlying notions.

III. FRAMEWORK FOR AGENT INITIALIZATION

In the framework proposed, cp. Figure 1, engineers provide two kinds of knowledge. On the one hand, they define product features that are available to customers as well as restrictions regarding the combination of features in the *Spec KB*. Additionally, they describe the capabilities of all resources available in terms of the processes and features these resources can realize within the *Capa KB*. Based on the specification KB, customers can configure their desired products, e.g., using a spreadsheet or via a web interface. To ensure that the configuration does not violate any restrictions imposed by engineers, query 1 (indicated by *Q1* in Figure 1) checks this product configuration against the specification KB. Based on the feedback offered by query 1, customers can revise their orders to resolve inconsistencies within the specification *(C2)*. If the configuration chosen by the customer is valid, query 2 returns possible process plans, which are stored as non-cyclic directed graphs, and serve as initial KBs for PAs. Analogously, query 3 returns the KBs necessary to initialize RAs. Our framework creates the initial KBs for PAs and RAs before the agents are actually deployed as parts of production systems, cp. upper box in Figure 1 marked *offline*. This reduces the necessary *online* communication because new agents do not have to create their entire environment model on the fly *(C1a)*. Additionally, we cope with the computational load of checking the RAs' capabilities against the features required by PAs offline, where more computational power is available *(C1b)*.

A. Requirements on the Agents' Initial KBs

The design of CPSoSs imposes several requirements on the agents' KBs. Intelligent agents require certain information that must be encoded. This information includes the description of the product in terms of specified features, the various resources, and the processes these resources are capable of. Also, the processes must be linked to the features they realize, the resources they can be executed by, and other processes they are related to, e.g., via a precedence relation. Additionally, customer requirements such as deadlines must be represented. Even though there are no real-time requirements towards the initialization of agents, the effort should not exceed the magnitude of minutes. This allows to update the agents' KBs several times a day. Also, engineers and customers require an interface to specify the required information that is easy to operate. Finally, the agents' initial knowledge should be encoded as automata to support integration with previous efforts regarding the application of intelligent PAs in the production domain [22]. To facilitate the agents' initialization, the automata should be provided in a standardized format.

B. Assumptions

Within the scope of this letter, we make several assumptions. The most significant thereof is that exactly one process is necessary to create a feature, even though there may be several processes that are capable of realizing the feature in question. This restriction can be lifted by adding part relationships for both features and processes, which would also allow the description of non-atomic features and processes. Also, we only consider precedence relationships among features, not those among processes. This is because the unambiguous mapping between

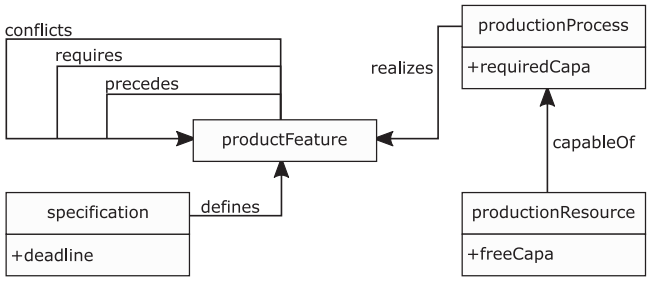

Fig. 2. Overview of the ontology used.

features and processes makes additional restrictions regarding process sequences redundant, as they can be inferred from the sequence in which the features are realized. So far, restrictions regarding precedence are only modeled in the terminological component of an ontology (TBox). This requires that the precedence relationships are generally valid. A possible extension would be to support individual restrictions on the instance level, i.e. within the assertional component of an ontology (ABox). Finally, we focus on the manufacturing part of production and assume that flexible transport units are available, which connect all manufacturing resources. However, additional restrictions resulting from non-flexible transport units can easily be added. Furthermore, we realize all logistics processes online, since we believe that they have the potential to compensate for delays in manufacturing.

C. Knowledge Base

The framework presented is based on a lean application ontology, cp. Figure 2. The ontology serves as a single source of truth *(C2)* because the initialization files for both PAs and RAs are created from it. By using common notions, cp. Section II-B, we allow engineers to combine this framework with existing approaches. The core universals are *specification*, *productFeature*, *productionResource*, and *productionProcess*. The *specification* represents the customer order that defines a combination of various *features* to unambiguously describe the desired product. Each *specification* has a *deadline* at which the product should be delivered to the customer. To ensure that the specification is valid, classes of *features* can be defined as *conflicting*. Additionally, engineers can express dependencies among classes of features using the object property *requires*. These two restrictions on combinations of *features* allow us to check the *specification* for inconsistencies. *Resources*, e.g., robots, are *capableOf* executing certain processes. Since a resource's capabilities may be restricted by its available capacity, *freeCapa* is included so that it can be checked against a process' *requiredCapa*. Every *process* serves the purpose of *realizing* a *feature*. There may exist dependencies among *features* that influence the process sequence. Such relations can be represented using the *precedes* relation between *features*. The ontology is formalized using the Web Ontology Language (OWL), which provides sufficient expressiveness while still being decidable. We intentionally kept the ontology lean and its degree of axiomatization low. Otherwise, the combination with existing and future KBs would be restricted unnecessarily because incompatibilities were more likely to arise.

D. Checking the Product Configuration for Inconsistencies

Within the ontology, product designers can specify restrictions regarding the configuration of products. The object property *conflicts* can be used to express exclusive *features* in the TBox. Analogously, the object property *requires* allows engineers to formalize that one *feature* cannot be chosen without another one. We implemented a simple SPARQL Protocol And RDF Query Language (SPARQL) query that identifies specifications which are incomplete or contain conflicting features in the ABox. The query's results can be fed back to the customers so that they can adapt their configuration accordingly. Alternatively, the encoded knowledge could also be applied to reduce the available decision points during the configuration process. We assume that all inconsistencies detected are resolved before the agents are initialized.

E. Synthesize Customer Order and Resource Capabilities

One way to encode a PA is to represent its possible process plans as an automaton that serves as a KB. This type of data structure allows various information to be encoded into the agents. The automaton is derived from the product specification and consists of states, transitions, and a global deadline that must be met.

The automaton's states represent the various states possible for a PA throughout the production process. More precisely, each state corresponds to an intermediate product defined by the feature that was manufactured last, upon completion of a manufacturing process [22]. This automaton design mirrors the product-centric view of a PA and has two major benefits, even though it contrasts with the process-centric view of an RA. First, it is easier to keep track of the features already completed. In production, several processes may be suitable for creating a product feature. If a process-centric view is used, all of them would need to be updated so that the same feature is not realized more than once. This would need to be achieved, e.g., by updating the guards for all of these processes. Secondly, the design chosen reduces the number of states and transitions by grouping them in a way that preserves the usability for our application. This way, a state space explosion can be avoided. An additional start and end state are added, so a PA's automaton has two more states than the respective specification has features.

Transitions indicate which feature can be realized next, depending on the current state. Hence, a transition exists for every tuple of states for which there is no restriction that states otherwise. To reduce the communication load and computational effort *(C1a, C1b)*, we increase the transitions' expressiveness by including information regarding the processes that can be executed to enter the next state, i.e. to realize the intended next feature. Since the same process may be executed on various machines, the transitions also include information regarding the resources that can be used to execute said process, and how much time the process takes to execute. Transitions are restricted by precedence relations among features, which apply analogously

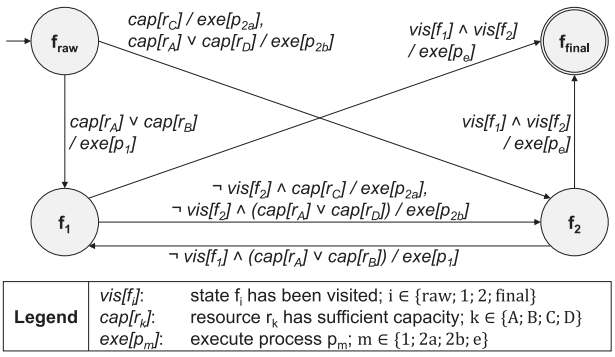

Fig. 3. Excerpt of an exemplary automaton that represents a PA's initial knowledge.

within the automaton. Two universal restrictions are that there may exist neither a state that precedes the start node, nor can the end node precede another state. The path through an automaton is further restricted by the use of counters so that the same state is not entered twice, which would correspond to the same feature being realized twice. Additionally, the PA can keep track of its current position by use of a variable.

Even though the RAs' capacities restrict the possible transitions, we choose not to consider them in the creation of the initial PAs' automata. This is because the capacities of resources are volatile, changing with the instantiation of additional PAs and RAs as well as in the case of technical failures. Instead, we provide the PAs with automata that include all theoretically possible paths so that they can figure out the best thereof during runtime.

The process plan automaton can be used to express additional information, too [18]. This includes, but is not limited to, prices for energy consumption and quality losses. Additionally, the RA schedules can be encoded within the automaton by use of clock constraints. For example, the deadline specified within the customer order is included as a global deadline for every PA. Analogously, the automaton can express manufacturing deadlines if a subsequent process has to be executed within a certain timespan.

An exemplary automaton is depicted in Figure 3. The specification defines two features, f_1 and f_2, which are complemented by the start state f_{raw} and the end state f_{final}. Feature f_1 can be realized by resource r_A or r_B via process p_1. Let only feature f_1 be completed for a product, i.e. the PA is currently in state f_1 but has not yet been in f_2. In this case, two different processes can be executed to realize f_2: p_{2a} or p_{2b}. While the process p_{2a} can only be run by resource r_C, both resources r_A and r_D are capable of completing process p_{2b}. The automaton can be extended to indicate the machine- and process-specific costs of realizing a feature as well as clock constraints.

For creating the RAs' KBs, we rely on the *Capa KB*. Relevant information for RAs are their available capacities, the features they can realize, and the processes they can execute to realize these features. Additionally, we include the costs of executing these processes, i.e. the *requiredCapa*. Such process execution duration times are typically stable and do not have to be continuously updated online.

F. Agent Initialization

Many different types of PAs have been proposed for MAS control [9], [40]. Each of the proposed PAs must be able to guide an associated physical part through the system based on its goals and communication with other agents. Therefore, the PA relies on the process plan provided to make a large fraction of its decisions. For example, once initialized by the system, a PA must understand the surrounding manufacturing environment [40], [41]. The PA can use the process plan to build an update version of the manufacturing environment in the vicinity of the associated physical part by querying the availability of the next desired features from the RAs. Using the example automaton in Figure 3, the PA initially requests nearby resources to accomplish f_1 or f_2. Using these requests, the RAs start to bid on how these features can be accomplished. These bids contain the resources that take the product to the RA, the schedules of these resources, and the expected time that each process should take. Note that the PA sends requests that contain multiple features that the RAs should respond to. While the exploration component of the PA is out of the scope of this letter, the process plan plays an important role in ensuring the effectiveness of the employed exploration technique.

After creating a model from these bids or using a model provided, the PA must choose its next actions using the process plan and the capabilities model [19], [40]. By incorporating the process plan it was initialized with offline, the PA is able check the accuracy of the model provided by RAs. By encoding possible resources for each process via the automaton events, the PA can perform a consistency check on RA responses. If a feature promised by the RA does not match up with the process plan, the PA can raise a flag about the RA, adding a layer of security to the agent-based controller. Additionally, before making any other requests, the PA can evaluate whether or not it is able to accomplish its goals based on the deadlines in its process plan. Note that the PA might have to change its decisions based on the status of its deadlines, e.g., if it does not meet a deadline, the PA might have to start negotiating with other resources (RAs or PAs) to find a different path through the system. Finally, the PA can start to schedule future events from the RAs and start to request operations from the RAs. An example of this type of intelligent PA can be found in [22], [40].

Agents decide on their next steps online by relying on their KB in combination with an objective function. For now, we assume a simple objective function to minimize the sum of production and transport times for the next production step s; with S being the set of possible next steps, cp. Equation (1). This objective function can be extended to include, e.g., quality losses and prices for energy consumption. These factors are weighted according to customer preferences or company specific guidelines. While the duration times for manufacturing are encoded in the agents' KBs, the transport duration times are determined online. The optimization during runtime is subject to restrictions originating in the agent's environment such as the position and workload of transport resources [40].

$$\min_{s \in S} \quad (t_{manufacturing_s} + t_{transport_s})$$
$$\text{s.t.} \quad \text{restrictions}_{\text{environment}} \tag{1}$$

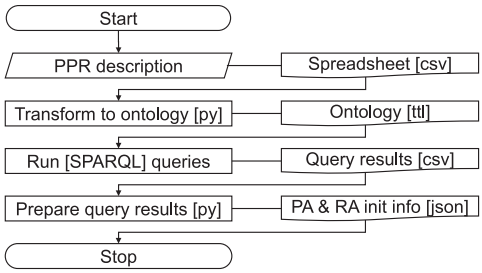

Fig. 4. Steps for creating the agents' initial KBs.

Note that during all of this communication, it is imperative that the RAs understands the PA requests. The initialization of the RAs' KBs allows for the RAs to match their capabilities with the various PA requests. By initializing the RA KB, the agent-based system maintains consistency over agents *(C2)*. This supports effective communication and negotiation between the agents in the system *(C1a, C1b)*.

IV. Implementation and Discussion

A. Workflow

As stated in Section III, the description of products, processes, and resources is key to the agent based production system. Engineers describe the resources available, the processes these resources can execute, and the features that can be realized with said processes. Customers on the other hand configure the product desired based on the features specified by the engineers. Within the scope of this letter, we use a semi-formalized Excel spreadsheet. To be less dependent on proprietary tools, we use a macro to export the file's sheets as csv files. In the next step, a python script transforms the csv files into an ontology compliant with OWL. To query the ontology, a reasoner is required that supports SPARQL queries. We use the knowledge graph platform stardog [42] to execute the queries described in Section IV-B. This way, we create automata for the PAs, which describe the states and transitions allowed as well as dictionaries that list the RAs' capabilities in terms of features and processes. These query results, too, are stored as csv files. This format allows us to easily parse the files by use of another python script and create a json file containing the initial information for every PA and RA. These json files are designed in a way that they can be used directly to initialize the PAs and RAs and thus get the production systems running. An overview of the workflow is depicted in Figure 4.

B. Queries

Within this framework, we use SPARQL to query the ontology developed. The framework's first query Q1 identifies whether a customer selected conflicting features and whether any dependencies among features are violated, cp. Algorithm 1. Customers can resolve these issues based on feedback regarding the kind of violation and the features concerned. Only when this query does not return issues any more is the specification accepted as valid.

Algorithm 1: Inconsistency Management for Specifications.

```
1:  procedure CONSISTENCY CHECKING (specs)
2:    for all specifications spec do
3:      for all tuples (f1, f2) of features in spec do
4:        if class(f1) conflicts class(f2) then
5:          return f1 conflicts f2
6:      for all features f1 in spec do
7:        if class(f1) requires c2 and ∄ instance(c2) then
8:          return f1 requires instance(c2)
```

Algorithm 2: Product Agent Initialization.

```
1:  procedure PA INIT (specs)
2:    for all specifications spec do
3:      globaldeadline = deadline
4:      states.update(start)
5:      for all features f in spec
6:        states.update(f)
7:      states.update(end)
8:    for all transitions t for tuples (f1, f2) in states do
9:      if ∄ precedence restriction for (f1, f2) then
10:       transitions.update(t)
11:       for all p in processes do
12:         if p can realize f2 then
13:           transitions[t].processes.update(p)
14:         for all r in resources do
15:           if r capable of p then
16:             transitions[t].processes[p].resources.
                update(r)
```

Algorithm 3: Resource Agent Initialization.

```
1:  procedure RA INIT (resources)
2:    for all resources r do
3:      get free capacity
4:      for all processes p do
5:        if r capable of p then
6:          r.processes.update(p)
7:          for all features f do
8:            if p can realize f then
9:              r.processes[p].features.update(f)
```

Queries Q2a and Q2b serve the purpose of extracting knowledge necessary for the PA, cp. Algorithm 2. First, Q2a identifies all states a PA can be in for every product specification. Q2a also returns the global deadline for each specification. As described in Section III-E, every PA's automaton includes one state per feature complemented by a start and an end node. Next, Q2b finds all $n * (n-1)$ possible transitions among the states, but removes those that would violate a precedence restriction. Associated to each transition, the query returns the combinations of processes and resources that can realize a feature and thus enable the PA to change into a subsequent state. Q2b deliberately does

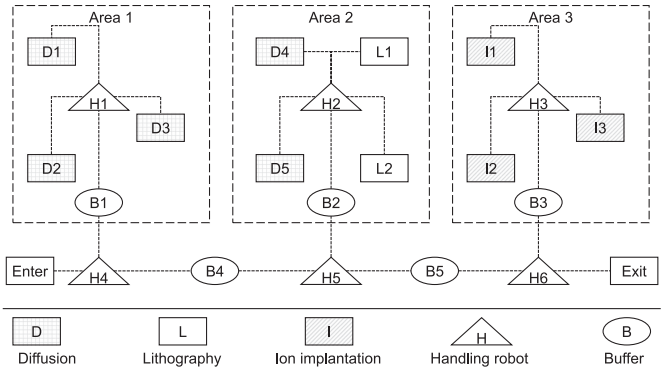

Fig. 5. Exemplary layout of a semiconductor fab.

not check the resources' capacities because these are subject to continuous changes as the production system operates.

To create the initial KBs for the RAs, we use queries Q3a and Q3b, cp. Algorithm 3. Q3a identifies the resources available in the system and their current capacities. This also allows engineers to easily analyze the resources' degree of capacity utilization. Q3b on the other hand returns the processes each resource is capable of as well as the features that can thus be realized. Additionally, Q3b gives back the capacity required for realizing the associated combination of feature and process.

All source files created in the scope of this research as well as the simulation used for the proof of concept can be found online [43], [44].

C. Feasibility Study

The framework proposed has been used to initialize PAs and RAs for the simulated semiconductor fabrication facility shown in Figure 5. The feasibility study can easily be scaled up to multiples of this layout. The manufacturing facility is a scaled-up version of the Intel Mini-Fab previously used to test an MAS planning architecture [45]. There are three different types of stations (diffusion, lithography, and ion implementation) that are connected using material handling robots and buffers. There are three possible configurations, specified in [45], for the process plan of a single part in the system. The entrance and exit points for this facility are shown in Figure 5.

For this example, the machines, robots, and buffers are each controlled by an individual RA. Within the *Capa KB*, engineers describe the resources' capabilities in terms of processes and features. E.g., the diffusion resource $D3$ can execute the two slightly different diffusion processes $PD3$ and $PD4$. While process $PD3$ can be used to realize the feature $Fd1$, process $PD4$ serves to complete the feature $Fd2$. Both $Fd1$ and $Fd2$ are of type $featDiff$. A similar methodology was used to map the lithogrophy and ion implementation processes and features. Additionally, engineers can restrict the combination of features allowed in customer specifications *Spec KB*, based on the universal process plans described in [45].

Once the engineers specified the *Capa KB* and the *Spec KB*, the customer followed the workflow described in Section IV-A to request new products from the system. If the requested specifications were found to be valid, json files were produced to initialize the PAs and RAs in the manufacturing MAS. An initializing agent would wait for new json files to appear. Once detected, the agent parses the files to initialize new PAs. In addition, the json files are provided to the RAs to ensure efficient communication between all agents in the system. Using this framework, the PAs and RAs are able to effectively communicate information that is required to store, handle, and manufacture the products requested.

D. Comparison With Existing Approaches

The approach presented in this letter differs from existing work as it specifically addresses the communication load *(C1a)* and computational effort *(C1b)* necessary during runtime of MAS based production systems. Using model-based PAs increases both flexibility and adaptability of the production system. However, existing approaches that use PAs and focus on flexibility and adaptability disregard the communication load *(C1a)* and the computational effort *(C1b)*. Also, these established approaches typically do not consider inconsistency management regarding the specification and KBs of the agents. Approaches based on SWT on the other hand mainly aim to create predefined process plans instead of a set of process plans, making the resulting production system more rigid. Also, most of these approaches do not consider PAs, which help greatly to increase flexibility and adaptability. Still, we ensure basic compatibility with these existing approaches as well as previous work for inconsistency management by using the same underlying notions. Compared to our own previous work [40], there are two major benefits of the approach presented in this letter. First, we reduced both the communication effort as well as the computational load for the decision making of the agents *(C1a, C1b)*. Secondly, we reduced the engineering effort by initializing the agents automatically in a consistent way *(C2)*.

V. Summary and Outlook

The framework presented allows engineers to automatically initialize intelligent agents for production systems by use of SWT. This helps to reduce the online communication load *(C1a)* and the computational effort *(C1b)* for both PAs and RAs because they already know their environments. On the one hand, the KBs of RAs describe their capabilities. On the other hand, every KB of a PA describes the set of possible production sequences of the respective product in the form of an automaton. To create these process plans, customer specifications are first checked for inconsistencies and are then synthesized with the available resources' capabilities *(C2)*. The underlying ontology's generic design allows it to be be combined with existing domain-specific KBs and, e.g., libraries for generic product feature mappings [46]. The framework's theoretical feasibility has been shown at the example of a semiconductor fabrication facility.

Future work will include more advanced feasibility checks that support an *engineer-to-order approach* instead of the current *configure-to-order* one. This should be complemented by approaches to cope with uncertainties in the production context. Additionally, we intend to close the loop between the "offline"

ontology and the agents' "online" KBs by continuously reporting changes of the agents' states. The flexibility gained through the framework may also be leveraged to increase the customers' benefits. The status of a PA can be used not only to let the customers track their orders' completion progress but also to allow late changes if customers change their minds regarding the product specifications. As long as certain features are not realized, the customers may change their specifications, which would result in an update of the respective PAs.

REFERENCES

[1] B. Vogel-Heuser, A. Fay, I. Schaefer, and M. Tichy, "Evolution of software in automated production systems: Challenges and research directions," *J. Syst. Softw.*, vol. 110, pp. 54–84, 2015.

[2] K. Barton, F. Maturana, and D. Tilbury, "Closing the loop in IoT-enabled manufacturing systems: Challenges and opportunities," in *Proc. Amer. Control Conf.*, 2018, pp. 5503–5509.

[3] V. Suraci, L. R. Celsi, A. Giuseppi, and A. Di Giorgio, "A distributed wardrop control algorithm for load balancing in smart grids," in *Proc. Med. Conf. Control Autom.*, 2017, pp. 761–767.

[4] F. D. Priscoli *et al.*, "Multi-agent quality of experience control," *Int. J. Control, Autom. Syst.*, vol. 15, no. 2, pp. 892–904, 2017.

[5] M. P. Fanti, B. Maione, S. Mascolo, and A. Turchiano, "Event-based feedback control for deadlock avoidance in flexible production systems," *IEEE Trans. Robot Autom.*, vol. 13, no. 3, pp. 347–363, Jun. 1997.

[6] P. Leitão, "Agent-based distributed manufacturing control: A state-of-the-art survey," *Eng. Appl. Artif. Intell.*, vol. 22, no. 7, pp. 979–991, Oct. 2009.

[7] B. Vogel-Heuser, J. Lee, and P. Leitão, "Agents enabling cyber-physical production systems," vol. 63, no. 10, pp. 777–789, 2015.

[8] P. Leitão, V. Maík, and P. Vrba, "Past, present, and future of industrial agent applications," *IEEE Trans. Ind. Inform.*, vol. 9, no. 4, pp. 2360–2372, Nov. 2013.

[9] P. Vrba *et al.*, "Rockwell automation's holonic and multiagent control systems compendium," *IEEE Trans. Syst, Man, Cybern.*, vol. 41, no. 1, pp. 14–30, Jan. 2011.

[10] L. Monostori *et al.*, "Cyber-physical systems in manufacturing," *CIRP Ann.*, vol. 65, no. 2, pp. 621–641, 2016.

[11] J. Lee, H.-A. Kao, and S. Yang, "Service innovation and smart analytics for industry 4.0 and big data environment," *Procedia CIRP*, vol. 16, pp. 3–8, 2014.

[12] A. J. Trappey, C. V. Trappey, U. H. Govindarajan, J. J. Sun, and A. C. Chuang, "A review of technology standards and patent portfolios for enabling cyber-physical systems in advanced manufacturing," *IEEE Access*, vol. 4, pp. 7356–7382, 2016.

[13] L. Ribeiro, J. Barata, and P. Mendes, "MAS and SOA: Complementary automation paradigms," in *Proc. Int. Conf. Inf. Technol. Balanced Autom. Syst.*, Boston, MA, USA, 2008, pp. 259–268.

[14] P. Leitão, S. Karnouskos, L. Ribeiro, J. Lee, T. Strasser, and A. W. Colombo, "Smart agents in industrial Cyber Physical systems," *Proc. IEEE*, vol. 104, no. 5, pp. 1086–1101, May 2016.

[15] M. J. Wooldridge, *An Introduction to MultiAgent Systems*, 2nd ed. West Sussex, U.K.: Wiley, 2009.

[16] D. McFarlane, V. Giannikas, A. C. Y. Wong, and M. Harrison, "Product intelligence in industrial control: Theory and practice," *Annu. Rev. Control*, vol. 37, no. 1, pp. 69–88, 2013.

[17] A. M. Farid and L. Ribeiro, "An axiomatic design of a multiagent reconfigurable mechatronic system architecture," *IEEE Trans. Ind. Inform.*, vol. 11, no. 5, pp. 1142–1155, Oct. 2015.

[18] C. Schoppmeyer, S. Subbiah, J. Manuel, D. L. Fuente, and S. Engell, "Dynamic scheduling of shuttle robots in the warehouse of a polymer plant based on dynamically configured timed automata models," *Ind. Eng. Chem. Res.*, vol. 53, no. 44, pp. 17135–17154, 2014.

[19] S. Rehberger, L. Spreiter, and B. Vogel-Heuser, "An agent-based approach for dependable planning of production sequences in automated production systems," *Automatisierungstechnik*, vol. 65, no. 11, pp. 766–778, 2017.

[20] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, 2nd ed. New York, NY, USA: Springer, 2009.

[21] P. Vrba, M. Radaković, M. Obitko, and V. Maik, "Semantic technologies: Latest advances in agent-based manufacturing control systems," *Int. J. Prod. Res.*, vol. 49, no. 5, pp. 1483–1496, 2011.

[22] I. Kovalenko, K. Barton, and D. Tilbury, "Design and implementation of an intelligent product agent architecture in manufacturing systems," in *Proc. Int. Conf. Emerg. Technol. Factory Autom.*, 2017, pp. 1–8.

[23] J. De Las Morenas, A. Garcia-Higuera, and P. Garcia-Ansola, "Shop floor control: A physical agents approach for PLC-controlled systems," *IEEE Trans. Ind. Inform.*, vol. 13, no. 5, pp. 2417–2427, Oct. 2017.

[24] W. Lepuschitz, A. Zoitl, M. Vallee, and M. Merdan, "Toward self-reconfiguration of manufacturing systems using automation agents," *IEEE Trans. Syst, Man, Cybern.*, vol. 41, no. 1, pp. 52–69, Jan. 2011.

[25] P. Marks, M. Weyrich, X. L. Hoang, and A. Fay, "Agent-based adaptation of automated manufacturing machines," in *Proc. IEEE Int. Conf. Emerg. Technol. Factory Autom.*, 2017, pp. 1–8.

[26] B. Vogel-Heuser and L. Ribeiro, "Bringing automated intelligence to cyber-physical production systems in factory automation," in *Proc. IEEE 14th Int. Conf. Autom. Sci. Eng.*, 2018, pp. 347–352.

[27] M. Obitko and V. Marik, "Ontologies for multi-agent systems in manufacturing domain," in *Proc. 13th Int. Workshop Database Expert Syst. Appl.*, 2002, pp. 597–602.

[28] F. Ocker, B. Vogel-Heuser, and C. J. J. Paredis, "Applying semantic web technologies to provide feasibility feedback in early design phases," *J. Comput. Inf. Sci. Eng.*, vol. 19, no. 4, 2019, Art. no. 041016.

[29] J. Nielsen, "Information modeling of manufacturing processes: Information requirements for process planning in a concurrent engineering environment," Ph.D. dissertation, Dept. Prod. Eng., KTH Roy. Inst. Technol., Stockholm, Sweden, 2003.

[30] E. M. Sanfilippo and S. Borgo, "What are features? An ontology-based review of the literature," *Comput.-Aided Des.*, vol. 80, pp. 9–18, Nov. 2016.

[31] J. J. Jung, "Ontology-based decision support system for semiconductors EDS testing by wafer defect classification," *Expert Syst. Appl.*, vol. 38, no. 6, pp. 7425–7429, Jun. 2011.

[32] J. L. Lastra and M. Delamer, "Semantic web services in factory automation: Fundamental insights and research roadmap," *IEEE Trans. Ind. Inform.*, vol. 2, no. 1, pp. 1–11, Feb. 2006.

[33] L. Ribeiro, J. Barata, M. Onori, and A. Amado, "OWL ontology to support evolvable assembly systems," *IFAC Proc. Vol.*, vol. 41, no. 3, pp. 290–295, Jan. 2008.

[34] S. Borgo and P. Leitão, "The role of foundational ontologies in manufacturing domain applications," in *On the Move to Meaningful Internet Systems*, R. Meersman and Z. Tari, Eds., Berlin, Germany: Springer, 2004, pp. 670–688.

[35] S. Borgo, A. Cesta, A. Orlandini, and A. Umbrico, "Knowledge-based adaptive agents for manufacturing domains," *Eng. Comp.*, vol. 35, no. 3, pp. 755–779, Jul. 2008.

[36] L. F. Lin, W. Y. Zhang, Y. C. Lou, C. Y. Chu, and M. Cai, "Developing manufacturing ontologies for knowledge reuse in distributed manufacturing environment," *Int. J. Prod. Res.*, vol. 49, no. 2, pp. 343–359, 2011.

[37] L. Mönch and M. Stehli, "An ontology for production control of semiconductor manufacturing processes," in *Multiagent System Technologies*. Berlin, Germany: Springer, 2003, pp. 156–167.

[38] E. C. Balta, Y. Lin, K. Barton, D. M. Tilbury, and Z. M. Mao, "Production as a Service: A digital manufacturing framework for optimizing utilization," *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 4, pp. 1483–1493, Oct. 2018.

[39] D. Yang, R. Miao, H. Wu, and Y. Zhou, "Product configuration knowledge modeling using ontology web language," *Expert Syst. Appl.*, vol. 36, no. 3, pp. 4399–4411, Apr. 2009.

[40] I. Kovalenko, D. Tilbury, and K. Barton, "The model-based product agent: A control oriented architecture for intelligent products in multi-agent manufacturing systems," *Control Eng. Pract.*, vol. 86, pp. 105–117, 2019.

[41] M. K. Lim, Z. Zhang, and W. T. Goh, "An iterative agent bidding mechanism for responsive manufacturing," *Eng. Appl. Artif. Intell.*, vol. 22, no. 7, pp. 1068–1079, 2009.

[42] Stardog Union, "Stardog: The enterprise knowledge graph platform," [Online]. Available: https://www.stardog.com/

[43] F. Ocker, "Multi agent ontology," [Online]. Available: https://github.com/felixocker/multi-agent-ontology

[44] I. Kovalenko, "Semiconductor simulation," [Online]. Available: https://github.com/ikovalenko92/SemiconductorSimulation

[45] H. J. Yoon and W. Shen, "A multiagent-based decision-making system for semiconductor wafer fabrication with hard temporal constraints," *IEEE Trans. Semicond. Manuf.*, vol. 21, no. 1, pp. 83–91, Feb. 2008.

[46] F. Ameri and S. Allen, "An ontological approach to integrated product and process knowledge modeling for intelligent design repositories," in *Smart Product Engineering*. Berlin, Germany: Springer, 2013, pp. 825–834.