






# Collaboration of Semantically Enriched Digital Twins based on a Marketplace Approach

Joel Lehmann<sup>1</sup> <sup>a</sup>, Andreas Lober<sup>2</sup> <sup>b</sup>, Alessa Rache<sup>1</sup> <sup>c</sup>,  
Hartwig Baumgärtel<sup>2</sup> <sup>d</sup> and Julian Reichwald<sup>1</sup> <sup>e</sup>

<sup>1</sup>Mannheim University of Applied Sciences, Paul-Wittsack-Str. 10, 68163 Mannheim, Germany

<sup>2</sup>Ulm University of Applied Sciences, Prittwitzstraße 10, 89075 Ulm, Germany

**Keywords:** Digital Twins, Cyber-physical System, Marketplace, Ontology, SPARQL, Manufacturing.

**Abstract:** Digital Twins are a key concept and an enabling technology within the digital transformation. They can communicate bilaterally with their physical counterpart. Thus the twins are reflecting each other's properties and state changes. Digital Twins add value to many use case domains, e. g., smart manufacturing. Nevertheless, today's implementations are mostly data and state representations of their physical counterparts, tying the digital twin to a passive and reactive role. The authors argue that semantically enhanced digital twins that autonomously perform actions and interactions on digital market spaces can add substantial value to various use cases. An architectural approach is proposed, and an implemented proof of concept shows the feasibility.

## 1 INTRODUCTION

Digital Twins (DTs) are a key concept within digital transformation. The concept asserts that all systems are dual in nature and have a digital representation besides their physical one (Grieves, 2019). Thus, a DT bridges the gap between the physical and digital space: data and state changes of the physical asset are reflected by the DT and vice versa. By design, DTs fit into highly distributed, heterogeneous environments. Since the digital representation may – in contrast to its physical counterpart – easily be extended by adding functionalities like, e.g., simulation systems or machine learning algorithms, such extended DTs can add substantial value to use cases from scenarios like, e.g., smart manufacturing, smart health, or smart cities. Nevertheless, current DT implementations are mainly based on simple data and state representations of the physical asset; hence they are tied to a passive role inside the system.


The authors argue that a more active role of a DT in communicating with other DTs, negotiating with them, and autonomously executing their decisions (which are by definition also executed in the physical space) can bring crucial benefits to the manu-


facturing domain. DTs of products or services would be able to interact with DTs of consumers on digital marketplaces and automatically negotiate on pricing, delivery times, or other product or service properties. They could also assemble themselves into complex products or modules if this is beneficial after negotiating. A system like this can leverage substantial value to specific use cases and scenarios. The authors propose a system where DTs represent data and state information of a physical asset. Moreover, they will – in a first step – be enhanced by semantic functionalities, which can be seen as an essential condition for any cross-domain negotiation and communication. The DTs will be implemented using a decentralized runtime environment, an essential element for executing autonomous actions in a second step.


After discussing the motivation and the related work in this field in section 2, the authors present their proposed architecture of the aforementioned system in section 3. After the description of the implementation, a demonstration use case is shown as a proof of concept in section 4. The work is concluded with a discussion and prospects our future work in section 5.


## 2 MOTIVATION & RELATED WORK


Production machines cannot easily be integrated into complex negotiation scenarios. Due to their limited

<sup>a</sup>  <https://orcid.org/0000-0001-8261-8362>

<sup>b</sup>  <https://orcid.org/0000-0003-3513-6103>

<sup>c</sup>  <https://orcid.org/0000-0001-7598-8672>

<sup>d</sup>  <https://orcid.org/0000-0002-6251-8898>

<sup>e</sup>  <https://orcid.org/0000-0002-4809-5710>

extensibility, these physical assets are unable to interact proactively. In contrast, the DT of a physical asset can be easily extended. However, DTs are often used as pure data and state representations with at most reactive behavior. With this contribution, the authors go beyond this by giving the DTs their own framework, paving the way for collaborative interaction between DTs within a marketplace scenario. Semantic technologies are a cornerstone of increased autonomy. Therefore, the authors will discuss both focal points of DTs and semantic representation of production skills to synthesize our architectural concept later.

## 2.1 Digital Twins

Digital Twins are a key concept and an enabling technology within the digital transformation. Michael Grieves first introduced the concept of DTs when he created and evolved an entirely new paradigm for Product Lifecycle Management and engineering methodologies commencing in 2003 (Grieves, 2019). Grieves' approach is based on the duality between physical and digital spaces, as depicted by figure 1. While on the one side, an instance of a real-life entity exists in the physical space, on the other side, its virtual representation coexists in the digital space. Both spaces are linked by a digital thread, consisting of a bidirectional stream of data and information. While the solid arrow shows the data obtained from the physical space, the dashed arrow represents the feedback information provided by the DT. Each DT is unambiguously related to his physical twin in a 1:1 relation. DTs are highly use-case driven. Depending on its specific application, the twin can be a complex representation of a physical entity with high-fidelity modeling or an abstracted minimalistic data construct (Grieves, 2021).

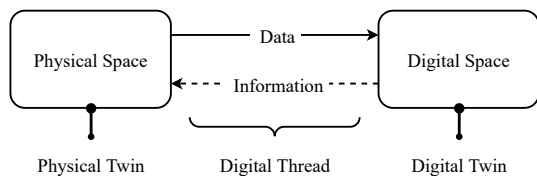


Figure 1: Concept of a Digital Twin According to Grieves.

(van der Valk et al., 2021) subdivide DTs into specific archetypes, which were subsequently validated through interviews with industry experts. Five archetypes are distinguished starting from a basic DT, only providing mandatory characteristics, extending to a fully autonomously interacting exhaustive Twin. They classify the twins into the following:

1. Basic Digital Twin
2. Enriched Digital Twin
3. Autonomous Control Twin
4. Enhanced Autonomous Control Twin
5. Exhaustive Twin

The expert interviews revealed that the autonomously interacting archetypes 3 - 5 have not yet been implemented apart from individually designed solutions. Regarding the more advanced archetypes 4 and 5, the research only provides conceptual approaches and pilot studies. Their conclusions are consistent with other studies addressing definitions and characteristics, stating that DTs are mostly still at an early stage of development. Only a few domains are already using specific implementations with maximum value exploitation. The concept and its potential are yet transferable to many other areas (Negri et al., 2017; Kritzinger et al., 2018; Barricelli et al., 2019; Fuller et al., 2020).

Extended DTs going beyond simple data representation are just emerging, with much elaboration still to be done (Saracco, 2019). Continuous cross-domain linking of DTs in the production process is necessary for a steady increase in competitiveness (Wagner et al., 2019). DTs are complex systems that integrate several technical disciplines. Choosing a universal development platform remains challenging due to various formats, protocols, and standards. Therefore, industrial practices require reliable infrastructures (Qi et al., 2021).

The German Industrie 4.0 initiative, as an affiliation of industrial companies and associations, drives the concept of DTs through the Asset Administration Shell (AAS). Standard IEC TS 62443-1-1:2009 defines an asset as a *physical or logical object owned by or under the custodial duties of an organization. This asset has either a perceived or actual value to the organization.* The AAS provides multi-vendor interoperability and consistent value chains throughout their lifecycle to fully exploit an asset's potential (P. I. 4.0, 2020).

Most recent publications focus on the implementation of passive AAS without autonomous capabilities. Consequently, reactive as well as proactive AAS implementations with intelligent decision-making and interaction with other AAS are still in their infancy (Sakurada et al., 2021). According to (Ocker et al., 2021), a suitable framework for provisioning proactive DTs by using AAS does not yet exist. Creating DTs via AAS is a challenge that requires a significant amount of manual effort because of its complex heterogeneous raw information.

(Vogel-Heuser et al., 2021) use DTs provided by the AAS to create a Multi-Agent System (MAS) and its required knowledge base. The respective agents of products and resources are intended to improve production processes in a decentralized approach. In their alike test case, (Rocha et al., 2019) demonstrated a MAS-based architecture representing a manufacturing system. The MAS is used to control the hardware of a production system and work orders through different agents. The approach shows a distributed agent-based control system demonstrating a modular design of a production system. MAS and DTs are conceptually and functionally very close. With the growing emergence of Industrial Cyber-Physical Systems (ICPS), new opportunities for MAS will also arise (Karnouskos et al., 2020). However, the MAS approaches are limited to the domain of one company, and it is not clear how this MAS would play out in a cross-company open marketplace scenario. Furthermore, MAS operate in a decentralized manner close to the edge based on traditional agent frameworks with several issues, whereas DTs primarily run centrally, for instance, within a cloud environment.

This work is based on Grieves' conceptual framework. His idea of a DT is entirely domain-independent and can easily be transferred to many other use cases. Hence, it is also feasible to add intelligent features to the DT. The subsequent elaboration should thus be as open as possible. However, a later evaluation of the implementation based on the AAS and MAS will not be ruled out but investigated in further publications.

## 2.2 Semantical Representation of Production Skills

Digital Twins are designed to realize the interaction of cyber and physical entities, causing the overall complexity of these systems to increase along with accelerating requirements regarding design, development, and testing. Several concepts built on a Service-oriented Architecture (SOA) have shown to be promising for representing knowledge and capabilities of production systems (Leitao, 2016). In contrast to the traditional component-based hierarchical approach, the SOA-concept is more complex but offers greater flexibility (Dorofeev and Wenger, 2019). These approaches for complex design processes can be classified as Model-Driven Engineering (MDE). Focusing on reducing the overall emerging complexity, MDE has emerged as a promising approach within process automation to serve a semantic representation provided by the underlying capabilities of the components, resources, and production systems.

For the support of MDE of software projects for manufacturing automation, a SysML-based approach is presented in (Vogel-Heuser et al., 2014), in which functional as well as non-functional requirements can be covered using a language profile. An ontology-based approach is presented in (Ameri and Dutta, 2006) and introduces the Manufacturing Service Description Language (MSDL). Thereby, core concepts and relations for the description of manufacturing processes are combined in an upper ontology. An opposing design has been presented in (Cândido and Barata, 2007) and provides a generic approach towards an ontology for describing entities in a manufacturing system based on a multi-agent system.

The IDEAS project introduces a configuration method for describing executable skills for evolvable assembly systems using a process-oriented semantic model. In this approach, object-oriented skills are defined using an assembly process type as well as a granularity level (Ferreira and Lohse, 2012). The capability and skill model of machines presented in (Köcher et al., 2020) is based on distinct ontology design patterns which are based on common industry standards for the direct description of executable skills and further provide an approach to enable the plug-and-produce principle. A semantic model in AutomationML is proposed in (Anandan et al., 2017) for representing skills in a computer-interpretable format to support high-performance plug-and-produce systems. Building on the automation of different machines by plug-and-produce, an approach by Pfrommer et al. is presented in (Pfrommer et al., 2015), in which the skill is defined as the capability of a resource using AutomationML and OPC UA to model and execute the skills. Another approach presented in (Backhaus and Reinhart, 2015) defines a skill as a vendor-neutral description of the functionality of a machine, where skills perform specific processes required to manufacture certain products.

(Järvenpää et al., 2016) provide a capability model approach by describing the skills by parameters and names. Taxonomies such as DIN 8580 are used by a process taxonomy to align capabilities and product requirements. Within the BaSys 4.0 project, skills are defined as a set of properties and made available via semantic annotations, thus providing another approach to modeling skills on a semantic basis (Perzylo et al., 2019).

A graph-based skill metamodel for CPS is presented in (Brovkina and Riedel, 2019), in which skills are specified as a composition of atomic and essential operations put independently to other operations. In (Bauernhansl et al., 2020), the semantic representation of different elements and capabilities of pro-

duction systems and the resulting representation and grouping of information for a highly flexible re-configuration of the system has been structured in an information model.

The overall objective of the presented approaches is to build a semantic representation based on the given information of an existing production system. However, the consistency of the engineering data is not or only partially ensured in these approaches. Furthermore, frameworks for marketplace scenarios are not considered in any cases.

### 3 ARCHITECTURE

Based on the authors' motivation to raise DTs to a higher level of collaboration and interaction and the fact, which emerged from the discussion of the related work, that there are no adequate frameworks for proactive DTs as well as collaborative marketplace approaches, the authors are subsequently introducing the general structure of their framework. The subsections distinguish between different types of DTs and elaborate the collaborative approach between them. Based on Grieves' concept, as shown in figure 1, the DTs will be further refined. The introduced digital spaces are depicted as two entities to distinguish the unique types, which will be renamed later.

#### 3.1 Asset Digital Twin

To efficiently interact with an asset settled in the operational technology (OT), the asset has to be provisioned as DT, enabling full access to be processed by information technology (IT). OT in this context refers to the isolated combination of hardware and software within an automated production facility. Due to still much-used encapsulated fieldbus systems or proprietary automation protocols, an OT/IT gateway is required in most cases. An asset interface can be established using standard IT protocols, such as Message Queuing Telemetry Transport (MQTT), to overcome this issue.

In order to give DTs room to negotiate, they need their habitat within IT. This environment is realized through a centralized approach in the digital space. It could also be transposed through a decentralized concept as a multi-agent system (MAS), addressed in further planned publications.

Figure 2 depicts the coherence with various companies participating in the digital space. Each company offers services through the DTs of its assets. Hence, the DTs are accumulated across company boundaries. Those lead away from traditional isolated

market spaces and market strategies towards centralized skill and capability trading open marketplaces.

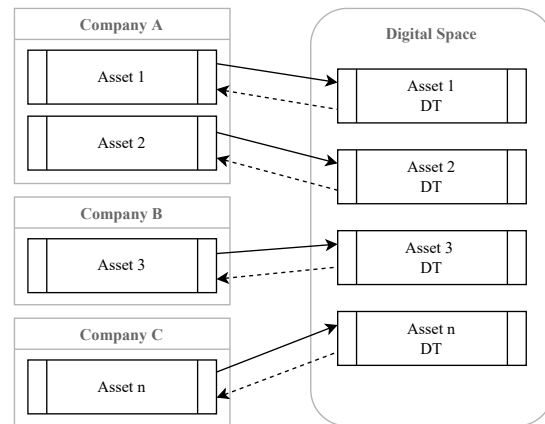


Figure 2: Interconnection of Production Assets and Their Digital Twins within the Digital Space.

A semantic description within each DT is necessary for implementing a proactive negotiation behavior between the twins. Particular attention is paid to the description of the skills and capabilities of an asset, as this forms the basis for reasoning required later. In addition to the unique semantic self-description, the assets need to continuously reconcile relevant operational parameters in their DTs, represented by the solid arrows. These DTs can be used to derive benefits through data analytics during the operation of the architecture, thereby increasing the efficiency and utilization of companies' operating resources. The negotiations between the DTs allow conclusions about ongoing operations and the execution of production processes triggered by the other type of DT introduced in the following subsection. The dashed arrows represent this fed-back information.

#### 3.2 Complex Product Demand Digital Twin

The second part of the authors' architecture consists of specific customer-centered, complex product demands. According to (Pfiesterer et al., 2016), a complex product consists of a combination of products and services that can be tailored to fit the customer's individual needs and contextual requirements. In order to make this demand to the architecture accessible, it must be composed into a Complex Product Demand Digital Twin (CPDDT). Thus, customers also participate directly in the digital space through their CPDDT.

Figure 3 depicts various customers who provide their specific complex product demands through their



CPDDT in the Digital Space. The complex product demands are created via an intuitive interface to make the customer's provisioning process as easy as possible. A composing service then transforms the complex product demand and finally provisions a semantically enriched CPDDT to the digital space, represented by the solid arrows. This digital space offers the possibility of contacting other DTs and establishing an interactive collaboration between them. The CPDDTs feedback, represented by the dashed arrow, requests missing descriptive data, final approval, or rejection from the customer. Similar to the provisioning process, the composing service converts the CPDDT information back into a human-readable format for the customer interface. Subsequently, this allows the production process to be triggered by the Asset DTs to satisfy the complex product demand.

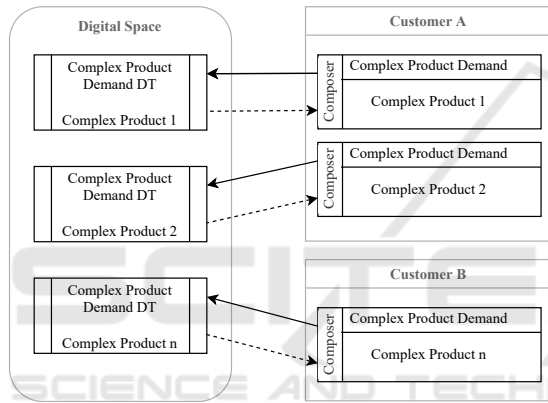


Figure 3: Interconnection of Complex Product Demands and Their Digital Twins.

### 3.3 Collaboration of Digital Twins

After the types of DTs have been introduced, collaborative interaction can be further discussed. The respective digital spaces have been renamed for simplicity to differentiate the various DT types. All entities are nonetheless located within a shared framework. Since the marketplace-oriented interaction mechanisms take place exclusively at the twin level, the link to the physical space, i.e., the assets on the one hand and the customer demands on the other, is neglected.

According to (Einav et al., 2016), peer-to-peer (P2P) market principles establish trade between fragmented buyers and sellers. These buyers and sellers are transposed into DTs located in supply- and demand space in the shown case. DTs can therefore be considered as peers within a marketplace of twins depicted by figure 4. The bidirectional arrow linking supply and demand space is intended to abstract the

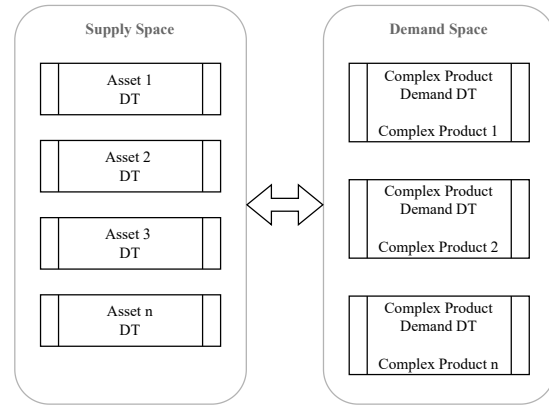


Figure 4: Digital Twins of Assets and Complex Product Demands.

P2P connection between each DT.

There are several challenges in designing marketplaces. The primary goal is to effectively match Asset DTs and CPDDTs while keeping matching problems at a minimum. Furthermore, there are general problems regarding pricing and trust, which will not be focused on further here (Pfisterer et al., 2016).

In order to satisfy complex product demands via its CPDDT, the semantic description is utilized. The CPDDT triggers queries on the underlying ontology of the framework to match asset DTs, which can meet with the specified sub-components or services. Thus, the CPDDT successfully combines the sub-components and services required to fulfill its demand with the other DTs. The following section discusses the required procedures in more detail.

## 4 IMPLEMENTATION

After showing the architectural overview, this section paves the way for proactive interaction between DTs and reinforces the authors' concept by implementing a simplified use case in the manufacturing domain. This demonstrator use case illustrates the collaboration more comprehensibly. Based on the motivation to create a marketplace-based interaction concept, the main foundation of semantic technologies comes into action here, which is essential for collaboration among DTs.

According to figure 5, the authors have instantiated different entities of DTs in supply- and demand space. On the left side, various kinds of asset DTs are shown, such as a Storage, a Punching Machine, a Drilling Machine, and a Laser Cutting Machine. This accumulation of production assets offers a bandwidth of skills and capabilities facing the needs of CPDDTs. Located on the right side is the CPDDT of sheet metal

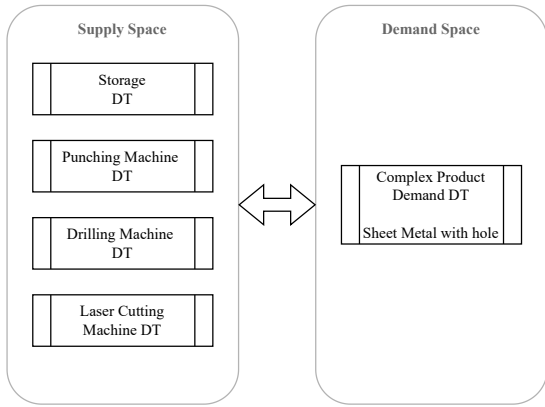


Figure 5: Demonstration Use Case.

with a hole. In order to satisfy this complex product demand, it first needs the specified sheet metal. Then it has to find an asset DT that can make the specified hole into the sheet metal. A functional ontology is built and service queries are implemented via SPARQL query. Not part of the implementation is the complete description of a DT since the focus of this paper is on the realization of the logical query of services based on an ontology. By Using this logical query, a reasoning mechanism to the DTs can be provided to search for the appropriate service on their own and form a collaborative partnership. In the demo showcase, the digital space is considered. This digital space is independent of the physical twin initially neglected in this example. The integration of the physical twin into the demo use case is planned and will be realized later.

In order to implement the previously presented use case, the authors have built an ontology that describes eight classes on the top-level hierarchy consisting of *BillOfMaterial*, *Product*, *ProductionPlan*, *ProductionProcess*, *ProductionStep*, *Resource*, *Service*, and *TypeOfMaterial*. The basic structure of the ontology is based on Pfrommer et al., who divided a system into products, processes, and resources (Pfrommer et al., 2013), within this work the ontology was extended by the classes *ProductionPlan*, *ProductionStep*, *BillOf-*

*Material*, and *TypeOfMaterial*. This extension of the basic ontology allows being more precise by modeling the digital marketplace scenario. The logic of the SPARQL query is based on Lober et al., who demonstrated a SPARQL query for transport services as well as semantic-based service discovery (Lober et al., 2020). Figure 6 shows the high-level structure of the ontology. The eight classes contain all the necessary information for the marketplace scenario by relating the basic classes *Services*, *Resources*, *ProductionProcesses*, and *Products*. The necessary capabilities are inferred from the production process via the production plan and the necessary production steps. The classes *BillOfMaterial* and *TypeOfMaterial* define the necessary information about raw materials and products relating to the production process. The CPDDT should find and connect to the relevant Asset DTs using the ontology.

The classes are divided into logical areas to describe the ontology more precisely. The first area (red) handles the product and the information that directly deals with the product (e.g., type of material and bill of material). The second area (green) deals with the production plan, the production process that can be defined, and the production steps it contains. The last section (yellow) describes the resources and the class of services offered by the resources required to produce the product.

The first area framed in red contains the classes *Product*, *BillOfMaterial*, and *TypeOfMaterial*. The class *Product* represents all types of products in this simple example. It has *RawMaterial*, *SemiFinishedProduct*, and *FinishedProduct* as sub-classes. Hence, some products can be raw material or semi-finished products as well as finished products depending on the level of the supply chain; all product individuals are linked directly to the main class via an *instanceOf* property. A product is assigned to *RawMaterial*, *SemiFinishedProduct*, or *FinishedProduct* by using various object properties to subclass via a reasoning mechanism, depending on its relationship to the product to be manufactured. This variability in the structure of the product class allows the dynamic

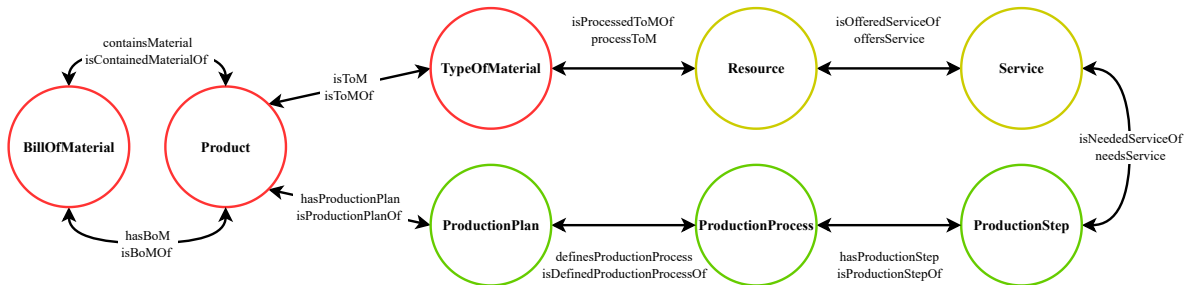


Figure 6: Top-Level Ontology Classes.

representation of the digital marketplace to be realized. The instances of the class *FinishedProduct* can have varying complexity, expressed in the depth of its partial structure, consisting of various semi-finished products and raw materials. In order to describe the product in its entire variability, various data properties such as *thickness* and *holeDiameter* are used to describe the properties of the product. In the shown use case, the product needs a resource capable of making a sheet with a thickness of 5mm and a hole diameter of 1.5mm. The CPDDT can be associated with the production process by logically linking the required production steps (e.g., ability to drill holes) through a given Production Plan using reasoning. These steps are necessary to create the product from raw material resources or components.

In order to describe the product more precisely, the class *BillOfMaterial* is used. The class defines the exact number of necessary raw material and semi-finished products to manufacture the product via its instances linked to the production steps. Furthermore, this class is used to assign an instance of the Product class to its subclass *RawMaterial* or *SemiFinishedProduct*. This assignment is realized via a combination of several object properties.

While the class *BillOfMaterial* establishes a link to the production steps, the class *TypeOfMaterial* is used to describe the product's type of material. The class is also directly linked to the resources via an object property (*isProcessedToMOf*).

The second area (outlined in yellow) deals with the production plan and its production process that can be derived from it, which contains individual production steps. The class *ProductionPlan* defines a production plan for a specific product and is linked to the class *Product* via the object property *isProductionPlanOf*. It defines the flow within the production process by specifying the required services. These services are fulfilled in the individual production steps.

While the production plan defines which services are required, the class *ProductionProcess* describes which production steps run within the process. The production process is assigned to precisely one production plan of a specific product.

The instances of the class *ProductionStep* represent several specific production steps related to the production service needed to produce a product (object property: *needProductionService*). Furthermore, the production step defines the input and output materials of the step via the object properties *needsInputMaterial* and *createOutputMaterial*. Via a Data property (*PositionInProcess*), the process sequence is explicitly defined, which is implicitly described by the

input and output materials. The output material of the last production step is the finished product.

The last area (framed in yellow) shows the classes *Resource* and *Service*. Both classes describe essential information for the supply space in the digital marketplace. Within the class *Resource*, possible physical devices are included, such as production machines, which log into the digital marketplace with specific capabilities (*ProductionServices*). Each resource offers one or more services and can use these services to serve specific production steps required to manufacture or transport a product. The object property *offersService* describes the resource assignment. The class has two sub-classes that can be extended depending on the use case. The sub-classes are shown in figure 7, including all instances of the top-level resource class. If a production step is defined as a production service or a transport service, these instances are assigned to the corresponding sub-classes via reasoning.

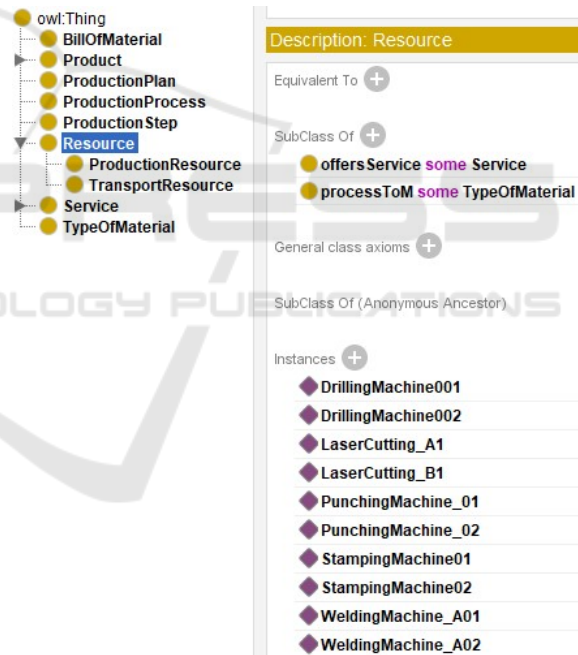


Figure 7: Subclasses of the Ontology.

All logistics or production services relevant to the use case, subdivided into sub-classes, are included in the *classService*. A resource always offers service and at the same time fulfills a production step, which is assigned to a production process in the class *ProductionProcess*. Via the inverse object property *offeredServiceBy* and object property *isNeededServiceOf*, the logical connections are described.

Figure 8 shows an example of the reasoning mechanism within the ontology. The logic structures the

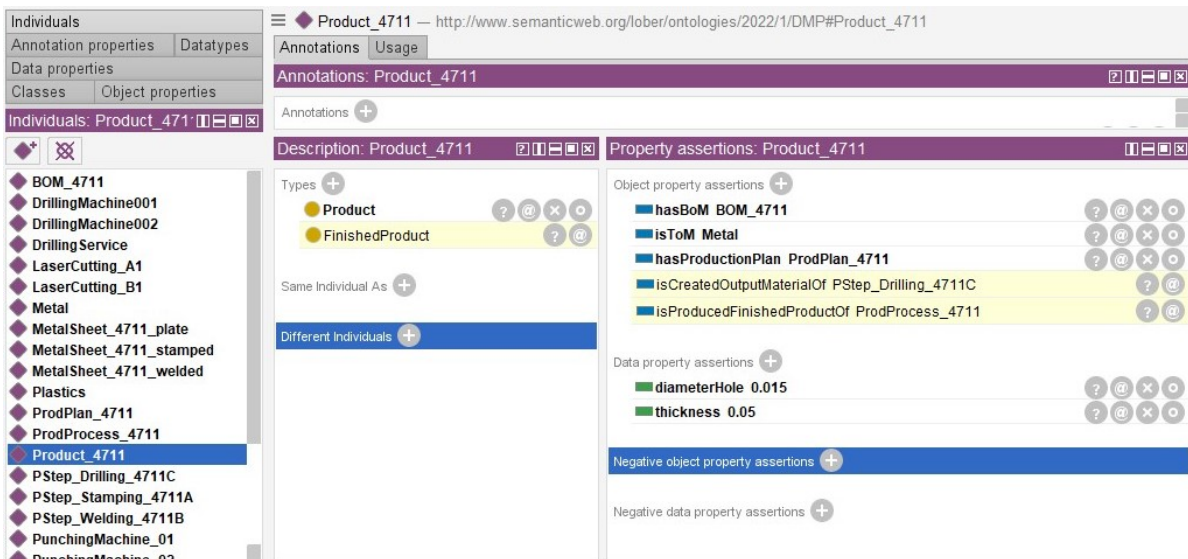


Figure 8: Example of Reasoning.

class *Product* to assign instances to the sub-class *FinishedProduct*. In the example, a reasoner instantiates the product *Product\_4711* as an instance of the subclass *FinishedProduct*. This logic is made accessible implicit via the inverse object properties *isProducedFinishedProductOf* and the corresponding *ProductionProcess* and *ProductionStep* (e.g., *ProdProcess\_4711* and *PStep\_Drilling\_4711C*). Explicitly, this connection is specified via the inverse object property *isCreatedOutputMaterialOf*. The decisive point here is the manufacturing process of the product. It determines the processing steps via the routing, for example, and thus defines whether a resource or a process (process step) covers the product side or the side of the producing system.

In the shown use case, a sheet with a thickness of 5mm and a hole diameter of 1.5mm is produced in three production steps. It is intended that only the drilling service can fulfill the desired depth and width. This is due to the fact that the machines entered (instances of the resources) have specific capabilities and cannot, for example, cut, drill, or punch through sheet metal. Using a SPARQL query on the created ontology will show that semantic filtering based on specific requirements of a complex product is one way to enable collaborative negotiation of DTs based on this.

According to the presented architecture, the respective classes establish a logical relation via specific object properties. The instances of the class *ProductionStep* are only assigned to different resources via their relationship to their products and services. This logic ensures that individual processes and steps can be easily added to the ontology without being limited to a subclass. This easy adding becomes essen-

tial as soon as a production process also contains a transportation process of a more complex product (assemblies and modules). Resources represent a more explicit example. A resource can be a production resource and a transport resource, which carries out work steps or a logistical process during manufacturing. Like the processes, the resources are assigned to the sub-classes of the resource class. This mapping is done via reasoning in the ontology, ensuring a flat class hierarchy. This flat class hierarchy is necessary to ensure a high degree of flexibility within the ontology.

If the use case has to be executed, a SPARQL query must be performed on the ontology, shown in the appendix. The SPARQL query filters the represented information on the ontology by specifying restrictions in the lower area of the query (WHERE) for identifying appropriate production resources. This query first filters on all described products and queries the corresponding necessary production plan, production processes, and corresponding production steps. The subsequent rows query the matching information, which production resource can fulfill the production steps and fulfill the specific variable properties of the product requirements. The variables' product properties are represented as data properties within the ontology and represent the variable information about the thickness and hole diameter of the product. Many production resources can be filtered out by specifying minimum and maximum hole depths that the machines can meet. The last filter represents the machinable material that the machines can process.

Table 1 shows that five production resources are sorted out of six possible resources due to non-



Table 1: Results of SPARQL Query for Suitable Production Resources.

Product	Product Process	Production Resource	Material	Minimum Thickness	Maximum Thickness	Product Thickness	Diameter hole Res.	Diameter hole Prod.	Possible Resource
Product_4711	ProdProcess_4711	DrillingMachine001	Sheet Metal	0.05	0.025	0.05	0.015	0.015	Yes
		DrillingMachine002	Sheet Metal	0.01	0.05		0.020		No
		LaserCutting_A1	Wood	0.02	0.06		0.015		No
		LaserCutting_B1	Wood	0.06	0.1		0.010		No
		PunchingMachine_01	Plastic	0.01	0.05		0.020		No
		PunchingMachine_02	Sheet Metal	0.1	0.15		0.1		No

matching properties. In the application example, the data properties filter for appropriate resources to perform the correct production process (*ProdProcess\_4711*). From the table, it can be seen that some resources are filtered as they are capable of processing inappropriate material. Other resources cannot serve the necessary thickness or diameter hole for the required product. Thus, only *DrillingMachine001* is output as a possible resource at the end of the query. This use case is intended to illustrate that complex filtering is possible in a production step which can provide the starting point for intelligent collaborative negotiation of DTs based on the semantic description of the capabilities of a production system.

## 5 CONCLUSIONS & FUTURE WORK

This paper presents a novel concept of leveraging DTs to a collaborative level based on a marketplace approach. The intention was to break through the limited extensibility of physical production machines by twinning them and empowering the DTs to negotiate proactively with others. After discussing our motivation and the related work in this field, we presented a high-level overview of our aforementioned architecture. Subsequently, a proof of concept was implemented using a simplified demonstration use case.

Combining the DT concept and semantic technologies enables contextualization and thus opens up entirely new possibilities. With this contribution, the authors go beyond DTs as pure data and state representations with at most reactive behavior by giving the DTs their own framework, paving the way for collaborative interaction within a marketplace scenario. For instance, the virtual representations of product demands and production machines can efficiently interoperate in the manufacturing domain. This combination facilitates cross-company interlinking through different industry sectors to the customer. Likewise, the customer experience for complex products increases due to the optimization of production time and

speed and convenient participation in the marketplace and usability. At the same time, higher utilization of production resources on the supply side is effectively possible by providing non-operational production resources on the marketplace. The maximum potential of a marketplace of twins arises with the increasing heterogeneity of the market participants.

As the architecture is still at an early stage, several limitations need to be addressed in the future. On the practical side, in an increasingly growing marketplace, flooding scenarios between the DTs need to be addressed, optimization approaches for recommendation and ranking must be considered. In addition, the *BillOfMaterial* and *TypeOfMaterial* classes of the ontology must be specified more granularly, enabling different Bill of Material types to be integrated as subclasses. The operational side raises problems concerning the transport and logistics of products between individual manufacturing steps, which must be investigated.

The research and development of the presented work form the basis for future research areas. In addition to the limitations described above, our future work will focus on the following three main areas. The initial research area intended to further expand on this work will be the specific structure and context of semantically enriched DTs, particularly asset DTs and CPDDTs. In addition, the detailed elaboration of negotiation mechanisms within the marketplace will be elaborated. Furthermore, agent-based methods constitute an important field of research that needs to be evaluated.

## ACKNOWLEDGEMENTS

This work is supported by a grant of the Karl Völker-Foundation.

## REFERENCES

Ameri, F. and Dutta, D. (2006). An upper ontology for manufacturing service description. volume 2006.

- Anandan, P., Ferreira, P., Lohse, N., and Guedes, M. (2017). An automationml model for plug-and-produce assembly systems. pages 849–854.
- Backhaus, J. and Reinhart, G. (2015). Digital description of products, processes and resources for task-oriented programming of assembly systems. *Journal of Intelligent Manufacturing*, 28.
- Barricelli, B. R., Casiraghi, E., and Fogli, D. (2019). A survey on digital twin: Definitions, characteristics, applications, and design implications. 7:167653–167671.
- Bauernhansl, T., Weyrich, M., Zarco, L., Müller, T., Marks, P., Schlegel, T., and Siegert, J. (2020). Semantic structuring of elements and capabilities in ultra-flexible factories. *Procedia CIRP*, 93:335–340.
- Brovkina, D. and Riedel, O. (2019). Skill-based metamodel for sustaining the process-oriented cyber-physical system description. pages 1–6.
- Cândido, G. and Barata, J. (2007). A multiagent control system for shop floor assembly. pages 293–302.
- Dorofeev, K. and Wenger, M. (2019). Evaluating skill-based control architecture for flexible automation systems. In *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, page 1077–1084. IEEE Press.
- Einav, L., Farronato, C., and Levin, J. (2016). Peer-to-peer markets. *Annual Review of Economics*, 8:615–635.
- Ferreira, P. and Lohse, N. (2012). Configuration model for evolvable assembly systems.
- Fuller, A., Fan, Z., Day, C., and Barlow, C. (2020). Digital twin: Enabling technologies, challenges and open research. *IEEE Access*, 8:108952–108971.
- Grieves, M. (2019). *Virtually Intelligent Product Systems: Digital and Physical Twins*, pages 175–200.
- Grieves, M. (2021). Intelligent digital twins: The role of AI and ML in the future of digital twins chief scientist of advanced manufacturing florida institute of technology.
- Järvenpää, E., Siltala, N., and Lanz, M. (2016). Formal resource and capability descriptions supporting rapid reconfiguration of assembly systems. pages 120–125.
- Karnouskos, S., Leitao, P., Ribeiro, L., and Colombo, A. W. (2020). Industrial agents as a key enabler for realizing industrial cyber-physical systems: Multiagent systems entering industry 4.0. *IEEE Industrial Electronics Magazine*, 14(3):18–32.
- Köcher, A., Hildebrandt, C., Fay, A., and Vieira da Silva, L. M. (2020). A formal capability and skill model for use in plug and produce scenarios.
- Kritzinger, W., Karner, M., Traar, G., Henjes, J., and Sihn, W. (2018). Digital twin in manufacturing: A categorical literature review and classification. *IFAC-PapersOnLine*, 51(11):1016–1022. 16th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2018.
- Leitao, Colombo, K. (2016). Industrial automation based on cyber-physical systems technologies: Prototype implementations and challenges. *Computers in Industry*, 81:11–25. Emerging ICT concepts for smart, safe and sustainable industrial systems.
- Lober, A., Baumgärtel, H., and Verbeet, R. (2020). Semantic service discovery in heterogeneous cyber-physical systems.
- Negri, E., Fumagalli, L., and Macchi, M. (2017). A review of the roles of digital twin in cps-based production systems. *Procedia Manufacturing*, 11:939–948. 27th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM2017, 27-30 June 2017, Modena, Italy.
- Ocker, F., Urban, C., Vogel-Heuser, B., and Diedrich, C. (2021). Leveraging the asset administration shell for agent-based production systems. *IFAC-PapersOnLine*, 54(1):837–844. 17th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2021.
- P. I. 4.0 (2020). Details of the asset administration shell part 1. Plattform Industrie 4.0, Federal Ministry for Economic Affairs and Energy (BMWi), Berlin, Germany.
- Perzylo, A., Grothoff, J., Lucio, L., Weser, M., Malakuti, S., Venet, P., Aravantinos, V., and Deppe, T. (2019). Capability-based semantic interoperability of manufacturing resources: A basys 4.0 perspective. *IFAC-PapersOnLine*, 52(13):1590–1596. 9th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2019.
- Pfisterer, D., Radonjic-Simic, M., and Reichwald, J. (2016). Business model design and architecture for the internet of everything. *Journal of Sensor and Actuator Networks*, 5(2).
- Pfrommer, J., Schleipen, M., and Beyerer, J. (2013). Pprs: Production skills and their relation to product, process, and resource. In *IEEE 18th Conference on Emerging Technologies & Factory Automation (ETFA), 2013*, pages 1–4, Piscataway, NJ. IEEE.
- Pfrommer, J., Štogl, D., Aleksandrov, K., Navarro, S., Hein, B., and Beyerer, J. (2015). Plug & produce by modelling skills and service-oriented orchestration of reconfigurable manufacturing systems. *at - Automatisierungstechnik*, 63:790–800.
- Qi, Q., Tao, F., Hu, T., Anwer, N., Liu, A., Wei, Y., Wang, L., and Nee, A. (2021). Enabling technologies and tools for digital twin. *Journal of Manufacturing Systems*, 58:3–21. Digital Twin towards Smart Manufacturing and Industry 4.0.
- Rocha, A. D., Tripa, J., Alemão, D., Peres, R. S., and Barata, J. (2019). Agent-based plug and produce cyber-physical production system – test case. In *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, volume 1, pages 1545–1551.
- Sakurada, L., Leitao, P., and De la Prieta, F. (2021). Towards the digitization using asset administration shells. In *IECON 2021 – 47th Annual Conference of the IEEE Industrial Electronics Society*, pages 1–6.
- Saracco, R. (2019). Digital twins: Bridging physical space and cyberspace. *Computer, IEEE Computer Society*, 52(12):58–64.
- van der Valk, H., Haße, H., Möller, F., and Otto, B. (2021). Archetypes of digital twins.
- Vogel-Heuser, B., Ocker, F., and Scheuer, T. (2021). An approach for leveraging digital twins in agent-based

production systems. *at - Automatisierungstechnik*, 69(12):1026–1039.

- Vogel-Heuser, B., Schütz, D., Frank, T., and Legat, C. (2014). Model-driven engineering of manufacturing automation software projects – a sysml-based approach. *Mechatronics*, 24(7):883–897. 1. Model-Based Mechatronic System Design 2. Model Based Engineering.
- Wagner, R., Schleich, B., Haefner, B., Kuhnle, A., Wartzack, S., and Lanza, G. (2019). Challenges and potentials of digital twins and industry 4.0 in product design and production for high performance products. *Procedia CIRP*, 84:88–93. 29th CIRP Design Conference 2019, 08-10 May 2019, Póvoa de Varzim, Portugal.

## APPENDIX

The following SPARQL query, as introduced in the implementation section, filters the represented information on the ontology by specifying restrictions in the *WHERE-Statement* to identify appropriate production resources:

```
PREFIX DMP: <http://www.semanticweb.org/lober/ontologies/2022/1/DMP#>

SELECT ?Product ?ProductionProcess ?Resource ?ProductionService

WHERE {
    ?Product DMP:hasProductionPlan ?ProductionPlan.
    ?Product DMP:thickness ?ProductThickness.
    ?Product DMP:diameterHole ?DiameterHole.
    ?Product DMP:hasToM ?TypeOfMaterial.
    ?ProductionPlan DMP:definesProductionProcess ?ProductionProcess.
    ?ProductionProcess DMP:hasProductionStep ?ProductionStep.
    ?ProductionStep DMP:needsProductionService ?ProductionService.
    ?ProductionResource DMP:offersProductionService ?ProductionService.
    ?ProductionResource DMP:processToM ?ProcessableToM.
        Filter (?ProcessableToM = ?TypeOfMaterial).
    ?ProductionResource DMP:minThickness ?MinThickness.
        Filter (?MinThickness <= ?ProductThickness).
    ?ProductionResource DMP:maxThickness ?MaxThickness.
        Filter (?MaxThickness >= ?ProductThickness).
    ?ProductionResource DMP:diameterHoleResource ?DiameterHoleRes.
        Filter (?DiameterHoleRes = ?DiameterHole.)
}
```