applied sciences

MDPI

Article

Ontology-Based Production Simulation with OntologySim

Marvin Carl May *, Lars Kiefer, Andreas Kuhnle and Gisela Lanza

wbk Institute of Production Science, Karlsruhe Institute of Technology (KIT), Kaiserstr. 12, 76131 Karlsruhe, Germany; uufrm@student.kit.edu (L.K.); andreaskuhnle@t-online.de (A.K.); gisela.lanza@kit.edu (G.L.)

* Correspondence: marvin.may@kit.edu; Tel.: +49-1523-950-2624

Abstract: Imagine the possibility to save a simulation at any time, modify or analyze it, and restart again with exactly the same state. The conceptualization and its concrete manifestation in the implementation OntologySim is demonstrated in this paper. The presented approach of a fully ontology-based simulation can solve current challenges in modeling and simulation in production science. Due to the individualization and customization of products and the resulting increase in complexity of production, a need for flexibly adaptable simulations arises. This need is exemplified in the trend towards Digital Twins and Digital Shadows. Their application to production systems, against the background of an ever increasing speed of change in such systems, is arduous. Moreover, missing understandability and human interpretability of current approaches hinders successful, goal oriented applications. The OntologySim can help solving this challenge by providing the ability to generate truly cyber physical systems, both interlocked with reality and providing a simulation framework. In a nutshell, this paper presents a discrete-event-based open-source simulation using multi-agency and ontology.

Keywords: ontology; production simulation; multi-agent; digital twin

Citation: May, M.C.; Kiefer, L.; Kuhnle, A.; Lanza, G. Ontology-Based Production Simulation with OntologySim. *Appl. Sci.* **2022**, *12*, 1608. https://doi.org/10.3390/app12031608

Academic Editor: Roque Calvo, José A. Yaguë-Fabra and Guido Tosello

Received: 4 January 2022
Accepted: 26 January 2022
Published: 3 February 2022

1. Introduction

Product differentiation and customer satisfaction today demonstrate a shift from a technical product focus towards customized, unique products. Increasing this product individualization leads to changes and increases in the complexity of production systems and amplifies the necessity for more flexible production systems [1,2]. Changes in production directly affect both modeling and the simulation of production systems, as these are carried out with the purpose of providing analyses and insights into the up-to-the-minute, real production system [3]. Most notably, to provide insights into complex systems and make decisions, for instance, regarding production changes [4]. To meet this new demand for flexibility of production system simulations [3], this paper presents an ontology-based simulation enabling the transfer of ontology advantages to simulation. The ontology's ability to dynamically map multi-dimensionality representations [5] offers new possibilities for simulation [6]. In this paper, the conceptualization of a simulation based on a knowledge graph as an instantiated ontology is presented. This is extended by introduction of the manifestation, the Owlready [7]-based open source solution OntologySim, which fully integrates an ontology into the simulation and offers a visualization via web development. The multi-agent-based OntologySim, thus, provides an ideal basis for the application of a digital twin. This research aims to present a combination of ontology with manufacturing simulation, which is available as an open-source solution to the general public and is distinguished from previous solutions by its flexibility and storability.A digital twin, as an up-to-the-minute representation of the real system [3], is instantiated based on the OntologySim as a digital master. Hence, the current production system shall be mapped in detail to the simulation, the ontology-based model that is both describing the state as a knowledge graph and generally modeling the ontological structure of the regarded

system, as shown in Figure 1. The underlying concept of this storable and expandable ontology-based simulation is discussed in more detail in Section 4.1.

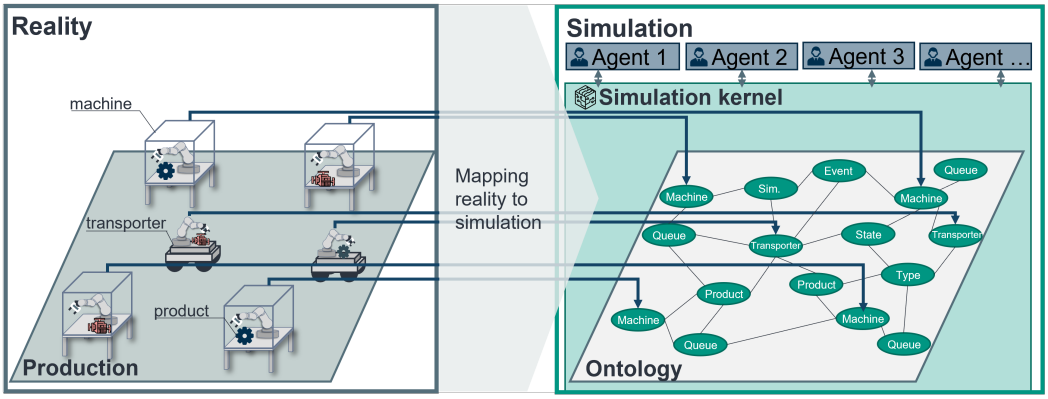

Figure 1. Outline of an ontology-based simulation model.

The paper is organized as follows. In Section 2, Foundation, the terms ontology and simulation are introduced. Section 3, Related Work, categorizes existing simulations and ontology-based simulations in the realm of production and beyond. In Section 4, Proposed Ontology-based Simulation, the unique features of an ontology-based simulation approach are presented and extended in Sections 5–7, introducing the manifestation OntologySim and the individual components and functionalities in more detail. This paper finishes with a discussion in Section 8 and a Summary and Outlook in Section 9.

2. Foundation

Given the novelty of the approach of using ontologies as means for simulating production systems and the necessity to elaborate both topics individually, this section presents an introduction to ontologies in Section 2.1, classical simulation in the context of production systems in Section 2.2, and a delimitation of simulations based on an ontology in Section 2.3.

2.1. Ontology

The term ontology was initially coined in philosophy and describes the semantic representation of existence. For today's problems, the term ontology is often associated with the term knowledge-base [8]. For the exact definitions of an ontology, there are multiple versions available, but, for this application, the following definition is quite appropriate [9]:

The formal description of an ontology is

$$O = (C, R, A, Top),$$

where C is the set of all concepts, R the set of all assertions, and A the set of all axioms. Top defines the top level of the hierarchy. R contains the two subsets H and N, where H is the "set of all assertions in which the relation is a taxonomic relation" [9], and N "a set of all assertions in which the relation is a non-taxonomic relation" [9]. An instantiated ontology can best described with the term knowledge graph, where concepts refer to vertices, and relations refer to assertions that signify the type and structure of the relation between two concepts.

The strengths of an ontology compared to object-oriented languages (OOL) are "reusability, interoperability, flexibility, consistency and quality checking and reasoning" [10]. In the field of OOL, there are only UML diagrams that offer a similarly good basis for reuse and interoperability. However, implemented applications are often incompatible, even if stated with the same UML diagram. Furthermore, reasoning, by design achievable with ontologies, offers classification, consistency checking, and knowledge extraction, which can only be achieved with significantly more effort in OOL [10].

To take full advantage of an ontology, five design principles have been established [8]. These are named and briefly explained below:

- *Clarity* pursues the goal to design definitions objectively and formally and to prefer fully consistent definitions whenever possible.
- *Coherence* is the property of being logical and consistent.
- *Extendability* allows the ontology to be extended or modified regarding new relations and vocabularies.
- *Minimal encoding bias* means that the chosen representation goes beyond the current implementation to ensure knowledge-sharing over the current use-case.
- *Minimal ontological commitment*: An ontology should have few requirements and restrictions on the modeled world to represent specialized and individual circumstances.

These design principles are also applied to the OntologySim, with a focus on "minimal ontological commitment" and "minimal encoding bias". The goal of the OntologySim is to represent diverse production types and characteristics in the best possible way to be able to map complex and individualized production plants with all their characteristics to an ontology. One classification of ontologies is based on a three-step approach that classifies an ontology based on the scope of concepts that are included: (1) general ontologies, including cross-divisional concepts, (2) top level ontologies, focused on a particular application, and (3) domain ontologies, that translate known class and data models [11]. Another classification can be based on the purpose of an ontology, which can be limited to communication enablers, where agents derive their communication convention from a commonly accepted ontology, the purpose of automatic reasoning to identify new concepts and relations or the purpose of representation through knowledge reusability [12].

Typical languages for the description and exchange of ontologies in the wake of the Semantic Web Initiative [13] are the Resource Description Framework (RDF), initially developed to describe metadata by describing resources with characteristics and links to other resources, and the Extensible Markup Language (XML), developed to annotate and describe data and documents. Their power can be vastly increased by using schemata, i.e., ontology languages, such as the Resource Description Framework Schema (RDFS), a set theory-based formalism for ontologies, and the Web Ontology Language (OWL) family, based on description logic and set theory [7,14].

In the following, two classification approaches for ontologies are presented in more detail, firstly, the distinction between static versus dynamic ontologies [15]. Secondly, from a programming perspective, the distinction between the SPARQL Protocol And RDF Query Language (SPARQL), a graph-based query language for the Web Ontology Language (OWL) [16], and ontology-oriented programming for OWL [7,17], as well as traditional Application Programming Interfaces (API), is made.

A static ontology assumes that the world is static, so only queries are possible, and "inference, classification or dynamic class creation" are not allowed [17]. Examples of static ontologies are given by Kalyanpur et al. [18] and Goldman [19]. Dynamic ontologies allow the creation of classes and instances during runtime [17]. They often use concepts that include state, state transition, and process and are inspired by finite state machines and Petri nets [10].

An overview of the classification of an ontology is visualized in Figure 2.

In the structure and access of an ontology, three types have been established, SPARQL, the traditional OWL API, and object-oriented OWL. The differences between the two types are explained using the Java OWL API and Owlready. SPARQL will not be discussed further in the following because of the following significant disadvantages. SPARQL is an RDF query language based on SQL (Structured Query Language), but, since it is not based on OWL and queries have to be written for each access [17,20], it is significantly slower than the alternatives. OWL API and Owlready use the OWL2 standard based on the W3C specification [21]. OWL2 is an extension to OWL concerning "richer data types, data ranges, qualified cardinality restrictions, asymmetric, reflexive, and disjoint properties" [21]. The RDF (Resource Description Framework) sets the standard grammar for formal "association

among resources" [10] and is based on an XML schema [10]. OWL extends the RDF concerning the "ability to express more information about the characteristics of properties and to define classes by grouping those instances that meet these characteristics" [10].

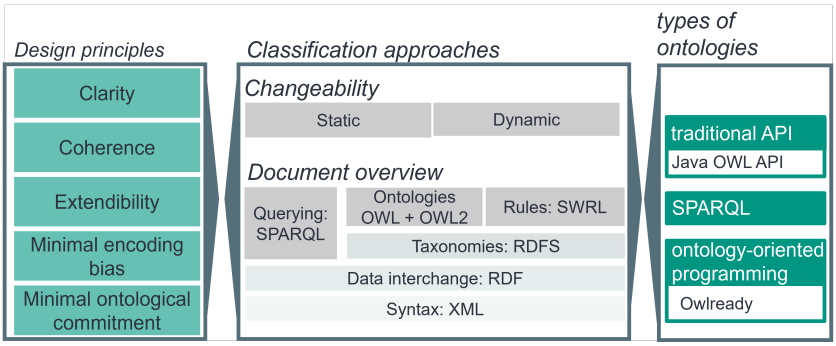

Figure 2. Summary of the OntologySim foundation, based on [8,10].

Traditional APIs define unified methods and classes to modify and customize the ontology. The Java-based OWL API allows loading and saving of different syntaxes, but with an independent internal syntax. The OWL API is widely used in Protégé-4 [22], SWOOP [23], and NeOnToolkit [24,25], which are tools capable of modeling ontologies, sometimes even graphically.

In ontology-oriented programming, the entities of the ontology classes, properties, and individuals are considered as classes, attributes, and instances in the object model [17]. The basic structure of Owlready is based on a SQLite3 database, which stores the optimized RDF quadstore [7]. If required, the ontology entities are loaded in Python, and a modification of the Python objects leads to an automatic update of the quadstore [7]. The advantages of ontology-oriented programming are lower programming effort and easy readability of the code [17]. Furthermore, Owlready combines the agility of object-oriented programming with the expressiveness of an ontology with good access speeds through relational databases [26]. These reasons were decisive for the use of Owlready2.

A comparison of the resource efficiency between OWL API and Owlready shows that Owlready requires less memory and has faster loading times than the OWL API for smaller applications (with about 60,000 classes). For larger ontologies (5 million triples), the loading time for OWL API is 100% faster. In terms of listening time, OWL API is significantly faster, although the gap increases exponentially with larger applications [17]. For this reason, when designing an ontology, care must be taken to ensure that the ontology remains compact, as the resource overhead increases exponentially, especially for listening for very large ontologies. For the application of the simulation in the presented OntologySim framework, for instance, with 5 machines, Owlready has 73 classes and 1225 objects, and it, thus, uses significantly fewer classes and edges than the benchmark test. For this reason, the problem regarding large, slow ontologies is of less relevance in this case. Owlready was introduced in 2017 and has been successfully used in biomedical informatics [17], for football games [27], and production simulation [28]. However, the differences between Liu [28] and OntologySim are discussed in Section 3.

2.2. Simulation

The term simulation refers to mapping a system to analyze its dynamic processes in a simplified replica (of the target system) and to obtain transferable knowledge [29]. When modeling the simulation, different approaches can be used either individually or in combination. There is a distinction between Discrete Event Simulation (DES), which has event-based step sizes, System Dynamics, which has a top-down approach over system changes per time, and Agent-based Simulation, which executes decision processes based on agents. Agent-based simulation can be further divided into multi-agent-based and single agent-based systems [30]. For the application in complex systems, multi-agent-based can be

advantageous because of the decentralized and robust structure [31–33]. The OntologySim is a discrete event-based, multi-agent system. The exact classification of the simulation is discussed in Section 4.

Besides the modeling type, a simulation consists of the following components according to the VDI3633 [29]: a simulation kernel, data management (see Figure 3), user interface, and interfaces to external databases. The simulation kernel contains the model world and necessary elements, particularly events to ensure automatic execution [29]. The last level, the interfaces to external databases, is not included in this simulation because the OntologySim attempts to represent as many applications as possible, and this interlocking with a particular external database conflicts with the many individual external databases prevailing in real production systems. Hence, a config file or owl file specifying the required information to be extracted before or while execution and API interfaces serve as external interfaces. For more detail, the reader is referred to Section 7.

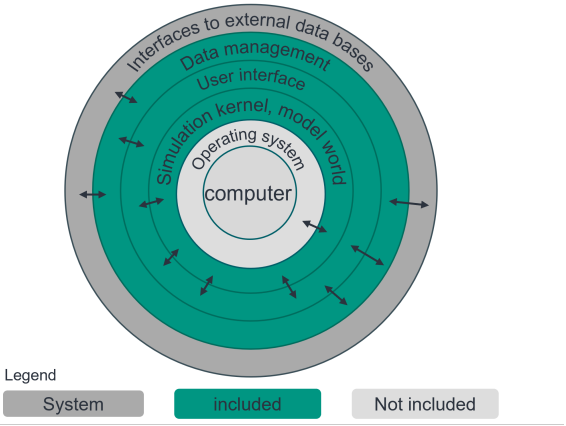

Figure 3. Components of the OntologySim simulation tool (DES), based on [29].

2.3. Term Ontology-Based Simulation

Ontology-based simulation as a term is used differently throughout the literature as the degree of ontology application is not clearly defined. The proposed classification scheme, as shown in Figure 4, is based on Reference [30,34] and has been adapted and integrated with respect to simulations based on an ontology. The Ontology Integration Level (OIL) for simulations continuously increases from production capacity to ontologies that store all subcategories and simulation specifications. It allows the classification of ontologies that are interwoven with an ontology-based simulation in the area of production simulations.

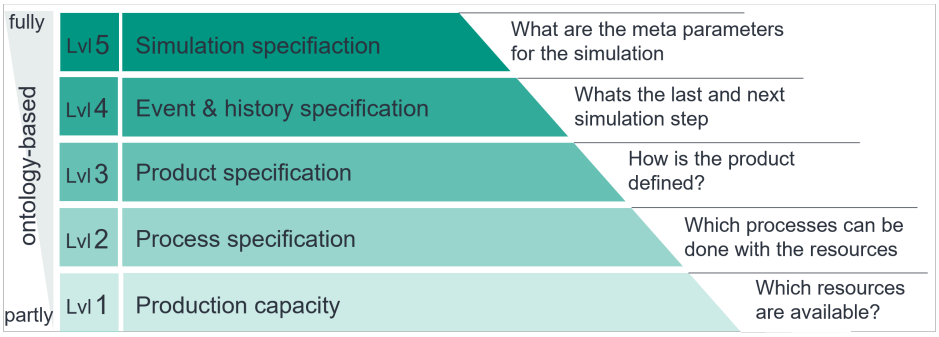

Figure 4. Examining the Ontology Integration Level for ontology-based production simulation.

The definition of each level is as follows:

- *Lvl1: Production capacity:* The ontology contains information about the capacity and structure of the ontology, such as machines, transporters, and queues.
- *Lvl2: Process specification:* The next level adds information regarding processes, setup behavior and resource change.
- *Lvl3: Product specification:* An ontology enables products to be produced in simulation. This means that production plans are included in the ontology and the current processing step of the product is known in the ontology at any time.
- *Lvl4: Event & history specification:* Events and data from the past lay the foundation for processes and allow the determination of the next simulation step.
- *Lvl5: Simulation specification:* The highest ontology integration level contains all information of the previous levels and focuses on all further necessary information, such as current time steps or work orders, to be contained in the ontology. Fully ontology-based simulations allow saving the ontology, and, thus, the simulation core and current status, at any time without information loss. This is possible when both the data management and the simulation kernel are integrated into the ontology.

According to this definition, OntologySim is a fully ontology-based simulation in ontology integration level 5.

3. Related Work

As introduced in Section 2.3, different integration levels for ontologies in simulation and production exist. The literature in this domain is reviewed and extended with other approaches of ontology-based simulations in Section 3.1. Then, available simulations for production systems are reviewed in Section 3.2.

3.1. Literature Review

In the literature, there are many approaches to integrate ontologies in production and simulation. The majority of approaches to date focus on using the ontology as a memory or database to build a simulation model. This is similar to handling the simulation data management with the help of an ontology, yet, the simulation itself remains unchanged. Based on a grounded theory literature review [35], a large set of papers dealing with both simulation and ontology or both digital twin and ontology in the domain of production research were identified. By manual analysis a total of 20 papers, the most relevant research approaches were selected (cf. Table 1), with priority given to approaches with higher Ontology Integration Levels and the requirement of including ontologies from a research perspective in production organization. In general, the existing approaches can be clustered as follows:

- *Concepts/schemata for production simulation:*
  This category is characterized by the fact that schemes and concepts for a simulation are presented, but the active use has not yet been implemented or has only been carried out for an example case. Some of the concepts differ greatly in scope and structure of the ontology. Thus, multi-agent approaches were designed, such as that by Karageorgos et al. [36], who define an agent-based approach to support logistics and production planning; and Mönch and Stehli [37], who store data based on domain-related predicates, such as machine structure and task-related predicates, such as scheduling. Additionally, there are concepts for the description of the factory layout [38] by ontological means. Furthermore, there are databases, such as ONKI [39], available, which hold production data sets and ontology schemata for production.
- *Real world application (CPS, MES):*
  These ontology approaches are characterized by the fact that a virtual representation of a production plant and a Manufacturing Execution System (MES) is extended by an ontology to maintain flexibility. In these approaches, the ontology often serves as a classic knowledge database. Examples are the use of ontologies for MES and OPC-UA interfaces [40] and a digital twin for a Cyber-physical Production system (CPS) [28]. A different approach is followed by the virtual factory data model presented by

Terkaj [41], which aims at representing factory objects, i.e., from products to machines, virtually, in a static way.

- *External data source to facilitate easier production simulation start:*
  These papers pursue the integration of different external data sources to create a uniform simulation data model through an ontology. It is noticeable for this type of concept that multiple ontologies are used for modeling [42–44]. The simulation models are either converted into executable models, as in Silver et al. [42], or the ontology serves as a knowledge database adjunct to the simulation model [43,44]. Benjamin et al. [43] introduce an ontology-driven framework based on scheduling, simulation and optimization ontology. Du et al. [44] implement a framework, where multiple databases are integrated into multiple ontologies and then combined to one core ontology model. A hybrid approach focuses on modeling external data sources, a static virtual model of the factory and its history, presented by Terkaj and Urgo [45], likewise uses ontologies and extends so far static representations with the system's history and evolution. Thus, these can be classified as a digital shadow [3]. An extension of this approach is described by Terkaj et al. [46], who use this continuously synchronized, ontology-based virtual model to enable "in situ" simulation of future system behavior. Hence, the approach aims at enabling a foresighted digital twin [3].
- *Fully ontology-based simulations:*
  A similar approach to the OntologySim is only provided by the paper of Warden et al. [47]. Here, an attempt was made to create a simulation for transport logistics utilizing several ontologies, which were divided into different layers. The goals regarding "scalable, portable and reusable domain models [...] fell short [...] and turned out to yield more pain than gain" [47].

The categorization of different literature examples is presented in Table 1

Table 1. Literature review.

	Ontology System						Real World		Simulation			
Approach by	**Ontology Schema**	**Multi Agent System**	**Multiple Ontology**	**Ontology as Data Storage**	**Application Framework**	**External Data Source**	**Cyber-Physical-System (CPS)**	**Manufacturing Execution System (MES)**	**Ontology Integration Level**	**Intervention**	**Visualization**	**Main Goal**
Concepts / schemata for production simulation												
Terkaj and Urgo [45]	●	○	○	●	●	●	○	○	4	○	○	Performance
Terkaj et al. [41]	●	○	○	●	○	○	○	○	3	○	○	Virtual model
Viljanen et al. [39]	●	○	○	○	○	○	○	○	-	○	○	Database
Mönch and Stehli [37]	●	●	○	●	●	○	○	○	3	○	○	MAS
Cheng et al. [6]	●	●	○	○	○	○	○	○	3	○	○	Concept
Scholz and Schabus [38]	●	○	○	●	○	○	○	○	1	○	○	Schema
Gurjanov et al. [48]	●	○	○	○	○	○	○	○	2	○	○	Product
Karageorgos et al. [36]	●	●	○	●	○	○	○	○	3	○	○	MAS
Fumagalli et al. [49]	●	○	○	●	○	○	○	◐	3	○	○	MES
Mazzola et al. [50]	●	○	○	○	○	●	○	○	2	○	○	Concept
Guizzardi and Wagner [51]	●	○	○	○	○	○	○	○	4	○	○	DES
External data source to facilitate easier production simulation start												
Du et al. [44]	●	●	●	●	●	●	○	○	3	○	●	AnyLogic
Terkaj et al. [46]	●	○	○	●	●	●	○	○	4	○	●	Speed
Benjamin et al. [43]	●	●	●	●	●	●	○	○	3	○	○	Framework
Silver et al. [42]	●	◐	●	○	●	●	○	○	3	○	○	Parsing
Real world application (CPS, MES)												
Liu et al. [28]	●	○	○	●	○	○	●	●	3	○	●	digital twin
Ansari et al. [52]	●	○	○	●	●	○	●	○	2	○	○	Problem
Katti et al. [40]	●	○	○	●	○	○	●	●	2	○	○	OPC UA
Chen and Tu [53]	●	○	○	●	○	○	◐	◐	1	○	●	CPS
Fully ontology-based simulations												
Warden et al. [47]	●	●	○	●	●	○	○	○	4	○	○	Scheduling

Legend: ● regarded ◐ rudimentary regarded ○ not regarded

All in all, there is still a research gap regarding fully ontology-based simulations that do not only use the ontology as a data storage. The use of the ontology as a control element, event handler, and data base has been little considered so far. However, promising advantages of continuing the ontology integration are flexibility [47], correct data through interlinkage with the real system [28], and extendability [45] among others. Thus, the need for researching and introducing concepts on Ontology Integration Level 5 arises.

3.2. Simulation Programs

Table 2 compares existing simulation systems on the market. For the comparison, a Python-based discrete-event simulation and, for simplification, two widely used commercial products, AnyLogic and Siemens Plant Simulation (PLM), are used. The selection of

the 3 simulation tools from the available simulation is based on the similar use in the production environment and the high popularity. Given the regarded scope of the comparison, commercial alternatives, such as Arena- or Petri net-based production system simulation, do not differ greatly. While commercial applications include visualizations, open source approaches, such as SimPy [54,55], seldom provide this kind of user-friendliness. Ease of use and first start is typically provided by graphical editors [56] or configurations, thus providing little flexibility. Furthermore, given DES cannot be interrupted, the simulation core changed, and then continued, a feature dubbed intervention. Furthermore, KPIs are typically self defined or calculated, omitting international standards, such as the ISO-22400-2:2014 standard [57].

Table 2. Comparison of a selection of existing simulation applications.

	User-Friendliness				Availability		Modularity
Simulation	**Visualization**	**Ease of Use**	**First Start**	**KPI Usage**	**Open Source**	**Intervention**	**Flexibility**
SimPy [54]	○	◐	●	◐	●	○	◐
AnyLogic [56]	●	◐	●	◐	○	○	◐
Siemens PLM	●	◐	●	◐	○	○	◐
Legend:	● implemented	◐ partly implemented	○ not possible				

In comparison to traditional approaches, the main advantages of the OntologySim lie in the combination of a storable, changeable DES on open source basis with visualizations. In the following chapter, the structure and advantages of the OntologySim are explained in more detail.

4. Proposed Ontology-Based Simulation

In this chapter, the OntologySim idea is explained in more detail. Furthermore, the OnologySim is classified according to VDI 3633 [29], and the unique selling propositions, that fit the previously outlined research gaps, are presented.

4.1. OntologySim Conceptualization

The main requirement in the conception and implementation of OntoloySim is to implement a flexible, modular simulation that can be saved and reloaded at any time. The process of changing and loading the simulation enables a direct linkage and runtime adjustement to the reality, which is shown graphically in Figure 5. Thereby, OntologySim enables a truly interlinked Digital Twin.

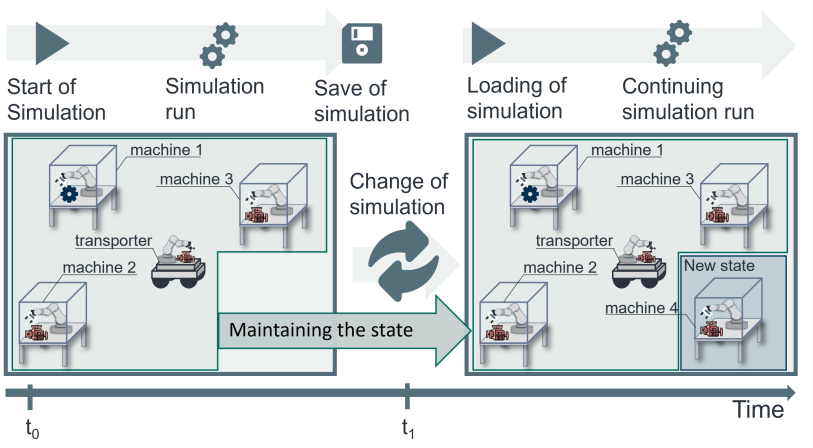

Figure 5. Main idea of OntologySim.

After configuration and the start of the simulation, the OntologySim can be saved at any time, with or without interruption. Since large parts of the simulation core are directly connected and modeled within the ontology and the fact that the ontology holds all relevant information. After saving the ontology in an "owl" file, changes can be made within or outside of the executable OntologySim, such as adding a machine or AGV (Automated Guided Vehicle). Loading the ontology guarantees the seamless continuation of the simulation at any time.

4.2. Unique Selling Proposition (USP)

Figure 6 summarizes the main points of differentiation from other simulations. The individual contents of the framework are explained in more detail in the various chapters: Sections 5–7.

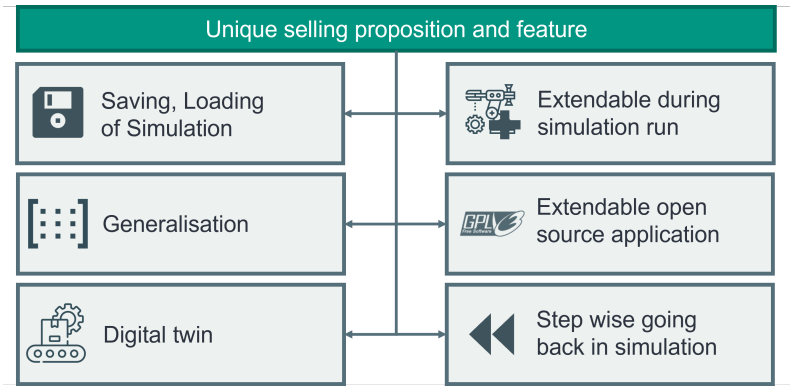

Figure 6. Unique position of the simulation.

The USP is based on the following advantages over existing discrete-event simulations:

- *Saving, Loading of Simulation:*
  Interrupting the simulation during runtime enables new use cases and possibilities, which are shown in Figure 7. When saving the simulation, the current state of the ontology is stored in a single OWL file. This saved data is sufficient to restart the simulation, as all essential information is contained in the ontology. In particular, the following new use cases are enabled:
  - The stored image can be run on other computational units and enables the easy extraction of information data.
  - After saving a defined state, several strategies, multi-agents, etc., can simulate different simulation runs, and the best agent configuration can be selected to facilitate research in this domain [31,55,58].
  - Furthermore, time saving is achieved because the initial run-up and warm-up of a simulation into a static state can be skipped, as a static stimulation can be loaded, that is "identical" to the real system state to increase inference accuracy required for real-world use cases [59,60].
  - A real world digital twin is enabled, as up-to-the-minute virtual models can be started with the exact same initial situation [3].

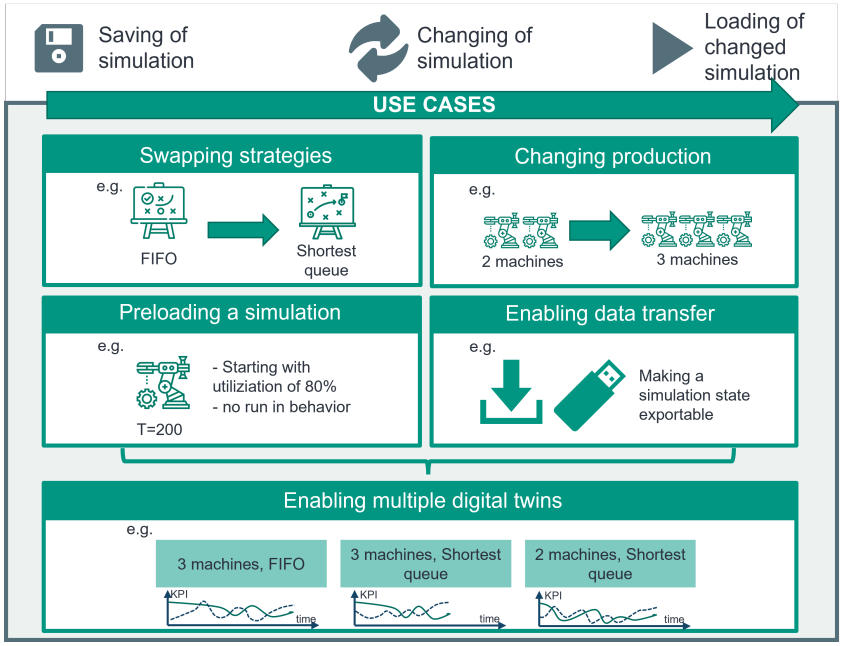

Figure 7. Use cases for saving and loading simulation.

- *Extendable during simulation run:*
  Saving and loading the simulation provides the basis for the simulation to be extended during a run. In addition to changing the strategy, adjustments can also be made to the manufacturing system which affect resources, such as machines, transporters, or processes. Thus, for example, transporters can be removed or added, or process times can be changed. It is also possible to add data and information that is not needed currently for the simulation. By adding supplementary nodes, it is possible to link external (real-world) data or to model the current description in detail. An example would be the addition of installation space sizes for the product, novel products added to the portfolio [33], or changes in the shopfloor management, as well as increased worker competencies [61].
- *Extendable open source application:*
  Another advantage compared to the majority of simulations reviewed in Section 3 is that OntologySim is implemented with the basic idea of being an open-source publication and software. Thus, the focus of the implementation is to make the simulation easily adaptable and to provide clearly defined and explained interfaces. This makes it possible to simply extend the simulation with self-implemented agents and access the ontology through standardized wrapper methods.
- *Generalization:*
  The interaction of the ontology and the Python modules enables a good generalization of simulation models. It is possible to model different production systems, be it line production, a workshop production or matrix production, or any combination. The ontology can be extended and customized to meet specific requirements. This includes the possible integration of information about tangible objects, such as products, intangible information, such as production planning, and control organization [33,58], or ever-changing, human-centered Industry 4.0 implementations [61].
- *Step wise going back in simulation:*
  Going back within a single simulation run in a step-by-step manner is another special feature of the OntologySim. Going back enables a better analysis of the simulation and increases the traceability and understanding of complex production systems [60]. This feature is made possible by the fact that, at any point in time, a defined state is available, and the past events are stored. These two properties are sufficient to recreate the past and explicitly analyze its states. By doing so, the analysis of ”what-

if" scenarios within a single simulation run can be evaluated without the need to instantiate many different simulations, for instance, for time-constraint adherence predictions [59,60].

- *Digital twin:*
  The OntologySim can ideally serve as a digital twin. It possesses the unique ability to serve as digital twin, digital master, and, to some degree, digital shadow, all at once. Starting from any current state in the real system, the ontology can be created and updated externally, manually, or by connection to data sources, such as MES or ERP systems. Then, an instantiation, or, in other terms, a simulation run, can be started directly within the OntologySim framework. This enables increasing digital twin capabilities for production systems [3].

4.3. Classification of simulation

This section classifies the simulation according to VDI3363 [29], as shown in Figure 8. The OntologySim is a discrete event-based simulation. The event-based programming has the advantage of simple implementation, high execution speed, and flexibility; for example, one event can easily trigger several other events [54]. Furthermore, the OntologySim is a multi-agent-based simulation. Each machine and transport unit represents an agent that can be instantiated as to decide independently.

Category of VDI3633	*Classification*			
Type of model description	Graphics-oriented	**Parameter-oriented**	Language-oriented	
Type of data structure for model configuration	**predefined**	**User-definable**		
Type of internal time-advance mechanism	Fixed-increment time advance	**Event-scheduling**	Process interaction	Transaction flow
Type of interaction possibilities during run time	batch processing (no interaction possible)	queries	**interventions (possible to change the model)**	

Figure 8. Classification of simulation according to VDI3363 [29].

5. Design Principles for the OntologySim

The OntologySim concept and the structure of the developed ontology presented below is intended to enable mapping diverse production systems, such as line production, workshop production, and matrix production, efficiently, in a modular way. The basic requirement for the OntologySim development is the ability to save the simulation state at any time, without data loss, and the ability to restart the simulation using the saved ontology. From this requirement follows that all entities have to be mapped in this ontology, achieving Ontology Integration level (OIL) 5. Thus, not only machines and transporters are stored in the ontology, but also set-up processes, defects and services, their statistics, the production plan for all products, and relevant future and past events. The storage of any entities in the ontology enables a high degree of flexibility since, on the one hand, relationships from zero to n are possible, and, on the other hand, a high degree of detail can be generated. Furthermore, the information and relationships within the ontology are dynamically adapted. Each executed event or simulation step executes changes to the ontology. Because past events are also stored in the ontology, it is possible to recreate past states from the current state of the ontology. This past information is also seamlessly available to enable machine learning-based production control, for instance, with reinforcement learning [55,58]. The concept and functioning of the ontology are exemplified by the entities Machine, Event, and Product type in the following.

5.1. Machine

The basic components of machines are queues, which are divided into different categories. One is the input and output queues, which serve as a buffer, and the "ProdQueue", in which products are processed. A queue can serve as both an input and output queue, but not as a "ProdQueue" and an input or output queue at the same time. The next level is the position, which serves as the interface to the products. Each position can be reserved in advance by exactly one product. In addition to the queues, the set-up processes and production processes are stored with respective distributions. The machine is also connected to events, past events ("EventLogger"), and various defects and maintenance entities. The exact configuration of a machine is always adapted to the individual use case so that, for example, the number of queues and positions can be flexibly varied. A simplified machine is visualized in Figure 9.

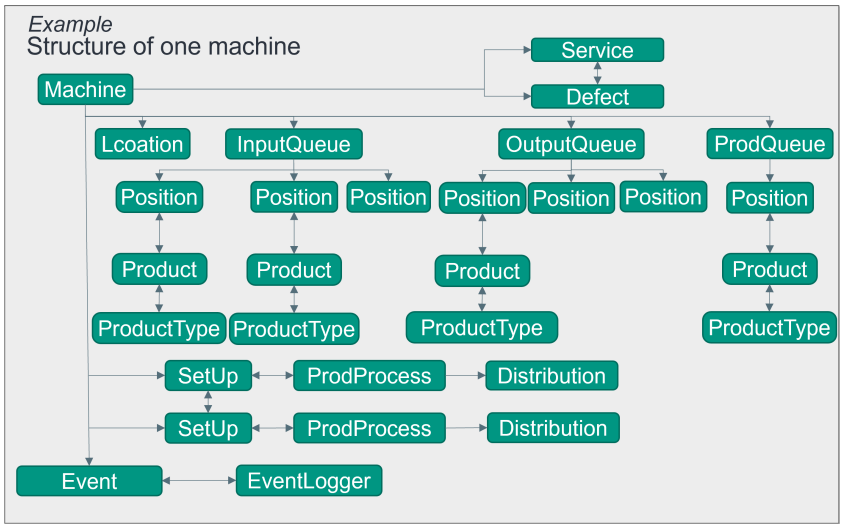

Figure 9. Concept of a machine in OntologySim.

5.2. Event

The states of the simulation are not controlled by state entities and are based solely on events and the information stored within the ontology. The state of an entity, e.g., a machine or transport unit, is defined by the currently valid and connected event. Each event contains information about the start time, duration, and type of event and is connected to the entities, which are influenced by the event. From this information, the state of each entity can be extracted at any time. Events that lie in the future can also be generated. However, the agents and control algorithms do not have access to the future events but only see the current state to preserve comparability in respect to the real world. Creating future events is useful for processing tasks at once in the simulation. For example, if the task is to produce a product on a machine, then two events are created, one to change the machine setup to the new process and the other to feed the product into the machine for processing. This possibility facilitates the implementation of the simulation.

It is always possible to infer in both directions (bidirectionally) in the simulation. This increases flexibility and allows easier access to entities. The disadvantage of this is that more data is stored, which hurts performance. As shown by Reference [17], the performance for compact ontologies is significantly better. For this reason, past events are only made available in one direction, which has the great advantage that, when querying the state and the next steps of a machine, only a few future events need to be queried, and it is not necessary to iterate over all past and future events. Figure 10 illustrates the differences between past and future events and their connectivity to entities.

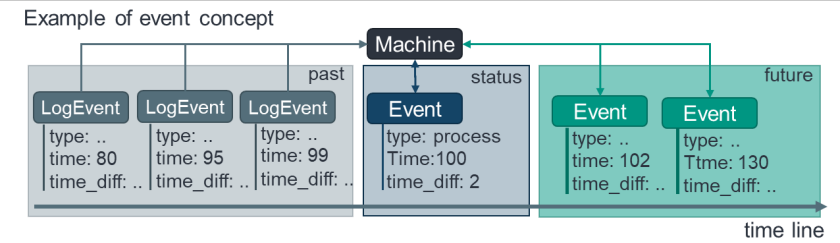

Figure 10. Explanation of the (semi-)bidirectional event-resource mapping.

5.3. Product Type

The mapping of product types and their current state is based on the structure of a Petri net, which is a feasible representation, for instance, in disassembly [62]. The state node in the ontology is equivalent to a place S, and the production process represents transitions T. Each product type has a production plan with start and end nodes. The arrangement and sequence of the process steps can be freely designed, as shown in Figure 11. The individual products always refer to their current state. The Petri net-like structure offers the advantages that different and complex production plans with many potential paths can be stored and iterated through with high performance and that information can be extracted quickly. For example, the calculation of the fastest path, the selection of the process with the shortest machine queue, and the number of production steps still required is possible. The graphic in Figure 11 shows a simple, yet relatively complex, production plan.

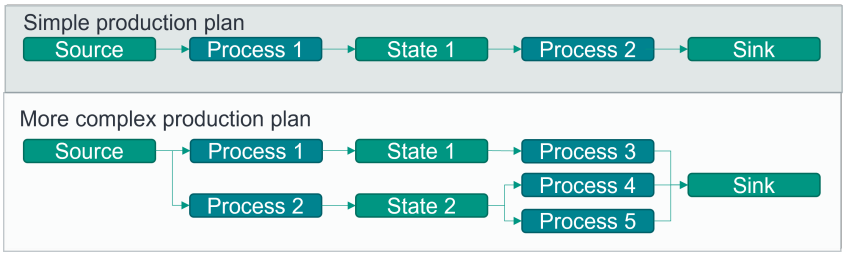

Figure 11. Exemplary concept of product type.

An overview of the classes and relationships and more in-depth information on how the ontology has been implemented in concrete terms are summarized in the ReadTheDocs documentation (https://ontologysim.readthedocs.io, accessed on 30 December 2021) for the OntologySim [63]. The code for the frontend and for the simulation are both published in two seperated github projects (ontologysim_react (https://github.com/larsKiefer/ontologysim_react, accessed on accessed on 27 December 2021) [64], ontologysim (https://github.com/larsKiefer/ontologysim, accessed on 27 December 2021) [65]).

6. Procedure of OntologySim

Based on the simulation steps as shown in the VDI3363 [29] and the adjustments regarding the application of an ontology, the following 4-stage pipeline was designed: Configuration of the simulation, Reasoning/loading of the simulation, Running through the simulation, Logging & storage of KPIs. The individual steps are briefly described below and visualized in Figure 12.

6.1. Configuration of the Simulation (1)

The configuration of the simulation is provided via config-files, owl-files, or an API interface. To simplify the configuration from lines to matrix production, different templates and examples can be used. Additionally, the OntologySim provides support for the configuration by visualization means [64].

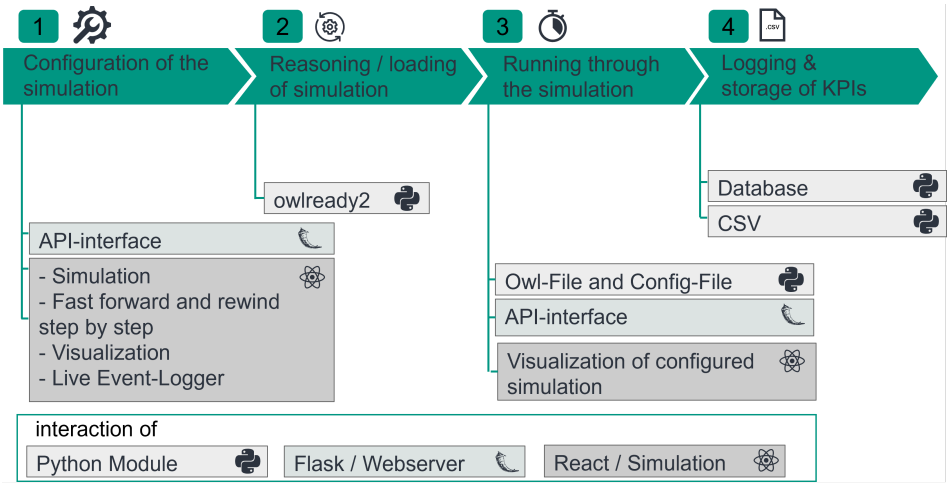

Figure 12. OntologySim pipeline.

6.2. Reasoning/Loading of the Simulation (2)

When starting the simulation, the model is created in Owlready2, and the reasoning is started [65]. For performance and resource efficiency reasons, the reasoning is only carried out at the beginning. All changes are made via wrapper methods so that the model remains consistent, and queries are executed directly on top of objects. This leads to a faster simulation procedure. However, the ability to perform reasoning on this ontology during or after simulation runs remains.

6.3. Running through the Simulation (3)

Running through the simulation offers two possibilities. One is the step-by-step run, and the other is the direct run [64], which is used for a fast simulation execution. Running through the simulation step-by-step allows you to go back and forth between simulation, live display of KPIs and events, and visualization of the production. Furthermore, the simulation can also be accessed via an API [63]; thus, the required data can be retrieved at any time. The goal of the step-by-step process is to better understand and analyze the decisions of the agents and algorithms in order to facilitate better algorithm or machine learning model design to enable further studies into the explainability of such systems [55].

6.4. Logging & Storage of KPIs (4)

After the simulation run, all KPIs and events can be obtained either as CSV files or an SQLite database [64,65]. The KPIs are based on the standard defined by Kang et al. [66]. For each run, up to 22 KPIs for machines, 8 KPIs for transporters, 11 KPIs for products, 1 KPI for queues, and 4 KPIs for the general simulation can be stored. There is one summary per KPI and element (machine, transporter, queue) and one per time interval. The configuration allows the time interval to be changed and KPIs to be added and removed.

7. Technical Description

7.1. Basic Building Block of the Ontologysim

The basis of the OntologySim is the Python library Owlready2. The ontology framework is used to store the data and the status of the production. A more detailed description of how Owlready works can be found in Section 2.1 and References [17,26].

The structure of the OntologySim is shown in Figure 13 below. Around the entire ontology, wrapper methods have been designed to standardize access to the ontology. The search queries are realized via unique IDs and via iteration through the connections. This approach has high-speed advantages over SPARQL and SWRL (Semantic Web Rule Language). Building on top of the wrapper methods, simulation, loggers, agents, and KPI modules are implemented. These modules form the basis for ontology-based simulation and contain, in addition to the ontology, the core logic of the simulation. To start the

simulation, either config files or owl files are loaded or a request to the webserver is sent. Nevertheless, the wrapper methods are generalized, so that only the config or owl is required to instantiate a simulation run. The agent integration is structured in such a way that standard agents, such as FIFO, Shortest Queue, etc., are pre-implemented, and their programmed agents can be easily created via a predefined interface.

Based on the basic simulation, a web server is integrated, which enables more targeted access to simulation data and lays the foundation for the visualization, as shown in Figure 13. The designed API interface is implemented with Flask, which is a stateful service due to the discrete running simulations. The basic functions of the API are to call KPIs, log data, and create and configure simulations. Together with the simulation module and the web server, this forms the backend of the software structure [64,65]. A frontend Framework, here React, is used for the visualization of the simulation. The data is retrieved from the Flask web server using Ajax calls and stored in the Redux Store. The Redux Store serves as the basis for displaying KPI diagrams, event logger tables, and simulations. The connection between frontend and backend is enabled using AJAX calls.

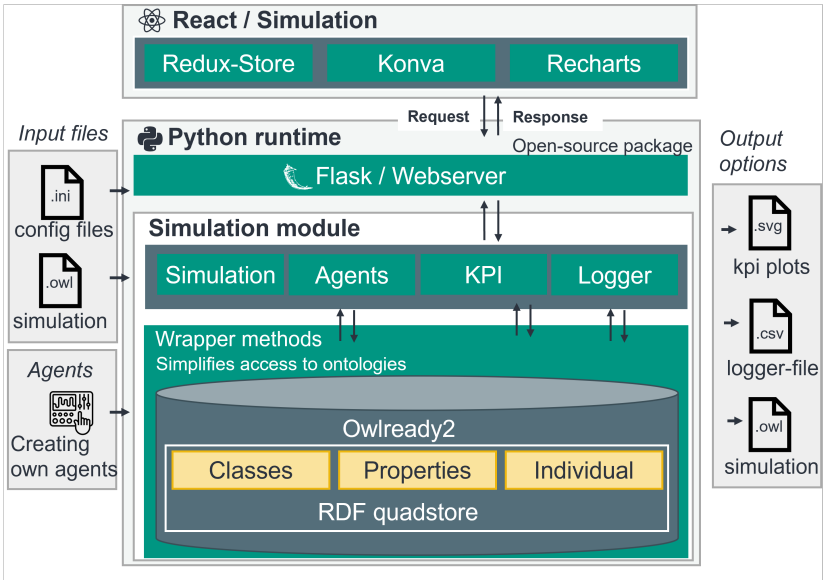

Figure 13. OntologySim structure.

7.2. Visualization

The website (Frontend) visualizes the information available during and after the simulation. In the following, the three most elementary pages are presented in more detail: (1) simulation with back and forward data, hover functions for more information (Figure 14); (2) dashboard and charts with KPIs (Figure 15); and (3) event logging including filtering and sorting of events (Figure 16).

The following elements are displayed in the graphic visualization of the simulation: Queue, Process Queue, Machine, and Transporter. In addition, the last event is displayed for the transporter and the machine. To be able to distinguish individual products, various product types are colored differently, and the progress is symbolized by the fill level of the circle and hovering over the product provides further information, such as queue time and the start of production. Furthermore, the visualization makes it possible to go back to steps in the production to better understand the agents' decisions.

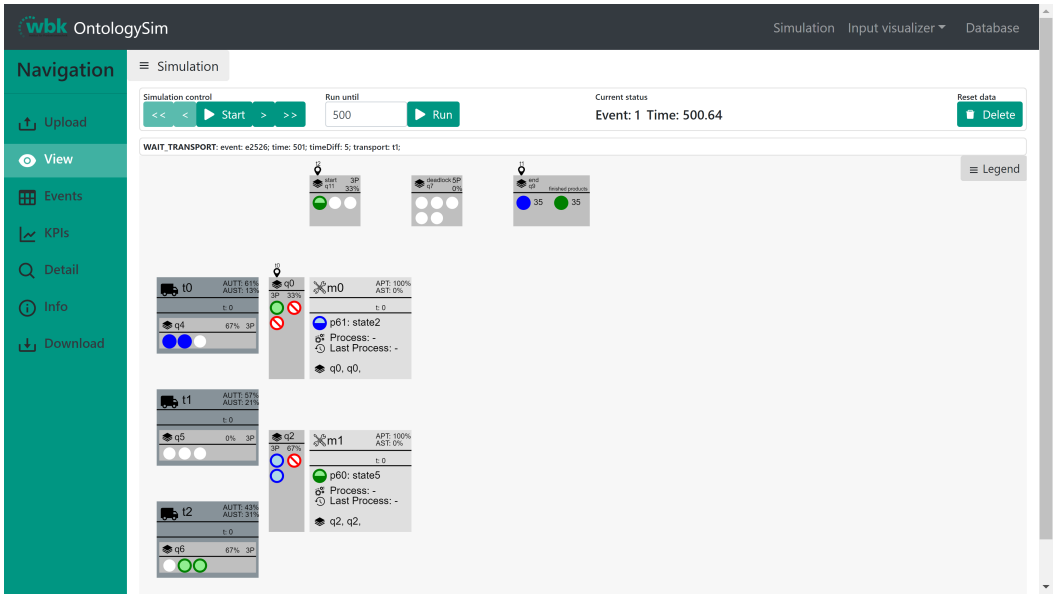

Figure 14. Simulation with back and forward data, hover functions for more information.

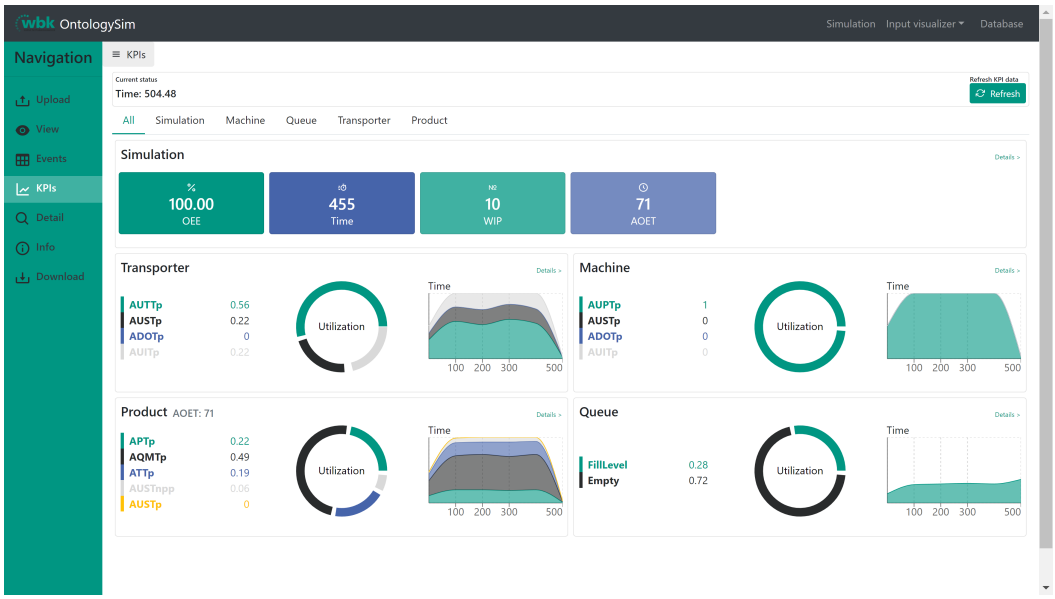

Figure 15. KPI Dashboard of a simulation run.

The KPI overview enables quick analysis of the production. For this purpose, the summarized value and the course of the KPIs are displayed during the simulation. Selected KPIs are visualized graphically, and the remaining KPIs are displayed in tabular form, as shown in Figure 15.

The event logging page allows the display of all past events and the filtering and sorting of events, as exemplified in Figure 16. For example, this enables to track the path of individual products exactly or to display only one product type on a machine.

All in all, the proposed framework is adaptable, as the strict division of the simulation core and visualization allows simple integration and adaption with existing systems and requirements. Thus, the overview is as presented in Figures 14–16.

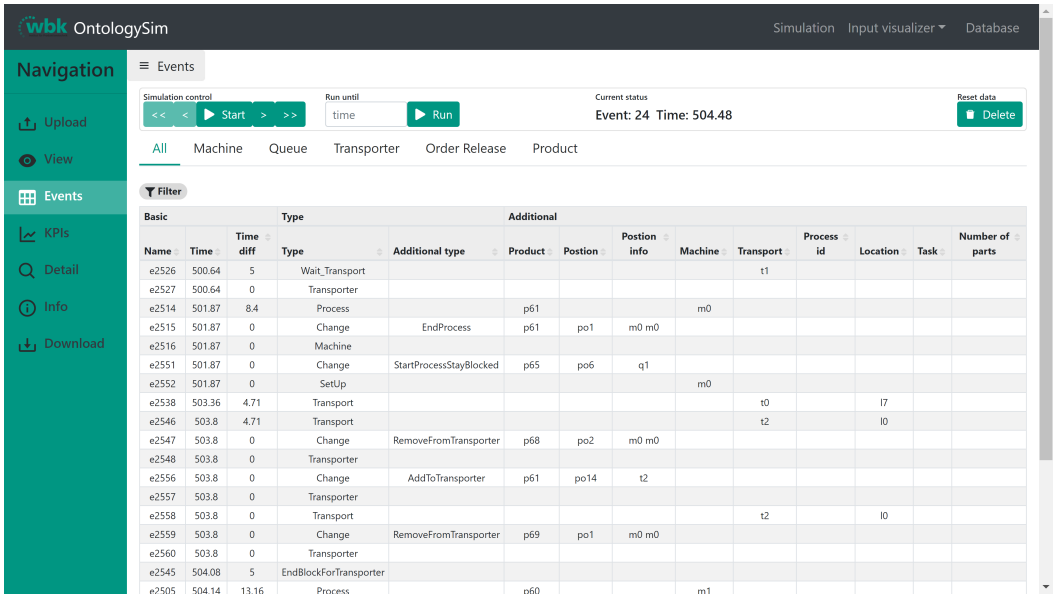

Figure 16. Event logger of simulation.

8. Discussion

Regarding the aim of "[presenting] a combination of ontology with manufacturing simulation, which is available as an open-source solution to the general public and is distinguished from previous solutions by its flexibility and storability", the proposed OntologySim satisfies all requirements outlined in the literature review in Section 3. As shown in Table 3, the OntologySim uses the underlying ontology (or several ontologies) both as a schema and as the underlying simulation storage and core. Thereby, it provides an application framework with the ability to support multi-agent systems and interventions. The latter is crucial and enabled by the truly ontology-based schema and storage and, thus, enables the OntologySim to achieve Ontology Integration Level 5.

As explained in Table 4, OntologySim can be compared to existing simulation frameworks in terms providing a visualization, changeability via GUI and integration of KPI calculation or flexibility. The latter is based on the ontology simulation core approach and, hence, additionally enables changeability of the simulation and interventions beyond the state-of-the-art. The challenge in developing ontology-based simulations, however, is to achieve high performance in terms of execution speed. The OntologySim is slower compared to SimPy, Anylogic, and other commercially available simulations. Slower, although massive, speed improvements have been achieved by avoiding SPARQL and SWRL rules. Nevertheless, further speed improvements are required to ensure the applicability in much larger systems and, thus, to enable more use cases the benefits of an OntologySim-enabled real digital twin. Furthermore, the implemented web application is not a stateless API because of the state storage in the ontology. In today's web applications, stateless APIs are used to enable multi-user operation. Hence, this multi-user operation should be included and examined in the application for further follow-up studies. Nevertheless, the provided OntologySim as an Ontology Integration Level 5 framework enables the application of truly interlocked digital twin, the analysis of "what if" situations in up-to-the-minute simulations, and the convenience for extendability of an open source software.

Table 3. OntologySim classification according to the literature review.

	Ontology System								Real World	Simulation		
Approach by	**Ontology Schema**	**Multi Agent System**	**Multiple Ontology**	**Ontology as Data Storage**	**Application Framework**	**External Data Source**	**Cyber-Physical-System (CPS)**	**Manufacturing Execution System (MES)**	**Ontology Integration Level**	**Intervention**	**Visualization**	**Main Goal**
Proposed Implementation as Fully ontology-based simulations												
OntologySim [64,65]	●	●	●	●	●	○	○	○	5	●	●	Simulation

Legend: ● regarded ◐ rudimentary regarded ○ not regarded

Table 4. Comparison of simulation application.

	Applicability				Availability	Modularity		
Simulation	**Visualization**	**GUI Changeability**	**Speed**	**KPI Standards**	**Open Source**	**Intervention**	**Changeability**	**Flexibility**
SimPy [54]	○	◐	●	◐	●	○	○	◐
AnyLogic [56]	●	◐	●	◐	○	○	○	◐
Siemens PLM	●	◐	●	◐	○	○	○	◐
OntologySim	●	◐	◐	●	●	●	●	●

Legend: ● implemented ◐ partly implemented ○ not possible

Despite the previously described advantages and approaches to circumvent the disadvantages of an ontology-based production simulation, in general, detrimental issues are as follows: Simulation speed is extenuated due to numerous knowledge graph or ontology queries. While queries, for instance, with SPARQL, are more flexible than existing frameworks, they are yet typically more complex to apply than semi-graphical GUIs. The direct integration into existing MES or ERP systems is not (yet) regarded.

9. Summary and Outlook

In order to enable fully autonomous digital twins that interact with real world entities and allow the digital representation of changing and flexible production systems, an ontology-based simulation model, the OntologySim, is presented. The OntologySim is an open-source fully ontology-based event-discrete simulation, which has high flexibility and modularity due to the developed ontology schema and the defined interfaces. Due to the fully ontology-based approach, it is possible to change and save simulations during runtime. The designed user interface allows a detailed analysis of the agents using KPIs, event, and simulation display. Since OntologySim is published as open source (AGPL-3.0 License), we hope to contribute to the growing collaboration and exchange in the production and simulation community.

Follow up research shall continue the development to enable to join and separate products. This would offer new possibilities to better map production processes in the industry and likewise enable portfolio external products and their influences on the production system to be analyzed [33]. Furthermore, work on the OntologySim is ongoing and feedback from the community is being incorporated. To further close the research gap as outlined before, experimental studies with real world use cases shall be continued to be conducted. Last, but not least, the advantages provided by the ontology core can be researched with the integration of autonomous production control, understandable reinforcement learning [55], and novel production planning approaches that make use of the available real-time data and experimentation ability in further research projects.

Author Contributions: Conceptualization, M.C.M., L.K., A.K. and L.G.; methodology, M.C.M., L.K., A.K. and L.G.; software, L.K. and M.C.M.; validation, L.K. and M.C.M.; formal analysis, L.K. and M.C.M.; investigation, L.K. and M.C.M.; resources, M.C.M., A.K. and L.G.; data curation, L.K. and M.C.M.; writing—original draft preparation, L.K. and M.C.M.; writing—review and editing, M.C.M., L.K., A.K. and L.G.; visualization, L.K. and M.C.M.; supervision,A.K. and L.G.; project administration, M.C.M..; funding acquisition, M.C.M., A.K. and L.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research work was undertaken in the context of DIGIMAN4.0 project ("DIGItal MANufacturing Technologies for Zero-defect Industry 4.0 Production", http://www.digiman4-0.mek.dtu.dk/, accessed on 27 December 2021). DIGIMAN4.0 is a European Training Network supported by Horizon 2020, the EU Framework Programme for Research and Innovation (Project ID: 814225).

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Nee, A.Y.; Ong, S.; Chryssolouris, G.; Mourtzis, D. Augmented reality applications in design and manufacturing. *CIRP Ann.* **2012**, *61*, 657–679.
2. Duray, R.; Ward, P.T.; Milligan, G.W.; Berry, W.L. Approaches to mass customization: Configurations and empirical validation. *J. Oper. Manag.* **2000**, *18*, 605–625.
3. May, M.C.; Overbeck, L.; Wurster, M.; Kuhnle, A.; Lanza, G. Foresighted digital twin for situational agent selection in production control. *Procedia CIRP* **2021**, *99*, 27–32.
4. Mourtzis, D. Simulation in the design and operation of manufacturing systems: state of the art and new trends. *Int. J. Prod. Res.* **2020**, *58*, 1927–1949.
5. Durán-Muñoz, I.; Bautista-Zambrana, M.R. Applying ontologies to terminology: Advantages and disadvantages. *Hermes-J. Lang. Commun. Bus.* **2013**, pp. 65–77.
6. Cheng, H.; Zeng, P.; Xue, L.; Shi, Z.; Wang, P.; Yu, H. Manufacturing ontology development based on Industry 4.0 demonstration production line. In Proceedings of the 2016 Third International Conference on Trustworthy Systems and Their Applications (TSA), Wuhan, China, 18–22 September 2016; pp. 42–47.
7. Lamy, J.B. Owlready: Ontology-oriented programming in Python with automatic classification and high level constructs for biomedical ontologies. *Artif. Intell. Med.* **2017**, *80*, 11–28.
8. Gruber, T.R. Toward principles for the design of ontologies used for knowledge sharing? *Int. J. -Hum.-Comput. Stud.* **1995**, *43*, 907–928.
9. Shamsfard, M.; Barforoush, A.A. Learning ontologies from natural language texts. *Int. J. -Hum.-Comput. Stud.* **2004**, *60*, 17–63.
10. Knublauch, H.; Oberle, D.; Tetlow, P.; Wallace, E.; Pan, J.; Uschold, M. A semantic web primer for object-oriented software developers. In *W3c Working Group Note W3C*; 2006
11. Wand, Y.; Weber, R. An ontological model of an information system. *IEEE Trans. Softw. Eng.* **1990**, *16*, 1282–1292.
12. Gruninger, Michael, L.J. Ontology: Applications and design. *Commun. ACM*, *45*, 39–41.
13. Berners-Lee, T.; Hendler, J.; Lassila, O. The semantic web. *Sci. Am.* **2001**, *284*, 34–43.
14. McGuinness, D.L.; Van Harmelen, F.; et al. OWL web ontology language overview. *W3C Recomm.* **2004**, *10*, 2004.
15. Jurisica, I.; Mylopoulos, J.; Yu, E. Ontologies for knowledge management: An information systems perspective. *Knowl. Inf. Syst.* **2004**, *6*, 380–401.
16. Pérez, J.; Arenas, M.; Gutierrez, C. Semantics and Complexity of SPARQL. In *The Semantic Web-ISWC 2006*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 30–43.
17. Lamy, J.B. Ontology-oriented programming for biomedical informatics. In *Transforming Healthcare with the Internet of Things*; IOS Press: Amsterdam, The Netherlands 2016; pp. 64–68.

18. Kalyanpur, A.; Pastor, D.J.; Battle, S.; Padget, J.A. Automatic Mapping of OWL Ontologies into Java. In *SEKE*; Citeseer: Princeton, NJ, USA, 2004; Volume 4, pp. 98–103.
19. Goldman, N.M. Ontology-oriented programming: Static typing for the inconsistent programmer. In *The Semantic Web-ISWC 2003*; Springer: Berlin/Heidelberg, Germany, 2003; pp. 850–865.
20. Seaborne, A.; Manjunath, G.; Bizer, C.; Breslin, J.; Das, S.; Davis, I.; Harris, S.; Idehen, K.; Corby, O.; Kjernsmo, K.; et al. SPARQL/Update: A language for updating RDF graphs. *W3c Memb. Submiss.* **2008**, *15*, 1–13.
21. World Wide Web Consortium. OWL 2 Web Ontology Language Document Overview. 2012 Available online: https://www.w3.org/TR/owl2-overview/ (accessed on 27 December 2021).
22. Knublauch, H.; Fergerson, R.W.; Noy, N.F.; Musen, M.A. The Protégé OWL plugin: An open development environment for semantic web applications. In *The Semantic Web-ISWC 2004*; Springer: Berlin/Heidelberg , Germany, 2004; pp. 229–243.
23. Kalyanpur, A.; Parsia, B.; Sirin, E.; Grau, B.C.; Hendler, J. Swoop: A web ontology editing browser. *J. Web Semant.* **2006**, *4*, 144–153.
24. Haase, P.; Lewen, H.; Studer, R.; Tran, D.T.; Erdmann, M.; d'Aquin, M.; Motta, E. The Neon Ontology Engineering Toolkit. 2008. Webpage: http://neon-toolkit.org/wiki/Main_Page.html (accessed on 27 December 2021).
25. Horridge, M.; Bechhofer, S. The owl api: A java api for owl ontologies. *Semant. Web* **2011**, *2*, 11–21.
26. Jean-Baptiste, L. Constructs, restrictions, and class properties. In *Ontologies with Python*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 135–156.
27. Adel, A.Z.; Zebari, S.; Jacksi, K. Football Ontology Construction using Oriented Programming. *J. Appl. Sci. Technol. Trends* **2020**, *1*, 24–30.
28. Liu, C.; Jiang, P.; Jiang, W. Web-based digital twin modeling and remote control of cyber-physical production systems. *Robot. -Comput.-Integr. Manuf.* **2020**, *64*, 101956.
29. Ingenieure, V.D. *VDI 3633 Simulation von Logistik-, Materialfluß- und Produktionssytemen—VDI 3633 Entwurf | Begriffsdefinitionen*; Beuth: Düsseldorf, Germany, 1996.
30. HrdliczNa, V. Leitfaden für Simulationsbenutzer in ProduNtion und LogistiN. *ASIM Mitteilungen* **1997**, *58*.
31. May, M.C.; Kiefer, L.; Kuhnle, A.; Stricker, N.; Lanza, G. Decentralized multi-agent production control through economic model bidding for matrix production systems. *Procedia CIRP* **2021**, *96*, 3–8.
32. Greschke, P. *Matrix-Produktion: Konzept einer Taktunabhängigen Fließfertigung*; BoD–Books on Demand Gmbh: Norderstedt, Germany, 2020.
33. May, M.C.; Schmidt, S.; Kuhnle, A.; Stricker, N.; Lanza, G. Product Generation Module: Automated Production Planning for optimized workload and increased efficiency in Matrix Production Systems. *Procedia CIRP* **2021**, *96*, 45–50.
34. Panetto, H.; Baïna, S.; Morel, G. Mapping the IEC 62264 models onto the Zachman framework for analysing products information traceability: A case study. *J. Intell. Manuf.* **2007**, *18*, 679–698.
35. Wolfswinkel, J.F.; Furtmueller, E.; Wilderom, C.P. Using grounded theory as a method for rigorously reviewing literature. *Eur. J. Inf. Syst.* **2013**, *22*, 45–55.
36. Karageorgos, A.; Mehandjiev, N.; Weichhart, G.; Hämmerle, A. Agent-based optimisation of logistics and production planning. *Eng. Appl. Artif. Intell.* **2003**, *16*, 335–348.
37. Mönch, L.; Stehli, M. An ontology for production control of semiconductor manufacturing processes. In *Multiagent System Technologies*; MATES 2003. Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2003; pp. 156–167.
38. Scholz, J.; Schabus, S. An indoor navigation ontology for production assets in a production environment. In *Geographic Information Science*; GIScience 2014. Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2014; pp. 204–220.
39. Viljanen, K.; Tuominen, J.; Hyvönen, E. Ontology libraries for production use: The Finnish ontology library service ONKI. In *European Semantic Web Conference*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 781–795.
40. Katti, B. Ontology-Based Approach to Decentralized Production Control in the Context of Cloud Manufacturing Execution Systems. Ph.D. Thesis, Technical University Kaiserslautern, Kaiserslautern, Germany, 2020.
41. Terkaj, W.; Pedrielli, G.; Sacco, M. Virtual factory data model. In Proceedings of the Workshop on Ontology and Semantic Web for Manufacturing, Graz, Austria, 24–27 July 2012; pp. 29–43.
42. Silver, G.A.; Miller, J.A.; Hybinette, M.; Baramidze, G.; York, W.S. An ontology for discrete-event modeling and simulation. *Simulation* **2011**, *87*, 747–773.
43. Benjamin, P.; Patki, M.; Mayer, R. Using ontologies for simulation modeling. In Proceedings of the 2006 Winter Simulation Conference, Monterey, CA, USA, 3–6 December 2006; pp. 1151–1159.
44. Du, J.; Jing, H.; Choo, K.K.R.; Sugumaran, V.; Castro-Lacouture, D. An ontology and multi-agent based decision support framework for prefabricated component supply chain. *Inf. Syst. Front.* **2020**, *22*, 1467–1485.
45. Terkaj, W.; Urgo, M. Ontology-based modeling of production systems for design and performance evaluation. In Proceedings of the 2014 12th IEEE International Conference on Industrial Informatics (INDIN), Porto Alegre, Brazil, 27–30 July 2014; pp. 748–753.
46. Terkaj, W.; Tolio, T.; Urgo, M. A virtual factory approach for in situ simulation to support production and maintenance planning. *CIRP Ann.* **2015**, *64*, 451–454.
47. Warden, T.; Porzel, R.; Gehrke, J.D.; Herzog, O.; Langer, H.; Malaka, R. In *Towards Ontology-Based Multiagent Simulations: The Plasma Approach*; ECMS: Kuala, Lumpur, 2010; pp. 50–56.
48. Gurjanov, A.; Zakoldaev, D.; Shukalov, A.; Zharinov, I. The ontology in description of production processes in the Industry 4.0 item designing company. *J. Phys.* **2018**, *1059*, 012010.

49. Fumagalli, L.; Pala, S.; Garetti, M.; Negri, E. Ontology-based modeling of manufacturing and logistics systems for a new MES architecture. In *IFIP International Conference on Advances in Production Management Systems*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 192–200.
50. Mazzola, L.; Kapahnke, P.; Vujic, M.; Klusch, M. In *CDM-Core: A Manufacturing Domain Ontology in OWL2 for Production and Maintenance*; KEOD: Setubal, Portugal, 2016; pp. 136–143.
51. Guizzardi, G.; Wagner, G. Towards an ontological foundation of discrete event simulation. In Proceedings of the 2010 Winter Simulation Conference, Baltimore, MD, USA, 5–8 December 2010; pp. 652–664.
52. Ansari, F.; Khobreh, M.; Seidenberg, U.; Sihn, W. A problem-solving ontology for human-centered cyber physical production systems. *CIRP J. Manuf. Sci. Technol.* **2018**, *22*, 91–106.
53. Chen, R.S.; Tu, M.A. Development of an agent-based system for manufacturing control and coordination with ontology and RFID technology. *Expert Syst. Appl.* **2009**, *36*, 7581–7593.
54. Matloff, N. Introduction to discrete-event simulation and the simpy language. *Dept. Comput. Sci. Univ. Calif. Davis* **2008**, *2*, 1–33.
55. Kuhnle, A.; May, M.C.; Schäfer, L.; Lanza, G. Explainable reinforcement learning in production control of job shop manufacturing system. *Int. J. Prod. Res.* **2021**, 1–23, https://doi.org/10.1080/00207543.2021.1972179.
56. Borshchev, A.; Brailsford, S.; Churilov, L.; Dangerfield, B. Multi-method modelling: AnyLogic. *Discrete-Event Simulation and System Dynamics for Management Decision Making*; Wiley: West Sussex, UK, 2014; pp. 248–279.
57. *ISO 22400-2*; Automation Systems and Integration—Key Performance Indicators (KPIs) for Manufacturing Operations Management—Part 2: Definitions and Descriptions. International Organization for Standardization: Geneva, Switzeralnd, 2014.
58. Overbeck, L.; Hugues, A.; May, M.C.; Kuhnle, A.; Lanza, G. Reinforcement Learning Based Production Control of Semi-automated Manufacturing Systems. *Procedia CIRP* **2021**, *103*, 170–175.
59. May, M.C.; Maucher, S.; Holzer, A.; Kuhnle, A.; Lanza, G. Data analytics for time constraint adherence prediction in a semiconductor manufacturing use-case. *Procedia CIRP* **2021**, *100*, 49–54.
60. May, M.C.; Behnen, L.; Holzer, A.; Kuhnle, A.; Lanza, G. Multi-variate time-series for time constraint adherence prediction in complex job shops. *Procedia CIRP* **2021**, *103*, 55–60.
61. Kandler, M.; May, M.C.; Kurtz, J.; Kuhnle, A.; Lanza, G. Development of a Human-Centered Implementation Strategy for Industry 4.0 Exemplified by Digital Shopfloor Management. In *Towards Sustainable Customization: Bridging Smart Products and Manufacturing Systems*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 738–745.
62. Wurster, M.; Michel, M.; May, M.C.; Kuhnle, A.; Stricker, N.; Lanza, G. Modelling and condition-based control of a flexible and hybrid disassembly system with manual and autonomous workstations using reinforcement learning. *J. Intell. Manuf.* **2022**, *33*, 575–591
63. Kiefer, L.; May, M.C. Read the Docs: OntologySim. 2021. Available online: https://ontologysim.readthedocs.io (accessed on 30 December 2021).
64. Kiefer, L.; May, M.C. OntologySim_react. 2021. Available online: https://github.com/larsKiefer/ontologysim_react (accessed on 27 December 2021).
65. Kiefer, L.; May, M.C. OntologySim. 2021. Available online: https://github.com/larsKiefer/ontologysim (accessed ion 27 Decemebr 2021).
66. Kang, N.; Zhao, C.; Li, J.; Horst, J.A. A Hierarchical structure of key performance indicators for operation management and continuous improvement in production systems. *Int. J. Prod. Res.* **2016**, *54*, 6333–6350.