Journal of Manufacturing Systems 68 (2023) 270–288

Contents lists available at ScienceDirect

Journal of Manufacturing Systems

journal homepage: www.elsevier.com/locate/jmansys

Technical paper

An Ontology-based Engineering system to support aircraft manufacturing system design

Rebeca Arista [a,c], Xiaochen Zheng [b,*], Jinzhi Lu [b], Fernando Mas [a,d]

[a] University of Seville, Seville, 41092, Spain
[b] École polytechnique fédérale de Lausanne, Lausanne, 1015, Switzerland
[c] Airbus SAS, Blagnac, 31700, France
[d] M&M Group, Cádiz, 11500, Spain

ARTICLE INFO

Keywords:
Ontology-based Engineering
Decision support
Ontology
Knowledge graph
Aircraft manufacturing system
Knowledge management
Cognitive Digital Twin

ABSTRACT

During the conceptual design phase of an aircraft manufacturing system, different industrial scenarios need to be evaluated against performance indicators in a collaborative engineering process. Domain experts' knowledge and the motivations for decision-making is a crucial asset for enterprises which is challenging to be captured and capitalised. Ontology-based Engineering (OBE) systems emerge as a new generation of Knowledge-based Engineering techniques with advancements of ontology engineering methods and computer science technologies. Ontologies enable to capture both explicit and implicit domain knowledge from historical records and domain experts. These Ontology-based Engineering systems can stand highly complex collaborative design processes involving multidisciplinary stakeholders and various digital tools. This paper proposes a tradespace framework with Ontology-based Engineering features included on top of existing Model-Based System Engineering and interoperability capabilities. These additional Ontology-based Engineering features reuse formalised knowledge via knowledge graph technologies and generative algorithms, changing the cognitive process from the designer, to an automatic process which generates design alternatives for the designer. The tradespace framework is demonstrated in a case study to design the aircraft fuselage orbital joint process, helping the designer to take better strategic decisions at conceptual phase and proving to be an advantageous paradigm for the design process.

1. Introduction

An aircraft manufacturing system design at concept phase starts by considering the concept product to be manufactured and the manufacturing system requirements for market competitiveness. With that, multiple scenarios are analysed and optimised against performance indicators in a collaborative engineering process with the product design, making key engineering decisions regarding the product functional design, manufacturing technologies to be used, product breakdown, assembly sequences, logistic plans, industrial resources to reuse, among others [1,2]. After that, several product and process design stages are carried out as described in the product development plan [3].

Modelling and simulation techniques are commonly used to analyse the different scenarios to provide decision support [4,5]. However, lack of interoperable tools hinders full trade-off analysis, resulting in partial assessments, local optimisations and decisions. It decreases the manufacturing system performance during its complete lifecycle. On another hand, the design process relies on the designers' knowledge [6] which is difficult to collect or capitalise due to limited methods and tools. Only part of the design process knowledge is captured explicitly using different documentation approaches and very little information persists from one design to another. Designers use modelling and simulation tools to define an imaginable number of relevant scenarios and carry out decisions based on their assessments and experience.

The combination of Ontology-based Engineering (OBE) techniques and interoperable digital design tools can improve the manufacturing system design by capturing and reusing domain knowledge for design automation [7]. These systems, including Model-Based System Engineering (MBSE) techniques, provide means for generative design and trade-off analysis, as well as complex modelling and simulations, thus to propose solutions to the designers and support decisions to get the best performance and flexible systems.

According to different definitions, meanings and fields of applicability, the term 'ontology' could be considered as: a classification, categorisation, and type of entities of a domain [8]; an exhaustive list of all that exist in a domain [9]; or a model that defines some aspects of

* Corresponding author.
E-mail address: xiaochen.zheng@epfl.ch (X. Zheng).

https://doi.org/10.1016/j.jmsy.2023.02.012
Received 20 September 2022; Received in revised form 3 February 2023; Accepted 20 February 2023
Available online 13 April 2023

Abbreviations

3LM	Three-Layer Model
BFO	Basic Formal Ontology
CDT	Cognitive Digital Twin
CPM	Core Product Model
DAG	Directed Acyclic Graph
DES	Discrete Event Simulation
DOLCE	Descriptive Ontology for Linguistic and Cognitive Engineering
DPDM	Distributed Product Data Management
FAL	Final Assembly Line
GOPPRRE	Graph, Object, Point, Property, Role, Relationship and Extension
iDMU	Industrial Digital Mock Up
IOF	Industrial Ontologies Foundry
KARMA	Kombination of ARchitecture Model specifiCAtion
KBE	Knowledge-based Engineering
LFT	Light Flex Track robot
M-SysML	Manufacturing Systems Modelling Language
MASON	MAnufacturing's Semantics ONtology
MBSE	Model-Based System Engineering
MfM	Models for Manufacturing
MOFLP	Mission, Operation, Function, Logic, Physic
MORFLP	Mission, Operation, Requirement, Function, Logic, Physic
MSO	Manufacturing System Ontology
OAM	Open Assembly Model
OBE	Ontology-based Engineering
PDP	Product Development Process
PLM	Product Lifecycle Management
PRONTO	PRoduct ONTOlogy
PSRL	Product Semantic Representation Language
RMPFQ	Resource, Material, Processes, Functions/Features, Quality
SOM	Semantic Object Model
STEP	Standard for Exchange of Product model data
XLR	eXtra Long Range

a domain [10]. Focusing on the ontologies applied to computer science and engineering, the use of ontologies can fall in three categories described next: ontology-based products, ontology-based applications, or ontology-based engineering systems.

Ontology-based products use ontologies applied to technological fields and are centred on the definition of the product. Some examples are: STEP (Standard for Exchange of Product model data) defined in ISO 10303 standard [11]; the Core Product Model (CPM) developed by NIST in 2004 [12] to describe the function, form and behaviour of a product or artefact; OAM (Open Assembly Model) including assembly and tolerances representation [13]; PRoduct ONTOlogy (PRONTO), DPDM (Distributed Product Data Management) and OntoPDM created by the academic community to model complex products [14]; and finally the open and principles-based ontologies created by initiatives like Industrial Ontologies Foundry (IOF) from which other domain dependent or application ontologies can be derived in a modular fashion [15].

Ontology-based applications use ontologies to enable semantic interoperability between product development systems and perform the integration of the information between well known commercial industrial software (ERP, CRM, PDM) more efficiently than the classical XML approach. As examples, Product Semantic Representation Language (PSRL) proposed by Patil et al. [16], an ontology to interoperate between CAD and CAPP systems was proposed by Dartigues et al. [17], and a novel ontology for Product Development Process (PDP) interoperability and information sharing proposed by Szejka et al. [18] which could be considered an approach to link ontology-based products with ontology-based applications.

Finally ontology-based engineering (OBE) systems describe a particular and well-defined domain from different perspectives: classification, interoperability, behaviour, and semantics. OBE systems cover the scopes of ontology-based product and applications adding the knowledge domain by integrating and transitioning from Knowledge-based Engineering (KBE) methodologies, and giving continuity between the KBE concepts. Some examples of OBE systems include: an integrated knowledge-based assembly planning system called "IKAPS" with suitable modelling environments for processing knowledge of various types in assembly design and planning [19]; and the work of Zheng et al. [20] which proposes a KBE system for defining robotic manufacturing system architectures, using an ontological knowledge model and a multi-attribute decision-making model to alleviate the burden on designers in this process and evaluate the architecture alternatives to determine the optimal solution.

Models for Manufacturing (MfM) is a recent OBE methodology [21] to capture manufacturing domain knowledge in ontologies by using different modelling tools. It is derived from MBSE methodology, using models and their associated behavioural abstraction for enabling a more robust engineering definition [22]. OBE systems implement these collaborative engineering tools in an interoperable environment, supporting the Cognitive Digital Twin (CDT) concept to guarantee at the end of the design phase a validated manufacturing system solution for the complete system lifecycle. CDT is an enhanced version of the current digital twin paradigm augmented with certain cognitive capabilities and support to execute autonomous activities; comprises a set of semantically interlinked digital models related to different lifecycle phases of the physical system including its subsystems and components; and evolves continuously with the physical system across the entire lifecycle [23].

This paper proposes a manufacturing system design tradespace framework based on OBE, MBSE and semantic technologies following MfM methodology guidelines, where an application ontology is developed to integrate assembly system domain knowledge, industrial requirements and system architecture model information. The same ontology is later used and enriched with design rules and constraints, to generate design alternatives for automatic assembly process design. The framework is demonstrated in a case study to support the design of the aircraft fuselage orbital joint process, highlighting the benefits of this type of design and decision support tools. The research motivations of this work are:

- to demonstrate the reusability of the explicit knowledge, which was captured in the OBE features of the tradespace framework following the MfM methodology, to automatically generate new design solutions of complex systems such as the aircraft manufacturing system.
- to move the cognitive and modelling process from the designer only, as proposed by current MBSE methodologies, to an automated modelling and decision support process enabling the designer to explore a wider design space of not just pre-conceivable scenarios.
- to propose and demonstrate a tradespace framework with the capacity of enabling trades between different industrial scenarios and design solutions, facilitating ontology-based requirement definition and requirement validation through simulations during manufacturing system design phase.

- to enable the CDT concept by leveraging from the potential of MBSE methods to trade promising design solutions, of semantic web technologies to capture explicit knowledge and enable tools integration, and of OBE features to add the cognitive part of the design and decision-making process.

The next sections of this paper are organised as follows. Section 2 describes the related work of aircraft manufacturing systems design and the use of OBE systems in this design process. Section 3 presents the architecture of the OBE system proposed as a tradespace framework to support the aircraft manufacturing system design. Section 4 covers the case study used to verify the feasibility of the proposed framework and the implementation of the OBE features in it. Section 5 discusses the findings and benefits of the combined capabilities of the tradespace framework. The paper ends in Section 6 with the conclusions and further work.

2. Related work

2.1. Aircraft manufacturing systems design

The aerospace industry was a pioneer in the 90s designing and industrialising aircraft using Concurrent Engineering techniques [24,25]. At that time, the industry pursued new working methods with the need to reduce time-to-market, introducing Product Lifecycle Management (PLM) methods, processes and tools.

The term Concurrent Engineering, also called "Simultaneous Engineering", was used for the first time in the United States in 1989. It is primarily an expression for the ambition to increase the competitiveness by decreasing the lead-time and still optimising quality and cost. The main methodology is to combine product and manufacturing system development. In this way Concurrent Engineering is a label for a development era of manufacturing technology [26]. Collaborative Engineering is a systematic approach to control lifecycle cost, production quality and time to market, by concurrently developing products and their related processes with response to customer expectations. Each decision-making ensures input and evaluation of all lifecycle disciplines, including suppliers [27]. Information Technology is applied to support information exchange where necessary [28].

Design practices, concurrent or not, tend to quickly converge on a solution point from the design space and modify until reaching design objectives [29]. To avoid the problem of selecting the wrong starting point, MBSE methods propose to consider a design space of possible solutions and perform multi-disciplinary analysis and optimisations, in order to gradually narrow and converge to the final solution [30,31].

Similar to an aircraft product, the aircraft manufacturing system has its own design and development process, running in parallel to the product one and interconnected during both systems lifecycles. It is divided in three phases: design, resilient production and dismantle. The design phase has the different maturity gates, creating a unique final deliverable with the product design in a Collaborative Engineering process, following concepts like the Industrial Digital Mock Up (iDMU) described by Mas et al. [32]. Reconfigurability is analysed in detail during the design phase, either to reuse existing resources or to include flexibility in the design of new resources. This allows reconfiguring the industrial resources during the resilient production phase or reuse them in a next development phase.

Remaining challenges to fully introduce collaborative engineering concepts in the aerospace industry lies in methods and tools that can support complex studies from early design considering product's technological complexity, physical dimension and customisation, assembly lines reconfigurability during their long lifecycle, and the manufacturing system resilience, among others. Until now only a partial set of the manufacturing system characteristics are considered in the different collaborative design methods and tools observed in the literature, with local trade-off studies and partial optimisations that do not propose an efficient capitalisation of historical knowledge of the company to support the different domain designers in their complex tasks. Such a multi-domain design process needs to be correctly addressed with special attention to manufacturing performance drivers, helping designers to explore and evaluate not only traditional manufacturing design options but infinite options from where to select the most promising ones.

2.2. Ontology-based engineering implementation in the manufacturing system design

Some of the first OBE systems addressing the manufacturing system design focused on concurrent integrated design environments interfacing knowledge between different CAD and CAM systems. Based on the generic product assembly model, STEP-based strategies and agent concepts are used for agent-based concurrent integration of design and assembly planning. Zha et al. [33] developed a knowledge intensive multi-agent system for cooperative/collaborative assembly modelling and process planning in a distributed design environment. A unified class of knowledge intensive Petri nets is defined using an object-oriented knowledge-based Petri net approach and used as an Artificial Intelligence protocol for knowledge objects cooperation formalisms.

Hunter et al. [34] used MOKA methodology to elicit knowledge using IDEF0 and UML to represent the fixture design process. Fixture functional requirements were defined and formalised considering material structure constraints, process constraints and resource constraints. Links of functional requirements with typical commercial fixture components were defined via tables and rules mapping, and a knowledge-based prototype application was developed to validate the proposal.

Considering mainly the integration of different software tools owing to the interoperability offered by a common data model, Colledani et al. [35] proposed a virtual factory environment providing to all the applications a common language to exchange data. This allowed integrating heterogeneous software tools supporting factory design activities over a common platform. On this platform basis, Terkaj and Urgo [36] presented an ontology-based data model supporting the design and performance evaluation of production systems. The formalisation of the performance history of a production system and its components are used to enrich the performance evaluation.

To support automated process selection in foundry-style manufacturing of military ground vehicles, Melkote [37] developed a digital library and modelling environment, containing the key characteristics and production capabilities of manufacturing processes and associated resources. A Manufacturing Systems Modelling Language (M-SysML) was developed to model the various manufacturing concepts involved. A complex knowledge graph-based approach is used to represent the relevant manufacturing knowledge of six major classes of manufacturing processes including material removal, metal forming, polymer and composite manufacturing, additive manufacturing, joining (welding), assembly (mechanical fastening), and finishing processes. The library is validated using several realistic test cases. In the same case study, Yukish et al. [38] addressed the foundry tradespace configuration and exploration, to assess the existence of a foundry that can fabricate the military ground vehicle, and can choose one option among multiple ones considering a multi-objective of costs and schedule which conflict with each other.

Mas et al. [3] and Ríos, et al. [39] described a knowledge-based application to support conceptual design of aeronautical assembly lines, documenting the process using IDEF0 and proposing knowledge models using UML, to support a Knowledge-based application prototype integrated with CAD/CAM commercial software systems. The authors test and validate this prototype in several Final Assembly Line design case studies, using time and cost indicators to support decision-making. Later, using MfM methodology, Mas et al. [40] created an ontology for aerospace assembly lines in Airbus, with the introduction of a

novel way to characterise the adherence concept in the design process of aerospace assembly lines. Based on this work, Arista et al. [41] defined an ontology capturing the industrial system design knowledge at concept phase, and demonstrating its application in a reconfiguration case study.

Page Risueno and Nagel [42] describe an OBE framework for modelling aircraft production, separating knowledge and its application to improve reuse and maintenance of knowledge. The prototype consists of a knowledge module to formalise production knowledge in OWL, an assessment engine connecting the knowledge stored in the ontologies and the design process, and an inference engine to execute the logic rules in SWRL and select an appropriate production process. The prototype is demonstrated for a stringer production process, and the assembly of a fuselage section. On a similar basis, Meski et al. [43] demonstrate knowledge-based decision support in the aeronautical mechanical machining industry. The authors presented a semantic-driven tradespace framework in [44] where MBSE and interoperability capabilities were used to accelerate the aircraft manufacturing system design, proving the feasibility of linking cross-domain information though an application ontology and helping the designer to perform complex trade-offs among defined design alternatives.

The different ontology-based implementations described above have improved the collaborative engineering process, providing means to design performant products and flexible industrial systems. Nevertheless, domain modelling complexity, the level of effort needed to capture and capitalise knowledge, and remaining technological limitations of ontology based applications, prevent from a wider implementation of OBE tools and techniques to support aircraft products industrialisation.

Motivated by the potential of OBE systems to support the manufacturing system design, and to address the current limitations, this work presents an extension of the tradespace framework presented in [44] with new OBE features that will allow to reuse formalised knowledge and change the cognitive process from the designer to the framework, proposing automatically design alternatives to be traded that could fulfil the design requirements. The architecture of this tradespace framework is described in the next section.

3. Ontology-based engineering system architecture

Focusing on the aerospace industry, the MfM methodology was proposed based on the MBSE methodology to support the generation and management of manufacturing ontologies [21]. The core of the MfM methodology is a three-Layer Model (3LM), consisting of a data layer, an ontology layer and a service layer. It enables OBE systems for knowledge acquisition, semantic processing and engineering support [45]. Based on this three-layer model, a tradespace framework is created to accelerate aircraft manufacturing system design. This framework extended the three-layer model by encapsulating the functions and activities into multiple functional modules. The architecture and key functional modules are depicted in Fig. 1.

The overall architecture and functions of each functional module have been introduced in a previous work [44]. To provide a general understanding of the tradespace framework, the functions of the key modules are briefly introduced as follows:

- The bottom Data layer holds different types of data sources that contain information and knowledge about manufacturing system requirements, historical system information, domain standards and specifications etc. It provides input to the Ontology layer for knowledge capturing and instantiating.
- The top Service layer provides different services to corresponding stakeholders through a set of functional modules including: the Requirement Management Module for requirement definition and requirement tracing; the Architecture Definition Module for industrial system definition and trade-off scenario definition; the Verification Module for design solution evaluation through discrete event simulations and 3D simulations; and the visualisation Module for presenting simulation results and trade-offs among different indicators to the end user for decision-makings.
- The Ontology layer in the middle uses an application ontology to capture domain knowledge from data sources of the Data layer and relevant experts. Meanwhile certain top-level ontologies and middle-level ontologies are adopted as the foundation of the application ontology to enhance its interoperability making it possible to reuse other existing ontologies such as MBSE ontologies and requirement ontologies. By this approach, the application ontology can be used to integrate knowledge about physical manufacturing systems, system design requirements and architecture models etc. An ontology-based knowledge graph database can be constructed to support new system design by instantiating the captured knowledge. For example, during the design phase of a new system, the industrial requirements defined by the Requirement Management module can be fed to the knowledge base and trigger corresponding querying and reasoning algorithms to automatically generate alternative design solutions for the new system. The generated solutions can then be fed to the Architecture Definition Module for creating system architecture models, and to the Verification Module for simulation and evaluation.

The implementation of the Ontology layer functional modules is based on an application ontology that is specifically developed according to the use case. Application ontologies are highly customised and heavily dependent on the corresponding application scenarios. Without a common specification they are difficult to interoperate and reuse. To improve the interoperability and reusability, it is necessary to follow the hierarchical strategy by reusing certain upper-level ontologies, which can be further divided into top-level and middle-level ontologies. A top-level ontology defines a set of general vocabularies that are commonly used across all domains. These vocabularies are properly structured and formally defined according to certain methodology. Examples of popular top-level ontologies include the Basic Formal Ontology (BFO) [46] and the Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) [47]. At a lower level, a middle-level ontology focuses on a certain application domain such as the MASON (MAnufacturing's Semantics ONtology) [48] and the MSO (Manufacturing System Ontology) [49] for manufacturing domain. The IOF [15] is an ongoing initiative aiming to co-create a set of open domain-level ontologies to support the manufacturing for industrial needs and to promote data interoperability. Depending on its scope, a middle-level ontology can be further narrowed down to certain domain and sub-domain ontologies.

The development of application ontology in this study follows the IOF hierarchical principle as shown in Appendix Fig. A.1 [15], and adopts the middle-level IOF-Core ontology as the foundation, which further refers to the BFO as the top-level ontology. IOF-Core contains a set of formally defined vocabularies that can be used for creating application ontologies in the manufacturing domain. The adoption of IOF-Core enables the developed application ontology to interoperate with other application ontologies that are also based on IOF-Core.

The Ontology layer is the core of the proposed tradespace framework. A first version of the framework was developed and presented in [45,50], addressing the interoperability problem of the tradespace system and the information transformation between different modelling languages, as represented in Fig. 2. At this stage, the designer was in charge of modelling the architecture model options using an MBSE methodology, and through the ontology layer the designer was able to automatically generate simulation models of each option, execute and compare its results to support the design decisions.

Later work was performed to enable design automation, collecting design rules and constraints from the designer's experience in the ontology layer. Through them, the tradespace framework now allows

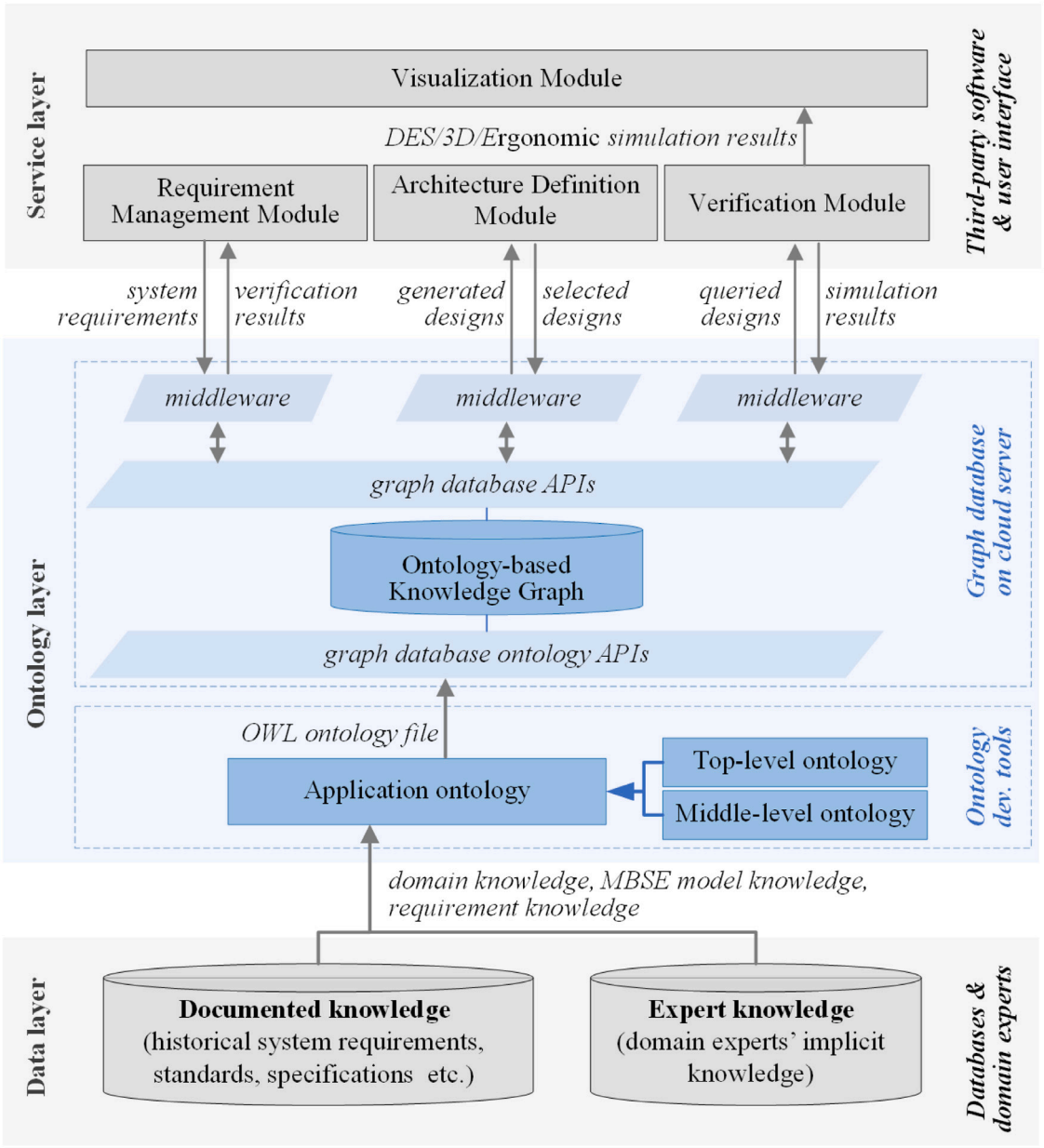

Fig. 1. Architecture and key functional modules of the proposed tradespace framework for manufacturing system design.

the designer to set a design problem boundary and by use of generative design algorithms automatically generate all possible design alternatives, removing the manual MBSE modelling activity where only selected design options were explored and modelled by the designer. The evolution of the tradespace framework into an OBE system, represented in Fig. 3, is supported by the previously existing functions of transformation between different modelling languages, as well as the automatic generation of simulation models, to execute and compare their performance. This wider design space exploration improves the analysis of promising design solutions while supporting the decision making process.

The ontology is developed based on common knowledge extracted from multiple existing assembly systems and is optimised by domain experts. In practical applications, the knowledge capturing process needs deep domain knowledge and is time consuming in many cases, however, once the ontology is developed, it can be reused in different cases. It can provide a common knowledge source to improve interoperability and reduce redundancies and human errors. The developed ontology can also be used as the basis to create a knowledge graph, which further enables the generation algorithm to create all possible design solutions automatically considering different variations.

In addition to domain knowledge, the ontology also enables integrating information about system requirements and architecture models. As shown in Fig. 1, the requirement management module, the architecture definition module and the verification module are all connected to the ontology-based graph database. This enables information consistency and automation of the entire design process from requirement definition, design solution generation, architecture modelling to requirement verification. Compared with the existing MBSE approaches, which is widely used in systems engineering domain, this ontology-based method allows a wider design space exploration, as well as advanced trade-off and decision making capabilities.

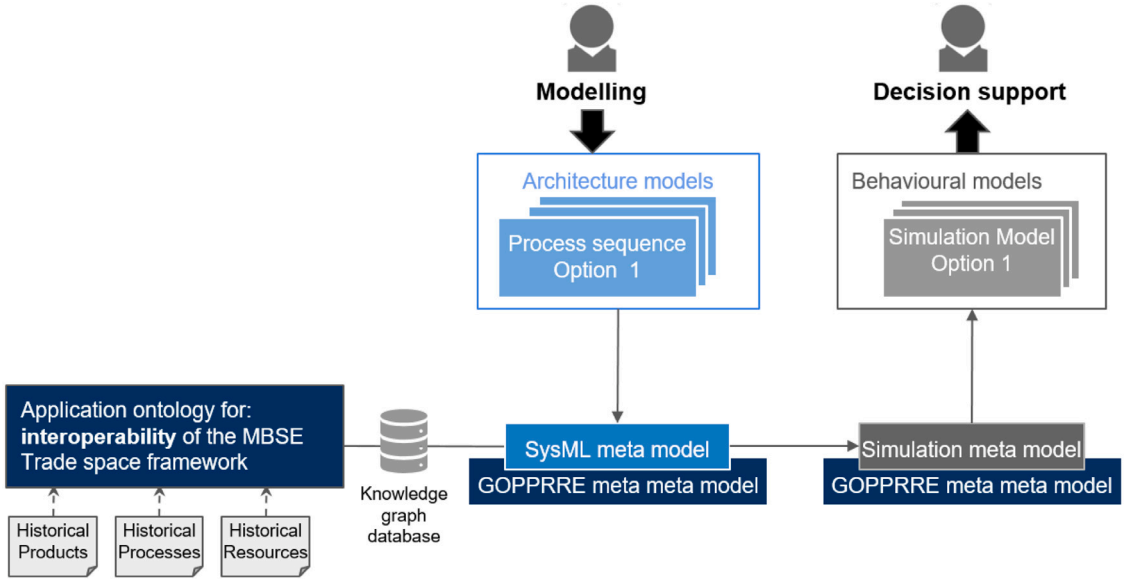

Fig. 2. Functional diagram of the interoperable MBSE tradespace system.

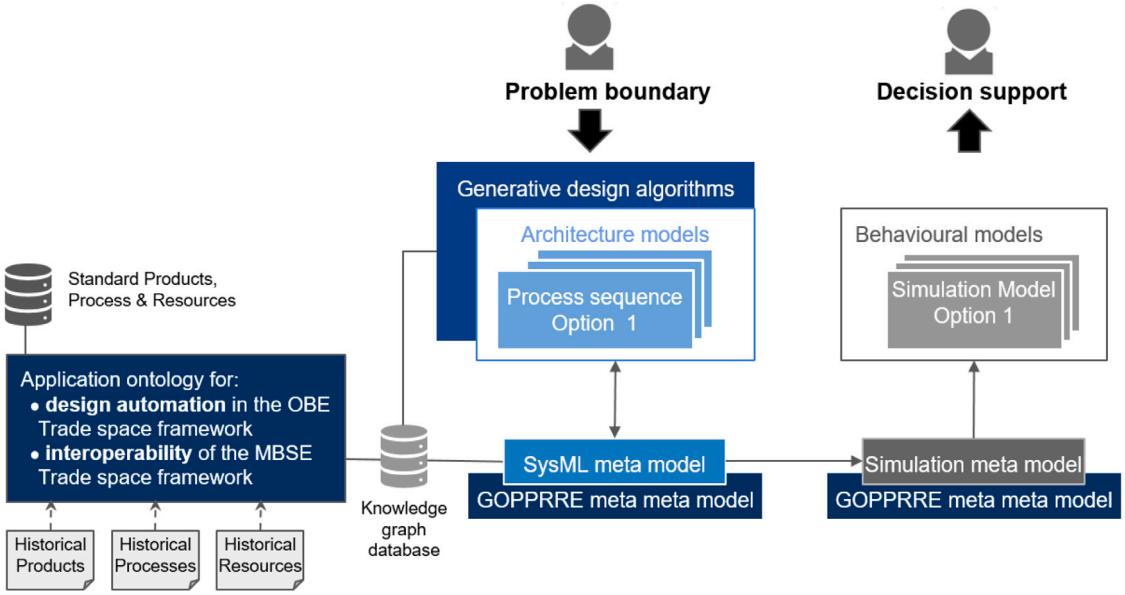

Fig. 3. Functional diagram of the Ontology-based Engineering and interoperable tradespace framework.

This paper focuses on the latest functionalities of the tradespace framework, describing the core capabilities including the application ontology developed, the ontology-based knowledge database construction, the automatic generation of design solutions and their interactions with different functional modules in the Service layer. The framework was validated through a representative case study of an assembly process design for a new aircraft.

4. Case study

To verify the feasibility of the proposed OBE-based design tradespace framework and demonstrate the implementation process, a case study is conducted based on the development of an aircraft fuselage assembly system. This section first introduces the application scenario of the case study; then presents the development of the application ontology and ontology-based knowledge graph; and finally introduce the generative design algorithm and the validation of the generated solutions.

4.1. Application scenario

The use case subject of this work is part of the manufacturing system design for a new aircraft model of the A320 family called "XLR" (eXtra Long Range). More specifically, it focuses on the design of the fuselage orbital joint process, to be performed between the front and rear aircraft fuselage sections through a riveting process, at one of the assembly stations of the aircraft Final Assembly Line (FAL). A representation of this process is shown in Fig. 4.

The fuselage orbital joint process is a critical assembly process in the FAL due to the product design tolerances that need to be defined and respected in this intersection, the accessibility to some areas to perform the joining process operations, and that during this assembly process no other activity can be performed on the aircraft. During the design of this assembly process, the designers need to trade between possible assembly process sequence scenarios, evaluating key performance indicators like process lead time, recurring and non-recurring costs, process automation level, autonomous quality level, among others.

Table 1
Examples of typical operations of the orbital joint process.

ID	Operation name	Rate	Type	Resource[a]	Time	Pre.
02	Set up working environment	Orbital	Manual	2MO, 1SP	10 min	01
03	Set in position Rails and LFT	1/2 Orbital	Auto	2AO, 1LFT, 1SP	10 min	02
04	Camera at stating holes	1/4 Orbital	Auto	1AO, 1LFT, 1SP	15 min	03
05	Drilling orbital 4.8	1/4 Orbital	Auto	1AO, 1LFT, 1SP	125 min	04
10	Drilling template installation	1/4 Orbital	Manual	2MO, 1SP	25 min	02
11	Fixation drilling template suite manual	1/4 Orbital	Manual	2MO, 1DT, 1SP	30 min	10

[a]The full name of the resource items see Table 2.

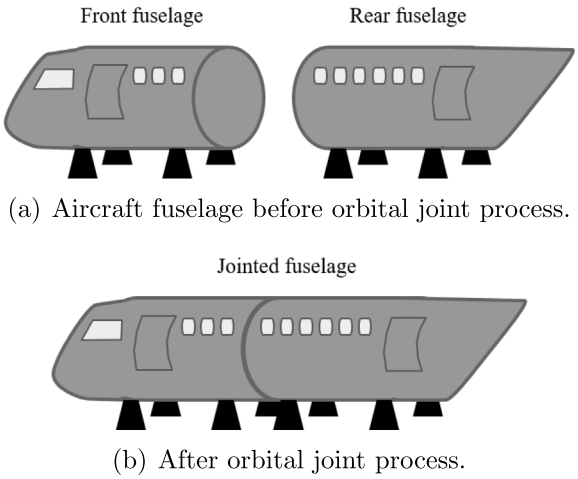

Fig. 4. Representation of the fuselage orbital joint process.

Table 2
Example of resources of orbital joint process.

Name	Abbr.	Cost	Calendar	Qty
Mechanical operator	MO	100 €/h	40 h/week	8
Automation operator	AO	100 €/h	40 h/week	8
Light Flex Track Robot	LFT	75 €/h	24 × 7	2
Station platform	SP	80 €/h	24 × 7	2
Hand drilling machine	HD	10 €/h	24 × 7	6
Drilling template	DT	10 €/h	24 × 7	5

4.1.1. Product, process and resources catalogue

The orbital joint process is composed of a set of operations such as positioning, fastening, drilling, riveting, cleaning and inspection. Each operation requires a certain number of resources such as a station platform, drilling machine, drilling template, workers, etc. The entire joining process can be made in a continuous single sequence, or split in orbit sections (e.g. upper and lower half-orbital or quarter-orbital) that can be done in parallel depending on the technical constraints and available number of resources to perform them.

Standard components of products, processes and resources are defined as catalogues. Specific attention is given to the standard processes defined, including standard parameters values, and the same for standard resources. These components are used as meta-objects to generate the objects of the different design options. It is worth mentioning that the operation details used in this use case are different from the ones used in real production. They have been shortened and anonymised to respect the data privacy regulations of the case owner. Some examples of the involved products, processes and resources are presented in Tables 1 and 2.

Some key operations like drilling and riveting, can be executed manually by mechanical operators or automatically by a robot called Light Flex Track Robot (LFT). Each option requires different resources and time duration, which will impact the final cost and lead time of the entire FAL. The combination of different operation sequences and technical options results in different design solutions for the orbital joint process. Moreover, certain constraints have to be considered, for example, the FLT covers half of the orbit, meaning that once chosen as an automatic option, the corresponding two quarter-orbital processes have to be sequentially executed. One example solution is presented in Fig. 5.

The target of the case study is to create a tradespace system based on the proposed framework and demonstrate it in the use case of the orbital joint process. It should be able to support automatic orbital joint process design and performing trade-offs according to the needs of different stakeholders.

4.2. Application ontology

4.2.1. Application ontology for interoperability of the trade-off system

For this case study, three application ontologies are developed including the assembly system domain knowledge ontology capturing knowledge about existing aircraft assembly systems, requirement ontology capturing knowledge about requirement specifications, and MBSE ontology capturing knowledge about architecture models. They are developed separately by experts from the corresponding domains based on the IOF-Core middle-level ontology and then integrated into one complete application ontology to link the knowledge about physical assembly systems, virtual models and related industrial requirements. The general hierarchical structure of the developed application ontology is depicted in Fig. 6, in which the key vocabularies of IOF-Core and BFO used by the application ontology are presented. The original IOF-Core ontology can be retrieved from the IOF website,[1] and the main classes defined in the application ontology are available on Github[2] in Turtle (.ttl) format.

As shown in Fig. 6, at the domain and application ontology levels, the three parts, i.e. domain knowledge, requirement and MBSE ontologies, are inserted to relevant classes that are defined in IOF-Core and BFO. More details are explained as follows.

- The Requirement ontology captures knowledge of different levels of requirements for the manufacturing system, including the top-level industrial requirements and regulations, functional and non-functional requirements. The obtained requirements can be traced to functions following the RFLP approach (Requirement, Function, Logical, Physical) [51]. Then they are included into a MOFLP model (Mission, Operation, Function, Logic, Physic) [52] by using requirements as links between mission and objectives as well as functions towards a MORFLP model. Some exemplary classes that are defined in the developed Requirement ontology are presented in Appendix Fig. B.1.
- The MBSE ontology aims to formalise a certain MBSE approach with ontology entities including classes, individuals and relationships. In this study. the GOPPRRE ontology [53] is reused together with the KARMA (Kombination of ARchitecture Model specificAtion) modelling language [54]. Meta-models from UPDM

[1] https://www.industrialontologies.org/top-down-wg/.
[2] https://github.com/zhengxiaochen/ontology_aircraft_system.

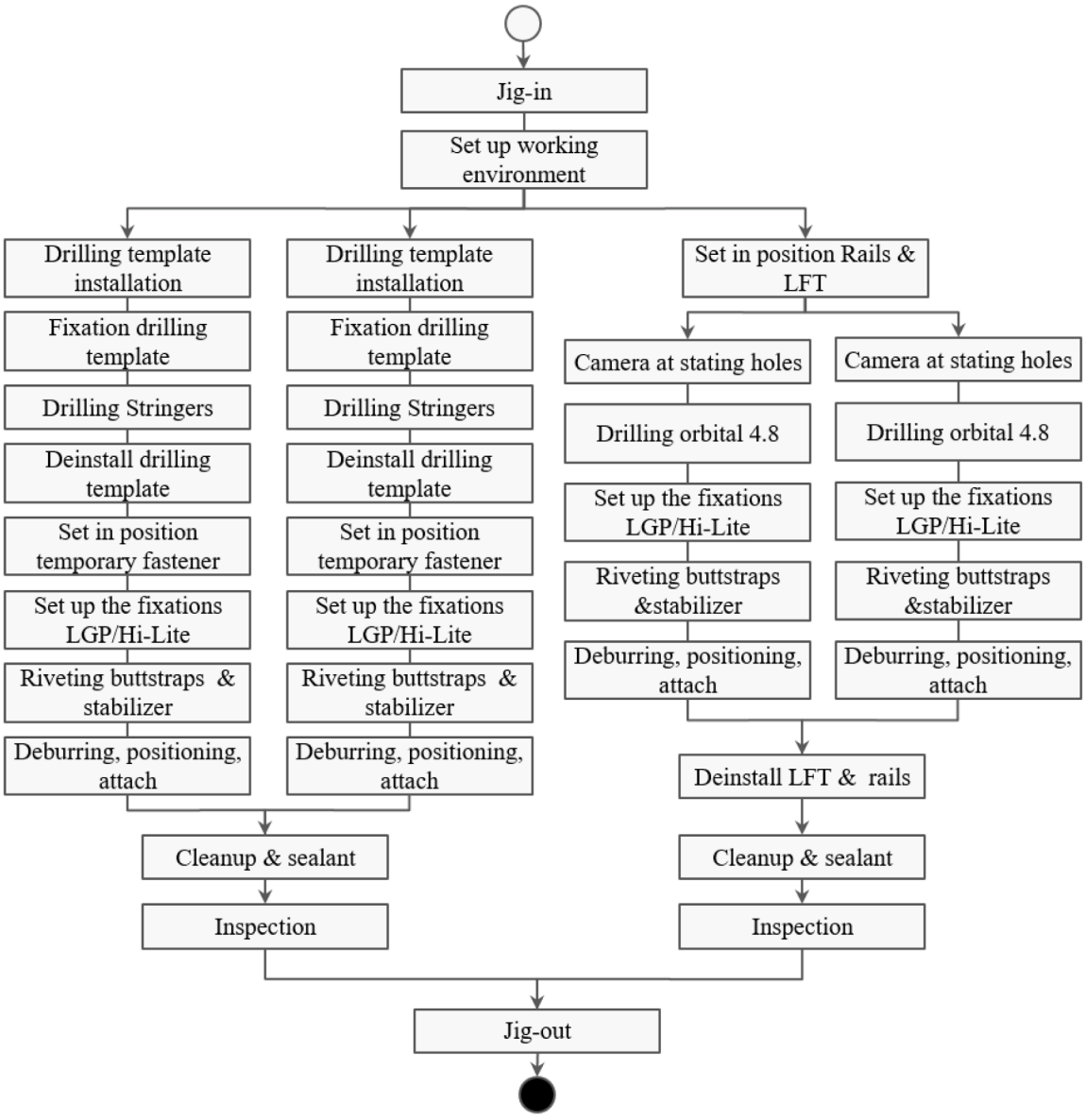

Fig. 5. Example of an orbital joint process sequence.

and SysML specifications for formalising the mission view, operation scenarios and diagrams for formalising the functional views, logic views and physical structure of the assembly processes are developed by using KARMA language. Some exemplary classes that are defined in the developed MBSE ontology are presented in Appendix Fig. B.2.

- The domain knowledge ontology captures knowledge about existing manufacturing systems from historical documented sources, domain experts' empirical knowledge and other public knowledge sources. Some exemplary classes that are defined in the developed domain knowledge ontology are presented in Appendix Fig. B.3. It is the main focus of this paper and its development process is further explained in the following dedicated section.

Aircraft manufacturing systems domain knowledge ontology. The knowledge sources for the domain knowledge ontology include the following three categories:

- The documented knowledge about historical manufacturing systems such as the assembly process specifications, material and resource catalogues, labour and resource cost etc. Vocabulary and relationships can be extracted from these specifications by decomposing each of the processes and operations as listed in Table 1. An example of the decomposition is illustrated in Fig. 7. The operation "Drilling orbital 4.8" is one of the typical assembly "automated" operations for the orbital joint process. It occurs "per 1/4 orbit"; has duration of "125 min"; requires resource "Light Flex Track Robot"; and happens after operation "Camera at stating holes". The operation and resource items are defined as classes in the ontology, and the other attributes are defined as properties of the operation as shown in Fig. 7. It is worth mentioning that the properties in ontology include object properties and data properties. The former link two ontology entities (classes or individuals), and the latter link a data type to an ontology entity.
- Domain experts' knowledge is collected to refine the ontology created based on documented sources. As an example, the available options to perform a certain operation, the typical structure of a process, the classification of the different operations and resources etc. This type of knowledge is usually common sense for domain experts and critical for knowledge reuse. It is challenging

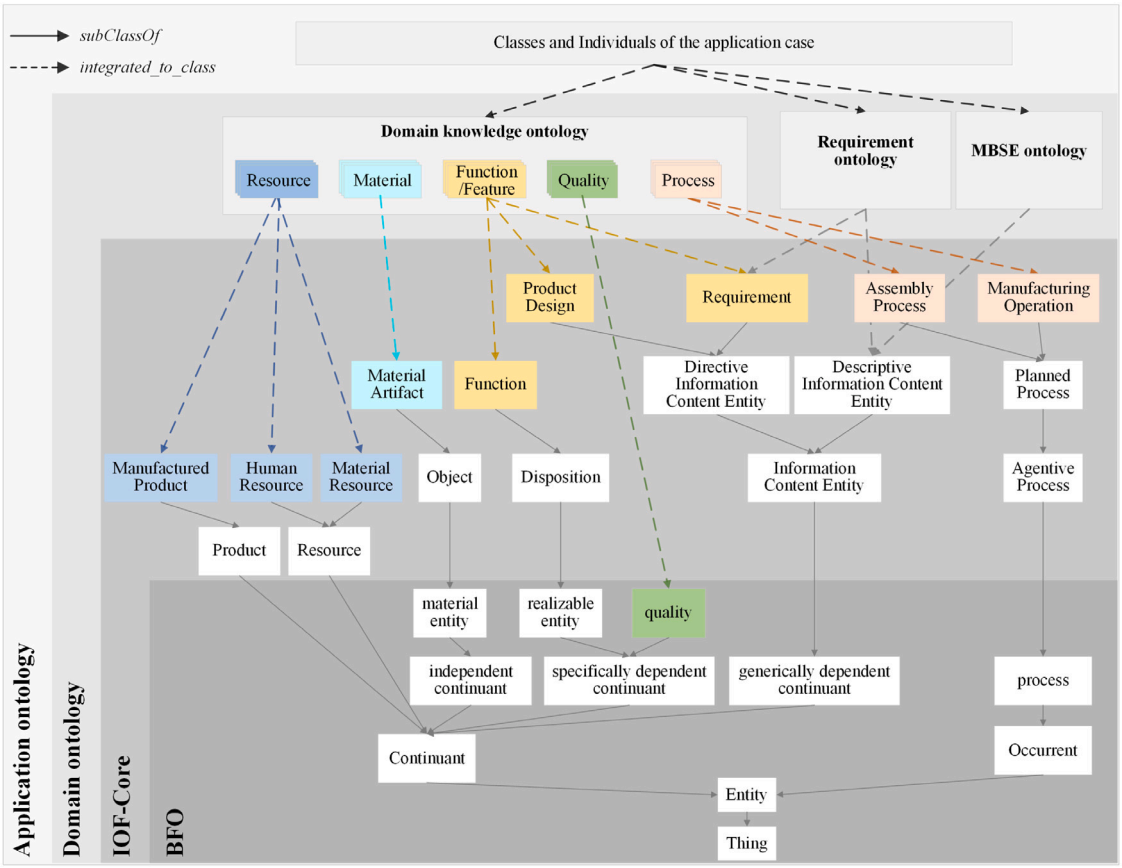

Fig. 6. Hierarchical structure of the application ontology based on IOF-Core and BFO.

to gather such knowledge since there are no efficient tools available. In this study, we mainly used surveys and interviews with multiple iterations.

- Public knowledge is the knowledge collected from public sources of relevant domains. For example, the RMPFQ (Resource, Material, Processes, Functions/Features, Quality) [50,55,56] model is used in this study to support the definition of the relationships between different classes of the ontology. RMPFQ is a quality-oriented semantic model that aims to identify the main influential factors related to product quality during manufacturing processes. More details about this model are introduced in [56]. In this case study, we mainly focus on the Resource, Material, Process and Quality aspects. During a typical assembly process, multiple Materials are assembled to realise certain Quality requirements through a planned Process consisting of a series of operations, which requires relevant manufacturing Resources.

4.2.2. Design process rules and constraints

Some essential knowledge about the targeted assembly process has been stored in the ontology which can be reused to create new solutions. In addition to the information about the operations, such as operation duration, required resources etc., the following rules and constraints are also included in the application ontology:

- The target assembly process joins two fuselages, named Front fuselage and Rear fuselage.
- The assembly process can be divided into upper and lower half-orbital subprocesses, each of which can be further divided into two quarter-orbital subprocesses, as shown in Fig. 8.
- Each resource at the target station has fixed available quantity and cost per use.
- The operations are linked to the assembly process as essential operations if they have no alternatives, such as “Set up working environment”; or as optional operations if they have alternatives, such as “Riveting buttstraps and stabiliser” which can be automated or manual. These relationships can be expressed with triples. For example:
  - “OrbitalJoiningProcess” - “hasSubProcess” - “LowerOrbital Joining”;
  - “OrbitalJoiningProcess” - “hasEssentialOperation” - “Set up working environment”;
  - “LowerOrbitalJoining” - “hasOptionalManualOperation” - “Riveting buttstraps and stabiliser manual”;
  - “Set up working environment” - “requiresResource” - “Station platform”.

These rules and constraints are specifically extracted for this case study. They are reusable in the scope of this case study for different orbital joint process designs of new aircraft programs, but they are not expected to be reused for other cases with different manufacturing processes and different application scenarios. For example, this case study is validated with the application scenario of the Airbus A320-XLR model, and the rules are reusable for the fuselage orbital joint process design of new derivative aircraft models (like the Airbus A350 freighter model) or for new aircraft families which require a fuselage orbital joint. They are added to complement the domain knowledge captured in the application ontology and enable the development of automatic process generation algorithms which is explained in the next sections. It is worth clarifying that most of the rules and constraints are already captured in the application ontology during the ontology development process, for example the sequence of certain operations and the required resource of each operation. These captured rules and constraints are the main foundation for the generative algorithm.

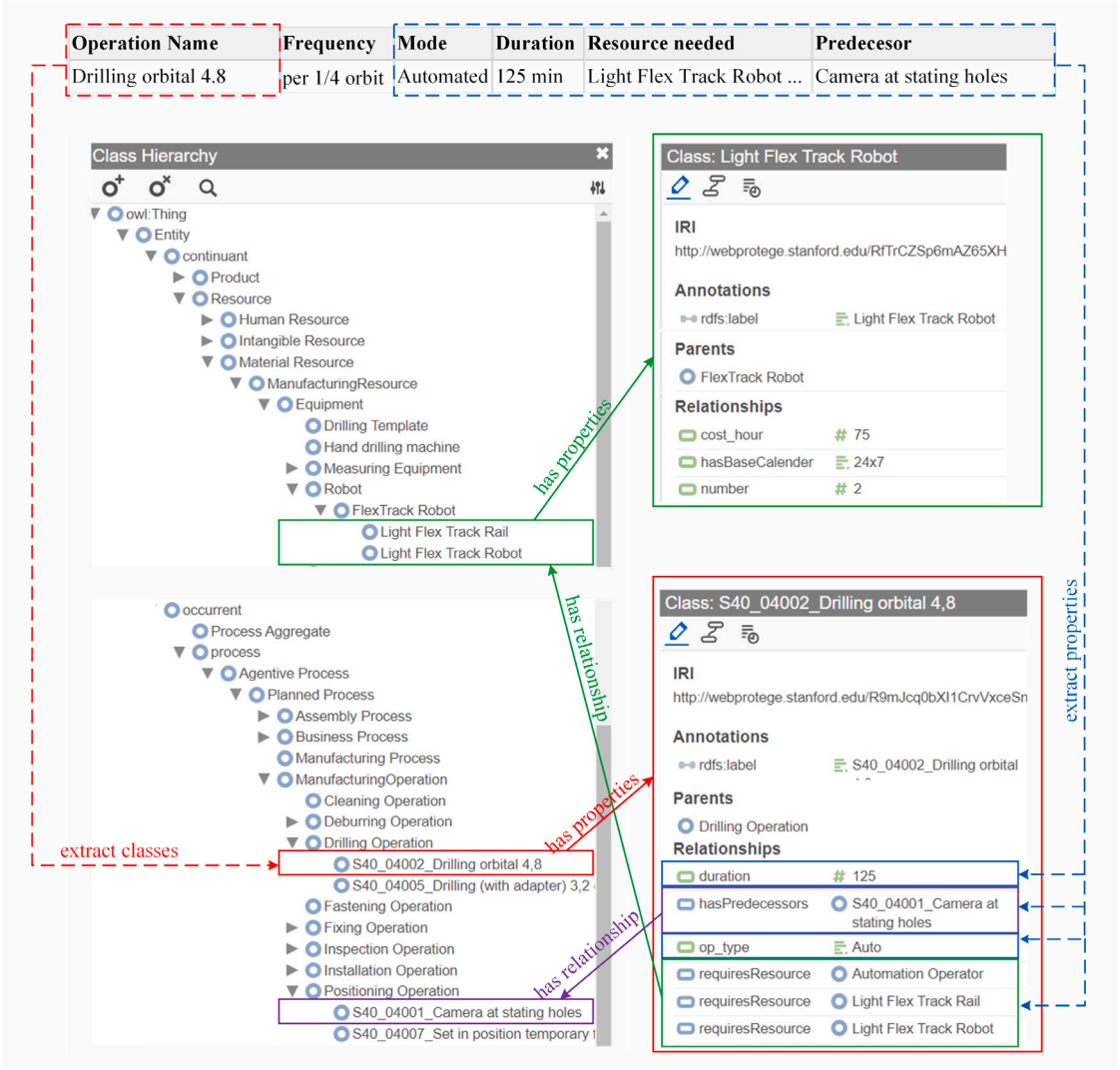

Fig. 7. Example of extracting ontology classes and properties from existing aircraft assembly processes.

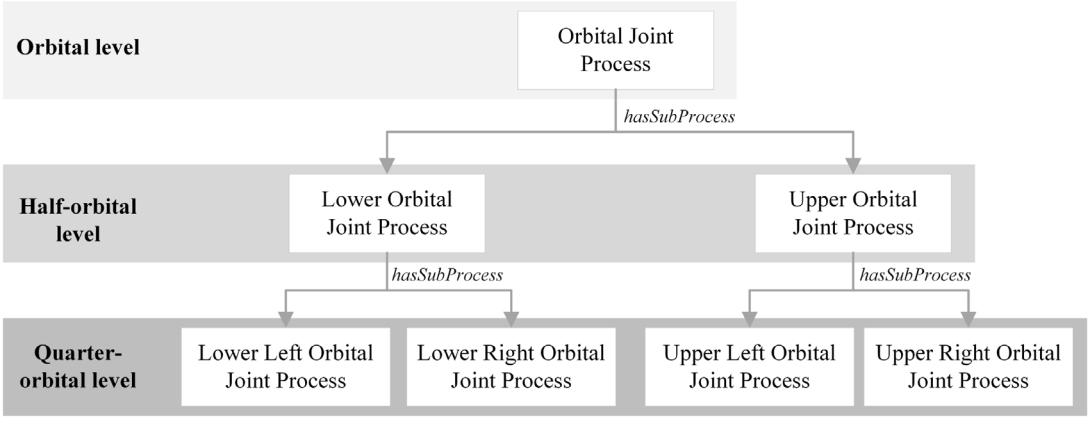

Fig. 8. Hierarchical structure of an orbital joint process.

4.3. Ontology-based knowledge graph database

The application ontology was developed using ontology editors Protégé and WebProtégé. Such ontology editors support basic reasoning functions enabled by different reasoners based on predefined inference rules. However, they are not sufficient for complex and large scale reasoning scenarios. To facilitate the knowledge reuse in this study, the developed application ontology is imported to a graph database named Neo4j which supports more flexible and powerful reasoning tasks. For example, it is impossible to add properties to relationships between entities (e.g. number of resources required by an operation) in ontology, whereas in Neo4j, the relationships are also stored as

nodes which allows to have customised properties. Moreover, Neo4j provides a graph query language named Cypher[3] to enable retrieving and editing graph data stored in Neo4j database. It also offers drivers for most programming languages making it possible to integrate Cypher queries into these languages thus to conduct more complex querying and reasoning tasks.

To facilitate the manipulation of ontologies, a plugin named Neosemantics[4] is provided by the Neo4j community which enables the use of RDF in the Neo4j graph database. It allows importing and exporting ontologies in various formats including OWL, RDF, XML and Turtle etc. Once imported to Neo4j, different querying and reasoning can be executed based on tailored query conditions and reasoning rules.

In this study, several stakeholders from different organisations are involved including the requirement engineers, system architects, simulation engineers and ontology experts. To enable remote access to the graph database, a cloud server deployed on Microsoft Azure cloud is used to hold the Neo4j database. Technical details about cloud server setup, graph database configuration and the source Cypher code for importing and editing the application ontology etc. are provided on GitHub[2].

4.4. Generative design algorithm

In this case study, the main target for knowledge reuse is to automatically generate possible solutions for the future aircraft assembly system based on predefined industrial requirements. A set of design rules are defined considering the system design process of designers.

- New solutions are created hierarchically, meaning that process-level operations, such as the orbital level operations will be queried first, such as “Jig in” and “Set up working environment”; then the half-orbital level operations, such as “Set in position Rails and LFT”; and finally the quarter-orbital operations, such as “Camera at stating holes” as listed in Fig. 5.
- Individuals of essential operations of each level are created for every solution; in contrast, when an optional operation is detected, an alternative solution will be created corresponding to the available option. In this case, the half-orbital assembly process has two options for the first operation i.e. “Set in position Rails and LFT” and “Drilling template installation” as listed in Fig. 5.
- For sequencing of the operations, the minimum unit is the quarter-orbital process because the sequence of the operations in a quarter is fixed, such as the manual operations “Drilling template installation” to “Deburring, positioning, attach” as listed in Fig. 5.
- For the automated half-orbital process, which starts with optional operation “Set in position Rails & LFT”, as shown in Fig. 5, the corresponding two quarter-orbital processes can only be executed sequentially, because the LFT robot covers half of the orbital. In contrast, manual options can be either sequential or concurrent.

There are two variables to be considered for generating new orbital joint process solutions:

- operation type, meaning the operations are conducted automatically with the FLT robot or manually by manually operators. This variable is determined at the half-orbital process level considering the last rule defined above, i.e. the LFT robot covers half of the orbital. It means that a half-orbital process cannot be composed of two different types of quarter-level processes.
- sequencing between subprocesses, which can be sequential or parallel. It occurs at half-orbital level between the upper and lower orbit joint processes, and at quarter-orbital level only under manual operation type since the automated operations must be sequential.

The combination of these two variables produces three alternatives for each of the two half-orbital joint processes: automated-sequential, manual-sequential and manual-parallel. They can then be connected sequentially or in parallel to form a complete orbital joint process.

Based on the aforementioned rules and variants, a customised Python script integrating a set of Neo4j-Cypher queries is developed to automatically generate all possible solutions based on the Neo4j graph database. The pseudocode of the algorithm is shown in Appendix Algorithm 1.

The reasoning of the generative algorithm is executed based on the knowledge graph stored in the Neo4j database instead of the ontology. More specifically, the ontology was developed in OWL format and imported to Neo4j as RDF format. The generation algorithm is based on the Cypher language provided by Neo4j which is embedded in the aforementioned Python program. Comparing to conventional ontology-based reasoning mechanisms, such as description logic-based or SWRL-based, the algorithm can be considered as a hybrid mechanism, i.e. the execution logic of the algorithm is similar to the description logic-based reasoning, and the generation of each new detailed entities (nodes and edges) is based on Cypher querying which is similar to SWRL-based reasoning. The key steps of the algorithm are explained in the next subsections.

4.4.1. Generate essential operations

The generation of the orbital joint processes starts with creating a new individual of the “S40_Orbital Joint Process” class defined in the ontology. In order to differentiate different alternatives, a prefix containing an incremental integer is added to the name of the generated individual, for example “N01_S40_Orbital Joint Process”. It can be performed with the following Cypher code:

```
MATCH (p:Class{label: 'S40_Orbital Joint Process'})
CREATE (o:Individual{label:N01_'+p.label})-[:isIndividualOf]->(p)
```

After creating the new process individual, a query is conducted in the Neo4j database to fetch the essential operation classes related to the “S40_ Orbital Joint Process” class and then create an individual for each of these essential operation classes.

```
MATCH (p)-[:hasEssentialOperation]->(e:Class)
CREATE (f:Individual{label:'N01_'+e.label})-[:isIndividualOf]
    ->(e), (o)-[:hasEssentialOperation]->(f)
```

The sequence of the operation individuals can also be inherited from the ontology:

```
MATCH (e)-[:hasPredecessor]->(e1)<-[:isIndividualOf]-(f1)
CREATE (f)-[:hasPredecessor]->(f1)
```

The individuals of the essential operations for the half-orbital process and their sequence can be created following the similar query steps. It is worth noting that two individuals need to be created for each essential operation class in the ontology corresponding to upper and lower half-orbital processes. Suffixes, such as “_1” and “_2”, can be added to the names to differentiate them from each other. This step is realised by the *“CreateEssetialOperations”* function as shown in the pseudocode. An example of a generated solution is shown in Fig. 9, in which the grey nodes (“Exx”) and their relationships (solid arrow) are created during this step.

[3] https://neo4j.com/developer/cypher/.
[4] https://github.com/neo4j-labs/neosemantics.

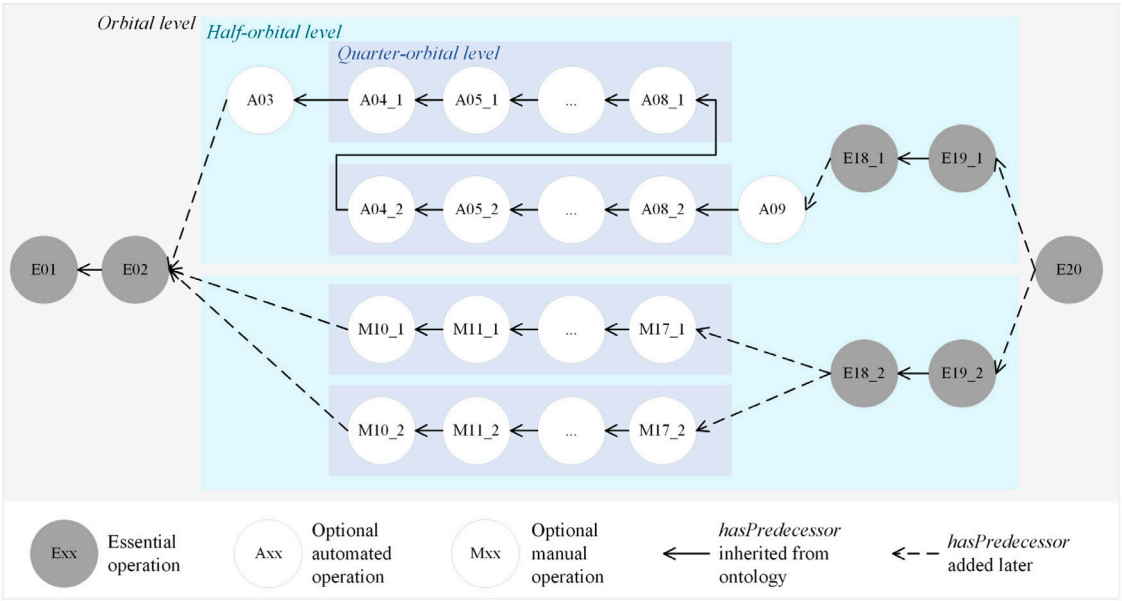

Fig. 9. An example of automatically generated orbital joint process.

4.4.2. Generate half-orbital processes and operations

The optional operations for half-orbital processes can be retrieved from the ontology using the properties "hasOptionalManualOperation" and "hasOptionalAutoOperation". Similar to essential operations, corresponding individuals are created for each of the retrieved classes. The difference is that when an optional operation is defined at the quarter-orbital level, two corresponding individuals need to be created to form a half-orbital process. Suffixes can be added to the names to differentiate them from each other. The created quarter-orbital level operations are first connected according to the "hasPredecessor" property in the ontology. Then they are connected sequentially or parallelly during different iterations which will be further explained in the following section.

This step is realised by the *"CreateAutoSubProcess", "CreateManSeqSubProcess"* and *"CreateManParSubProcess"* functions as shown in the pseudocode. The corresponding Cypher codes are available in the source file of the Python script (Github). As the example shown in Fig. 9, the white nodes ("Axx" and "Mxx") and their relationships (solid arrow) are created during this step.

4.4.3. Connect quarter-orbital and half-orbital processes

During the previous steps, corresponding individuals of essential and optional operation classes are created by retrieving knowledge stored in the ontology. They are partially connected (i.e. solid arrows in Fig. 9) based on the "hasPredecessor", producing multiple operation segments. To form a complete orbital joint process, these operation segments need to be connected. It requires identifying the beginning and ending operation of each operation segment. This can be realised by pattern matching with the "hasPredecessor" property. If an operation only has input "hasPredecessor" relationship without output relationship, it can be identified as beginning operation, such as "E01", "E18_1(2)", "A03" and "M10_1(2)" in Fig. 9. In contrast, if an operation only has output "hasPredecessor" relationship, it is an ending operation such as "E02", "E19_1(2)", "A09" and "M17_1(2)" in Fig. 9.

Once the beginning and ending operations are identified, they can be connected with each other using the "hasPredecessor" relationship (dotted arrows in Fig. 9). Different connections can be added producing different processes alternatives. For example, in Fig. 9, the quarter-orbital level "M10_1" and "M10_2" operations are connected to the "E02" operation producing a parallel manual half-orbital process. Alternatively, "M10_2" can be connected to "M17_1" and only "M10_1" connected to "E02" to create a sequential manual half-orbital process. It is worth clarifying that the dotted and solid arrows used in Fig. 9 aim to indicate that they are generated at different steps in the algorithm. They represent the same meaning in the generated processes. In some special cases, some isolated operations can be generated, such as "E20", which has no neighbouring operations. Their location in the generated process needs to be specially marked.

4.4.4. Generate resources and other properties

After all the operations of a complete process are generated, the required resources for each operation can be retrieved from the corresponding class in the ontology querying with the "requiresResource" property. For each process, one individual of each resource class is created and linked to relevant operations through the "requiresResource" relationship. An example of the generated process is shown in Fig. 10, in which the grey nodes represent the generated operations, the green nodes represent resources, the blue node represents the process name, the pink nodes represent the relevant classes defined in the application ontology, and the lines represent different types of relationships. The other properties such as the duration of each operation, quantity and cost of required resources etc., are also added in this step through the *"AddResourceTime"* function in the algorithm.

4.4.5. Generate process alternatives based on different combinations of operation types and sequences

The last step of the algorithm is to iterate among different combinations of operation types and sequences at the half-orbital level. Three options are available for each half-orbital (upper and lower) process, and they can be connected sequentially or parallelly. In this study the upper and lower orbitals are treated as different processes. Therefore, totally 18 alternative solutions are automatically generated. This is realised through three nested loops in the main thread of the algorithm.

4.5. Validation of the ontology-based engineering system

The aforementioned application ontology, the knowledge graph database and the generative design algorithm enables the main functions of OBE system. In industrial applications, the aircraft manufacturing system design starts from the definition of system boundary by the system architect. The performance requirements for the manufacturing system are first defined. For example, in the case study the defined

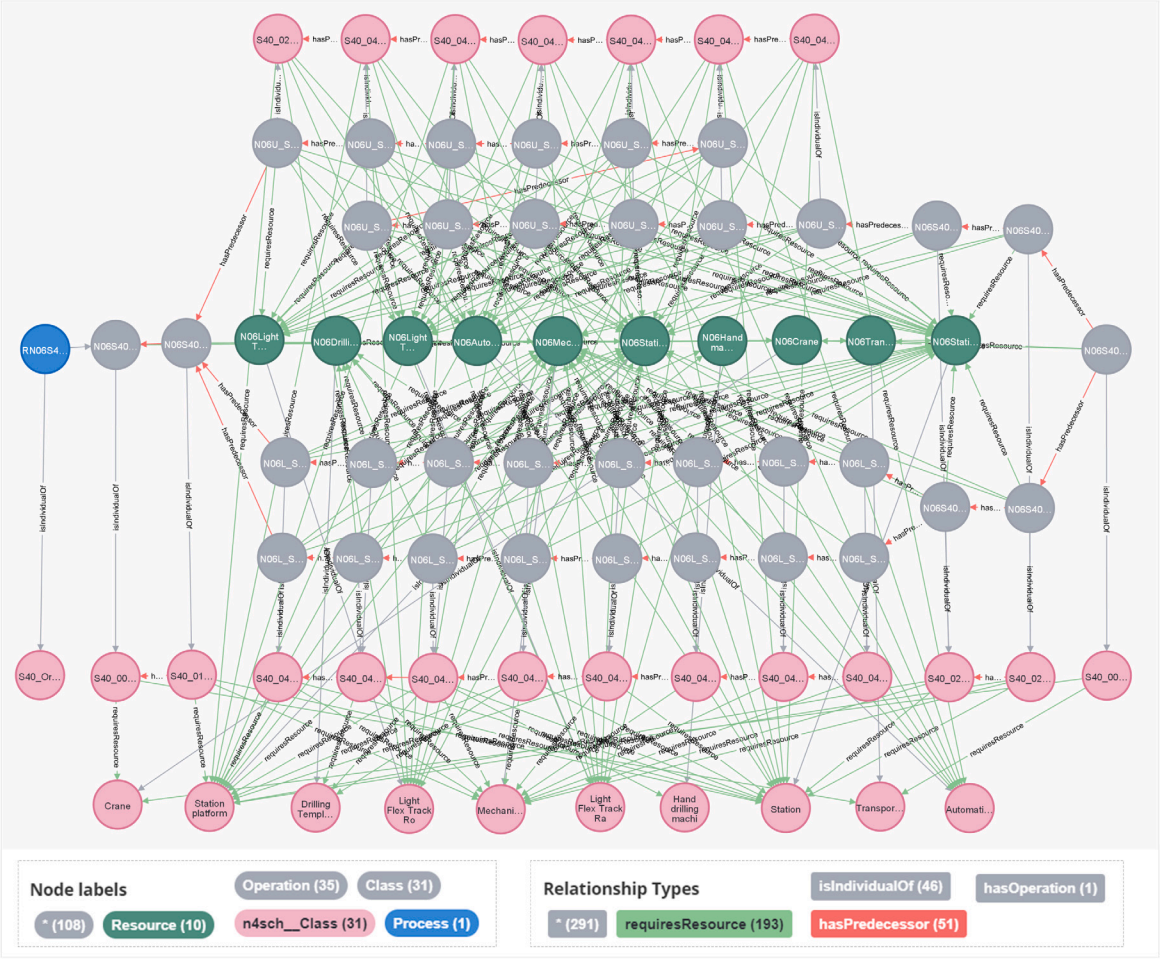

Fig. 10. An example of the generated process in Neo4j graph database. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

industrial requirements include minimum throughput of 150 aircrafts per year, automation level higher than 60% from the total number of operations, and total process lead time below 350 h. The requirement management system is used by the designer in the Service layer to define these requirements. Based on the requirement ontology, the corresponding requirement individuals are generated and added to the Neo4j graph database in OWL format.

Next, the generative design algorithm previously is triggered, generating all possible solutions for the future assembly system. The standard product, process and resources catalogue is used to create the instances of all possible process sequence alternatives that fulfil the requirements defined, providing as well the standard values to the properties of the process and resources elements. In this case study, 18 solutions of the assembly process are generated. Each solution includes a set of individuals (e.g. operations and resources) and the relationships between them. Each pair of them can be represented by a vertex and an edge respectively in the graph database. Thus, the topology of an assembly process can be represented by a Directed Acyclic Graph (DAG) which leads to a generalised representation that also fits other scenarios in the same context.

The different process sequence options can be exported and visualised as architecture models in the MBSE modelling tool in the Service Layer. The transformation between knowledge graphs and architecture models is a separate research topic in the MBSE domain which is out of the scope of this study. The approaches used in this study have been explored in some previous research articles [53,54,57]. However, some key information is briefly introduced as follows. The system architecture modelling tool used in this study is a commercial off-the-shelf software named MetaGraph.[5] The architecture models created using KARMA language based on the GOPPRRE ontology. KARMA language enables the creation of metamodels including SysML diagrams for formalising the functional views, logic views and physical structure of assembling processes. The automatically generated 18 assembly system solutions are exported from Neo4j as OWL models and then transformed to architecture models using a plugin of the MetaGraph software. Then the designer can filter, manipulate and reconfigure the imported architecture models in different formats such as SysML activity diagrams and BPMN diagrams. The selected models are then send back to the graph database following the reversed workflow using the same tools.

Later on, the designer trigger the simulation models generation, to simulate the process sequence option and compare their performance values. A customised parser is developed to fetch the process sequence options and requirement individuals from the Neo4j graph database hosted on the cloud server. The returned graph entities are parsed and information about the assembly process are extracted corresponding to different solutions. The extracted information is transformed into certain intermediate formats, such as DataFrame and JSON, to cache the extracted data. The topology of each solution is represented by an adjacency matrix, which enables the creation of a tool-agnostic model to be consumed by different simulation models. This final step follows

[5] http://www.zkhoneycomb.com/productinfo/101503.html.

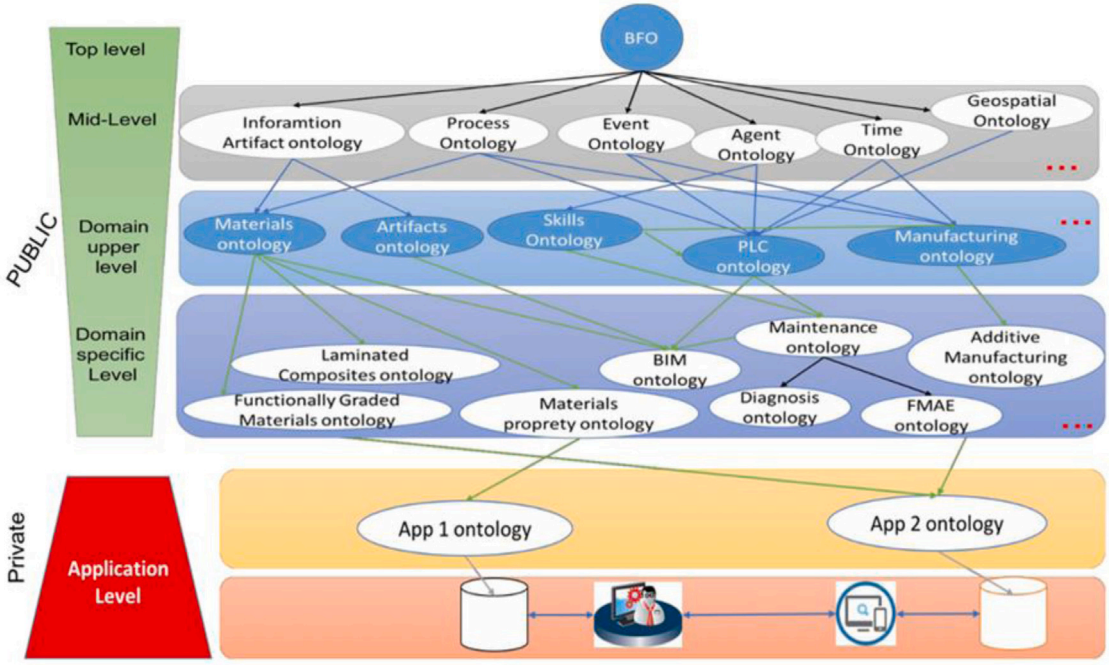

Fig. A.1. The IOF hierarchical ontology development principle.

the co-modelling approach [58] which allows different simulations to utilise the same model.

Two types of simulation, including Discrete Event Simulation (DES) and 3D simulation, are performed in this study. The DES models mainly focus on the duration of each operation and the architecture information (e.g. operation predecessor and successor etc.). In DAG, each operation has an in-degree and out-degree, which corresponds to its predecessors and successors. The DES engine recognises the topology and simulates the lead-time. The 3D simulation simulates the real working environment, operation details and interactions between operators and parts. It configures the required data like operation duration, 3D model paths and coordination information. Subsequently, the routes, actions of operators and interaction between operators and parts as well as resources are simulated. In addition, by enriching the data of virtual objects, such as the weight of jigs and tools, ergonomic factors can be calculated regarding the operator's body posture and working time.

Finally, the results of the simulations are visualised through a graphical user interface, showing the value of the performance indicators of each process sequence option. The designer is able to compare the options between them, and verify if an option fulfilled or not the defined requirement in the problem boundary defined. More details about the simulation process have been introduced in previous studies [45,59] based on a similar case.

5. Discussion

The proposed OBE functionalities of the tradespace framework were implemented to support the design automation of an aircraft orbital joint process using pre-existing knowledge of the design process, the selected toolchain and following MfM-3LM architecture. These functionalities were supported by existing interoperability functionalities of transformation between different modelling languages and automatic generation of simulation models to execute and compare results. To summarise, this study demonstrates the reuse of knowledge captured in the Ontology layer of the 3LM architecture, to automatically generate new solutions of complex systems like the aircraft assembly system, by means of querying and reasoning enabled by graph databases and relevant algorithms. Information and knowledge is first captured with application ontologies which can be integrated under the same upper-level ontologies such as the IOF-Core and BFO used in this study.

Moreover, this work reveals a step-change for MBSE methodologies, automating the manufacturing system design modelling through ontology-based knowledge capture and reuse. By this mean, a wider design space exploration can be explored and not only imaginable options by the designer, improving the analysis of promising design solutions and supporting the trade-off and decision making process.

In addition, this study leverages on the potential of MBSE and semantic web technologies to semantically integrate various systems and sub-system from concept design throughout different lifecycle phases, by bringing the cognitive side of ontology-based engineering systems, essential to realise the CDT concept [23] in the future. CDT is an ambitious target of intelligent systems which requires information integration not only during the design phase of the system lifecyle, but also extend to the following manufacturing, operation and maintenance phases. It also consider the integration across different system levels and hierarchical levels within an enterprise. Some existing work [60–62] in relevant fields provides potential methods for realising this target.

Finally, this work demonstrates the proposed framework's capability of enabling trade-off between different industrial scenarios and design solutions, facilitating ontology-based requirement definition and requirement validation through simulations during manufacturing system design phase, thus to support decision-making.

This case study served as proof-of-concept, therefore different considerations need to be taken for the implementation of such OBE tradespace framework:

- The application scenario was based on a simplified assembly system design which covers only one process in one station of the aircraft final assembly line. The problem space considered only operation types (automated and manual) and resource availability constraints (shared resources among operations), leading to a design space with a controllable number of alternatives. The real application factors of a complex design problem lead to an

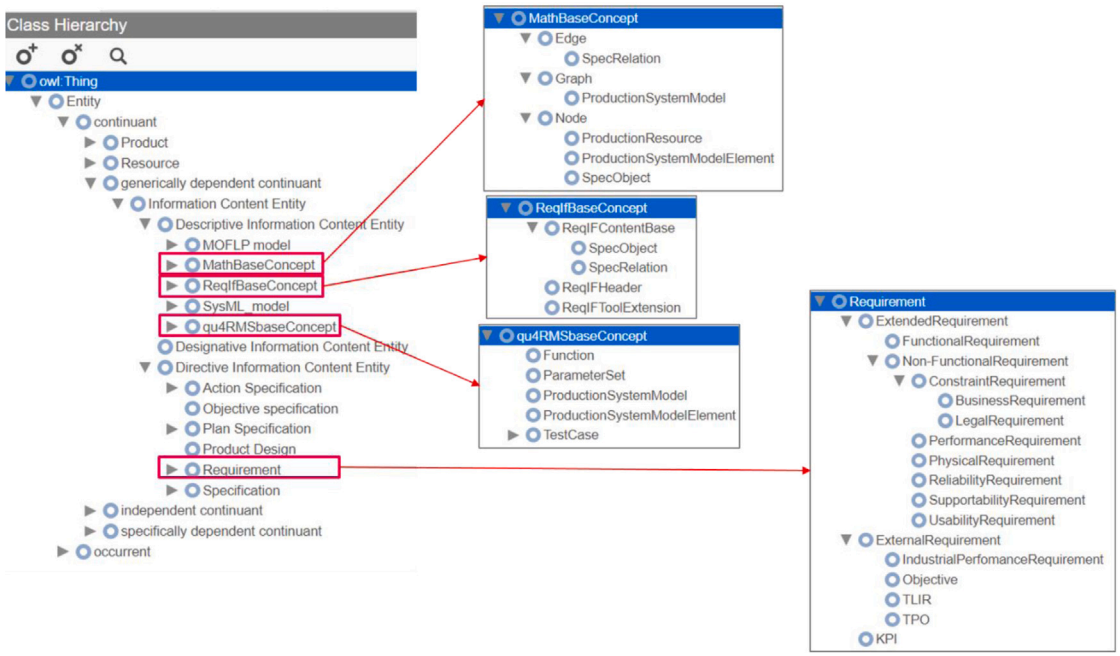

Fig. B.1. Classes defined in the Requirement application ontology.

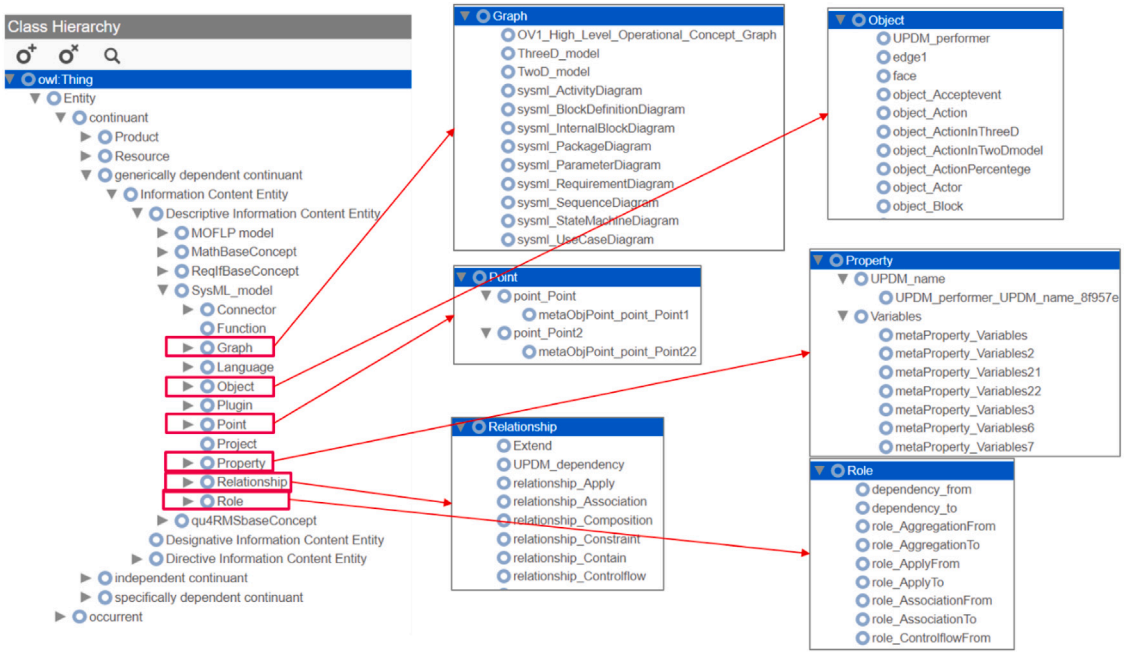

Fig. B.2. Classes defined in the MBSE application ontology.

exponential number of alternative solutions. In this case, it is necessary to adopt more advanced graph computing algorithms.

- The interaction between different functional modules need to be unified and integrated into one common platform for industrial applications, avoiding the use of intermediate files and middleware provided by different technical providers.
- The redundancy and mismatching of existing ontologies is a challenge for ontology-based applications. In this study, we adopted the upper-level ontologies like BFO and IOF-Core to partially mitigate this challenge. Some research efforts have also been made attempting to address this challenge such as the ongoing projects OntoCommons[6] and OntoTrans.[7]
- In this study, the development of the ontology and the generation of new entities through reasoning are independent. The new entities are stored in the knowledge graph and will not impact the ontology. However, the maintenance and update of the ontology

[6] https://ontocommons.eu/.
[7] https://ontotrans.eu/.

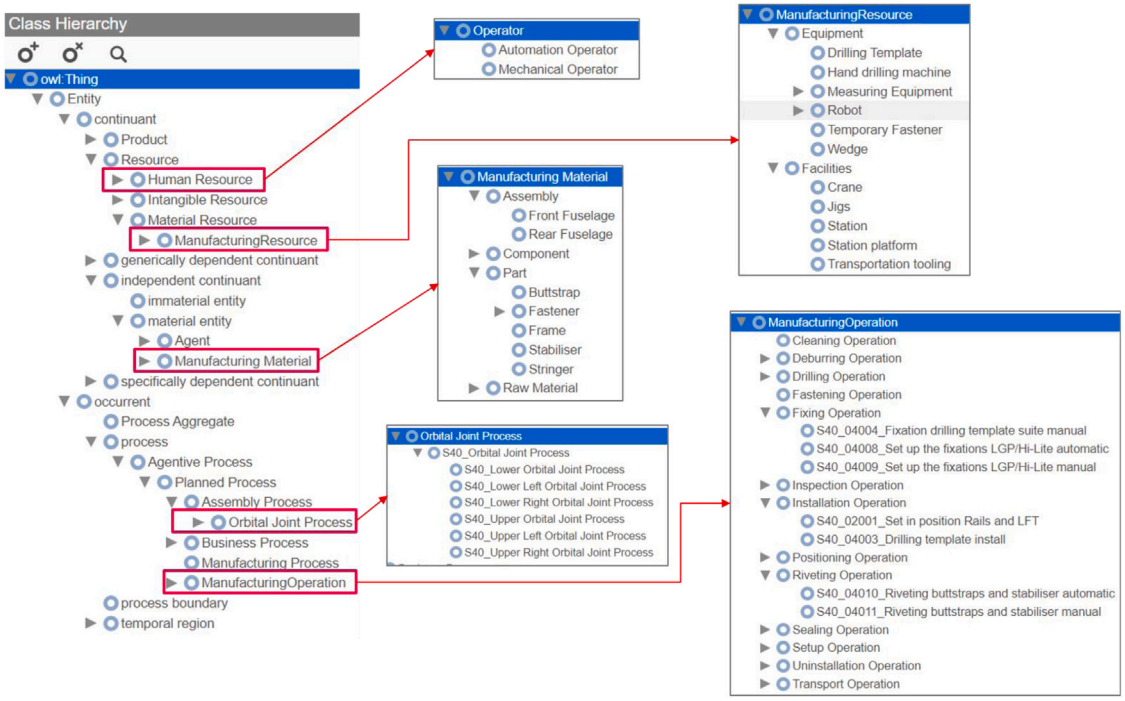

Fig. B.3. Classes defined in the Domain Knowledge application ontology.

using the new generated knowledge is a limitation to be addressed in future research.

6. Conclusion

This paper proposes an OBE system to support the aircraft manufacturing system design. The OBE features included on top of existing MBSE and interoperability capabilities of the tradespace framework, have proven to be an advantageous paradigm for the design process supporting decision-making. This enables to change the cognitive process performed by the designer when modelling the process sequence options to the OBE features (application ontology and generative design algorithm). The designer is now in charge to define a problem boundary set in the form of performance requirements to be fulfilled by the assembly process, and trigger the automatic generation of design alternatives reusing formalised knowledge in the OBE features. The alternatives can be further verified in the tradespace framework through automatically generated DES and 3D simulations and validated by comparing the simulation results against the performance requirements defined. To summarise, the main contributions of this work include:

- combining and enriching the knowledge base used for semantic integration of the MBSE tradespace framework with additional knowledge to generate design alternatives for an automatic design process;
- moving the cognitive process from the designer to the design toolchain using an ontology-based system;
- providing the designer the possibility to analyse a larger design space and support the decision-making process;
- enabling information consistency of the entire design process from requirement definition, design solution generation, architecture modelling to requirement verification based on an ontology-based system;
- exploring a possible solution to enhance the cognitive capability of industrial systems by combining ontology-based systems and MBSE methods.

The ontologies developed as part of this proof-of-concept still need to fulfil the OBE objective to describe a particular and well-defined knowledge domain from different perspectives. More efforts are needed to define methodologies and tools which can enable knowledge capturing considering different expert/user-oriented formalisation stages. Methodologies like MfM can improve capturing knowledge from designers and domain experts, while machine learning techniques such as Natural Language Processing technology and graph computing might provide possible solutions for explicit knowledge.

The real application factors of a complex design problem as the manufacturing system will lead to an exponential number of alternative solutions to be analysed, reaching unsolvable limits of the design space. Heuristic methods and genetic algorithms, as well as integrated simulation–optimisation loops are already providing solutions up to a certain level. In the authors' opinion, the cognitive process captured by OBE systems and its reasoning capabilities can provide possible solutions to control and manage the complexity of the generative design and decision-making processes.

Finally, the aircraft manufacturing system design is a highly complex process involving multiple domains and hierarchical system levels at different lifecycle phases. The strategic design decisions taken at conceptual phase should be broadened to every stage of the lifecycle in order to reach the full capabilities of a Cognitive Digital Twin.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Rebeca Arista reports financial support was provided by European Union. Xiaochen Zheng reports financial support was provided by European Union. Jinzhi Lu reports financial support was provided by European Union.

Data availability

The authors confirm that the data supporting the findings of this study are available within the article. Additional data such as source code and software instructions are available on GitHub.[8]

[8] https://github.com/zhengxiaochen/ontology_aircraft_system

Acknowledgements

The work presented in this paper has been partially supported by the EU H2020 project QU4LITY (825030) - Digital Reality in Zero Defect Manufacturing, and EU H2020 project FACTLOG (869951) - Energy-aware Factory Analytics for Process Industries.

The authors would like to recognise Fraunhofer IAO, Visual Components, Seville University, EPFL and Airbus colleagues for their support during the development of this work, and thank them for their contribution.

Appendix A. Hierarchical ontology network defined by IOF

See Fig. A.1.

Appendix B. Examples of the classes defined in the application ontology

See Figs. B.1–B.3.

Appendix C. Pseudocode of the assembly process generation algorithm

Algorithm 1 The assembly process generation algorithm — Part 1

```
Import library GraphDatabase from Neo4j Python API
#define a connection class to connect to the graph database
Class Neo4jConnection:
    Define init(self, uri, username, password)
    Define query(self, query_string)
    return query_response

#function to generate orbital and half-orbital essential operations
Define CreateEssetialOperations(Neo4jConnection, Name_prefix):
    string1 = "Cypher string for creating orbital essential operations"
    string2 = "Cypher string for creating half-orbital essential operations"
    #names of orbital level essential operations
    name1 = Neo4jConnection(string1)
    #names of half-orbital level essential operations
    name2 = Neo4jConnection(string2)
    return name1, name2

#function to generate automated-sequential half-orbital process
Define CreateAutoSubProcess(Neo4jConnection, Name_prefix):
    string3 = "Cypher string for creating 2 quarter-level automated operations and connect them sequentially"
    #names of the first and last operations
    name3= Neo4jConnection(string3)
    return name3

#function to generate manual-sequential half-orbital assembly process
Define CreateManSeqSubProcess(Neo4jConnection, Name_prefix):
    string4 = "Cypher string for creating 2 quarter-level manual operations and connect them sequentially"
    #names of the first and last operations
    name4 = Neo4jConnection(string4)
    return name4
```

Algorithm 2 The assembly process generation algorithm — Part 2

```
#function to generate manual-parallel half-orbital assembly process
Define CreateManParSubProcess(Neo4jConnection, Name_prefix):
    string5 = "Cypher string for creating 2 quarter manual operations"
    #names of the first and last operations of the 2 created operations
    name5 = Neo4jConnection(string5)
    return name5

#function to add required resources and duration to created operations
Define AddResourceTime(Neo4jConnection, Name_prefix):
    string6 = "Cypher for adding resource/duration to created operations"
    Neo4jConnection(string6)

#the main thread
if name == "_main_":
#create connection to Neo4j database
Neo4jConnection = Neo4jConnection(uri, username, password)
#increasing integer to differentiate created solutions
Nameprefix = 1
#available alternatives for half-orbital level processes
OptionsVector = ["automated-sequential", "manual-sequential", "manual-parallel"]
UpperProcess in OptionsVector: LowerProcess in OptionsVector: Sequence in ["sequential", "parallel"]:
#generate essential operations
call CreateEssetialOperations()
#generate two half-orbital processes
call CreateAutoSubProcess() or CreateManSeqSubProcess() or CreateManParSubProcess() accordingly
connect half-orbital processes with essential operations
#add relevant resources and duration to created operations
call AddResourceTime()
```

References

[1] Roy R, Williams G. Capturing the assembly process planning rationale within an aerospace industry. In: Proceedings of the 1999 IEEE international symposium on assembly and task planning (ISATP'99)(Cat. No. 99TH8470). IEEE; 1999, p. 319–24. http://dx.doi.org/10.1109/ISATP.1999.782978.

[2] Maganha I, Silva C, Klement N, dit Eynaud AB, Durville L, Moniz S. Hybrid optimisation approach for sequencing and assignment decision-making in reconfigurable assembly lines. IFAC-PapersOnLine 2019;52(13):1367–72. http://dx.doi.org/10.1016/j.ifacol.2019.11.389.

[3] Mas F, Rios J, Menendez JL, Gomez A. A process-oriented approach to modeling the conceptual design of aircraft assembly lines. Int J Adv Manuf Technol 2013;67(1–4):771–84. http://dx.doi.org/10.1007/s00170-012-4521-5.

[4] Askin RG, Askin RG, Standridge CR, Standridge CR. Modeling and analysis of manufacturing systems. John Wiley & Sons Incorporated; 1993.

[5] Mourtzis D. Simulation in the design and operation of manufacturing systems: state of the art and new trends. Int J Prod Res 2020;58(7):1927–49. http://dx.doi.org/10.1080/00207543.2019.1636321.

[6] Davies M. Knowledge–Explicit, implicit and tacit: Philosophical aspects. In: International encyclopedia of the social & behavioral sciences, Vol. 13. 2015, p. 74–90.

[7] Mas F, Ríos J, Gómez A, Hernández JC. Knowledge-based application to define aircraft final assembly lines at the industrialisation conceptual design phase. Int J Comput Integr Manuf 2016;29(6):677–91. http://dx.doi.org/10.1080/0951192X.2015.1068453.

[8] Smith B. Ontology. In: The furniture of the world. Brill; 2012, p. 47–68. http://dx.doi.org/10.1163/9789401207799_005.

[9] Horrocks I. Ontologies and the semantic web. Commun ACM 2008;51(12):58–67, URL https://dl.acm.org/doi/fullHtml/10.1145/1409360.1409377.

[10] McCarthy J. Circumscription—a form of non-monotonic reasoning. Artificial Intelligence 1980;13(1–2):27–39. http://dx.doi.org/10.1016/0004-3702(80)90011-9.

[11] Pratt MJ, et al. Introduction to ISO 10303—the STEP standard for product data exchange. J Comput Inf Sci Eng 2001;1(1):102–3. http://dx.doi.org/10.1115/1.1354995.

[12] Fenves SJ, Foufou S, Bock C, Sriram RD. CPM2: a core model for product data. J Comput Inf Sci Eng 2008;8(1). http://dx.doi.org/10.1115/1.2830842.

[13] Baysal MM, Roy U, Sudarsan R, Sriram RD, Lyons KW. The open assembly model for the exchange of assembly and tolerance information: overview and example. In: International design engineering technical conferences and computers and information in engineering conference, Vol. 46970. 2004, p. 759–70. http://dx.doi.org/10.1115/DETC2004-57727.

[14] El Kadiri S, Kiritsis D. Ontologies in the context of product lifecycle management: state of the art literature review. Int J Prod Res 2015;53(18):5657–68. http://dx.doi.org/10.1080/00207543.2015.1052155.

[15] Industry-Ontology-Foundry. Industry Ontology Foundry (IOF) - Technical principles. Tech. rep., 2020, URL https://www.industrialontologies.org/technical-principles/.

[16] Patil L, Dutta D, Sriram R. Ontology-based exchange of product data semantics. IEEE Trans Autom Sci Eng 2005;2(3):213–25. http://dx.doi.org/10.1109/TASE.2005.849087.

[17] Dartigues C, Ghodous P, Gruninger M, Pallez D, Sriram R. CAD/CAPP integration using feature ontology. Concurr Eng 2007;15(2):237–49. http://dx.doi.org/10.1177/1063293X07079312.

[18] Szejka AL, Júnior OC, Loures ER, Panetto H, Aubry A. Proposal of a model-driven ontology for product development process interoperability and information sharing. In: IFIP international conference on product lifecycle management. Springer; 2016, p. 158–68. http://dx.doi.org/10.1007/978-3-319-54660-5_15.

[19] Zha XF, Du H. A PDES/STEP-based model and system for concurrent integrated design and assembly planning. Comput Aided Des 2002;34(14):1087–110. http://dx.doi.org/10.1016/S0010-4485(01)00186-5.

[20] Zheng C, An Y, Wang Z, Qin X, Eynard B, Bricogne M, Le Duigou J, Zhang Y. Knowledge-based engineering approach for defining robotic manufacturing system architectures. Int J Prod Res 2022;1–19. http://dx.doi.org/10.1080/00207543.2022.2037025.

[21] Mas F, Racero J, Oliva M, Morales-Palma D. A preliminary methodological approach to models for manufacturing (MfM). In: IFIP international conference on product lifecycle management. Springer; 2018, p. 273–83. http://dx.doi.org/10.1007/978-3-030-01614-2_25.

[22] Holland OT. Model-based systems engineering. In: Modeling and simulation in the systems engineering life cycle. Springer; 2015, p. 299–306.

[23] Zheng X, Lu J, Kiritsis D. The emergence of cognitive digital twin: vision, challenges and opportunities. Int J Prod Res 2021;1–23. http://dx.doi.org/10.1080/00207543.2021.2014591.

[24] Delpiano M, Fabbri M, Garda C, Valfrè E. Virtual development and integration of advanced aerospace systems: Alenia aeronautics experience. Tech. rep., ALENIA AERONAUTICA SPA TORINO (ITALY); 2003, URL https://apps.dtic.mil/sti/pdfs/ADP014152.pdf.

[25] Pardessus T. Concurrent engineering development and practices for aircraft design at Airbus. In: Proceedings of the 24th ICAS conf., Yokohama, Japan, Vol. 1. 2004, URL http://icas.org/ICAS_ARCHIVE/ICAS2004/PAPERS/413.PDF.

[26] Sohlenius G. Concurrent engineering. CIRP Ann 1992;41(2):645–55.

[27] Mousavi BA, Azzouz R, Heavey C, Ehm H. A survey of model-based system engineering methods to analyse complex supply chains: a case study in semiconductor supply chain. IFAC-PapersOnLine 2019;52(13):1254–9. http://dx.doi.org/10.1016/j.ifacol.2019.11.370.

[28] Kiritsis D. Semantic technologies for engineering asset life cycle management. Int J Prod Res 2013;51(23–24):7345–71. http://dx.doi.org/10.1080/00207543.2012.761364.

[29] Jørgensen N. The Boeing 777: development life cycle follows artifact. In: World conference on integrated design and process technology. IDPT, Citeseer; 2006, p. 25–30.

[30] Batarseh O, McGinnis L, Lorenz J. 6.5.2 MBSE supports manufacturing system design. In: INCOSE international symposium, Vol. 22. Wiley Online Library; 2012, p. 850–60. http://dx.doi.org/10.1002/j.2334-5837.2012.tb01375.x.

[31] Oliva M, Mas F, Eguia I, Valle Cd, Lourenço EJ, Baptista AJ. An innovative methodology to optimize aerospace eco-efficiency assembly processes. In: IFIP international conference on product lifecycle management. Springer; 2020, p. 448–59. http://dx.doi.org/10.1007/978-3-030-62807-9_36.

[32] Mas F, Menéndez JL, Oliva M, Gómez A, Ríos J. Collaborative engineering paradigm applied to the aerospace industry. In: IFIP international conference on product lifecycle management. Springer; 2013, p. 675–84. http://dx.doi.org/10.1007/978-3-642-41501-2_66.

[33] Zha XF, Lim SY, Lu WF. A knowledge intensive multi-agent system for cooperative/collaborative assembly modeling and process planning. J Integr Des Process Sci 2003;7(1):99–122, URL https://dl.acm.org/doi/abs/10.5555/1242339.1242346.

[34] Hunter R, Rios J, Perez J, Vizan A. A functional approach for the formalization of the fixture design process. Int J Mach Tools Manuf 2006;46(6):683–97. http://dx.doi.org/10.1016/j.ijmachtools.2005.04.018.

[35] Colledani M, Pedrielli G, Terkaj W, Urgo M. Integrated virtual platform for manufacturing systems design. Procedia CIRP 2013;7:425–30. http://dx.doi.org/10.1016/j.procir.2013.06.010.

[36] Terkaj W, Urgo M. Ontology-based modeling of production systems for design and performance evaluation. In: 2014 12th IEEE international conference on industrial informatics. INDIN, IEEE; 2014, p. 748–53. http://dx.doi.org/10.1109/INDIN.2014.6945606.

[37] Melkote S. Development of iFAB Manufacturing Process and Machine Library. Tech. Rep, Georgia Institute of Technology; 2012, URL https://apps.dtic.mil/sti/citations/ADA565243.

[38] Yukish M. Configuring and exploring the foundry trade space. Tech. rep., PENNSYLVANIA STATE UNIV STATE COLLEGE APPLIED RESEARCH LAB; 2012, https://apps.dtic.mil/sti/citations/ADA564500.

[39] Ríos J, Jiménez J, Pérez J, Vizán A, Menéndez J, Más F. KBE application for the design and manufacture of HSM fixtures. Acta Polytech 2005;45(3). http://dx.doi.org/10.14311/698.

[40] Mas F, Racero J, Oliva M, Morales-Palma D. Preliminary ontology definition for aerospace assembly lines in Airbus using Models for Manufacturing methodology. Procedia Manuf 2019;28:207–13. http://dx.doi.org/10.1016/j.promfg.2018.12.034.

[41] Arista R, Mas F, Morales-Palma D, Oliva M, Vallellano C. A preliminary ontology-based engineering application to industrial system reconfiguration in conceptual phase. In: Proceedings, Vol. 1613. 2020, p. 0073, http://ceur-ws.org ISSN. URL http://ceur-ws.org/Vol-2969/paper20-FOMI.pdf.

[42] Page Risueno J, Nagel B. Development of a knowledge-based engineering framework for modeling aircraft production. In: AIAA aviation 2019 forum. 2019, p. 2889. http://dx.doi.org/10.2514/6.2019-2889.

[43] Meski O, Belkadi F, Laroche F, Ritou M, Furet B. A generic knowledge management approach towards the development of a decision support system. Int J Prod Res 2021;59(22):6659–76. http://dx.doi.org/10.1080/00207543.2020.1821930.

[44] Zheng X, Hu X, Arista R, Lu J, Sorvari J, Lentes J, Ubis F, Kiritsis D. A semantic-driven tradespace framework to accelerate aircraft manufacturing system design. J Intell Manuf 2022;1–24. http://dx.doi.org/10.1007/s10845-022-02043-7.

[45] Hu X, Arista R, Zheng X, Lentes J, Sorvari J, Lu J, Ubis F, Kiritsis D. Ontology-based system to support industrial system design for aircraft assembly. IFAC-PapersOnLine 2022;55(2):175–80. http://dx.doi.org/10.1016/j.ifacol.2022.04.189.

[46] Arp R, Smith B. Function, role, and disposition in basic formal ontology. Nat Preced 2008;1. http://dx.doi.org/10.1038/npre.2008.1941.1.

[47] Borgo S, Ferrario R, Gangemi A, Guarino N, Masolo C, Porello D, Sanfilippo EM, Vieu L. DOLCE: A descriptive ontology for linguistic and cognitive engineering. Appl Ontol 2022;(Preprint):1–25. http://dx.doi.org/10.3233/AO-210259.

[48] Lemaignan S, Siadat A, Dantan J-Y, Semenenko A. MASON: A proposal for an ontology of manufacturing domain. In: IEEE workshop on distributed intelligent systems: Collective intelligence and its applications (DIS'06). IEEE; 2006, p. 195–200. http://dx.doi.org/10.1109/DIS.2006.48.

[49] Negri E, Fumagalli L, Macchi M, Garetti M. Ontology for service-based control of production systems. In: IFIP international conference on advances in production management systems. Springer; 2015, p. 484–92. http://dx.doi.org/10.1007/978-3-319-22759-7_56.

[50] Zheng X, Lu J, Arista R, Hu X, Lentes J, Ubis F, Sorvari J, Kiritsis D. Development of an application ontology for knowledge management to support aircraft assembly system design. In: FOMI 2021: 11th international workshop on formal ontologies meet industry, held at JOWO 2021: Episode VII the Bolzano summer of knowledge, September 11–18, 2021, Bolzano, Italy, Vol. 1613. 2020, p. 0073, URL http://ceur-ws.org/Vol-2969/paper21-FOMI.pdf.

[51] Mejía-Gutiérrez R, Carvajal-Arango R. Design verification through virtual prototyping techniques based on systems engineering. Res Eng Des 2017;28(4):477–94. http://dx.doi.org/10.1007/s00163-016-0247-y.

[52] Li T, Lockett H, Lawson C. Using requirement-functional-logical-physical models to support early assembly process planning for complex aircraft systems integration. J Manuf Syst 2020;54:242–57. http://dx.doi.org/10.1016/j.jmsy.2020.01.001.

[53] Lu J, Ma J, Zheng X, Wang G, Li H, Kiritsis D. Design ontology supporting model-based systems engineering formalisms. IEEE Syst J 2021;1–12. http://dx.doi.org/10.1109/JSYST.2021.3106195.

[54] Lu J, Wang G, Ma J, Kiritsis D, Zhang H, Törngren M. General modeling language to support model-based systems engineering formalisms (Part 1). In: INCOSE international symposium, Vol. 30. Wiley Online Library; 2020, p. 323–38. http://dx.doi.org/10.1002/j.2334-5837.2020.00725.x.

[55] Zheng X, Psarommatis F, Petrali P, Turrin C, Lu J, Kiritsis D. A quality-oriented digital twin modelling method for manufacturing processes based on a multi-agent architecture. Procedia Manuf 2020;51:309–15. http://dx.doi.org/10.1016/j.promfg.2020.10.044.

[56] Zheng X, Petrali P, Lu J, Turrin C, Kiritsis D. RMPFQ: A quality-oriented knowledge modelling method for manufacturing systems towards cognitive digital twins. Front Manuf Technol 2022;2. http://dx.doi.org/10.3389/fmtec.2022.901364.

[57] Ding J, Reniers M, Lu J, Wang G, Feng L, Kiritsis D. Integration of modeling and verification for system model based on KARMA language. In: Proceedings of the 18th ACM SIGPLAN international workshop on domain-specific modeling. New York, NY, USA: Association for Computing Machinery; 2021, p. 41–50. http://dx.doi.org/10.1145/3486603.3486775.

[58] Gomes C, Thule C, Broman D, Larsen PG, Vangheluwe H. Co-simulation: a survey. ACM Comput Surv 2018;51(3):1–33. http://dx.doi.org/10.1145/3179993.

[59] Hu X, Arista R, Lentes J, Lu J, Zheng X, Sorvari J, Ubis F, Kiritsis D. Ontology-centric industrial requirements validation for aircraft assembly system design. In: 10th IFAC conference on manufacturing modelling, management and control, Nantes, France, June 22-24. 2022.

[60] Chryssolouris G. Manufacturing systems: Theory and practice. Springer Science & Business Media; 2013.

[61] Chirumalla K, Bertoni A, Parida A, Johansson C, Bertoni M. Performance measurement framework for product–service systems development: a balanced scorecard approach. Int J Technol Intell Plan 2013;9(2):146–64.

[62] Mourtzis D, Panopoulos N, Angelopoulos J. Production management guided by industrial internet of things and adaptive scheduling in smart factories. In: Design and operation of production networks for mass personalization in the era of cloud technology. Elsevier; 2022, p. 117–52. http://dx.doi.org/10.1016/B978-0-12-823657-4.00014-2.