

Doctoral theses at NTNU, 2024:212

Liang Zhang

Digital transformation supported by Knowledge-Based Engineering and semantic modeling for automating engineering tasks

Doctoral thesis

NTNU
Norwegian University of Science and Technology
Thesis for the Degree of
Philosophiae Doctor
Faculty of Engineering
Department of Mechanical and Industrial
Engineering



Norwegian University of
Science and Technology

Liang Zhang

Digital transformation supported by Knowledge-Based Engineering and semantic modeling for automating engineering tasks

Thesis for the Degree of Philosophiae Doctor

Trondheim, May 2024

Norwegian University of Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering



Norwegian University of
Science and Technology

NTNU

Norwegian University of Science and Technology

Thesis for the Degree of Philosophiae Doctor

Faculty of Engineering

Department of Mechanical and Industrial Engineering

© Liang Zhang

ISBN 978-82-326-8016-0 (printed ver.)

ISBN 978-82-326-8015-3 (electronic ver.)

ISSN 1503-8181 (printed ver.)

ISSN 2703-8084 (online ver.)

Doctoral theses at NTNU, 2024:212

Printed by NTNU Grafisk senter

Preface

This PhD thesis is submitted in partial fulfillment of the requirements of the degree of Philosophiae doctor (Ph.D.) at the Norwegian University of Science and Technology (NTNU) in Trondheim. The presented research was carried out at the Department of Mechanical and Industrial Engineering (MTP) between September 2020 and December 2023. The Ph.D. research was directed by Associate Professor Andrei Lobov as the main supervisor and Associate Professor Anna Olsen as the co-supervisor.

The Ph.D. work was supported by NTNU MTP project nr. 81148038. The aim of the Ph.D. research is to explore the advancing engineering knowledge representation and integration principles in Knowledge-Based Engineering (KBE) domain. This dissertation is presented as a collection of publications. The collection of publications represent the culmination of established work and theory developed towards building and defending the thesis idea of *Digital transformation supported by Knowledge-Based Engineering and semantic modeling for automating engineering design tasks.*

Acknowledgement

I would like to express my sincere gratitude to all those who supported me throughout my PhD journey and contributed to the completion of this thesis. First of all, I wish to thank my esteemed supervisor Andrei Lobov for all the things he did for me. He offered me the opportunity to pursue my Ph.D. degree, which I believe is one of the most important turns in my life. He taught me that there is typically no fixed answer so we as researchers can provide our own version, and even a little expansion of the boundary of current human knowledge is valuable. He consistently granted me the freedom to steer my research direction and offered inspiration and guidance toward our goals. We discussed a lot about the KBE research, sometimes even heated debates, and he always patiently heard my puzzle. Without his support, I can not finish this thesis. Additionally, I also wish to thank my co-supervisor Anna Olsen, for her contributions in refining my journal paper and providing encouragement.

Then I would like to express my deep gratitude to my dear wife, Jian Li, for her unwavering support throughout my PhD journey. Since her arrival in Norway in 2021, my life has become more vibrant and convenient. The birth of our son, JunZhe Zhang, in St. Olavs hospital on January 6, 2023, added a new dimension to our lives. Thanks to Junzhe, an angel baby, for bringing us happiness and not consuming too much of our energy in taking care of him. My wife dedicated herself to caring for our baby and managing our daily life, allowing me more time to focus on my PhD research. Occasionally, I would discuss my research with her. Her insights as an engineer influenced my work, and her encouragement made me confident in the research. With her companionship and support, I have cherished memories from my pursuit of a PhD. I cannot imagine the completion of this thesis without my wife.

In addition, I would thank all the friends I met in Norway for helping me or appearing in my life. Special thanks to Aunt Mei, Wei Zhang's mother, for her generous assistance and companionship with my wife, especially after the birth of my son. I am grateful to Ye family for providing me with my first place to live when I was a newcomer to Norway. I will cherish the memories of the shared flight to Norway with Yu Wang and Wei Zhang, marking the beginning of our friendship. The trip to Trolltunga with Zhuo Xu and Ge Li allowed us to witness the breathtaking scenery, creating lasting memories. Fixing my bike in the laboratory with Xu He and Di Wan, and the joyful dinners with Fan Gao, Xiaoming Ran, Qingbo Wang, and Anni Cao, are also moments I will never forget. I extend my appreciation to my classmates Tuan, Brikene, Ambra, Eman, Sara, Ashley, and Haakon, for studying together throughout this journey. Special thanks to my neighbor, Grandpa Roy, for his assistance with the drainage pipe and for sharing the interesting adventures of his life.

I am also grateful to the individuals who assisted me at the Department of Mechanical and Industrial Engineering (MTP), NTNU. I extend my thanks to the administration team of MTP for their guidance and support throughout my journey. Special thanks to Kari Elise Dahle for her useful assistance in the visa application and guidelines in the initial stage of my PhD.

Finally, I would like to express my gratitude to my parents for their support for my PhD study. It is a pity that we have not been able to meet in Norway due to the inconveniences brought by COVID-19.

These three years have been filled with memorable and happy times, making me truly love this place. I humbly acknowledge all the help I received throughout my PhD journey.

Liang Zhang
December 2023, Trondheim, Norway

Abstract

The culture of product design is shifting from case-by-case development to the Knowledge-Based (KB) paradigm. This shift aims to foster knowledge sharing and reuse across different stages and groups in engineering. Within this context, engineering knowledge encompasses both product data and design rules. The paradigm shift requires the digital transformation of product design from document-centric to knowledge-centric paradigm. Despite existing research primarily focusing on product data representation, there remains a notable gap in addressing a comprehensive framework formalizing design knowledge, particularly in terms of representing various types of design rules.

Knowledge-Based Engineering (KBE) is a technology that employs knowledge representation languages for describing facts and rules, with the goal of automating engineering tasks through reasoning abilities, including design, analysis, and optimization. KBE often integrates with a CAD kernel for geometry manipulation, and its explicit product data representation facilitates data exchange with Computer-Aided Analysis (CAA) tools, enhancing its data processing and computation capabilities.

The semantic web is designed to provide a common framework for sharing and reusing data across various sources. It aims to transform the existing “web of documents” to be a “web of data”, aligning with the objectives of KBE paradigm. Consequently, the semantic web stack can be utilized to represent product data and universal design rules. Leveraging the semantic representation of design knowledge, various applications can be developed to encapsulate case-specific rules. In summary, KBE and semantic modeling offer a potential framework for the digital transformation of the product design paradigm.

This thesis aims to propose a practical framework that leverages KBE and se-

semantic modeling. This framework is designed to facilitate the transition from a document-centric paradigm to a knowledge-based paradigm in engineering design. A key feature of this framework is its ability to capture and formalize designers' intent, enabling the rapid generation of product variants in response to changes in intent. Compared with hard-coded ad-hoc software applications, this framework offers better interoperability and extendability. Consequently, the integration of various digital tools for diverse engineering tasks is facilitated, supporting long-term knowledge reuse. The exploration of interaction with Natural Language Processing (NLP) tools for user-friendly workflow composition highlights its promising potential.

The thesis is presented as a collection of publications with the goal of facilitating the transition towards a knowledge-based paradigm in engineering design. The structure of the thesis consists of five chapters: Chapter 1 provides a comprehensive overview of the background and the research questions that underpin this Ph.D. work. Chapter 2 delves into the theoretical framework and examines related works that have preceded this research. Chapter 3 summarizes the primary contributions of each individual research paper. Chapter 4 elaborates the connections between the individual research papers and the three derived research questions. Finally, Chapter 5 concludes the thesis and provides perspectives for future research.

Contents

Preface	I
Acknowledgement	III
Abstract	V
1 Introduction	1
1.1 Background and research goal	1
1.2 Research questions	3
1.3 Research scope	4
1.4 Contributions	4
1.5 Thesis structure	5
2 Theoretical foundation, Related Works and Definitions	7
2.1 Product design	8
2.2 Semantic Web Stack	11
2.3 Knowledge-Based System	14
2.4 Interpretation of Rules	16
2.5 Knowledge-Based Engineering	17

2.6	Interoperability and Extendability	21
2.7	Service-Oriented Architecture	21
2.7.1	Service composition	22
2.7.2	Semantic Service Discovery	22
2.8	Low-code Workflow Platform	23
3	Contributions Overview	25
3.1	Paper A: A parametric model of umbilical cable with Siemens NX considering its reliability	25
3.1.1	Objective	25
3.1.2	Relevance to the thesis	26
3.1.3	Contributions	26
3.1.4	Results	27
3.2	Paper B: An ontology-based KBE application for supply chain sustainability assessment	27
3.2.1	Objective	27
3.2.2	Relevance to the thesis	28
3.2.3	Contributions	28
3.2.4	Results	29
3.3	Paper C: Extending design automation by integrating external services for product design	29
3.3.1	Objective	29
3.3.2	Relevance to the thesis	30
3.3.3	Contributions	30
3.3.4	Results	31
3.4	Paper D: Interoperability in automating engineering tasks: An illustration with pipe routing application	32
3.4.1	Objective	32

3.4.2	Relevance to the thesis	32
3.4.3	Contributions	33
3.4.4	Results	33
3.5	Paper E: A Low-code KBE Solution for Engineering Design: a Pipe Routing Case Demonstration	34
3.5.1	Objective	34
3.5.2	Relevance to the thesis	35
3.5.3	Contributions	36
3.5.4	Results	36
3.6	Paper F: Semantic Web Rule Language-based approach for imple- menting Knowledge-Based Engineering systems	37
3.6.1	Objective	37
3.6.2	Relevance to the thesis	38
3.6.3	Contributions	38
3.6.4	Results	39
4	Discussion	43
4.1	RQ1: Benefits to use KB paradigm and why not just programming languages	43
4.2	RQ2: What knowledge to represent and what approaches for it . .	50
4.3	RQ3: User-friendly design workflow composition	53
4.4	Additional remarks	56
5	Conclusions and Future Work	57
5.1	Conclusions	57
5.2	Future work	58
	Bibliography	59

Appendices	73
Paper A: A parametric model of umbilical cable with siemens NX considering its reliability	75
Paper B: An ontology-based KBE application for supply chain sustainability assessment	83
Paper C: Extending design automation by integrating external services for product design	93
Paper D: Interoperability in automating engineering tasks: An illustration with pipe routing application	101
Paper E: A Low-code KBE Solution for Engineering Design: a Pipe Routing Case Demonstration	111
Paper F: Semantic Web Rule Language-based approach for implementing Knowledge-Based Engineering systems	139

List of Figures

- 1.1 KBE design paradigm. a) Comparison with conventional design paradigm; b) Schematic diagram of a KBE application. 3
- 1.2 The relationship between research questions and contributions. . . 6

- 2.1 Schematic representation of 3 types of MDO system implementations. (Adapted from Ref. [1]) 12
- 2.2 A simple version Semantic Web Stack. (Paper F) 13
- 2.3 The relation between KBE and CDS. 19
- 2.4 Formalization at different levels. 20

- 3.1 Result demonstration [2]. (a) Green: reliability check passed. (b) Red: reliability check failed. 27
- 3.2 The demonstration of the KBE application [3]. a) Different teams input their data into the unified knowledge base. b) The calculation process. c) The query to get the assessment result. d) The assessment results of different supply chains. 29
- 3.3 External services wrapped as Procedural Rules. (Adapted from Ref. [4].) 31

3.4	Four wood connection designs with different fasteners: (a) bolt; (b) lag screw; (c) nail; (d) wood screw. Green: connection capacity check passed; Red: connection capacity check failed. (Adapted from Ref. [4].)	32
3.5	The proposed architecture for automating engineering tasks powered by semantic data [5].	34
3.6	The KBE pipe router interoperates with different tools. a) AVEVA. b) Web browser. c) Siemens NX. [5]	35
3.7	The BPMN workflows generated from NL description. a) From high quality text; b) From low quality text. (Paper E)	37
3.8	The recommendation process for relevant services based on input and output check. (Paper E)	37
3.9	The proposed architecture based on KBE and semantic web stack. (Paper F)	39
3.10	Schematic diagram of a transmission shaft. (Paper F)	40
3.11	The development and debugging in local ontology editor Protege. (a) Class definitions; (b) Object property; (c) Data property; (d) Inferred geometric model of shaft in AVEVA PML; (e) Inferred geometric model of shaft in Siemens NX KF. (Paper F)	41
4.1	The illustration of the three types of KBE applications. a) “Classic” KBE. b) “Extended” KBE. c) “Flexible” KBE.	45
4.2	The comparison between ad-hoc data parsers and a unified knowledge representation [3].	46
4.3	The designers’ intent is at knowledge level.	48
4.4	A reference architecture of “flexible” KBE extended with BPMN.	51
4.5	The flow to semantically represent the design knowledge.	52
4.6	The NLP-based workflow composer and the process to execute a workflow (Paper E).	55

List of Abbreviations

AI	Artificial Intelligence
AMAAD	Adaptable Methodology for Automation Application Development
API	Application Programming Interface
BPMN	Business Process Model and Notation
CAA	Computer-Aided Analysis
CAD	Computer-Aided Design
CAE	Computer-Aided Engineering
CAx	Computer-Aided technologies, x for Analysis, Design and etc.
CommonKADS	Common Knowledge Acquisition and Design Support
CRUD	Create, Read, Update and Delete
GA	Genetic Algorithm
GUI	Graphic User Interface
ICARE	Illustrations, Constraints, Activities, Rules and Entities
JSON	JavaScript Object Notation
KBE	Knowledge-Based Engineering
KBS	Knowledge-Based System

KB	Knowledge-Based, Knowledge Base
KNOMAD	Knowledge capture; Normalisation; Organisation; Modeling; Analysis; and Delivery
MBD	Model-Based Development
MDD	Model-Driven Development
MDO	Multi-disciplinary Design and Optimization
MOKA	Methodology for Knowledge-Based Engineering Applications
NL(P)	Natural Language (Processing)
OOP	Object-Oriented Programming
OO	Object-Oriented
OPC UA	Open Platform Communications Unified Architecture
OWL(-S)	Web Ontology Language (for Services)
PML	Programmable Macro Language (for AVEVA)
RDF	Resource Description Framework
SOA	Service-Oriented Architecture
SPARQL	SPARQL Protocol and RDF Query Language
STEP	Standard for the Exchange of Product Data
SWRL	Semantic Web Rule Language

Chapter 1

Introduction

This chapter is dedicated to outlining the background and research goals of the study, providing insight into the research questions, research scope, and research outcomes. Additionally, the comprehensive structure for the remainder of the thesis is presented herein.

1.1 Background and research goal

In recent times, enterprises in the engineering domain are under pressure to swiftly and effectively adapt to evolving market requirements [6, 7]. This has prompted new initiatives in the engineering domain, including customized production, Industry 4.0, smart factories, and more. Consequently, there is a pressing need to elevate the level of automation across the entire product lifecycle. Given its pivotal role in determining the functional quality of the product, as well as the value of manufacturing and service, product design emerges as the most important component of the product life cycle [8]. Simultaneously, advancements in hardware and software have facilitated the interconnection of diverse systems, accelerating the digital transformation of product design.

Digital transformation is a multifaceted concept encompassing various dimensions, including technological, organizational, and social aspects [9], or more succinctly, technology and actor dimensions [10]. Despite the diverse interpretations, a common thread lies in leveraging information and communication technology to foster new capabilities in business, extending beyond trivial automation [11]. This involves the integration of a range of modern information and digital technologies, such as Artificial Intelligence (AI), data analytics, digital twins, industrial robots, and blockchain [12]. The shift from a document-centric paradigm to a

model-centric or knowledge-based (KB) paradigm becomes essential for harnessing unstructured data and isolated information systems [9].

Presently, many companies continue to adhere to the conventional document-centric approach for product design [13]. Distinct engineering groups operate within their respective domains and employ individual software tools. Upon completing their tasks, these groups pass on their work downstream by transferring documents, which are subsequently read and translated into formats compatible with the downstream software. In some cases, engineers may develop monolithic tools to aid their specific processes. However, the diverse business processes and the heterogeneous nature of data types lead to interoperability challenges [14, 15], hindering the seamless integration of these tools. As a result, human involvement persists throughout the workflow, limiting the level of automation and hindering the adoption of digital technologies in product design.

Knowledge-Based Engineering (KBE) and semantic modeling can play pivotal roles in the digital transformation of product design. KBE advocates a design paradigm focusing on the modeling of product design knowledge, with the primary objective of enabling efficient reuse of design knowledge for the rapid generation of product variants, as shown in Figure 1.1 a). Initially, KBE focused on the parametric modeling of geometric knowledge within products and later expanded its scope to encompass non-geometric knowledge. The KBE paradigm involves building applications to represent design knowledge and automatically execute predefined engineering tasks. Figure 1.1 b) depicts a schematic diagram of a KBE application. These applications explicitly represent product data and design processes using specific knowledge representation languages. This representation enables them to automatically generate expected outcomes, such as geometric models and reports, by processing given inputs, such as requirements and constraints, in explicit forms.

Concurrently, semantic modeling offers a foundational software technology for representing the captured design knowledge. Semantic data involves abstracting entities with descriptive information and depicting relationships among entities to structure data based on its inherent meaning. The Semantic Web serves as a common framework for building semantic models, enabling data to be shared and reused across application, enterprise, and community boundaries [16]. Its goal is to transition the existing “web of documents” into a “web of data”, creating a global database where machine-interpretable metadata facilitates the interconnection of relevant information. This vision aligns with the core principles of KBE, which prioritize knowledge capture and formalization to enable efficient knowledge reuse.

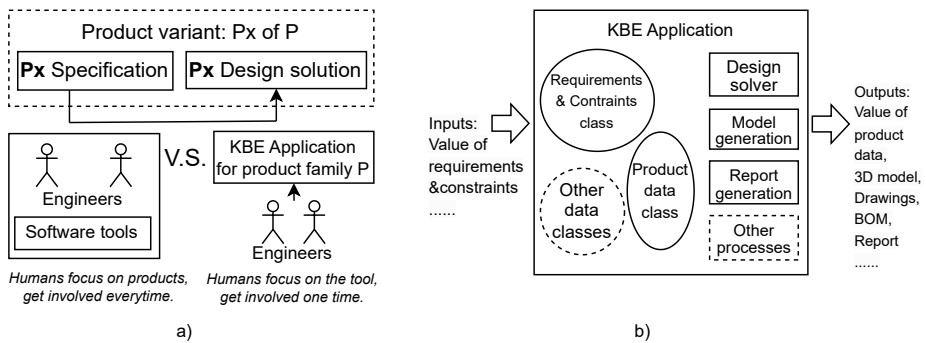


Figure 1.1: KBE design paradigm. a) Comparison with conventional design paradigm; b) Schematic diagram of a KBE application.

The goal of this research is to propose a practical framework to facilitate the transition from document-centric paradigm to knowledge-based paradigm in engineering design, leveraging the principles of KBE and semantic modeling. The knowledge-based paradigm emphasizes the building of knowledge models to capture the designers' intent, enabling the rapid generation of product variants in response to changes in intent. This paradigm involves the digital transformation of the product design knowledge, encompassing both the product data model and the design process model. Meanwhile, semantic modeling provides promising underlying technology for knowledge representation. By transforming the design culture and toolchain, a design platform can be developed to execute current engineering tasks efficiently. Moreover, various digital tools can be seamlessly integrated with this platform by reusing existing engineering knowledge. This evolving capability enables sustainable acquisition of new functionality, facilitating prompt responses to evolving market requirements.

1.2 Research questions

To achieve the research goal, three research questions (RQs) are identified to facilitate the knowledge modeling of product design. RQ1 elaborates on the motivation, necessitating the introduction of RQ2. While RQ3 offers a solution for enhancing the user experience, cooperating with the application guided by RQ2. These RQs are outlined below:

RQ1: *What are the benefits of using KB paradigm in product design and analysis tasks? Why not just use general-purpose programming languages for design automation?*

As the KBE paradigm requires a cultural shift in product design from case-by-case to knowledge-centric approaches, demonstrating tangible benefits is crucial to motivate designers. How do knowledge representation languages differ from general-purpose programming languages? Are they simply newly invented “programming languages”?

RQ2: *How to represent different components in a design problem in context of KBE and semantic model? What is the knowledge to be represented and what representation approaches to use for it?*

How can the philosophy of KBE contribute to the digital transformation of a product design process? How can the semantic web stack be utilized to construct knowledge models and KBE applications?

RQ3: *How to compose design workflow in a user-friendly manner with help of low-code platform and NLP?*

The design applications have to be adaptable and open to incorporating new capabilities since design processes involve diverse disciplines and evolving requirements. The integration with low-code platforms and NLP could serve as a demonstration of this adaptability. Furthermore, these tools can in turn facilitate the user-friendly composition of workflows that represent new capabilities.

1.3 Research scope

This research focuses on software technologies that facilitate the digital transformation of engineering design. Specifically, it explores the use of the KBE paradigm to guide a cultural shift from case-by-case to knowledge-centric product design, leveraging semantic modeling for knowledge representation. While the approaches to solve specific engineering problems are not in the scope of this research. In other words, the research does not aim to create new algorithms for specific engineering problems.

1.4 Contributions

The following papers constitute the contributions of this research. Each of them offers contributions to one or multiple RQs, as illustrated in Figure 1.2.

[Paper A]. Zhang, Liang, Brikene Berisha, and Andrei Lobov. "A parametric model of umbilical cable with siemens NX considering its reliability." IFAC-PapersOnLine 54.1 (2021): 187-192.

[Paper B]. Zhang, Liang, Anna Olsen, and Andrei Lobov. "An ontology-based KBE application for supply chain sustainability assessment." Resources, Environ-

ment and Sustainability 10 (2022): 100086.

[Paper C]. Zhang, Liang, and Andrei Lobov. "Extending design automation by integrating external services for product design." 2021 IEEE 19th International Conference on Industrial Informatics (INDIN). IEEE, (2021).

[Paper D]. Zhang, Liang, and Andrei Lobov. "Interoperability in automating engineering tasks: An illustration with pipe routing application." IECON 2023-49th Annual Conference of the IEEE Industrial Electronics Society. IEEE, 2023.

[Paper E]. Zhang, Liang, and Andrei Lobov. "A Low-code KBE Solution for Engineering Design: a Pipe Routing Case Demonstration.", 2024, Under review.

[Paper F]. Zhang, Liang, and Andrei Lobov. "Semantic Web Rule Language-based approach for implementing Knowledge-Based Engineering systems.", Advanced Engineering Informatics (2024), Accepted.

1.5 Thesis structure

The structure of this thesis is as follows:

Chapter 1 introduces the research by outlining the background and identifying the research objectives, questions, as well as presenting the research scope and the contributions.

Chapter 2 explains the related theoretical foundation and the related works, introducing the digital technologies adopted for the digital transformation of engineering design.

Chapter 3 provides a comprehensive summary of the contributions in terms of the objectives and relevance to the research.

Chapter 4 discusses how the contributions answer the RQs and as well as gives some additional remarks about the RQs based on the understanding obtained through the research.

Chapter 5 concludes the thesis by summarising the finds of the research and provides suggestions for future research in this domain.

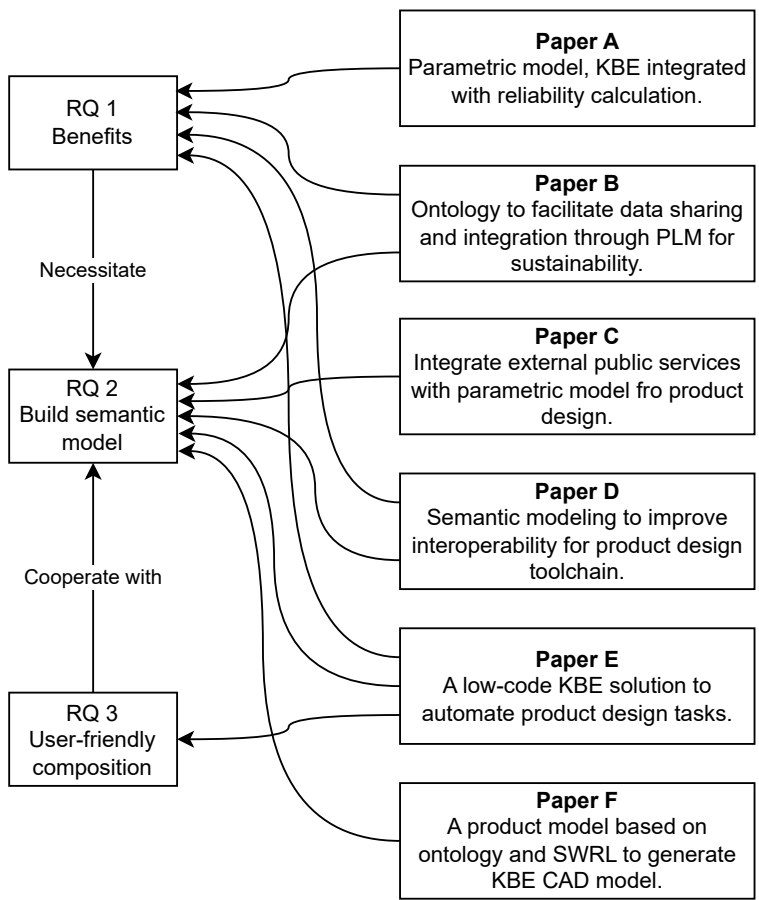


Figure 1.2: The relationship between research questions and contributions.

Chapter 2

Theoretical foundation, Related Works and Definitions

The digital transformation of product design involves the digitalization of the design knowledge, emphasizing the need to formalize a typical product design process. A current trend in product design is the transition from traditional experience-based design to Knowledge-Based (KB) design [8], making KBE methodology a natural candidate for facilitating design knowledge representation. The complexity of multi-disciplinary design often necessitates collaboration with different design teams, highlighting the importance of good extendability and interoperability in design applications. These qualities ensure easy functionality extension and seamless interaction with various software tools.

The rapidly evolving web technology provides promising solutions for building such applications. The semantic web stack, including ontology, Semantic Web Rule Language (SWRL), and SPARQL (SPARQL Protocol and RDF Query Language), serves as the underlying technology for knowledge representation. Service-Oriented Architecture (SOA) guides developers in creating loosely-coupled functional modules, providing a solution to integrate these black boxes building blocks, promoting the required extendability and interoperability. Additionally, the low-code workflow platform is gaining attention for its user-friendly interaction, a key factor for the success of an application. This chapter introduces these theories and related works, and defines some important concepts, forming the theoretical foundation of the proposed solution for the digital transformation of product design.

2.1 Product design

Ref. [1] gives a good description on design process. A product design process can be conceptualized as the identification of a set of design variables and the determination of specific values for each variable within that set, with the aim to meet the functional and performance requirements under the constraints of the variables. Functional requirements define what the design should do, and performance requirements define how well the design should do. For complex products, the design can not be defined at beginning stage of design process, in some industries such as aircraft and naval architecture, the design process can be divided into three stages based on the level of complexity:

1. Conceptual design: analyse the requirement specification to determine the main concerning concepts of the design and find a range of alternatives. The methods in this period can be mainly based on statistics, e.g., some previous designs as reference.
2. Preliminary design: determine the main design variables by a comprehensive analysis and optimization.
3. Detailed design: conduct the design for every subsystems of the product at detailed level.

The design process can be formalised as a constraint satisfaction problem (CSP), which consists of three components [17]:

1. Variable set **X**: The design variables represents the product, spanning a design space. $\{X_1, \dots, X_i\}$
2. Domain set **D**: Each domain D_i consists of a set of allowable values, $\{v_1, \dots, v_k\}$ for variable X_i .
3. Constraint set **C**: Specify allowable combinations of values of variables.

Thus, many search algorithms can be employed to solve the CSP, such as backtracking search, local search and etc.

In real product designs, there are often some design objectives for which it must seek the minimum (or maximum) values. Thus, this type of design problems can be formalised as constraint optimization problem (COP). COP can be seen as a CSP that includes objective functions to be optimized [17]. A COP for design process involves design variables, feasible domains, objective functions, constraints,

dependent/intermediate variables, the functions to calculate these variables. A typical form is as follows:

$$\text{Minimize } f_m(\mathbf{X}) \quad \text{for } m = 1, 2, \dots, M \quad (2.1)$$

$$\text{Subject to } g_i(\mathbf{X}) = 0 \quad \text{for } i = 1, 2, \dots, I \quad (2.2)$$

$$h_j(\mathbf{X}) \leq 0 \quad \text{for } j = 1, 2, \dots, J \quad (2.3)$$

$$\mathbf{X} = [X_1, X_2, \dots, X_n] \quad (2.4)$$

$$X_i \in D_i \quad \text{for } i = 1, 2, \dots, n \quad (2.5)$$

, where $f_m(\mathbf{X})$ are the objective functions, \mathbf{X} or X_1, X_2, \dots, X_n are design variables, D_i are feasible domains, $g_i(\mathbf{X})$ are equality constraints and $h_j(\mathbf{X})$ are inequality constraints. The performance requirements are often reflected as dependent/intermediate variables, which are represented in $g_i(\mathbf{X})$ and $h_j(\mathbf{X})$.

Most COPs for design process are multi-objective optimization (MOO), which means the objective functions may be contradicts to each other. Thus, there is not a single solution but a solution set called Pareto-optimal set. The solving methods for MOO can be roughly classified into two types:

1. Classic methods: To convert the multi-objectives to single object, e.g., weighted sum method, ϵ -constraint method, weighted metric method.
2. Intelligent methods: E.g., particle swarm optimization (PSO), multi-objective genetic algorithms.

If a product design problem can be formalised like the above-mentioned form, it is not far from solving it. However, the realities of modern design can be complex [1]:

1. Multi-objectives function: The design attributes are no longer separate functions. They must be grouped together to achieve a realistic optimum design. Furthermore, these functions commonly involve analytic processes executed through specialized software tools rather than simple arithmetic functions.
2. Multi-disciplinary: The disciplines in different domains interact and may not be taken into account at same level of detail.
3. Multi-team participation: The experts from different domains need to cooperate to handle the multi-disciplinary problem. These experts often work with their own tools (such as FEA, CFD and etc.) to support decision-making.

4. Distributed design team: It is often not possible to keep all teams connected on the variable level, and other teams have to be used to avoid divergence.
5. Lead time domination: The balance between the pressure to shorten the lead time and the effort to improve the final design quality always exists. A logic manner to store the previous design solutions is important so that the new design can be generated rapidly by adapting the previous designs.

Due to these features, modern product design often involves multi-disciplinary design and optimization (MDO). These characteristics pose challenges in realizing a concise model of the product design problem. Ref. [1] provides three types of MDO system implementations, as illustrated in Figure 2.1. In the figures, \mathbf{x} represents design variables, \mathbf{p} denotes parameters (constants) such as loads specified in requirements, and $\mathbf{y}(\mathbf{x}, \mathbf{p})$ indicates the behavioral variables obtainable through analysis process.

The geometry-less MDO system (Figure 2.1 (a)) is applicable in cases where clear mathematical models of the product exist, limiting its capability to reference-based product design. The grid-perturbation method (Figure 2.1 (b)) is suitable for cases where only small perturbations are allowed, but topology variations are not possible. The model-in-loop MDO system (Figure 2.1 (c)) is an advanced generative approach to implement MDO since the models can be automatically updated in the loop, but a high-fidelity product modeling system is necessary. Although the representativeness of these three types for all product design systems can be debated, it can be seen that there is shared data involved in a product design process, such as CAD models, analysis models, requirements, analysis results, and design solutions. And there are some disciplinary processes that can be reused for specific repetitive design and analysis tasks.

Traditional design paradigms, such as manual interaction with software and ad-hoc programs, are difficult to handle MDO design tasks. For instance, some analysis tools require the geometric model as input for conducting analytical processes, but automatic generation of a geometric model from design variables is not supported in manual interaction paradigm. Ad-hoc programs, being hard-coded for specific tasks, face challenges in extending functions or interoperating with other tools. Therefore, a methodology supporting MDO is needed to facilitate the transition from traditional design paradigms by representing all data and disciplinary processes in the design tasks in “proper forms”. This enables the generation of the MDO workflow in Figure 2.1 (c) with ease, so that the MDO workflows can be formalised freely and be performed automatically. The “proper forms” should address the interoperability and extendability issues to facilitate the digitalization of product design process, as well as the digital transformation at organizational

level. KBE is an approach designed to represent design knowledge, facilitating the construction of a high-fidelity product modeling system with fine interoperability and extendability performance.

2.2 Semantic Web Stack

Semantic data refers to data enriched with explicit semantic content, surpassing the scope of relational data by including additional information describing the semantics of entities and relationships, encompassing connections and consistency constraints [18]. With the aim to make the content of World Wide Web more machine-understandable by adding semantic metadata, Tim Berners-Lee coined the term "Semantic Web" in 1999 [16]. The World Wide Web Consortium (W3C) describes Semantic Web as "*The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries*" Consequently, the Semantic Web is often referred to as 'Linked Data' or 'Web of Data,' aiming to transform the Web into a global database with machine-interpretable metadata that interconnects related information. This conforms to the philosophy of KBE, which underscores the importance of enabling knowledge reuse via the capture and systematization of knowledge. The well-known Semantic Web stack was introduced to standardize data and explicitly establish relationships among data elements. This Semantic Web architecture incorporates underlying technologies such as ontology, SWRL (Semantic Web Rule Language) and SPARQL (SPARQL Protocol and RDF Query Language) for semantic data processing, as illustrated in Figure 2.2.

Digital resource can be roughly classified into two types. Sometimes it is referred to as a document to emphasize that it is human readable, or as object to emphasize that it is something which is more machine readable in nature [19]. Semantic data can make data more machine-readable, thus it is believed to play a crucial role in enabling effective tool integration and fostering collaboration across various domains. In medical informatics, there is a trend to move from a data model technocentric approach to semantic data representation in order to meet the need for data sharing between different systems, such as public health, research [20]. In the manufacturing field, the standards like OPC UA (Open Platform Communications Unified Architecture) [21] facilitates seamless communication and interoperability among diverse manufacturing systems and devices by employing a standardized and semantically rich data model.

Within the design domain, there has been a notable shift from document-centric approaches to data-centric design, exemplified by techniques such as model-based system engineering (MBSE) and Knowledge-Based Engineering (KBE) [22, 23, 24]. These advancements underscore the importance of semantic data representa-

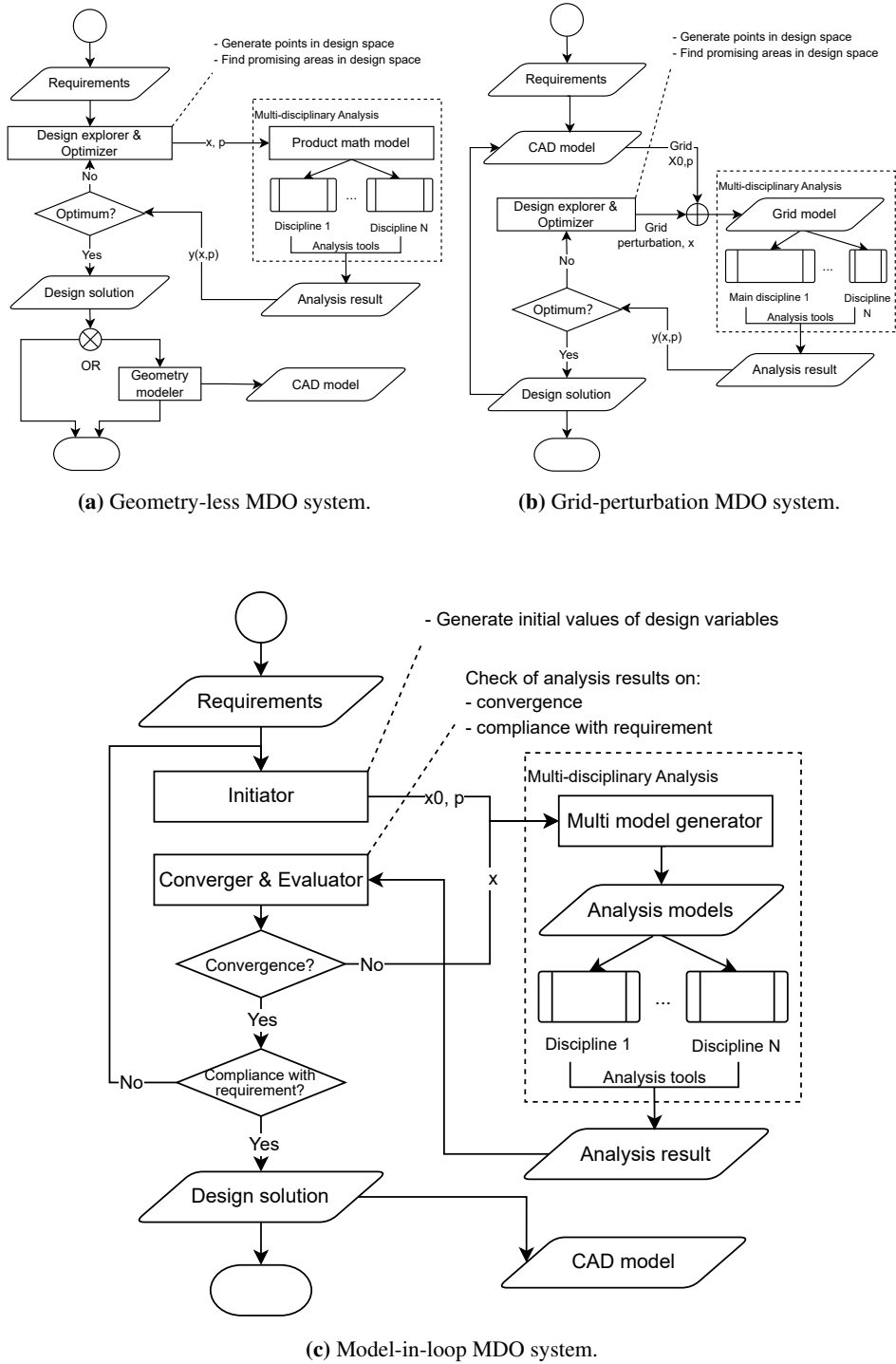


Figure 2.1: Schematic representation of 3 types of MDO system implementations. (Adapted from Ref. [1])

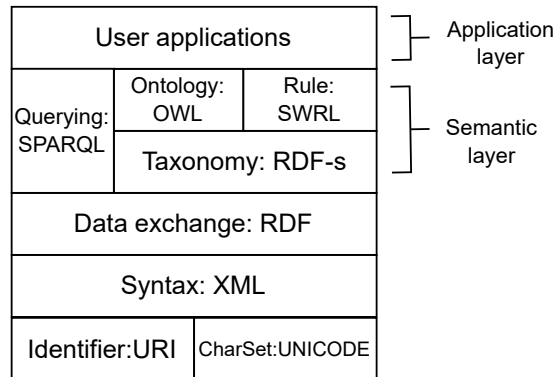


Figure 2.2: A simple version Semantic Web Stack. (Paper F)

tion in tool integration and collaboration enhancement. Embracing a data-centric approach and harnessing semantic data representation can significantly improve reusability and interoperability among software tools and systems, thereby enhancing automation and decision-making processes [25, 26].

Ontology, as a powerful tool, can serve as a repository for storing and organizing semantic data[27]. According to Ref. [28] and Ref. [27], an ontology is a formal, explicit specification of a shared conceptualization. The term “conceptualisation” refers to the abstract model of some phenomenon by extracting the relevant information from the real-world phenomenon containing infinite information. The term “formal” indicates that the representation should be in some sort of well understood logic to make itself machine-readable [27, 29]. “Explicit” refers to the fact that the type of concepts used, and the constraints on their use are explicitly defined [27], which means the relations and attributes related to the objects are pre-defined. While the term “shared” reflects that the knowledge captured in the ontology is accepted by the related community rather than private to some individuals [27, 29, 30].

Ontology serves as a formal and standardized means to define concepts, relationships, and properties within a specific domain, fostering the harmonization of different data sources. Ontology elements often include classes, object properties, data properties, individuals and axioms. By defining common concepts, relationships, and properties in problem domains, ontology significantly enhances interoperability among diverse data sets, enabling the seamless integration and exchange of information across various systems and domains [31, 32, 33, 34].

Moreover, since ontology is stored in graph form, the ontology-based knowledge

base is easily extendable, which proves valuable in applications with evolving data schemas. In summary, ontology is considered a supporting technology for establishing the knowledge base, providing a unified repository for storing and exchanging knowledge in the product design process. OWL is a formal language built on top of RDF (Resource Description Framework) for representing and sharing ontologies. It allows to describe concepts in an unambiguous manner based on set theory and logic [35]. Therefore, OWL (Web Ontology Language) is frequently employed to specify the data schema in KBE applications.

SWRL is a rule language for the Semantic Web that allows users to define rules that can be applied to RDF triples for inference. It provides a formalism for expressing rules that enhance the expressive power of ontologies, particularly those expressed using the OWL. SWRL was developed as a W3C Member Submission and provides a set of built-in rule constructs that can be used to create rules for reasoning about ontologies.

SPARQL is a query language and protocol specifically designed for querying and manipulating RDF data. SPARQL provides users a standardized and efficient way to perform queries on RDF datasets, retrieve specific information, and extract meaningful patterns from linked data. Therefore it plays a crucial role in the Semantic Web ecosystem. SPARQL and SWRL can be used together to query and infer information from semantic data.

2.3 Knowledge-Based System

Knowledge-Based System (KBS) is the AI systems based on a general-purpose search mechanism trying to string together elementary reasoning steps to find complete solutions [36]. Sometimes, it is also called expert system because it aims to capture human experts' knowledge to do reasoning. A KBS consists of user interface, knowledge base and inference engine [37], among which the latter two are the distinguishing features. A knowledge base is a set of sentences. Each sentence is expressed in a language called a knowledge representation language and represents some assertion about the world. Inference engine is to infer new sentences from old ones [36].

The first KBS is the DENDRAL program [38] developed by Ed Feigenbaum, Bruce Buchanan and Joshua Lederberg at Stanford. It is used to solve the problem of inferring molecular structure from the information provided by a mass spectrometer. In this KBS, the knowledge of human chemists is represented as if-then rules, thus the analysis on substructures of molecule can be automated. For example, a rule to identify a ketone (C=O) subgroup (which weighs 28) is as follow

[38, 36]:

If M is the mass of the whole molecule;
 $\wedge x_1$ and x_2 are the positions of two peaks;
 \wedge (a) $x_1 + x_2 = M + 28$;
 \wedge (b) $x_1 - 28$ is a high peak;
 \wedge (c) $x_2 - 28$ is a high peak; and
 \wedge (d) At least one of x_1 and x_2 is high

Then there is a ketone subgroup. (2.6)

By embodying many well-known but not explicitly-expressed patterns of peaks in the spectrum in the form of machine-processable rules, this KBS can suggest common substructures in the molecule, thus improving the efficiency of analysis of spectrum.

The fact representation in DENDRAL is relatively simple, consisting of the position of low/high peak and mass described in first order logic. The following is a fact description from another KBS example in Ref. [17]:

$$\begin{aligned} \text{Relevant}(\text{page}, \text{query}) &\iff \exists \text{store}, \text{home} \quad \text{store} \in \text{OnlineStores} \\ &\wedge \text{Homepage}(\text{store}, \text{home}) \\ &\wedge \exists \text{url}, \text{url}_2 \quad \text{RelevantChain}(\text{home}, \text{url}_2, \text{query}) \\ &\wedge \text{Link}(\text{url}_2, \text{url}) \\ &\wedge \text{page} = \text{Contents}(\text{url}). \end{aligned} \quad (2.7)$$

This rule defines a webpage that is relevant to a user-input query. The symbol \wedge means the logic relation “AND”. Even the logical definitions of basic fact (e.g., OnlineStores, Homepage, Link and etc.) are given, it is not enough to apply an inference algorithm to obtain a set of relevant webpages for the query. Because the function (or operator) “Contents(url)” is not defined and seems improper to be defined in description logic. This HTTP procedure is better to be implemented as **procedural attachment**, which is obviously different from the reasoning procedure based on description logic.

From the two examples, it can be seen that a core concept of KBS systems is the utilization of domain-specific knowledge representation languages to represent domain fact or rules, rather than general-purpose programming languages (GPL). The rule representation language involves syntax and semantics. The syntax defines how the fact is expressed based on the given words, while semantics defines the meaning of these words, whereby the words are operators and operands. Many

rule description languages are based on first order logic or description logic, such as Prolog (first order logic), OWL and SWRL (description logic). The operands are often described as an individual of a user-defined set, while the operators are “predicate” relations.

After the rules are expressed in KB, KBS calls the inference engine to interpret the rules, applies them on KB and generates new facts. The KBS processes the rules at two different levels: knowledge level and implementation level. At knowledge level, only the facts and goals are specified so that the KBS can generate new facts to achieve the goal. In these scenarios, the semantics of operators can be understood by inference engine, thus all the processes can be handled by the inference engine. However, the semantics of operators in some scenarios can not be understood by inference engine, for instance the operator “Contents(url)” in rule 2.7. These operators are often proper to be implemented as black-box functions for some reasons, hiding the details to the inference engine. Procedural attachment are the special-purpose methods imposed for these particular black-box functions [39].

2.4 Interpretation of Rules

The interpretation of rules can vary significantly. According to Ref. [40], *“In the broadest sense, a rule could be any statement which says that a certain conclusion must be valid whenever a certain premise is satisfied Using the term “rule” as a synonym for “first-order Horn implication” has become common practice in connection with the Semantic Web..... Yet another kind of rules that is very relevant in practice is known as production rules..... Rule languages of this type apply a more operational interpretation of rules, i.e. they view rules as program statements that can be executed actively.”* Obviously, the differences between “deduction rules” and “program statements” have been noticed: *“It can be argued that the deduction rules of virtually any calculus could be expressed as logical rules of some suitable logic. But this logic is typically required to be very expressive, making it difficult or impossible to implement general-purpose reasoners that can process the logical theory that was derived from a set of deduction rules.”*

The author of this thesis agrees that it is impractical to represent all rules as “deduction rules” or “rules of inference”. But a product design system should be capable of representing both “deduction rules” and “program statements”. In this context, “deduction rules” are often associated with simple design rules, implemented using specific rule languages and stored in the ontology-based KB. On the other hand, “program statements” refer to complex design rules or processes, also called procedural rules in this thesis, implemented as black-box functions using general-purpose programming languages. For this latter type of rules, only the de-

scriptive information or metadata can be semantically stored in the ontology-based KB. The procedural attachment of a KBS belongs to this type. From the aspect of semantic web stack, the procedural rules operate at the application layer based on the semantic layer. This dual interpretation aims to accommodate the diverse nature of rules within a product design system.

2.5 Knowledge-Based Engineering

KBE is the evolution of KBS towards the specific needs of the engineering domain to reduce time and costs of product development [41]. The name of KBE may be ambiguous, as it seems to indicate the existence of engineering that is not based on knowledge. Actually, the term “knowledge” refers to rules, hence this name is highlighting that the KB approach focuses on the reuse of engineering rules (knowledge) by knowledge management techniques, e.g., capture, formalization, representation and integration [3].

KBE originally refers to the merger of AI and CAD technology [42, 41]. It takes advantages of the knowledge representation and reasoning competence of KBS, enhanced with geometry manipulation ability by cooperating with a CAD kernel. Additionally, the parametric modeling approach required by the declarative representation facilitates the data exchange with computer aided analysis (CAA) tools, which in turn enhances its data processing and computation ability. Considering the typical requirement to provide a geometric model for product design, effectively representing geometric data and the automatic generation of geometric models often emerge as critical tasks for KBE applications. KBE is not intended to replace CAA tools but rather to offer solutions for modeling product data and design processes. This enables the automatic transformation of a given product data model into the specific data required to run various analysis tools, in such a way that the design processes can be performed automatically [1].

As the boundary between KBE and other types of design software can be blur with regards to the data schema and rules at the implementation level [26], there are different KBE definitions. Ref. [1] defines it as

“KBE is engineering using product and process knowledge that has been captured and stored in dedicated software applications, to enable its direct exploitation and reuse in the design of new products and variants.”

Ref. [41] gives another definition:

“KBE is a technology based on the use of dedicated software tools called KBE systems, which are able to capture and systematically reuse product and process engineering knowledge, with the final goal of reducing time and costs of product development by means of the following:

- Automation of repetitive and non-creative design tasks;

- *Support of multidisciplinary design optimization in all phases of the design process.*”

These two definitions are based on the terms “knowledge”, “Knowledge-Based System”, thus it may bring questions like “What is knowledge capture?”, “How does it differ from common software applications?”

This thesis gives a definition focusing on the differences with common software applications:

KBE application is a software system that employs knowledge representation languages for fact and rule description and rule processors for new fact generation, with the aim to automate engineering tasks, such as design, analysis, optimization, where the fact is product data representation and rules are design processes.

The knowledge of engineering tasks includes how to represent the product (product data representation), and how to determine the values of design variables (design processes). The product data representation consists of the geometric and non-geometric design variables, performance variables and other explicit attributes of the product. while the design processes or rules include the functions to calculate performance variables, the functions to represent the constraints and other relations between variables. The rules can be simple descriptions such as “if-then” rules and simple arithmetic formulas, or they can take the form of complex black-box processes executed by invoking external tools, which are called procedural rules. The knowledge-representation language can be a new language defined by developers, or a library based on general-purpose programming language, as long as it is machine-processable and can represent the items of problem domain at the knowledge level.

There are two major fields focusing on design automation, namely KBE and Computational Design Synthesis (CDS). According to Ref. [43], “*CDS is a research area focused on approaches to automating synthesis activities in design. Resulting methods may be fully automated or interactive with the goals of automatically generating a range of alternatives, sparking creativity and innovation, automating tedious or time-consuming engineering tasks, and simply exploring the creative abilities of computational systems.*” Ref. [44] argues that CDS aims to support the early design stages through functional modelling and simulation-based design, while KBE focuses on design automation tasks involving geometric representation, parametric modelling and advanced CAD applications. However, if consider any repeatable processes as procedural rules, e.g., a simulation to give performance result, KBE can be seen as any type of automation of repetitive tasks in product development based on the reuse of knowledge [25]. In this sense, KBE and CDS have many similarities.

Figure 2.3 gives the author’s understandings about the relation between KBE and

CDS. Although both approaches aim to automate design tasks through computational methods, an explicit representation of knowledge is necessary in KBE, while it is not the case for CDS. Recently, CDS also imposes generative grammars to computationally encode knowledge about creating designs [45], aligning CDS and KBE more closely. However, there are still many hard-coded ad-hoc programs mainly focus on making the different computational tools work synthetically, without caring about integrating with external tools outside existing systems. On the other hand, there are some KBE applications based on pure reasoning by inference engine without integrating with external programs. These KBE applications are mainly built for some simple product design problems whose design process does not involve complex numeric simulation.

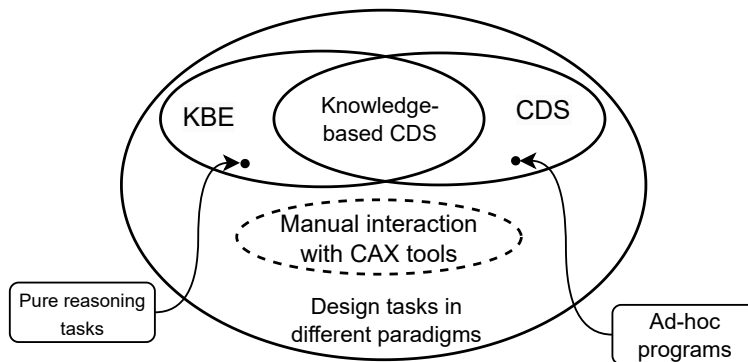


Figure 2.3: The relation between KBE and CDS.

The study of KBE dates back to the 1980s [41]. Several commercial parametric modeling languages have since been established for constructing KBE CAD models, including ICAD, GDL by Genworks, AML by Technosoft, and Knowledge Fusion by Siemens NX. While these tools offer significant conveniences to developers operating within the KBE design paradigm, it is not obligatory to build a KBE application exclusively on these languages. As long as the KBE application furnishes a declarative language for product model representation, capable of seamless cooperation with a CAD environment, it remains viable.

Meanwhile, there are already some existing framework to develop KBE application, e.g. MOKA [46], CommonKADS [47], AMAAD [48], KNOMAD [49]. Among them, MOKA is a highly-cited framework. It divides the lifecycle of a KBE system into six phases: identify, justify, capture, formalize, package, activate. MOKA proposes an informal model for knowledge capture and a formal model for formalization. The informal model represents the knowledge in ICARE

format: Illustration, Constraint, Activity, Rule and Entity forms, while the formal model represents the knowledge about product model and process model in UML [50]. MOKA has been implemented by some studies (e.g., Ref. [51, 52, 53]), these design cases show the potential of KBE in engineering design, especially in swiftly generating product variants. The typical approach involves documenting design information in MOKA ICARE forms, formalizing knowledge in UML, and utilizing this knowledge through KBE applications via the CAD system's API. But MOKA does not consider the knowledge captured and formalized in other forms, so it is hard to take advantage of the technical legacies, i.e., the monolithic ad-hoc tools and documents.

The process of addressing a real-world problem through various formalizations can be depicted in Figure 2.4. The KB method impose a knowledge level where domain experts can work with the domain-specific languages, which are generally more accessible than programming languages. While, the ad-hoc programs can be seen as the formalization at implementation level where the knowledge is represented and encoded in programming languages. Given that knowledge representation involves two levels at different levels of abstraction, the question arises as to which level is appropriate for a specific piece of knowledge. Despite this, few studies mention the limited expressive capability of domain-specific languages. This leads to RQ2: "How to represent different components in a design problem in context of KBE and semantic model? What is the knowledge to be represented and what representation approaches to use for it?"

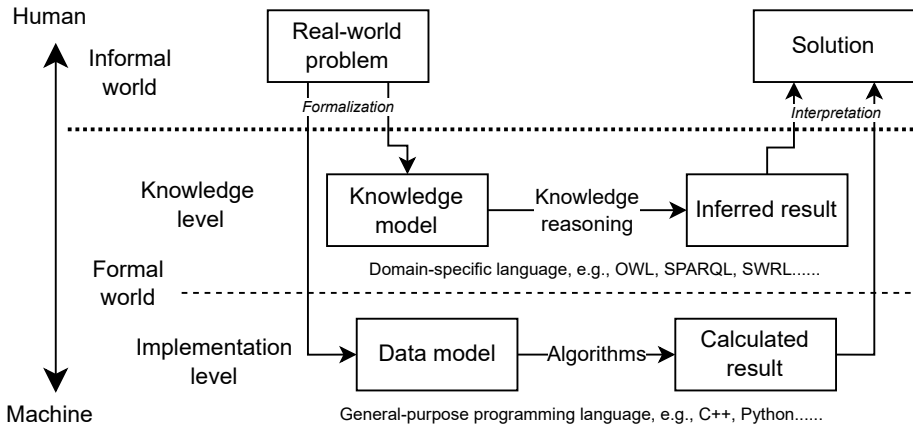


Figure 2.4: Formalization at different levels.

This thesis endeavors to broaden the intersection of KBE and CDS by presenting

ad-hoc program developers with a methodology to convert their tools into interoperable procedural rules. Consequently, integrating legacy digital assets with a comprehensive KBE application becomes feasible, thereby promoting knowledge reuse across multiple disciplines and teams.

2.6 Interoperability and Extendability

Interoperability issue exists in many multidisciplinary scenarios involving different systems [19, 54, 20, 55]. Interoperability is seen as a key success factor to build a digital system. International standard ISO/IEC/IEEE 24765:2017 [56] provides some definition of interoperability and extendability. In the standard, interoperability is defined as “3. the capability to communicate, execute programs, and transfer data among various functional units in a manner that requires the user to have little or no knowledge of the unique characteristics of those units.”, while extendability is defined as “1. the ease with which a system or component can be modified to increase its storage or functional capacity.” Ref. [57] provides a qualitative metric to evaluate the interoperability and extendability of software based on the three dimensions of the system development methodology. It can be assumed that the extension of functionality can be achieved by interoperating with other systems, thus a better interoperability can bring better extendability [5].

2.7 Service-Oriented Architecture

Service-Oriented Architecture (SOA) is a terminology in software engineering, referring to an architectural style that focuses on discrete services instead of a monolithic design. In SOA, an assembly of services communicates with each other, offering several advantages over monolithic design. It promotes the creation of modules that are loosely coupled and highly cohesive, enhancing interoperability and reusability [58]. These attributes align with the objectives of a well-designed KBE application, making SOA a noteworthy area of interest for KBE researchers. The continuous evolution of web-based technologies further enhances the appeal of SOA in KBE, with a focus on interoperability [59, 60, 61].

In recent years, microservice has become one of the latest architectural trends in the field of software engineering [62]. It can be seen as a variant of the SOA, providing finer-grained services than SOA. A microservice is a cohesive, independent process interacting via messages, with the aim to strip away unnecessary levels of complexity in order to focus on the programming of simple services that effectively implement a single functionality [63]. This feature inspires us to implement a procedural rule as a microservice since a microservice can be easily reused and the microservice style application is easy to extend. A difference is that each microservice usually maintains its own private database, whereas the method pro-

posed in this thesis encourages the use of a unified ontology-based KB for better interoperability.

2.7.1 Service composition

In a SOA system, service composition is an important topic. Service composition is the process of creating a new service by combining two or more existing services to produce the ideal service to fulfill the request [64]. It is useful when no single web service can satisfy the functionality required by the user. As a workflow can be seen as a composite service [65], the research on service composition can offer valuable insights into user-friendly workflow generation.

Given that a workflow can be viewed as a composite service [65], research on service composition offers valuable insights into workflow auto-generation. The typical approach to service composition involves analyzing the request, with a focus on input/output data or other metadata. Services are then discovered by recursively fulfilling output data until the initial input data is satisfied, and the identified services are returned in a specific order [66, 67].

Service composition approaches can be broadly categorized into two types: graph-based and AI Planning [68]. Graph-based approaches employ graph search techniques to generate service compositions, considering quality of services (QoS) or other metrics [67, 69]. On the other hand, AI planning-based approaches treat composition as a planning problem, incorporating metrics such as concept similarity, semantic matchmaking between web service parameters, and more [70, 71].

Despite all these efforts, automatic web service composition remains a challenging task. After 20 years since the beginning of this century, it appears to have reached a stagnating state where only little progress is made [72]. This means the semi-automatic tool for service composition are still necessary in many cases. NLP-based service search can be a promising exploration considering the growing capability of NLP technology [73].

2.7.2 Semantic Service Discovery

Although service discovery and composition are related tasks, they are often treated separately [68]. In certain microservices contexts, service discovery refers to obtaining the location (IP address and port) of the invoked service instance for load balancing purposes [74]. However, this thesis defines service discovery as the process of locating or recommending relevant services from a service repository based on provided keywords, aiming to identify services offering the desired functionality [73].

Semantic technologies such as ontologies and SPARQL have been employed in

service discovery, as demonstrated in references like Ref. [75, 69, 76]. This process generally involves searching the metadata of services within a service repository to find matches based on requested input and output data types. Service discovery is also implemented in domains like IoT [77], manufacturing [78]. However, there has been limited research on service discovery for product design, possibly due to the greater flexibility and complexity of services in design processes compared to those in machine-related contexts, particularly in terms of input and output data types.

The semantic representation of services is graph-based, illustrating concepts as nodes and relationships as edges. SPARQL query execution can be viewed as a process of matching a graph described by the query to corresponding ontologies, resembling a more sophisticated “syntactic search”. Alternatively, natural language, being closer to human understanding, can define relationships and concepts. Consequently, NLP can be employed to facilitate the discovery of services in a more user-friendly and intuitive manner.

2.8 Low-code Workflow Platform

The term “low-code” was firstly coined by Forrester Research in 2014, emphasizing visual interfaces to enable people, without a technological background, to create and deploy business apps without complex engineering [79]. Therefore, these platforms also offer enterprises a more economical way to fulfil the market and/or enterprises internal requirements. Since a significant portion of the code is already developed by developers in advance, users only need to visually configure the applications and make necessary adjustments to fulfill their specific needs [80]. The rapid and easy development brings many benefits such as addressing the shortage of professional developers, aligning business processes and IT tools, better privacy due to less development outsourcing and etc [6, 81, 82, 83].

Additionally, low-code platforms provide visualized languages for representing workflows and a workflow engine for executing these composed workflows. This capability enables seamless integration with pre-defined functional blocks, streamlining the process of generating workflows in user-friendly manner. This type of visualized languages can act as the knowledge presentation, especially for procedural rules. BPMN is a standardized language that enables businesses to visually depict their internal processes, providing organizations with a common way to communicate and understand these procedures. Its graphical notation makes it particularly well-suited for use within low-code platforms.

Some research have explored the automatic generation of BPMN diagrams from natural language (NL) text. The typical steps include pre-processing, syntax and

semantic analysis, fact type extraction, mapping to BPMN elements, and generating spreadsheet-based descriptions [84]. However, according to Ref. [85, 84], the auto-generation of BPMN from complex text description remains challenging. This challenge leads to the consideration of implementing NLP-based BPMN generation in scenarios with predefined fundamental building blocks, aiming to reduce the search space [76]. Within a low-code platform, each BPMN block can be associated with a specific service. Consequently, the NLP-based search for identifying blocks with the highest textual similarity can be equated to the service discovery process.

Chapter 3

Contributions Overview

This chapter introduces the significance of each paper to the thesis and outlines the contributions of the six papers to the RQs. The relationship of each paper with the RQs is illustrated in Figure 1.2. All papers (Paper A-F) contribute to RQ1 by demonstrating the benefits of the KBE paradigm in terms of reusability, extendability, and interoperability. These advantages highlight the necessity of implementing the KBE paradigm in product design, leading to the exploration of RQ2—how to construct KBE applications to transform the conventional design paradigm? Papers B-F contribute to RQ2 by showcasing various approaches to represent different types of knowledge, encompassing product data and design processes. Given the representation of knowledge in semantic formats, the user-friendly composition of workflows for specific design problems becomes crucial. This issue is encapsulated in RQ3, which is addressed by Paper E.

3.1 Paper A: A parametric model of umbilical cable with Siemens NX considering its reliability

3.1.1 Objective

Umbilical cable, serving as one of the key control equipment in the subsea oil and gas production system, is a typical customized product tailored to specific usage parameters, such as the installation site. The design of its cross-section can be considered as a variant derived from previous products. This type of design includes minor modification to previous products and some routine performance evaluation. Thus, a digital tool for this repetitive process is necessary.

This paper aims to showcase the benefits of integrating KBE CAD models with

performance calculation programs, with a focus on reliability in this context. This integrated approach yields valuable tools for designers, enabling the generation of product variants and providing a swift, visualized presentation of performance evaluation results in real-time. Such tools prove particularly advantageous in the design of products, like the cross-section of an umbilical cable, where variants are derived from previous iterations.

3.1.2 Relevance to the thesis

The Knowledge-Based Engineering (KBE) paradigm diverges significantly from the manual interaction with conventional CAD tools. Therefore, it is important to showcase the advantages of making this transition and provide illustrative scenarios where KBE excels, which aligns with RQ1.

This paper serves as the main response to addressing RQ1. It reveals the advantages of introducing a Knowledge-based Engineering (KBE) approach, including the rapid generation of variant CAD model and enabling the integration with automated calculations for product reliability. Through this, the paper contributes to the understanding of the transformative benefits brought about by KBE in the design process.

3.1.3 Contributions

This paper demonstrates a KBE CAD model extended with complex performance evaluation, specifically illustrated through reliability calculations in this paper. Compared with the conventional way, the benefits of KBE paradigm are significantly demonstrated in the scenarios where the design is a variant derived from previous products. The key contributions can be summarized as follows:

1. It demonstrates that KBE modeling empowers the swift generation of variant designs, coupled with seamless integration with external programs for complex performance evaluations like reliability calculations.
2. It highlights that KBE models transcend traditional geometric considerations, encompassing non-geometric parameters during the design process.
3. Product data can be transferred between upstream and downstream processes automatically, which saves significant time to get through the design loop.
4. The parametric models introduced lay a foundation for the prospective automation of design and optimization processes, reflecting the forward-looking implications of adopting KBE methodologies.

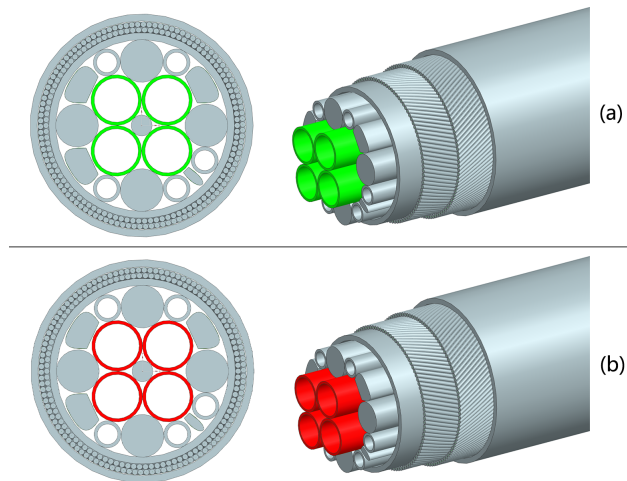


Figure 3.1: Result demonstration [2]. (a) Green: reliability check passed. (b) Red: reliability check failed.

3.1.4 Results

This paper establishes a parametric model of steel tube umbilical cable considering its reliability. By leveraging parametric modeling and KBE, the conceptual design stage can incorporate not only geometric considerations but also non-geometric aspects. Designers can generate a CAD model of steel tube umbilical cable according to specific projects requirements. Additionally, the performance evaluation results, e.g., reliability in this paper, can be seen in an intuitive manner, see Figure 3.1. With the help of this tool, the time spent on design loop of the cross-section can be reduced.

3.2 Paper B: An ontology-based KBE application for supply chain sustainability assessment

3.2.1 Objective

Supply Chain Sustainability Assessment (SCSA) involves knowledge sharing among various stages and teams. With the growing significance of sustainable development, evaluating the sustainability of product supply chains has become a pivotal concern within product lifecycle management. However, challenges persist in sharing and integrating knowledge across different stages of the product: (1) Misunderstanding: Divergent teams employ varied terminology and conventions, potentially leading to misunderstandings unless explicitly articulated. (2) Not machine-

readable: Human intervention is necessary to comprehend data and create wrappers for converting it into a format compatible with domain applications. Because many applications in specific domains are developed without prior consideration for interoperability and integration with others. (3) Not human-readable: Updating or querying data in traditional machine-readable data formats (such as SQL databases) requires programming skills rather than a user-friendly approach for humans.

This paper endeavors to demonstrate a proof of concept for an ontology-based framework intended to improve the effectiveness of knowledge sharing and integration in PLM applications, with a specific emphasis on collaborative tasks like sustainability assessment. While the core objective of this framework is to enhance software development efficiency through ontology-based knowledge representation, it does not inherently contribute to the advancement of calculation methods or assessment frameworks for supply chain sustainability evaluations.

3.2.2 Relevance to the thesis

This paper showcases the execution of supply chain sustainability assessment within the KBE paradigm. The demonstration highlights the ability of ontology-based knowledge representation to streamline program development for collaborative engineering tasks, thus contributing RQ1. Meanwhile, the paper also demonstrate the method to semantically represent multi-disciplinary knowledge from various sources, contributing to the investigation of RQ2.

3.2.3 Contributions

The ontology-based knowledge representation facilitates the multi-disciplinary knowledge sharing and integration, simplifying the PLM software development, promoting the digital transformation and design automation. Specifically, the contributions are summarised as follows.

1. Ontology-based knowledge representation can provide a unified and flexible data format for the teams in collaboration, which can reduce the misunderstanding caused by various data formats.
2. Ontology-based knowledge representation with SPARQL query language makes the data exchange more human-readable than the conventional data exchange patterns.
3. A unified knowledge base can reduce the time spent on writing data wrappers, which increases the efficiency of software development.
4. A unified and flexible knowledge representation makes a unified knowledge

base possible. Thus, all the data involved in the collaborative engineering tasks can be reached globally.

5. High-efficient software development contributes to supply chain sustainability management.

3.2.4 Results

This paper introduces the development of an ontology-based Knowledge-Based Engineering (KBE) application for supply chain sustainability assessment, as demonstrated in Figure 3.2. The illustration emphasizes how semantic representation can enhance knowledge sharing and integration. Simultaneously, the paper explains the method of constructing KBE applications for collaborative engineering tasks, providing an example architecture for reference.

<pre> 1 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#> 2 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> 3 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> 4 PREFIX owl: <http://www.w3.org/2002/07/owl#> 5 PREFIX susass: <http://www.semanticweb.org/lianz/ ontologies/2021/10/sustainability_assessment#> 6 insert 7 { susass:Asset_cost_AC_sc1 a susass:Asset_cost_AC. 8 susass:Asset_cost_AC_sc1 a owl:NamedIndividual. 9 susass:Asset_cost_AC_sc1 susass:hasMembershipDegreeDistribution "[0.565,0.325, 0.085, 0.035, 0]".} 10 WHERE{} </pre>	<pre> \$python SusAssessCalculator.py synthetical score: 54.54, scores for each factor: [68.76, 42.35, 57.94, 60.88, 65.38, 63.07] comment: good </pre>												
a)	b)												
<pre> 1 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#> 2 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> 3 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> 4 PREFIX owl: <http://www.w3.org/2002/07/owl#> 5 PREFIX susass: <http://www.semanticweb.org/lianz/ ontologies/2021/10/sustainability_assessment#> 6 select ?supply_chain ?comment 7 WHERE { ?supply_chain susass:hasComment ?comment .} </pre>	<table border="1"> <thead> <tr> <th></th> <th>supply_chain</th> <th>comment</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>susass:comment_SC3</td> <td>"excellent"</td> </tr> <tr> <td>2</td> <td>susass:comment_SC2</td> <td>"general"</td> </tr> <tr> <td>3</td> <td>susass:comment_SC1</td> <td>"good"</td> </tr> </tbody> </table>		supply_chain	comment	1	susass:comment_SC3	"excellent"	2	susass:comment_SC2	"general"	3	susass:comment_SC1	"good"
	supply_chain	comment											
1	susass:comment_SC3	"excellent"											
2	susass:comment_SC2	"general"											
3	susass:comment_SC1	"good"											
c)	d)												

Figure 3.2: The demonstration of the KBE application [3]. a) Different teams input their data into the unified knowledge base. b) The calculation process. c) The query to get the assessment result. d) The assessment results of different supply chains.

3.3 Paper C: Extending design automation by integrating external services for product design

3.3.1 Objective

The representation of product data can be accomplished through a semantic schema. However, the case for process knowledge or procedural rules is more complex

compared to product data. Process knowledge or procedural rules refers to the sequence of operations to determine attribute values of instances. While knowledge-based systems or semantic languages offer specific modeling languages with built-in operations, such as basic math calculations and string manipulation, the implementation of complex rules at the procedural level often necessitates the use of general-purpose programming languages. This is particularly true when intricate rules require support from external libraries. Implementing such rules from scratch using knowledge representation languages can be challenging due to the absence of readily available libraries. This is the typical scenario for procedural attachment in a KBS.

This paper introduces a proof-of-concept for a KBE designer integrated with external services, using a simple case of wood-to-wood connection with different fasteners as an illustration. The procedural rule wraps an online calculator, provided by a professional organization, for connection capacity calculation. Simultaneously, the arguments of the online calculator are formalized as concepts in the rule language. The CAD model of the wood-to-wood pair is developed using a parametric method, enabling the rapid generation of design variants. Designers can manipulate the domain-specific language to formulate a design, after which validation is conducted, and the CAD model is generated to visually represent the design result.

3.3.2 Relevance to the thesis

In this case of wood-to-wood connection design, the external service acts as a procedure to calculate the performance variables, which are a component in a design problem. This paper showcases how to wrap an external service as a procedural rule and integrate with KBE application. Meanwhile, it shows again the capability of KBE CAD model to rapid generate design variant. In summary, this paper contributes to RQ2 by showing the approach to representing a procedural rule.

3.3.3 Contributions

This paper demonstrates an automatic design tool based on KBE method and an online professional calculator. This tool can quickly generate a 3D model (DFA files) according to the customers' demands. The contributions to the RQs are summarised as follows.

1. This case illustrates that ad-hoc calculation tools can be viewed as reusable design knowledge represented in a hard-coded manner.
2. This type of knowledge can be wrapped as procedural rules and the integrated with KBE CAD model within a SOA KBE application.

3. Additionally, it demonstrates that extracting calculation results from an external service not originally designed for SOA can be time-consuming and unrobust.
4. The advantages of representing CAD model in KBE manner are shown again. Rapid generation of product variants and automated performance evaluation become feasible.
5. It underscores the imperative of translating human-readable knowledge into machine-readable formats. For instance, dimensions of standard components such as bolts are often stored in files optimized for human comprehension rather than conforming to universally accepted digital schemas. This practice poses a hindrance to the rapid development of digital systems.

3.3.4 Results

A proof-of-concept KBE designer integrated with external services is demonstrated in this paper. Through wrapping external services as procedural rules like the manner in Figure 3.3, the hard-coded knowledge can be reused in a KBE application. With the help of KBE CAD templates, the product variants can be rapidly generated and the performance evaluation can be conducted automatically. Some design results generated by the KBE designer are shown in Figure 3.4.

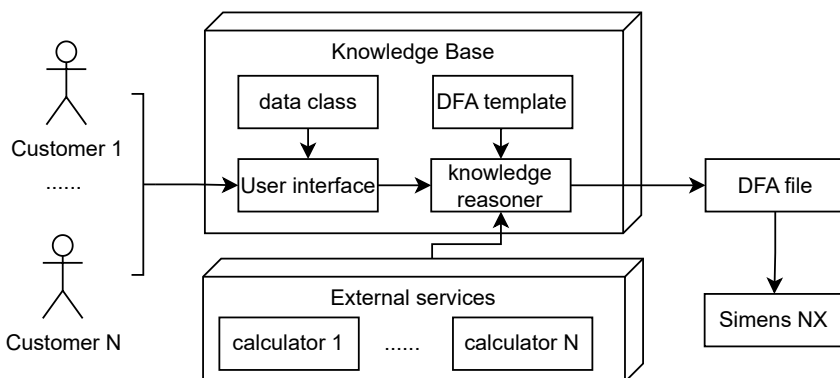


Figure 3.3: External services wrapped as Procedural Rules. (Adapted from Ref. [4].)

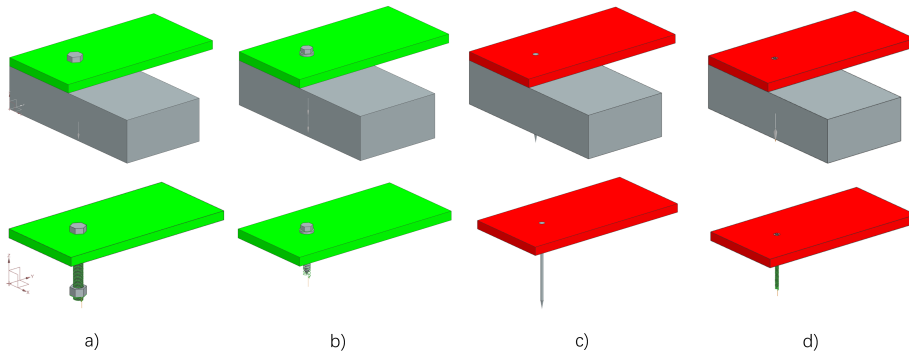


Figure 3.4: Four wood connection designs with different fasteners: (a) bolt; (b) lag screw; (c) nail; (d) wood screw. Green: connection capacity check passed; Red: connection capacity check failed. (Adapted from Ref. [4].)

3.4 Paper D: Interoperability in automating engineering tasks: An illustration with pipe routing application

3.4.1 Objective

The modern design involves multi-disciplinary optimization and multi-team participation. Thus there may be different tool preferences among engineers in different problem domains or also organizations working within the same problem domain. A successful digital transformation of product design should provide an interoperable solution to be able to support different engineering tools and formats in order to form a toolchain.

This paper presents an approach for building such an interoperable solution supporting toolchain. The approach is illustrated using a pipe routing application based on genetic algorithm (GA) and A* algorithm. The output can be handled by end users in web browser, Siemens NX and AVEVA tools. The solution can also be extended to include other tools. The evaluation of the proposed approach is conducted using a qualitative metric that considers the three dimensions of the system development methodology.

3.4.2 Relevance to the thesis

This paper proposes an approach for building an interoperable solution to support different engineering tools and formats and illustrates using a pipe routing application based on genetic algorithm (GA) and A* algorithm. This research contributes

to RQ1 by showing how this semantic representation can promote interoperability for a toolchain forming. And meanwhile it contributes to RQ2 by showing how to represent design knowledge in semantic schema.

3.4.3 Contributions

This paper proposed an approach to build an interoperable solution to integrate different software tools, which is illustrated with a KBE pipe router case. This approach can enhance the interoperability of the KBE application, facilitating to form the toolchain for automating engineering design tasks. The contributions to the RQs are summarised as follows.

1. The ontology-based representation can store the semantics of the design variables in the product data model, facilitating the harmonization and integration of diverse data sources.
2. To bridge different data schemas, SPARQL queries and case-specific adaptors are suggested as an intermediate layer.
3. Retrieving semantic data through SPARQL and adaptors proves to be a more straightforward process than parsing data from various formats. Consequently, integrating legacy design tools with the proposed KBE architecture becomes more effortless.
4. Compared with ad-hoc solutions, the proposed KBE solution has better interoperability and extendability.

3.4.4 Results

This paper presents an approach for constructing an interoperable KBE solution. The method advocates the use of semantic representation and ontology-based storage for the product data model, thereby enhancing the interoperability of KBE applications. By employing SPARQL queries and case-specific adaptors as an intermediate layer, integrating different software tools becomes more straightforward, facilitating well-informed decision-making during the design stage. The architecture of the KBE application, developed using the proposed approach, is illustrated in Figure 3.5. To exemplify the effectiveness of the approach, a KBE application for a pipe routing case is developed.

This KBE pipe router can route pipes using the A* algorithm and visualizes the results in three software tools: AVEVA, a web browser, and Siemens NX, as depicted in Figure 3.6. Ad-hoc tools typically represent equipment as cuboids to align with the A* routing algorithm's requirement for discrete cuboids. Developers of

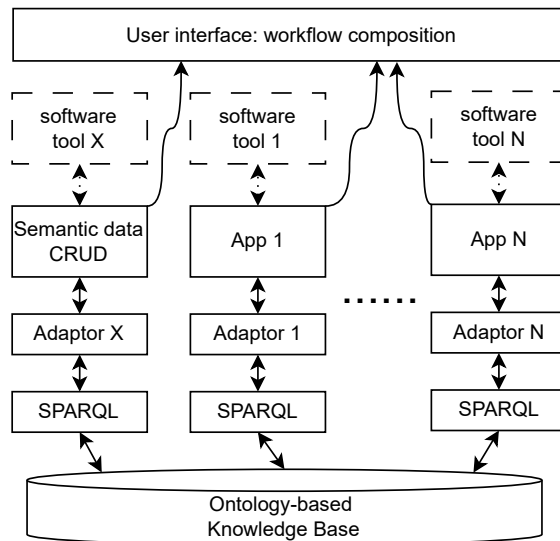


Figure 3.5: The proposed architecture for automating engineering tasks powered by semantic data [5].

ad-hoc tools then write parsers to extract this data from raw non-semantic formats like STEP. However, when integrating with tools that render equipment as cylinders, such as the tool shown in Figure 3.6 b), developers face the challenge of parsing STEP files and extracting relevant cylinder components, which can be a tedious process. Conversely, if the equipment is represented in a semantic schema, data retrieval becomes more straightforward, as the data is already within the KB. Utilizing semantic data representation, the KBE pipe router can exchange information and interact with different software tools, fostering a smooth workflow and enhancing collaboration among engineering teams utilizing diverse software environments.

3.5 Paper E: A Low-code KBE Solution for Engineering Design: a Pipe Routing Case Demonstration

3.5.1 Objective

The solving of a product design problem can be seen as a workflow consisting of some predefined subroutines. Low-code platform provides a visualized language to represent workflows. From aspect of KBE, these predefined subroutines are procedural rules. Thus, a user-friendly composition of design workflow based on these procedural rules becomes important for a KBE application for design

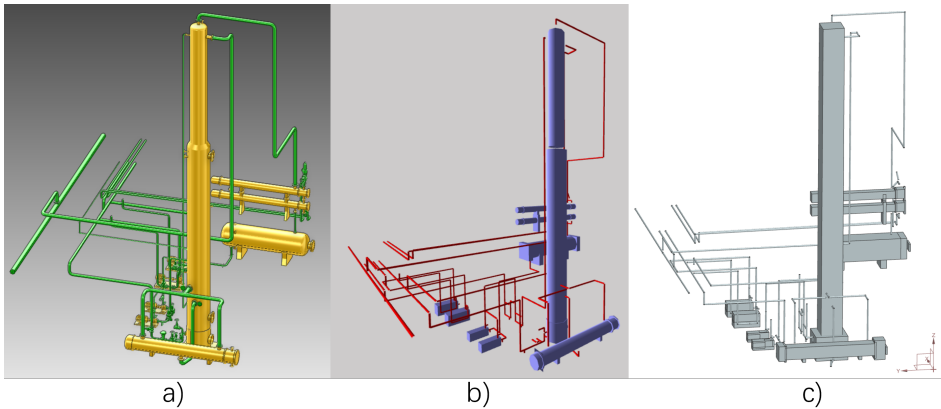


Figure 3.6: The KBE pipe router interoperates with different tools. a) AVEVA. b) Web browser. c) Siemens NX. [5]

automation.

The design variables of design problems can be represented in semantic schema, while how to integrate procedural rules with low-code platform at both knowledge level and implementation level needs to be explored. SOA has the potential to facilitate the reuse of functions, which can be seen as procedural rules in context of KBE. Meanwhile, Natural Language Processing (NLP) technology has the potential to convert the user requirement in natural language (NL) to the executable workflow in a formalised format such as Business Process Model and Notation (BPMN) diagram, bringing higher automation and user-friendly interaction to the low-code KBE solution.

This paper presents a practical approach for constructing a low-code KBE solution by employing semantic modeling of product data and design rules, SOA, and BPMN. Furthermore, this solution is extended with NLP technology to semi-automate the creation of customized, executable workflows.

3.5.2 Relevance to the thesis

This paper illustrates the development of a low-code KBE solution that can be further enriched by NLP technology. Through this KBE application, a user-friendly design workflow can be composed, contributing to all three RQs. The semantically-enhanced synergy of KBE and low-code platforms is demonstrated, addressing RQ1. An approach to constructing a low-code KBE application is proposed, thus contributing to RQ2. Additionally, the exploration of utilizing NLP technology to

compose an executable workflow contributes to RQ3.

3.5.3 Contributions

This paper introduces a pragmatic approach to building a low-code KBE solution, emphasizing semantic modeling of product data and design rules. Notably, this approach enables end-users to easily compose workflows, addressing the shortage of professional software developers in many engineering design scenarios. The key contributions can be summarized as follows:

1. The approach enables the representation of product data and design rules in a unified and semantically-enhanced manner.
2. The adoption of Web APIs as carriers for design rules in engineering design tasks enhances interoperability and reusability.
3. Developers are provided with a universal method to extend the solution's functionality, efficiently integrate new tools, and enhance development productivity.
4. The approach simplifies user engagement at different levels, e.g., direct CAD manipulation, ontologies, NLP.
5. The semantically enhanced service deception can facilitate service discovery and recommendation.
6. End users can easily compose workflows for repetitive tasks in a user-friendly manner with the assistance of an NLP-based workflow composer, even without the involvement of professional software developers. The approach, facilitated by visual interfaces and reusable services, offers the flexibility needed to efficiently manage specific yet routine engineering design tasks.

3.5.4 Results

The proposed approach can facilitate the development of a low-code KBE application, which provides end users user-friendly workflow composition. The unified semantic representation of product data and design rules facilitates service discovery and recommendation, enabling the user-friendly workflow composition. Figure 3.7 illustrates the conversions from NL workflow description to an executable BPMN workflow by the NLP-based workflow composer. If the NL workflow description is of high quality, the composer can provide a complete and directly executable workflow. While if the user gives low quality workflow description, the composer engages in reasoning to identify relevant web APIs based on input and

output checks. The rule for input and output checks is to ensure that all inputs of the web APIs originate from the user, see an example in Figure 3.8.

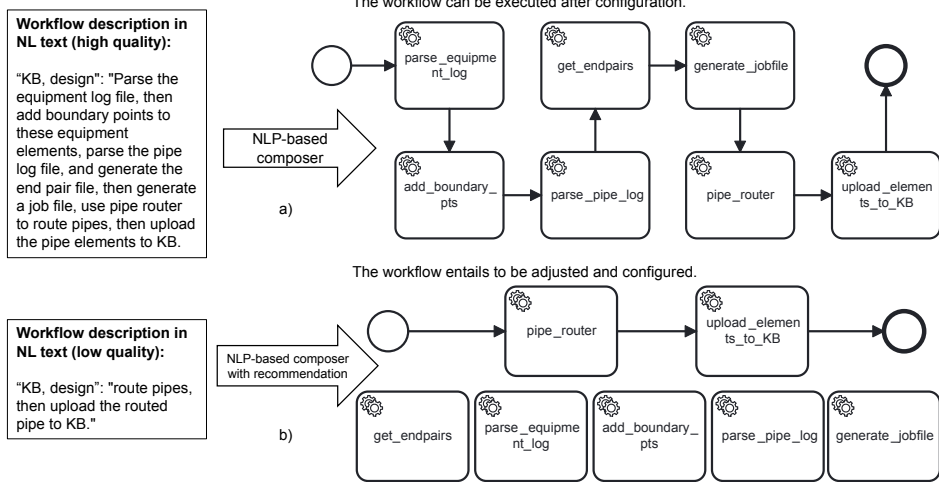


Figure 3.7: The BPMN workflows generated from NL description. a) From high quality text; b) From low quality text. (Paper E)

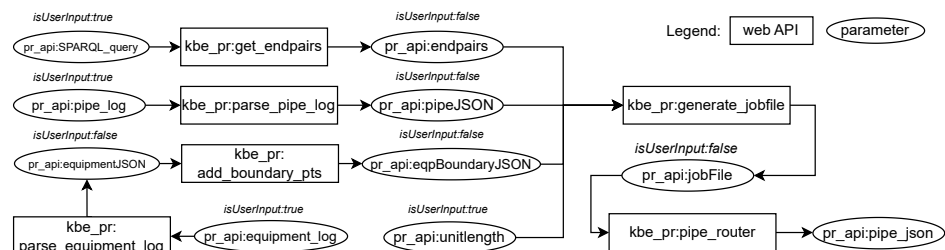


Figure 3.8: The recommendation process for relevant services based on input and output check. (Paper E)

3.6 Paper F: Semantic Web Rule Language-based approach for implementing Knowledge-Based Engineering systems

3.6.1 Objective

The Semantic Web stack, including OWL and SWRL, offers promising formats for representing product data and rules, facilitating knowledge sharing and reuse in engineering. However, many KBE applications in product design only treat

ontology-based knowledge bases as graph databases, overlooking the reasoning capabilities provided by the semantic web stack. Specifically, geometric model construction is often encapsulated as a black-box process. This type of reuse often leads to fragmented solutions, where relevant knowledge, particularly regarding geometric model reconstruction, is scattered across multiple sources.

This article introduces an approach for creating a cohesive knowledge base using OWL and SWRL, leveraging the reasoning capabilities offered by the semantic web stack. Notably, geometric model (re)construction can be achieved using KBE language code snippets and the string processing capabilities of SWRL. To demonstrate this approach, a shaft design case, frequently used in research on product design, serves as a demonstrator to provide a conceptual proof.

3.6.2 Relevance to the thesis

This paper demonstrates the potential for seamless integration and knowledge sharing in the field of product design through the application of the semantic web stack and KBE. In the paper, design rules are broadly categorized into two types: universal rules and case-specific rules. The options for implementing design rules include SWRL and black-box procedural rules, with the choice depending on the universality of the rules. This discussion contributes to RQ2, providing a practical guideline for selecting the rule implementation.

3.6.3 Contributions

This paper introduces an approach that utilizes KBE and semantic web stack to build KBE application for product design. To validate the feasibility of the proposed approach, a case study involving the design of a four-section shaft is conducted as conceptual proof. The key contributions can be summarized as follows:

1. Design rules can be broadly categorized into two types: universal rules and case-specific rules.
2. The implementation manner for rules is determined based on their universality, with SWRL suitable for universal rules and black-box functions for case-specific rules.
3. The proposed approach allows the construction of a cohesive knowledge base that encapsulates universal rules, leveraging the reasoning capabilities provided by the semantic web stack.
4. The (re)construction of geometric models can be achieved using KBE language code snippets and the string processing capabilities of SWRL. The

resulting CAD models can be generated in KBE languages and visualized through interaction with the CAD kernel.

5. This high-cohesive knowledge representation form can offer easy sharing and reuse together with robust interoperability, facilitating future functionality extension.

3.6.4 Results

The proposed approach in the paper is illustrated in Figure 3.9. The core ideas of the proposed approach include (1) utilizing the semantic web stack to build the knowledge model of product design, encompassing product data and design rules; and (2) codifying the geometric models in KBE languages and constructing CAD models through the string processing capabilities offered by SWRL.

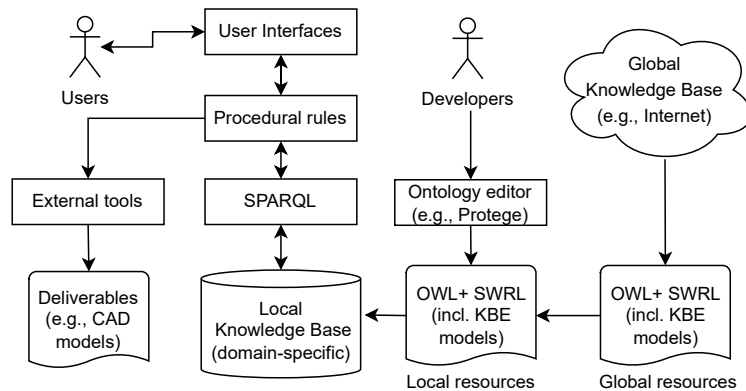


Figure 3.9: The proposed architecture based on KBE and semantic web stack. (Paper F)

In the case study, an KBE application based on semantic web stack is developed. Protege [86] serves as the ontology editor for editing the model in OWL and debugging the SWRL rules. The local KB is implemented using Apache Jena Fuseki SPARQL server [87], augmented with the open-source SWRL reasoner Openllet [88]. The semantic model of a transmission shaft is illustrated in the schematic Figure 3.10. The different components of a shaft are abstracted as parametric models in an object-oriented manner. The development and debugging of this model can be done in the local ontology editor Protege. Figure 3.11 (a), (b), and (c) show the TBox of the semantic representation of product data. Based on the concepts in TBox, a shaft instance can be created in ABox. The universal rules for deriving the inferred properties are implemented as SWRL rules, including the construction of the CAD model represented in KBE languages, as shown in Figure 3.11 (d) and

(e). Then, this complex of OWL and SWRL can be uploaded to the ontology-based knowledge base. A user application, including the design processes represented as procedural rules, can be built upon the knowledge base containing the semantic representation of design knowledge.

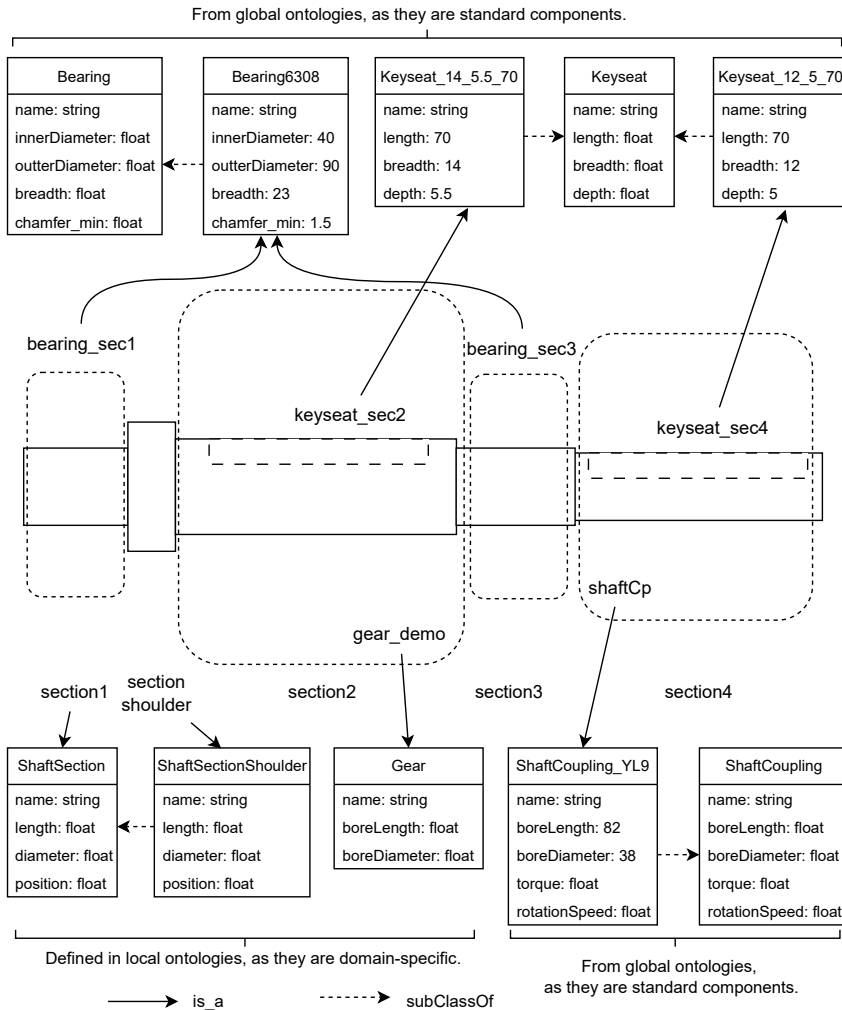


Figure 3.10: Schematic diagram of a transmission shaft. (Paper F)

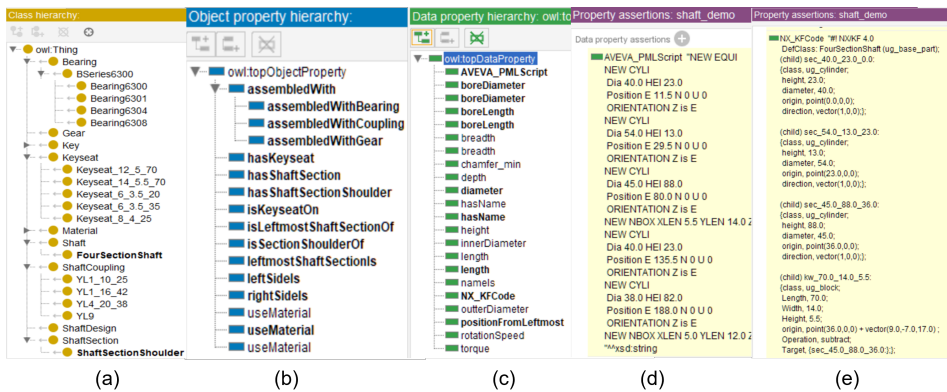


Figure 3.11: The development and debugging in local ontology editor Protege. (a) Class definitions; (b) Object property; (c) Data property; (d) Inferred geometric model of shaft in AVEVA PML; (e) Inferred geometric model of shaft in Siemens NX KF. (Paper F)

Chapter 4

Discussion

This chapter provides answers to the RQs based on the contributions introduced in Chapter 3. Meanwhile, additional remarks are offered with regards to the understanding of KBE.

4.1 RQ1: Benefits to use KB paradigm and why not just programming languages

The KB paradigm for product design and analysis tasks is significantly different from traditional paradigms, such as manual interaction with conventional CAD tools and hard-coded ad-hoc software application. The benefits derived from the KBE paradigm stem from these distinctions in terms of reusability, interoperability and extendability.

In comparison to the conventional CAD paradigm, where human interaction with GUI is prevalent, a KBE application offers the advantages of automation in engineering tasks and the reusability of knowledge. The tacit knowledge inherent in GUI interactions is often not codified or explicitly represented, limiting the potential for knowledge reuse and hindering the automation of engineering tasks. Conversely, as demonstrated in the paper introduced in Chapter 3, KBE applications codify design knowledge. In this context, human engineers need only input a limited amount of product knowledge, such as the values of design variables or constraints. Subsequently, the repetitive design processes can be executed by programs, leading to the automatic generation of product variants.

Compared with hard-coded ad-hoc software application, the benefit of KBE mainly lies in interoperability and extendability. To gain a better understanding, it's helpful to explore the characteristics of ad-hoc applications and the various types of

KBE applications.

Ad-hoc software applications are hard-coded, thus the design knowledge is encoded in programming languages. In this context, users lack intermediate languages to articulate design problems, restricting them to engineering tasks predefined by the developers of the hard-coded software. Since the encoded knowledge is at the implementation level, users face challenges in modifying the knowledge unless they are skilled to alter the source code, a task often too complex for engineers. Essentially, the extendability of ad-hoc applications is hindered by their hard-coded nature. Moreover, in terms of interoperability, ad-hoc applications exhibit poor performance. Designed without considering cooperation with other software, these applications often employ different data schemas, necessitating inevitable data parsing and conversion. Both functionality extension and interaction with other software require programming skills typically absent in users.

KBE applications can be categorized into three types, all employing parametric Object-Oriented (OO) representation of product data, see the illustration in Figure 4.1. The distinguishing factor lies in how the rules are executed. The first type is “classic” KBE, where all the rules are represented in a rule language processable by its rule engine. This is suitable for simple design problems where the design processes involve only built-in operators of rule languages, such as if-then rules, simple string processing, basic arithmetic calculations and some domain-specific operators supported by the rule engine. Users can define new rules themselves using the built-in operators provided by the rule language. However, when design processes become more complex, involving black-box processes executed by external program tools, such as intricate arithmetic calculations, functionality extensions to the rule engine become necessary.

Developers can implement these complex design processes as procedural rules or attachments, adding them as new operators to the rule engine. The second type, termed “extended” KBE, emerges as a result of functionality extensions to the rule engine, see Figure 4.1 b). Users can compose new rules using the extended rule language. However, if users need to modify procedural rules (e.g., if the granularity of the operator is unsatisfactory), developers (programmers) are required.

Sometimes, developers might choose not to integrate codified design processes into the original rule engine to avoid making it unrobust. Instead, the codified design processes are encapsulated as invocable functions, integrated with another rule engine. And a user interface for rule composition is provided to the user to define new rules. These types of procedural rules can also be called workflows. These workflows act as a bridge, retrieving knowledge from the knowledge base, executing calculations, and updating the knowledge base with the results. While

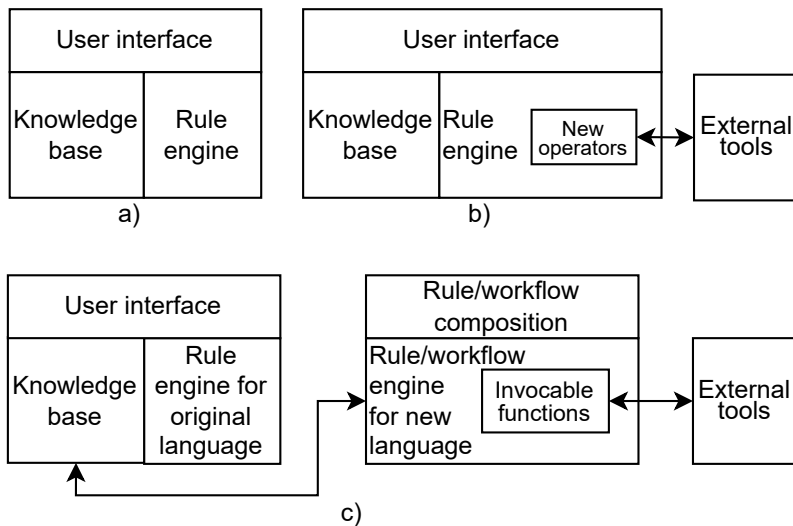


Figure 4.1: The illustration of the three types of KBE applications. a) "Classic" KBE. b) "Extended" KBE. c) "Flexible" KBE.

users can automate design tasks by representing different parts of their design problems in various languages, modifications and additions to functions still require intervention from developers. This is the third type of KBE, which can be called "flexible" KBE applications in this thesis, as it provides flexible manner to extend the functionality, see Figure 4.1 c).

While both ad-hoc applications and KBE applications facilitate the reuse of design knowledge and automation of predefined design problems, they differ significantly in terms of extendability and interoperability. The comparison between ad-hoc and KBE applications is outlined in Table 4.1, addressing the first half of RQ1, "What are the benefits of using the Knowledge-Based paradigm in product design and analysis tasks?" For simple design problems involving built-in operators, KBE users can extend functionality by modifying or adding rule expressions using the rule language. The interoperation with other KBE applications is straightforward as they share a unified product representation and rule language. For instance, integrating with CAD software that supports KBE features, like Siemens NX, is relatively simple, as the representations in different applications can be mapped with ease. In the context of design problems that entail complex processes, developers can extend functionality according to the framework of current KBE, introducing new operators or invocable functions. The interoperation with other tools is streamlined for developers due to the parametric representation of product

data in languages at the knowledge level.

In contrast to KBE applications, the functionality extensions of ad-hoc applications almost always necessitate developer intervention due to their hard-coded nature. Ad-hoc tools are typically designed without considering interoperability with other tools, requiring developers to handle the parsing and conversion of data schemas from various sources. Unfortunately, this process is often time-consuming and error-prone, given the absence of a unified parsing method. For instance, the lack of a common vocabulary in data schemas can result in misunderstandings and miscommunication between systems. Unless ad-hoc tools share a standardized data schema, the efficiency of data parsing becomes a persistent challenge. However, if a well-designed standardized data schema is implemented for a specific design case, this schema can already be viewed as a knowledge representation language. In this context, the data becomes shareable knowledge, representing the facts of the design case. Figure 4.2 illustrates that the inconvenience of ad-hoc data parsers compared to a unified knowledge representation.

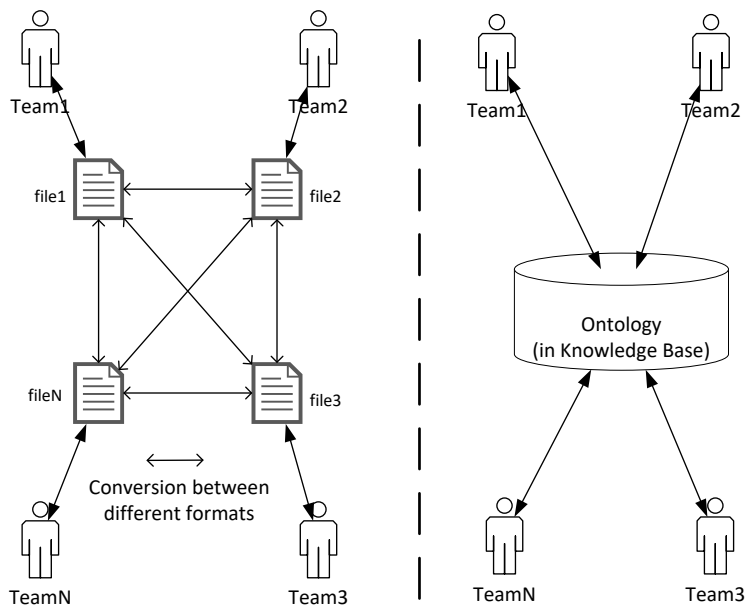


Figure 4.2: The comparison between ad-hoc data parsers and a unified knowledge representation [3].

In addition to data interoperability, extending knowledge represented in knowledge representation languages like OWL is simpler due to its support for open properties. Inserting a new attribute to an instance in OWL can be easily done without

modifying the class definition, as demonstrated in the SPARQL code in Listing 4.1. On the other hand, statically-typed languages like Java, C++, C# do not support this feature, as illustrated in the code snippet in Listing 4.2, which will throw an error. It can be envisioned that extending knowledge represented in these languages is not as straightforward as knowledge representation languages supporting open properties, as the modification to the class definition may cause unexpected troubles. Similar functionality can be achieved with Python, but it is not specifically designed for representing complex relations between various entities.

```
PREFIX kbe: <http://example/kbe.owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
INSERT {
  kbe:block1 kbe:hasDensity "7.8"^^xsd:float
}
WHERE {}
```

Listing 4.1: Add a new attribute to an instance using SPARQL without affecting other instances.

```
#include <iostream>
#include <string>

class Block {
public:
  std::string name;
  // Constructor with name parameter
  Block(const std::string& n) : name(n) {}
};

int main() {
  // Create a Block instance
  Block block1("block_test");

  // Attempting to dynamically add a 'density' property at runtime
  // This is not idiomatic and not supported in a typical C++ scenario
  block1.density = 7.8; // Error: 'class Block' has no member named '
  density'

  // Print the block1's name
  std::cout << "Name:_" << block1.name << std::endl;

  return 0;
}
```

Listing 4.2: Adding new attributes to an instance is not supported in C++.

This feature is very useful for recording designers' intent. As shown in Figure 4.3, designers' intent operates at the knowledge level, which can be effectively represented in semantic formats. In conventional manual interaction or ad-hoc solutions, capturing knowledge, i.e., recording designers' intent, is often not supported. The loss of designers' intent brings inconvenience in data parsing to find

it back, hindering the automation of design workflow. Additionally, designers' intent is flexible, meaning that it is often extended or modified. Thus, it is crucial to insert new attributes without modifying the class definition, necessitating the use of knowledge representation languages other than general-purpose programming languages. Noteworthy, as knowledge representation languages are supposed to operate at the knowledge level, although it is possible to do some operations at the implementation level, it is not proper to do so. Otherwise, it is essentially inventing another general-purpose programming language, which is unnecessary.

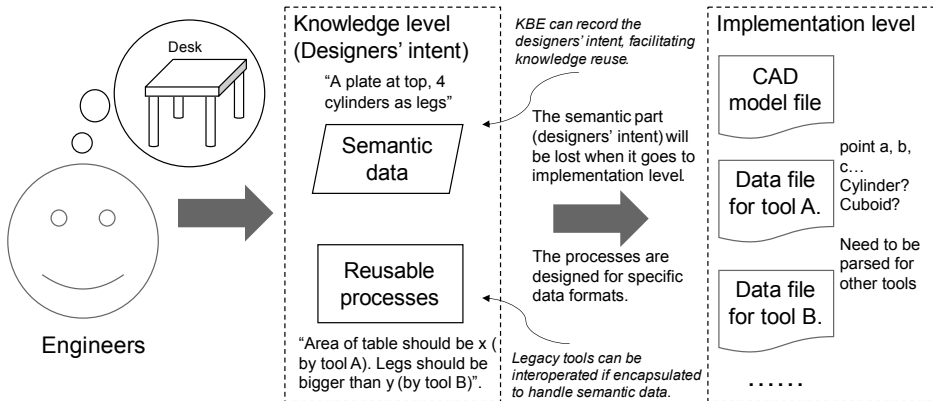


Figure 4.3: The designers' intent is at knowledge level.

The discussion highlights that a knowledge representation language provides users with a way to work at the knowledge level without delving into the details at the implementation level. Simultaneously, this language offers developers a unified "data schema", thereby enhancing the interoperability of KBE applications. This insight addresses the second half of RQ1, "Why not just use general-purpose programming languages for design automation?" In essence, the specialized knowledge representation language proves essential in bridging the gap between user-level domain knowledge and developer-level programming implementation, fostering a more effective and interoperable design automation environment.

Table 4.1: The comparison between ad-hoc and KBE applications

	Ad-hoc	“Classic” KBE	“Extended” KBE	“Flexible” KBE
Reusability /automation	Support	Support	Support	Support
Extendability	Users can not. Developers add new data classes and functions with programming languages.	Users add new data classes and rules with knowledge language. No need for developers.	Users add new data classes with knowledge language. Developers add new operators with programming languages.	Users add new data classes with knowledge language. Developers add new callable functions with programming languages.
Interoperability	Users can not. Developers write data parsers to integrate with other ad-hoc tools.	Users can translate the data and rules to other KBE systems. Developers can convert its data to integrate with other tools easily due to the explicit representation.	Developers can convert its data to integrate with other tools easily due to the explicit representation.	Developers can convert its data to integrate with other tools easily due to the explicit representation.
Remarks	Hard-coded. Difficult to support multi-disciplinary design.	Only for simple design problems, if without using procedural rules.	It is compact, but the extension to rule engine may make it unrobust.	Flexible to extend and interoperate with other systems.
Examples	Ref. [89]	Paper F	Ref. [39]	Paper A-E

4.2 RQ2: What knowledge to represent and what approaches for it

According Section 2.1, there are different components in a MDO workflow need to be represented, such as CAD models, analysis models, requirements, analysis results, and the disciplinary processes for specific repetitive design and analysis tasks. These knowledge can be classified into two types: the product data and the design rules. The knowledge of product data describes what variables can represent the product and its performance or other attributes, such as design variables, dependent/intermediate variables including performance variables, objective variables, constraint variables and etc. While the knowledge of design rules describes the restrictions to these variables, such as the feasible domain of some variables, the constraints to some variables, the functions to calculate some variables and etc.

Considering the important role of knowledge representation languages in a KBE application, it can be broadly stated that creating a KBE application involves representing design knowledge in existing knowledge representation languages, extending existing languages, or importing/creating new languages tailored to users' cases. These correspond the three types of KBE applications described in RQ1 discussion.

The architecture of "flexible" KBE, as suggested in this thesis, is a notable departure from the conventional design paradigm where design knowledge is often documented in materials like CAD models, documents, and isolated tools. The proposed KBE architecture encapsulates design processes as invocable functions integrated with a workflow language, specifically BPMN. This approach enables users to work at the knowledge level, as depicted in Figure 4.4. The figure shows a suggested, and thus, reference architecture listing suggested key languages to implement each of the components in the architecture.

Figure 4.5 further illustrates the transformation of product data and design processes into semantic formats for the recommended KBE architecture, as detailed in paper E and F. The design rules can be broadly categorized into two types: universal rules and case-specific rules. Universal rules are those that apply to classes in all cases, regardless of specific instances. It is reasonable to assume that universal rules typically have simple forms. For example, if a shaft section is paired with a bearing, then the design diameter of this section must equal the bore diameter of the bearing, regardless of the specific type of bearing. These rules primarily involve reasoning or simple arithmetic calculations, utilized to derive values of inferred attributes and relations. Consequently, they can be effectively represented using description logic-based SWRL. Storing SWRL rules alongside assertions in OWL format facilitates a high-cohesive form of knowledge representation, greatly

enhancing ease of sharing and reuse. This is “representing design knowledge in existing knowledge representation language”. The processes represented in the semantic layer operate at the knowledge level, aiming to enhance reusability and interoperability.

In contrast, case-specific rules are those that apply only to a particular case or subset of cases, typically having more intricate forms. For instance, a design process to select a bearing can be codified as a reusable rule given its operational conditions. However, this type of rule only applies in the designers’ own cases, as it is usually tailored to their specific case conditions and designers’ preferences. If other designers wish to reuse it, they typically have to adapt it according to their own preferences, such as preferred bearing series, diameter range, and so on. As there are more factors to consider for specific cases, these rules normally involve complex mathematical calculations, large-scale data processing, or processes with external tool dependencies. They are normally used to generate assertions as input to KB. Consequently, these rules are often implemented as black-box procedural rules, as encapsulating them into SWRL is considered uneconomical or challenging. Typically, it is sufficient to reuse the knowledge encapsulated in these case-specific rules in their black-box forms, as these rules do not universally apply to all cases. For this type of complex design processes, web API and BPMN are employed to compose a workflow capable of generating assertions to update the KB. This aligns with the notion of “importing new languages tailored to the users’ cases”. The processes codified in the application layer operate at the implementation level, with the aim of executing intrinsic functionalities.

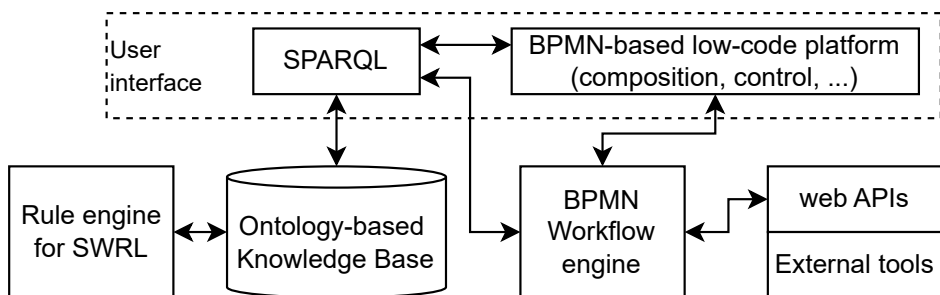


Figure 4.4: A reference architecture of “flexible” KBE extended with BPMN.

The product data model includes both geometric and non-geometric aspects. In the context of semantic representation, starting with the geometric aspects is suggested [5]. If the geometric model is not already represented in CAD software, the MOKA method provides an ICARE format to assist in capturing and formal-

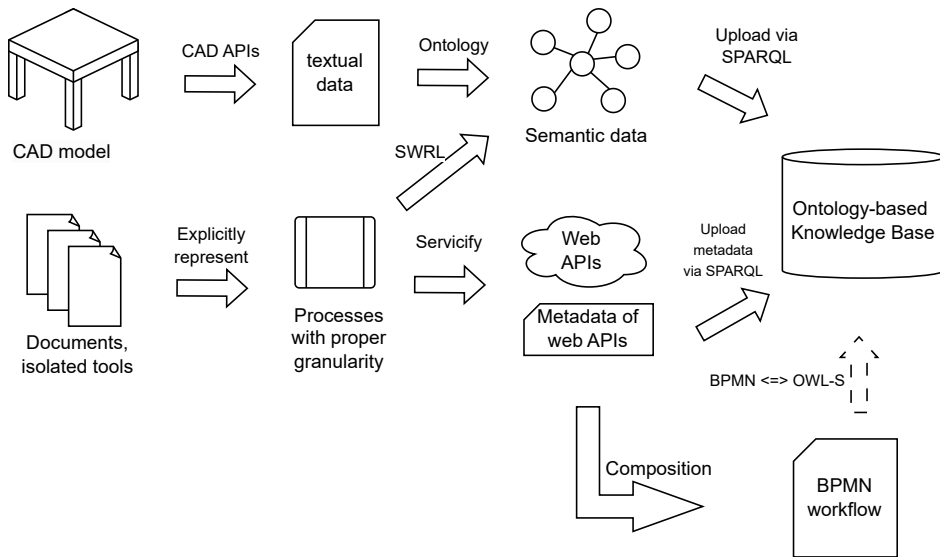


Figure 4.5: The flow to semantically represent the design knowledge.

izing geometric knowledge from scratch. This knowledge can then be written in a semantic format, as demonstrated in some examples in Paper A and C. On the other hand, when the geometric model is already represented in CAD software, the necessary ontology elements of the geometric model, such as classes, object properties, and data properties, can be identified by examining the attributes of objects within the CAD software. Examples of this process can be found in Paper D and E. A practical guide can be followed, illustrated in Fig. 4.5:

1. Identify the essential attributes required for reconstructing the geometric CAD model within the KBE context, considering the principle of minimal commitment [90].
2. Utilize APIs offered by CAD software to extract the necessary data from the geometric CAD model and convert it into a textual format suitable for further processing.
3. Construct a semantic model of the product data, incorporating both geometric and non-geometric aspects, using predefined classes and relationships defined in an ontology.
4. Upload the resultant semantic data to the KB, facilitating interactions with other tools.

Regarding design rules documented in text or implemented as isolated program code, SWRL, web API and BPMN workflow are suitable carriers for their implementation based on their complexities. For rules implemented as web APIs, unlike SWRL rules that are already stored in the KB, their metadata needs to be represented in a semantic format and stored in the KB. In this sense, KB also works as the service registry within SOA. A practical guide for handling this can be followed, as illustrated in Fig. 4.5:

1. Explicitly represent the processes described in documents and codify them into program code.
2. Decompose processes already written in program code into appropriate levels of granularity. Striking a balance between fine-grained and coarse-grained processes or services is crucial to minimize communication overhead while maintaining loose coupling [91].
3. For universal (often simple) rules involving only the built-in operators of SWRL, represent them in SWRL.
4. Encapsulate the complex (often case-specific) processes into web APIs and represent their metadata in a semantic format. OWL-S is a natural choice for metadata representation when working with ontology-based knowledge bases. The OWL-S ontology describes the overall structure of a web service using three main components: profile, process, and grounding.
5. Upload the semantic metadata of these web APIs to KB, thus facilitating the reusability of these processes and enabling efficient workflow composition.
6. For the composite rules constructed with finer-grained rules, which can be seen as workflow, represent them as BPMN workflows. If necessary, these workflows can be converted to OWL-S format and stored in KB.

4.3 RQ3: User-friendly design workflow composition

The semantic representation of product data and design processes forms the basis for users to articulate their engineering tasks, specifically the composition of workflows involving finer-grained rules. This composition occurs within a BPMN-based low-code platform, as depicted in Figure 4.4. Many low-code platforms provide a GUI for manual workflow composition, offering better user-friendliness compared to text-based rule editors. The continuous advancement of NLP technology further facilitates a more user-friendly approach to workflow composition.

The proposed approach, detailed in paper E, centers on the integration of an NLP handler to comprehend users' intentions and subsequently select appropriate services for the desired workflow. Figure 4.6 from paper E provides a visual representation of how the NLP-based workflow composer semi-automatically generates a workflow and how the workflow is executed.

The user articulates an informal workflow in NL text and submits the text to the service composer. The service composer processes the NL input to extract relevant features of the workflow, such as "serviceCategory". Metadata of related web APIs is obtained through a SPARQL query, filtering by attributes like "serviceCategory" in this example, and stored locally. The efficiency and accuracy of service discovery are enhanced by reducing the search space for the NLP-based workflow composer through SPARQL-based filtering [76].

The matchmaking process encompasses a similarity comparison between the NL text and the text description in the metadata of the filtered services. The service with the highest similarity is then chosen to construct the workflow. In situations where the NL workflow description is of high quality, the composer can efficiently generate a complete and directly executable workflow.

However, users unfamiliar with the rules might provide lower quality workflow descriptions. In such cases, a recommendation process can be initiated to suggest potential candidates capable of establishing connections between the input and output of selected services. The criterion for input and output checks is to ensure that all inputs of the web APIs originate from the user, as exemplified in Figure 3.8. Moreover, considering that the metadata of web APIs contains information beyond the text description, more sophisticated matchmaking could be made. For example, Ref. [92] proposes a matching techniques that operate on OWL-S process models, taking into account not only the input/output service profile. It can be envisioned that better matchmaking can be expected considering the potentials of advanced NLP technologies like LLM.

However, users unfamiliar with the rules might provide lower-quality workflow descriptions. In such cases, a recommendation process can be initiated to suggest potential candidates capable of establishing connections between the input and output of selected services. The criterion for input and output checks is to ensure that all inputs of the web APIs originate from the user, as exemplified in Figure 3.8. Moreover, considering that the metadata of web APIs contains information beyond the input/output, more sophisticated matchmaking could be made. For example, Ref. [92] proposes matching techniques that operate on OWL-S process models, taking into account not only the input/output service profile. It can be envisioned that better matchmaking can be expected considering the potentials of advanced

NLP technologies like LLM.

Subsequently, a list of web APIs is generated and represented as service tasks in the BPMN workflow draft. If the automatically generated BPMN draft is unsatisfactory, users have the option to make adjustments at this stage. Users can configure the workflow drafts, refining input and output variables and introducing glue code to relevant service tasks.

After these manual adjustments and configurations, a formal BPMN workflow is ready and can be executed by the low-code workflow engine. The low-code workflow engine identifies service providers, obtains the results of the services in the workflow, and, upon executing the BPMN diagram, returns the results, thereby completing the user-defined design workflow.

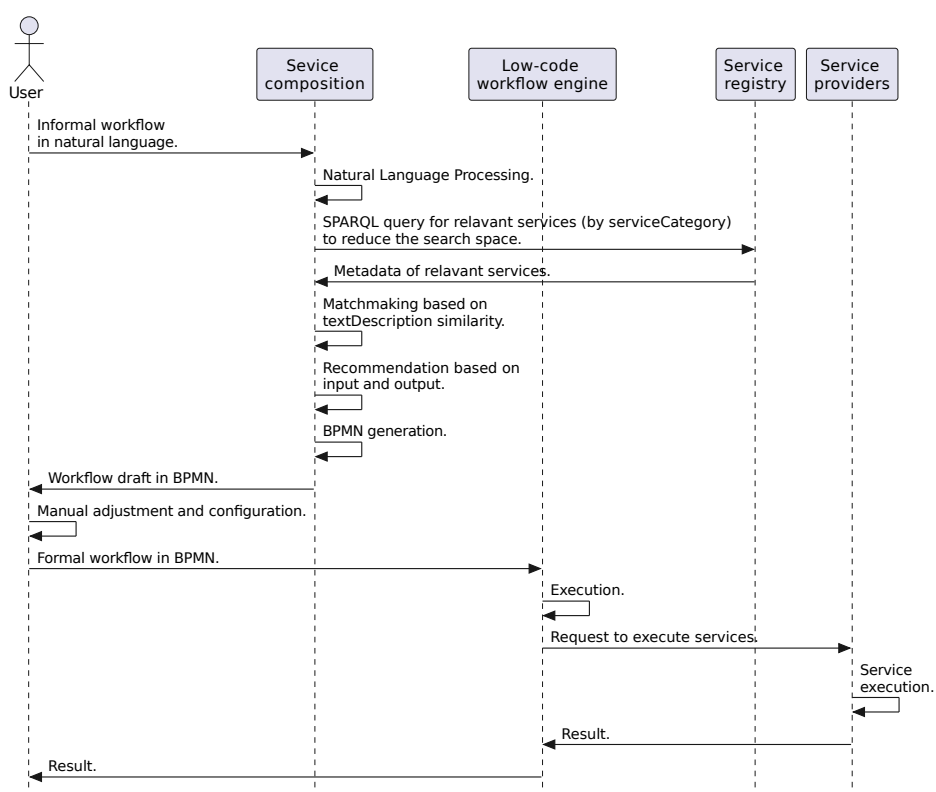


Figure 4.6: The NLP-based workflow composer and the process to execute a workflow (Paper E).

4.4 Additional remarks

This thesis elaborates on the extension of "classic" KBE to incorporate more intricate design processes. In the context of "flexible" KBE applications, the differences from "classic" KBE are evident. For instance, critical calculations are performed by external tools rather than the knowledge inferring engine. The challenges in addressing engineering tasks often stem from the absence of tools rather than issues related to product representation. As these tools constitute a substantial portion of the overall application, considering this scenario, can we still categorize this type of applications as KBE?

In this thesis, the author advocates for categorizing certain tools as KBE, acknowledging the existence of potential ad-hoc components. The "Sorites paradox" or "sand heap paradox" is invoked as an analogy to illustrate the challenge of defining the boundaries and addressing vagueness in the concept of KBE. It is a thought experiment presenting the following paradox: nobody would deny 1000000 grains of sand can be called a heap of sand, and if a heap of sand minus one grain is still a heap. But 1 grain is obviously not a heap. So, when does the heap of sand stop being a "heap"? Similarly, it could raise the question of when a system ceases to be called a KBE when incorporating ad-hoc components.

This thesis avoids delving into profound philosophical discussions. Instead, it underscores that the benefits of KBE come from knowledge representation languages. Any system that represents product data in semantic forms and provides users with languages to define new rules for solving engineering problems can be considered a KBE system. These languages can vary, ranging from those designed for specific cases to universal knowledge representation languages like OWL and SWRL, or even libraries that encapsulate domain knowledge using general-purpose programming languages. Following this concept, KBE developers strive to hide knowledge details at the implementation level and determine the suitable granularity of processes to formulate domain-specific languages at the knowledge level. This approach grants users the freedom to represent their problems using the provided languages, thus realizing the benefits outlined in RQ1.

Chapter 5

Conclusions and Future Work

In this concluding chapter, the contributions to the RQs are summarized, and suggestions for future research are provided.

5.1 Conclusions

This thesis aims to contribute to the digital transformation of the product design paradigm through KBE and semantic modeling. To address this goal, three RQs) are identified to delve into key concerns during the implementation of KBE systems. The findings from six papers presented in this thesis collectively provide insights and contribute to answering the RQs.

For the first RQ, the benefits of implementing KBE over common ad-hoc solutions are explained. Papers A, B, D, E, and F showcase engineering tasks conducted in the KBE paradigm. In contrast to conventional manual interactions with CAX tools, a KBE system enables the reuse of captured knowledge, facilitating rapid generation of variants through templated representations. Compared to hard-coded ad-hoc software, the advantages of KBE primarily lie in interoperability and extendability, making it a promising approach for developing platforms that can seamlessly integrate with other digital tools.

Addressing RQ2, papers B-F illustrate the semantic representation of different components in a design problem, encompassing two knowledge types: product data and design rules. Paper E proposes a reference architecture using the semantic web stack for KBE system implementation. The recommendation is to model product data in an OO style and represent it in ontologies. Design rules are suggested to be implemented using SWRL, web APIs, and BPMN, selected based on their universality and complexities. SWRL is deemed suitable for universal

rules, while black-box functions implemented through web APIs and BPMN are preferred for case-specific rules. Semantic representation empowers users to utilize a KBE system where they can address current design challenges using existing capabilities and extend functionalities by integrating new tools through the reuse of semantically represented knowledge.

Finally, RQ3 explores user-friendly design workflow composition with the assistance of low-code platforms and NLP. Paper E demonstrates that once a KBE application is established using semantically represented design knowledge, various tools such as low-code platforms and NLP programs can be integrated to the platform effortlessly. Thus, users can compose customized workflows by manipulating BPMN language in low-code platforms. Additionally, an NLP-based composer can be developed to semi-automatically generate executable workflows from natural language descriptions.

In conclusion, the KBE design paradigm emphasizes the reusability of design knowledge, fostering a culture within design groups to formalize and codify knowledge instead of relying on case-by-case design approaches. This paradigm enables human knowledge to be processed by machines, facilitating the digital transformation of product design. Due to its strong performance in interoperability and extendability, the KBE paradigm lays the foundation for developing systems that can grow in complexity to meet future demands within a unified architecture.

5.2 Future work

This thesis advocates for the adoption of the KBE design paradigm and semantic modeling to propel the technological advancement of digital transformation in product design. This technological shift is expected to induce a corresponding transformation in cultural dimensions, moving from traditional experience-based design to digital design grounded in knowledge engineering. The unified architecture established within the KBE paradigm provides flexibility to meet future demands by extending the functionality of the current system or seamlessly inter-operating with other digital tools.

In future research, several promising directions could enhance and facilitate the KBE design paradigm:

1. User-friendly modeling tool: Developing a user-friendly modeling tool for end-users to create semantic representations of product models, especially CAD models, can significantly improve the overall user experience. This tool could simplify the process of capturing and translating designers' intent into semantic formats, thus broadening its accessibility.

2. Large-scale transformation of past designs into semantic formats: The creation of datasets is crucial for training data-driven AI tools. Developing conversion tools to transform a substantial number of document-centric historical product designs into semantic formats can establish such datasets. This endeavor facilitates the effective utilization of historical data and presents significant potential for enhancing data-driven automated design tools.
3. Fully-automated workflow generation: The synergy of a low-code platform and a large language model presents an opportunity to automatically discover and configure services (web APIs) for the customized workflows. For instance, the effectiveness of tools like ChatGPT [93] in composing sequences of web APIs based on NL descriptions has been observed by the author, even though the configuration aspect of these web APIs was not thoroughly examined. If both service discovery and configuration can be successfully achieved, the workflow creation process will be streamlined, ultimately enhancing user productivity.

Bibliography

- [1] Jaroslaw Sobieszczanski-Sobieski, Alan Morris, and Michel Van Tooren. *Multidisciplinary design optimization supported by knowledge based engineering*. John Wiley & Sons, 2015.
- [2] Liang Zhang, Brikene Berisha, and Andrei Lobov. A parametric model of umbilical cable with siemens nx considering its reliability. *IFAC-PapersOnLine*, 54(1):187–192, 2021.
- [3] Liang Zhang, Anna Olsen, and Andrei Lobov. An ontology-based kbe application for supply chain sustainability assessment. *Resources, Environment and Sustainability*, 10:100086, 2022.
- [4] Liang Zhang and Andrei Lobov. Extending design automation by integrating external services for product design. In *2021 IEEE 19th International Conference on Industrial Informatics (INDIN)*, pages 1–6. IEEE, 2021.
- [5] Liang Zhang and Andrei Lobov. Interoperability in automating engineering tasks: An illustration with pipe routing application. In *IECON 2023-49th Annual Conference of the IEEE Industrial Electronics Society*, pages 1–8. IEEE, 2023.
- [6] Raquel Sanchis, Óscar García-Perales, Francisco Fraile, and Raul Poler. Low-code as enabler of digital transformation in manufacturing industry. *Applied Sciences*, 10(1):12, 2019.
- [7] M Varl, J Duhovnik, and J Tavčar. Changeability and agility enablers in one-of-a-kind product development and design processes. *Research in Engineering Design*, pages 1–18, 2021.

- [8] Chan Qiu, Jianrong Tan, Zhenyu Liu, Haoyang Mao, and Weifei Hu. Design theory and method of complex products: a review. *Chinese Journal of Mechanical Engineering*, 35(1):103, 2022.
- [9] Andreia De Bem Machado, Silvana Secinaro, Davide Calandra, and Federico Lanzalonga. Knowledge management and digital transformation for industry 4.0: a structured literature review. *Knowledge Management Research & Practice*, 20(2):320–338, 2022.
- [10] Swen Nadkarni and Reinhard Prügl. Digital transformation: a review, synthesis and opportunities for future research. *Management Review Quarterly*, 71:233–341, 2021.
- [11] Sascha Kraus, Paul Jones, Norbert Kailer, Alexandra Weinmann, Nuria Chaparro-Banegas, and Norat Roig-Tierno. Digital transformation: An overview of the current state of the art of research. *Sage Open*, 11(3):21582440211047576, 2021.
- [12] Morteza Ghobakhloo. The future of manufacturing industry: a strategic roadmap toward industry 4.0. *Journal of manufacturing technology management*, 29(6):910–936, 2018.
- [13] Pedro Company, Jorge D Camba, Stanislao Patalano, Ferdinando Vitolo, and Antonio Lanzotti. A functional classification of text annotations for engineering design. *Computer-Aided Design*, page 103486, 2023.
- [14] Asma Ladj, Zhiqiang Wang, Oussama Meski, Farouk Belkadi, Mathieu Ritou, and Catherine Da Cunha. A knowledge-based digital shadow for machining industry in a digital twin perspective. *Journal of Manufacturing Systems*, 58:168–179, 2021.
- [15] Francesco Longo, Giovanni Mirabelli, Letizia Nicoletti, and Vittorio Solina. An ontology-based, general-purpose and industry 4.0-ready architecture for supporting the smart operator (part i–mixed reality case). *Journal of Manufacturing Systems*, 64:594–612, 2022.
- [16] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific american*, 284(5):34–43, 2001.
- [17] Stuart J Russell and Peter Norvig. *Artificial intelligence a modern approach*. London, 2010.
- [18] Joan Peckham and Fred Maryanski. Semantic data models. *ACM Computing Surveys (CSUR)*, 20(3):153–189, 1988.

-
- [19] Oreste Signore et al. Representing knowledge in the semantic web. In *Open Culture Conference (organised by the Italian office of W3C)*, pages 27–29, 2005.
- [20] Christophe Gaudet-Blavignac, Jean Louis Raisaro, Vasundra Touré, Sabine Österle, Katrin Cramer, and Christian Lovis. A national, semantic-driven, three-pillar strategy to enable health data secondary usage interoperability for research within the swiss personalized health network: methodological study. *JMIR Medical Informatics*, 9(6):e27591, 2021.
- [21] Open Platform Communications Foundation. Opc unified architecture. <https://opcfoundation.org/about/opc-technologies/opc-ua/>. [Accessed 10-Jun-2023].
- [22] Aditya Akundi and Viviana Lopez. A review on application of model based systems engineering to manufacturing and production engineering systems. *Procedia Computer Science*, 185:101–108, 2021.
- [23] Marc-Henri Bleu-Laine, Mayank V Bendarkar, Jiacheng Xie, Simon I Brice, and Dimitri N Mavris. A model-based system engineering approach to normal category airplane airworthiness certification. In *AIAA Aviation 2019 Forum*, page 3344, 2019.
- [24] Nancy J Lindsey, Mahdi Alimardani, and Luis D Gallo. Reliability analysis of complex nasa systems with model-based engineering. In *2020 Annual Reliability and Maintainability Symposium (RAMS)*, pages 1–8. IEEE, 2020.
- [25] Wim JC Verhagen, Pablo Bermell-Garcia, Reinier EC Van Dijk, and Richard Curran. A critical review of knowledge-based engineering: An identification of research challenges. *Advanced Engineering Informatics*, 26(1):5–15, 2012.
- [26] Patricia Kügler, Fabian Dworschak, Benjamin Schleich, and Sandro Wartack. The evolution of knowledge-based engineering from a design research perspective: Literature review 2012–2021. *Advanced Engineering Informatics*, 55:101892, 2023.
- [27] Rudi Studer, V Richard Benjamins, and Dieter Fensel. Knowledge engineering: Principles and methods. *Data & knowledge engineering*, 25(1-2):161–197, 1998.
- [28] Thomas R Gruber. A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2):199–220, 1993.

- [29] Peter Mika and Hans Akkermans. Analysis of the state-of-the-art in ontology-based knowledge management. *Vrije Universiteit, Amsterdam, Tech. Rep*, 2003.
- [30] Lan Yang, Kathryn Cormican, and Ming Yu. Ontology-based systems engineering: A state-of-the-art review. *Computers in Industry*, 111:148–171, 2019.
- [31] Akshay Raju Kulkarni, Maurice Hoogreef, and Gianfranco La Rocca. Combining semantic web technologies and kbe to solve industrial mdo problems. In *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, page 3823, 2017.
- [32] Yusuf Sermet and Ibrahim Demir. Towards an information centric flood ontology for information management and communication. *Earth Science Informatics*, 12(4):541–551, 2019.
- [33] Douglas Orellana and William Mandrick. The ontology of systems engineering: towards a computational digital engineering semantic framework. *Procedia Computer Science*, 153:268–276, 2019.
- [34] Lan Yang, Kathryn Cormican, and Ming Yu. Ontology learning for systems engineering body of knowledge. *IEEE Transactions on Industrial Informatics*, 17(2):1039–1047, 2020.
- [35] Michael DeBellis. New Protégé Pizza Tutorial. <https://www.michaeldebellis.com/post/new-protege-pizza-tutorial>, September 2021. [Accessed Oct-2023].
- [36] Stuart Russell and Peter Norvig. Artificial intelligence: a modern approach, 4th us ed. *University of California, Berkeley*, 2021.
- [37] HK Lin, JA Harding, and WC Tsai. A rule-based knowledge system on semantic web for collaboration moderator services. *International Journal of Production Research*, 50(3):805–816, 2012.
- [38] Bruce Buchanan, Georgia Sutherland, and Edward A Feigenbaum. Heuristic dendral: A program for generating explanatory hypotheses. *Organic Chemistry*, page 30, 1969.
- [39] Luc Steels. Procedural attachment. *Proceedings of the Artificial Intelligence Memo*, 543, 1979.
- [40] Markus Krötzsch. *Description logic rules*, volume 8. IOS Press, 2010.

-
- [41] Gianfranco La Rocca. Knowledge based engineering: Between ai and cad. review of a language based technology to support engineering design. *Advanced engineering informatics*, 26(2):159–179, 2012.
- [42] Craig B Chapman and Martyn Pinfold. Design engineering—a need to re-think the solution using knowledge based engineering. *Knowledge-based systems*, 12(5-6):257–267, 1999.
- [43] Matthew I Campbell and Kristina Shea. Computational design synthesis. *AI EDAM*, 28(3):207–208, 2014.
- [44] Eugen Rigger, Kristina Shea, and Tino Stankovic. Task categorisation for identification of design automation opportunities. *Journal of Engineering Design*, 29(3):131–159, 2018.
- [45] Amaresh Chakrabarti, Kristina Shea, Robert Stone, Jonathan Cagan, Matthew Campbell, Noe Vargas Hernandez, and Kristin L. Wood. Computer-Based Design Synthesis Research: An Overview. *Journal of Computing and Information Science in Engineering*, 11(2):021003, 06 2011.
- [46] CORDIS. Methodology and software tools oriented to kbe applications, 1999. [Online; accessed Jan-2023].
- [47] August Th Schreiber, Guus Schreiber, Hans Akkermans, Anjo Anjewierden, Nigel Shadbolt, Robert de Hoog, Walter Van de Velde, and Bob Wielinga. *Knowledge engineering and management: the CommonKADS methodology*. MIT press, 2000.
- [48] Christian Van der Velden, Cees Bil, and Xinghuo Xu. Adaptable methodology for automation application development. *Advanced Engineering Informatics*, 26(2):231–250, 2012.
- [49] Richard Curran, Wim JC Verhagen, Michel JL Van Tooren, and Ton H van der Laan. A multidisciplinary implementation methodology for knowledge based engineering: Knomad. *Expert Systems with Applications*, 37(11):7336–7350, 2010.
- [50] Melody Stokes et al. *Managing engineering knowledge: MOKA: methodology for knowledge based engineering applications*, volume 3. Professional Engineering Publishing London, 2001.
- [51] Samar Ammar-Khodja, Nicolas Perry, and Alain Bernard. Processing knowledge to support knowledge-based engineering systems specification. *Concurrent Engineering*, 16(1):89–101, 2008.

- [52] Wojciech Skarka. Application of moka methodology in generative model creation using catia. *Engineering Applications of Artificial Intelligence*, 20(5):677–690, 2007.
- [53] Pai Zheng, Gang Zhao, Víctor Hugo Torres, and José Ríos. A knowledge-based approach to integrate fuzzy conceptual design tools and moka into a cad system. In *2012 7th International Conference on Computing and Convergence Technology (ICCCT)*, pages 1285–1291, 2012.
- [54] Jonathan Tatcher. *Development of semantic data models to support data interoperability in the rail industry*. PhD thesis, University of Birmingham, 2016.
- [55] Chao Yang, Yuan Zheng, Xinyi Tu, Riku Ala-Laurinaho, Juuso Autiosalo, Olli Seppänen, and Kari Tammi. Ontology-based knowledge representation of industrial production workflow. *Advanced Engineering Informatics*, 58:102185, 2023.
- [56] ISO/IEC/IEEE. Iso/iec/ieee international standard - systems and software engineering–vocabulary. *ISO/IEC/IEEE 24765:2017(E)*, pages 1–541, 2017.
- [57] Andrei Lobov. On extendibility and interoperability properties in knowledge-based engineering. In *AIP Conference Proceedings*, volume 2605. AIP Publishing, 2023.
- [58] Douglas K. Barry and David Dick. Chapter 3 - web services and service-oriented architectures. In Douglas K. Barry and David Dick, editors, *Web Services, Service-Oriented Architectures, and Cloud Computing (Second Edition)*, The Savvy Manager’s Guides, pages 15–33. Morgan Kaufmann, Boston, second edition edition, 2013.
- [59] Vijay Srinivasan. An integration framework for product lifecycle management. *Computer-aided design*, 43(5):464–478, 2011.
- [60] Dazhong Wu, David W Rosen, Lihui Wang, and Dirk Schaefer. Cloud-based design and manufacturing: A new paradigm in digital manufacturing and design innovation. *Computer-aided design*, 59:1–14, 2015.
- [61] Alexander Ortner-Pichler and Christian Landschützer. Integration of parametric modelling in web-based knowledge-based engineering applications. *Advanced Engineering Informatics*, 51:101492, 2022.
- [62] Guozhi Liu, Bi Huang, Zhihong Liang, Minmin Qin, Hua Zhou, and Zhang Li. Microservices: architecture, container, and challenges. In *2020 IEEE*

- 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pages 629–635. IEEE, 2020.
- [63] Nicola Dragoni, Saverio Giallorenzo, Alberto Lluch Lafuente, Manuel Mazara, Fabrizio Montesi, Ruslan Mustafin, and Larisa Safina. Microservices: yesterday, today, and tomorrow. *Present and ulterior software engineering*, pages 195–216, 2017.
- [64] Neha Thakur, Avtar Singh, and AL Sangal. Cloud services selection: A systematic review and future research directions. *Computer Science Review*, 46:100514, 2022.
- [65] Jinghai Rao and Xiaomeng Su. A survey of automated web service composition methods. In *International Workshop on Semantic Web Services and Web Process Composition*, pages 43–54. Springer, 2004.
- [66] Ourania Hatzii, Dimitris Vrakas, Nick Bassiliades, Dimosthenis Anagnostopoulos, and Ioannis Vlahavas. The porsce ii framework: Using ai planning for automated semantic web service composition. *The Knowledge Engineering Review*, 28(2):137–156, 2013.
- [67] Yixin Yan, Bin Xu, and Zhifeng Gu. Automatic service composition using and/or graph. In *2008 10th IEEE Conference on E-Commerce Technology and the Fifth IEEE Conference on Enterprise Computing, E-Commerce and E-Services*, pages 335–338. IEEE, 2008.
- [68] Ahmed Abid, Mohsen Rouached, and Nizar Messai. Semantic web service composition using semantic similarity measures and formal concept analysis. *Multimedia Tools and Applications*, 79:6569–6597, 2020.
- [69] Pablo Rodriguez-Mier, Carlos Pedrinaci, Manuel Lama, and Manuel Mucientes. An integrated semantic web service discovery and composition framework. *IEEE transactions on services computing*, 9(4):537–550, 2015.
- [70] Wenqiang Li, Xuemei Dai, and Hao Jiang. Web services composition based on weighted planning graph. In *2010 First International Conference on Networking and Distributed Computing*, pages 89–93. IEEE, 2010.
- [71] Freddy Lécué, Samir Salibi, Philippe Bron, and Aurélien Moreau. Semantic and syntactic data flow in web service composition. In *2008 IEEE International Conference on Web Services*, pages 211–218. IEEE, 2008.
- [72] Marco Aiello. A challenge for the next 50 years of automated service composition. In *International Conference on Service-Oriented Computing*, pages 635–643. Springer, 2022.

- [73] Neha Agarwal, Geeta Sikka, and Lalit Kumar Awasthi. A systematic literature review on web service clustering approaches to enhance service discovery, selection and recommendation. *Computer Science Review*, 45:100498, 2022.
- [74] Chris Richardson and Floyd Earl Smith. Microservices: From design to deployment. online, NGINX, July 2016.
- [75] Joerg Becker, Oliver Mueller, and Manuel Woditsch. An ontology-based natural language service discovery engine—design and experimental evaluation. *ECIS 2010 Proceedings*, 166, 2010.
- [76] Pooja Thapar and Lalit Sen Sharma. Implementing sparql-based prefiltering on jena fuseki tdb store to reduce the semantic web services search space. In *Evolutionary Computing and Mobile Sustainable Networks: Proceedings of ICECMSN 2021*, pages 319–333. Springer, 2022.
- [77] Meriem Achir, Abdelkrim Abdelli, Lynda Mokdad, and Jalel Benothman. Service discovery and selection in iot: A survey and a taxonomy. *Journal of Network and Computer Applications*, 200:103331, 2022.
- [78] Hartwig Baumgärtel, Patrick Moder, Nour Ramzy, and Hans Ehm. Service-based semiconductor manufacturing using the digital reference ontology for global service discovery. In *IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society*, pages 4533–4540. IEEE, 2020.
- [79] Clay Richardson and John Rymer. New Development Platforms Emerge For Customer-Facing Applications. Technical report, Forrester, Cambridge, MA, USA, 2014.
- [80] OutSystems. The State of Application Development. Is IT Ready for Disruption? Technical report, OutSystems, Boston, MA, USA, 2019.
- [81] Apurvanand Sahay, Arsene Indamutsa, Davide Di Ruscio, and Alfonso Pierantonio. Supporting the understanding and comparison of low-code development platforms. In *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 171–178. IEEE, 2020.
- [82] Fahim Sufi. Algorithms in low-code-no-code for research applications: a practical review. *Algorithms*, 16(2):108, 2023.
- [83] Daniel Pinho, Ademar Aguiar, and Vasco Amaral. What about the usability in low-code platforms? a systematic literature review. *Journal of Computer Languages*, page 101185, 2022.

-
- [84] Sholiq Sholiq, Riyanarto Sarno, and Endang Siti Astuti. Generating bpmn diagram from textual requirements. *Journal of King Saud University-Computer and Information Sciences*, 34(10):10079–10093, 2022.
- [85] Ana Ivanchikj, Souhaila Serbout, and Cesare Pautasso. From text to visual bpmn process models: Design and evaluation. In *Proceedings of the 23rd ACM/IEEE international conference on model driven engineering languages and systems*, pages 229–239, 2020.
- [86] Mark A Musen. The protégé project: a look back and a look forward. *AI matters*, 1(4):4–12, 2015.
- [87] Apache.org. Apache Jena Fuseki SPARQL server. <https://jena.apache.org/documentation/fuseki2/>, Accessed Nov-2023.
- [88] Miguel2617@Github.com. Integrate Openllet SWRL reasoner to Apache Jena Fuseki: Step by step guide. <https://gist.github.com/Miguel2617/eea8c79e209727233026ea47b72d2d65>, Accessed Nov-2023.
- [89] Wentie Niu, Haiteng Sui, Yaxiao Niu, Kunhai Cai, Weiguo Gao, et al. Ship pipe routing design using nsga-ii and coevolutionary algorithm. *Mathematical Problems in Engineering*, 2016, 2016.
- [90] Thomas R Gruber. Toward principles for the design of ontologies used for knowledge sharing? *International journal of human-computer studies*, 43(5-6):907–928, 1995.
- [91] Douglas K Barry and David Dick. Chapter 3-web services and service-oriented architectures. *Web Services, Service-oriented Architectures, and Cloud Computing*, pages 15–33, 2013.
- [92] Adel Boukhadra, Karima Benatchba, and Amar Balla. Ranked matching of owl-s process model for distributed discovery of sws in p2p systems. In *2014 17th International Conference on Network-Based Information Systems*, pages 106–113. IEEE, 2014.
- [93] OpenAI. Chatgpt.com. <https://chat.openai.com/>, Accessed: Sep-2023.

Appendices

Paper A

A parametric model of umbilical cable with Siemens NX considering its reliability

© 2021 The Authors. Originally published as:

Zhang, Liang, Brikene Berisha, and Andrei Lobov. "A parametric model of umbilical cable with siemens NX considering its reliability." IFAC-PapersOnLine 54.1 (2021): 187-192.

URL: <https://doi.org/10.1016/j.ifacol.2021.08.022>

A Parametric Model of Umbilical Cable with Siemens NX considering its Reliability

Liang Zhang*, Brikene Berisha*, Andrei Lobov*

* Norwegian University of Science and Technology, Dep. of Mechanical
and Industrial Engineering, Norway (e-mail: {liang.zhang,
brikene.berisha, andrei.lobov}@ntnu.no).

Abstract: Umbilical cable is one of the key control equipment, for example, in the subsea oil and gas production system. That can be seen as a customized product related to specific parameters of use cases, e.g. installation site. In this paper, we first apply the calculation method for the reliability of the umbilical cable by advanced first-order second-moment method (AFOSM) and Monte Carlo method. Secondly, we demonstrate the use of Siemens NX and its framework for engineering knowledge representation called Knowledge Fusion (KF) to generate the parametric model of the umbilical cable design considering its reliability. Finally, we reveal the advantages of introducing a Knowledge-based Engineering (KBE) approach to integrate the CAD models extended with automated calculations for product reliability.

Copyright © 2021 The Authors. This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0>)

Keywords: Parametric Model, KBE, Knowledge Fusion, CAD, Product Design, Customized Product, Umbilical, Reliability, AFOSM, Monte Carlo Simulation

1. INTRODUCTION

One of the focuses on Industry 4.0 is to meet the growing demand for customized products. In the domain of subsea oil and gas production, the cross-section of the steel tube umbilical cable always varies depending on the specific project's requirements, which is a typical customized product. A steel tube umbilical cable often consists of some structural and functional components, as shown in figure 1. The outside is the outer sheath, used to protect the tensile armors from seawater. Tensile armors, often appearing in double layers to balance the torque, are the main components to bear the tension load. The functional components are assembled into an inner core covered by an inner sheath, including central and external tubes, electrical cable, optical fiber, fillers (Lu et al. (2014)). Since the application water depth goes deeper, the traditional deterministic design method using the safety factor is difficult to satisfy the design requirements. Therefore the design method based on reliability is drawing more and more attention in the structural design of umbilical (Yan et al. (2017)).

The conventional CAD system focuses on geometrical aspects of a design, lacking the capability to consider non-geometrical aspects. Although CAD is adjusted toward interactive operations with limited ability to automate its operations via scripts, it is still difficult to generate robust parametric product models that permit topology changes and freedom to make adaptive modifications (Sobieszcanski-Sobieski et al. (2015)).

The KBE method, as the evolution of conventional CAD systems, can quickly generate different configurations and variants of a given product, and manage, learn and grow

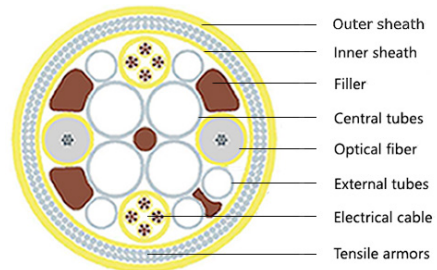


Fig. 1. Typical cross-section of steel tube umbilical cable (adapted from Yan et al. (2017))

on a large amount of multidisciplinary knowledge. This is a remarkable feature when developing a large number of almost identical parts, which is not practical to do “by hand” in a traditional way. The KBE method can also support the integration of heterogeneous sets of analysis tools in the Multidisciplinary Design Optimization (MDO) framework by automating the generation of the necessary disciplinary abstractions. This can relieve the optimizer from the burden of managing spatial integration constraints, which can be intrinsically guaranteed by properly defined generative models (Sobieszcanski-Sobieski et al. (2015)).

As mentioned above, steel tube umbilical cable is a customized product depending on the project's requirement. However, only some minor modifications to the previous design solution is enough in most cases. After the initial design is provided by designing engineers, some routine analysis is necessary to conduct by analyzing engineers to guarantee the design to meet the requirement. Reliability

performance is one of the routine analysis. The conventional design method with the traditional CAD model lacks flexibility in doing this kind of task. If the initial design needs to be modified, the designing engineers have to draw the model again, which can be time-consuming and tedious. And the same applies to the analyzing engineers as the data can not be transferred between different phrases. In such situations, the KBE method gives an advantage to complete the design loop in a time-saving way by providing a parametric model.

In this paper, we choose the cross-section design of steel tube umbilical cable considering its reliability as the example to demonstrate the benefits to implement the KBE method instead of the conventional CAD approach in a customized product design. Firstly, the calculation method for the reliability of the umbilical cable by advanced first-order second-moment method (AFOSM) and Monte Carlo method is introduced. Secondly, the workflow to generate a parametric model of umbilical cable is demonstrated with Siemens NX and its engineering knowledge representation module called Knowledge Fusion (KF). Finally, the benefit and current limitation of introducing KBE approach to integrate the CAD models extended with automated calculation function is discussed and the advice on future research about KBE method in product design is given.

2. STATE OF THE ART

2.1 KBE

KBE is the engineering using product and process knowledge that has been captured and stored in dedicated software applications, to enable its direct exploitation and reuse in the design of new products and variants (Sobieszcanski-Sobieski et al. (2015)). KBE systems are the software tools to reuse engineering knowledge combining the rule-based reasoning capabilities of knowledge-based systems (KBS) with the CAD-like geometry manipulation and data processing capabilities. A typical KBE system provides the user with a programming language, typically object-oriented and one integrated or tightly connected CAD engine. The programming language allows the user to capture and reuse engineering rules and processes, while the object-oriented modelling approach serves to abstract the system as collections of objects, defined by parameters and behaviour, connected by relations (Sobieszcanski-Sobieski et al. (2015)). Many CAD software introduced KBE features to support the rapid development of products. The KBE is supported in Siemens NX by Knowledge Fusion (KF), which is an interpreted, object-oriented language (Siemens website (2016)).

Some KBE CAD models have been established by researchers. Wang et al. (2012) used Knowledge Fusion (KF) to read the airfoil data files and generate the impeller model automatically, then used Fluent to analyse the auto-generated model to get proper aerodynamic parameters, which greatly improved the efficiency of modelling and flexibility of the CAD system. Gujarathi and Ma (2010) proposed a “common data model” (CDM) containing all the required parametric information for both CAD and CAE analysis, which is expected to integrate CAD and CAE processes. Soulat (2012) employed open-source KBE

tools to develop an application to generate, visualize, and export aircraft configuration geometries, which improved the automation and optimization of the aircraft design process. Zhang et al. (2008) integrated KBE and modular design method to realize parametric drive and detailed design of the main frame of a tunnel boring machine, which effectively shortens the developing cycle. Lobov et al. (2020) proposed the use of Knowledge Fusion for generation of robot trajectories to support faster transition from a product information in CAD and KF till the robot code that should weld the product. Some KBE models considered the analysis and optimization function, however, the amount of this kind of cases is not large.

2.2 Reliability Estimation of Steel Tube Umbilical

Structural reliability is to apply reliability engineering theories to structural analysis, which might replace traditional deterministic ways of design and maintenance (Choi et al. (2006)). This design method has been introduced in many structural analysis fields, e.g. architectural design (Dey et al. (2018)), mechanical engineering, and marine engineering (Yong Bai (2016)). Khan and Ahmad (2007) studied the riser fatigue reliability with response surface method (RSM) in conjunction with the First Order Reliability Method (FORM) and compared the result with the Monte Carlo simulation method. Li and Low (2012) investigated the influence of soil uncertainties on the SCR fatigue reliability, and concluded that the efficient first-order reliability method (FORM) and inverse-FORM (IFORM) analysis are fairly accurate compared with Monte Carlo simulation. Yan et al. (2017) investigated the reliability of the steel tube umbilical cable undertaking the most dangerous load cases with AFOSM and Monte Carlo simulation considering the uncertainty of components in the cross-section, and applied particle swarm optimization algorithm to find an optimized design with a higher reliability index.

It can be seen that the advanced first-order second-moment (AFOSM) method is widely used to estimate the failure probability and reliability index during the reliability analysis process. Meanwhile, the Monte Carlo Simulation (MCS) is commonly considered as the benchmark to verify the result obtained from AFOSM. Therefore, in this paper, AFOSM and MCS are applied as the calculator estimating the reliability of the product, to demonstrate the potential of KBE to integrate different knowledge used in design process.

3. METHODOLOGY OF RELIABILITY ESTIMATION

3.1 Structural Reliability of the Umbilical Cable

Reliability is commonly defined as the ability of an item to perform its function in a certain period under a certain condition. Due to manufacturing errors, the geometric parameters of the umbilical cable will have errors from the design value (table 1, from Yan et al. (2017)), which brings the structural failure probability under the design load during its life span. And the central tubes are considered as the most dangerous part. The reliability of the umbilical cable in its design life span can be defined as $R = 1 - P_f$, where P_f is the failure probability, determined by the

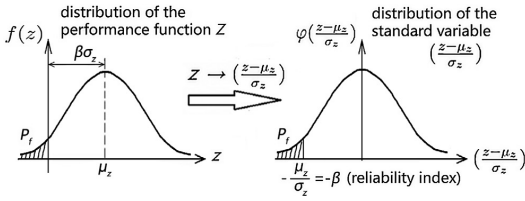


Fig. 2. Illustration of the reliability index β

distribution of the performance function z . When $z \leq 0$, the structure will fail, i.e. $P_f = P(z \leq 0)$.

Table 1. The statistics of 5 variables

Items	μ	σ
Out diameter of central tubes	29.14 mm	0.043
Thickness of central tubes	1.87 mm	0.062
Out diameter of external tubes	15.52 mm	0.043
Thickness of external tubes	1.41 mm	0.047
Yields stress of the material	550 MPa	22.6

Assuming the performance function z conforms to the normal distribution $N(\mu_z, \sigma_z^2)$ with the probability density function of $f(z)$, the failure probability is equivalent to the area in shade in Figure 2. Obviously, the transformation $\frac{z - \mu_z}{\sigma_z}$ conforms to the standard normal distribution $N(0, 1)$ with the probability density function of $\varphi(\frac{z - \mu_z}{\sigma_z})$. Thus, the relation between the failure probability and distribution of performance function can be represented by (1).

$$P_f = P(z \leq 0) = P\left(\frac{z - \mu_z}{\sigma_z} \leq \frac{-\mu_z}{\sigma_z}\right) = \Phi\left(\frac{-\mu_z}{\sigma_z}\right) \quad (1)$$

Define reliability index $\beta = \frac{\mu_z}{\sigma_z}$, then $P_f = \Phi(-\beta)$ (Hasofer and Lind (1974)). Here, the reliability index β represents the distance between the critical value and the mean value. The larger the reliability index β is, the safer the structure is.

3.2 The Performance Function of the Umbilical Cable

In this specified case, only steel components (central and external tubes, tensile armors) are considered to affect the reliability estimation. The central tubes are considered as the most dangerous components based on the engineering experience and mechanics analysis. Without consideration of plastic deformation, if Von Mises stress of the central tubes exceeds the yield stress of the steel, the central tubes are highly likely to break and the pressured liquid inside will leak to outside. Based on the assumption, considering the uncertainty of the 5 variables (table 1), the performance function is defined as

$$z = \gamma\sigma_y - \sigma_{equal} = g(x_1, x_2, \dots, x_5) \quad (2)$$

, where γ is the utilization ratio of the material of central tube, σ_y is the yield stress, and Von Mises stress is

$$\sigma_{equal} = \sqrt{\frac{1}{2}((\sigma_a - \sigma_r)^2 + (\sigma_a - \sigma_h)^2 + (\sigma_h - \sigma_r)^2)}. \quad (3)$$

Since the wall thickness of the central tube is much less than its radius, the central tube can be seen as a thin-wall cylinder model (see Figure 3). Considering a thin-wall

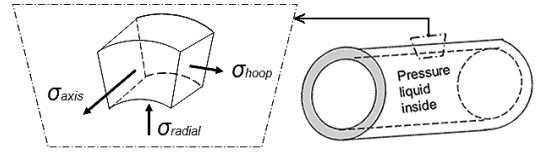


Fig. 3. Force analysis of the central tube as thin-wall cylinder

cylinder model subjected to internal pressure, the hoop, radial and longitudinal(axial) stress($\sigma_{hoop}, \sigma_{radial}, \sigma_{axial}$ respectively) produced in the wall can be calculated by (4), where p is the pressure of the inside liquid, D, d are outside and inside diameter of central tube.

$$\begin{aligned} \sigma_{hoop} &= p \frac{D^2 + d^2}{D^2 - d^2} \\ \sigma_{radial} &= -p \\ \sigma_{axial} &= \sigma_t + \sigma_M + \sigma_e. \end{aligned} \quad (4)$$

The stress in axial direction (σ_{axial}) is a sum of three components: the stress caused by internal pressure(σ_e), the stress caused by bending moment(σ_M), and the stress caused by tensile load(σ_t), which can be calculated by (5), where $T, \frac{1}{\rho}$ are the tension load and the curvature of the umbilical at the end, calculated by a professional simulation software(OrcaFlex)(Yan et al., 2017).

$$\begin{aligned} \sigma_e &= p \frac{\pi d^2}{4} \frac{1}{A_t} \\ \sigma_M &= E_t \frac{D}{2} \frac{1}{\rho} \\ \sigma_t &= \frac{T}{K} \frac{K_t}{A_t}. \end{aligned} \quad (5)$$

The tension stiffness of the umbilical cable and its components can be calculated by (6), where K is the overall tension stiffness of the umbilical cable, A_t, A_o are the cross-section area of a central tube and an external tube, K_t, K_o are the tension stiffness of a central tube and an external tube, E_t is Young's modulus of steel, α_o, α_i are outside and inside helix angle between tensile armor and the axis, n_o, n_i are the number of outside and inside tensile armors.

$$\begin{aligned} K &= 4K_t + 5K_o + n_i E_t A \cos^3 \alpha_i + n_o E_t A \cos^3 \alpha_o \\ A_t &= \pi \frac{D^2 - d^2}{4} \\ K_t &= E_t A_t \\ K_o &= E_t A_o \end{aligned} \quad (6)$$

3.3 Advanced First Order Second Moment method

The Advanced First Order Second Moment (AFOSM) method is a widely used method to estimate the reliability of structure with a performance function of insignificant non-linearity. AFOSM has high efficiency and acceptable accuracy in most cases. It is adapted from the Mean value First Order Second Moment (MFOSM) method, which is easy to apply but has shown to be inferior to AFOSM (Madsen et al., 2006).

It is beneficial to introduce the MFOSM method firstly. Starting from a simple case, consider a structure with a linear performance function $z = f(x_1, x_2, \dots, x_n) = \sum_{i=1}^n c_i x_i$, where f is a linear function, c_i are constant,

x_i are independent random variables conforming to normal distribution $N(\mu_i, \sigma_i^2)$. According to the property of normal distribution, $z \sim N(\sum_{i=1}^n c_i \mu_i, \sum_{i=1}^n c_i^2 \sigma_i^2)$. Then the reliability index of this system can be calculated by

$$\beta = \frac{\mu_z}{\sigma_z} = \frac{\sum_{i=1}^n c_i \mu_i}{\sqrt{\sum_{i=1}^n c_i^2 \sigma_i^2}}.$$

The reliability index of a system with a linear performance function is easy to calculate. However, the performance function in many cases are nonlinear, thus the distribution of performance function is not easy to calculate directly. Consider a general case: a structure with a general performance function $z = g(x_1, x_2, \dots, x_n)$, where x_i are independent random variables conforming to normal distribution $N(\mu_i, \sigma_i^2)$. The performance function can be expanded by a Taylor series at the mean value point $\boldsymbol{\mu}$ and linearized by taking the two first-order items:

$$z \approx z^* = g(\mu_1, \mu_2, \dots, \mu_n) + \sum_{i=1}^n \frac{\partial g(\mu_1, \mu_2, \dots, \mu_n)}{\partial x_i} (x_i - \mu_i). \quad (7)$$

According to the property of normal distribution, plus the independence of variables x_i , the relations (8) can be get based on the linearized performance function (7):

$$\begin{aligned} \mu_z &= g(\boldsymbol{\mu}) \\ \sigma_z^2 &= \sum_{i=1}^n \sum_{j=1}^n \frac{\partial g(\boldsymbol{\mu})}{\partial x_i} \frac{\partial g(\boldsymbol{\mu})}{\partial x_j} \text{cov}(\mathbf{X}_i, \mathbf{X}_j) \\ &= \sum_{i=1}^n \left(\frac{\partial g(\boldsymbol{\mu})}{\partial x_i} \sigma_i \right)^2 \end{aligned} \quad (8)$$

$$\text{, thus } \beta = \frac{\mu_z}{\sigma_z} = \frac{g(\boldsymbol{\mu})}{\sqrt{\sum_{i=1}^n \left(\frac{\partial g(\boldsymbol{\mu})}{\partial x_i} \sigma_i \right)^2}}.$$

As the performance function is extended at the mean value point and uses a first-order Taylor series and the first and second moments of the input variables, this method is called Mean value First Order Second Moment (MFOSM) methods.

The essential of MFOSM method is to find the limit state surface in variable space by the approximated Taylor expansion at the mean value point. However, since the mean value is not on the limit state surface, a more accurate result can be gained by a Taylor expansion at a point which is on the limit state surface. The Advanced First Order Second Moment method (AFOSM) is developed based on this logic. AFOSM is to find a point which is on the limit state surface and meanwhile has the shortest distance to the mean value point. The point of meeting the criteria is called the design point and is usually solved by an iteration method. The steps are as follows.

1. Choose the initial value of the design point (\mathbf{x}^*) by $x_i^* = \mu_i (i = 1, 2, \dots, n)$.
2. Calculate sensitive coefficient (α_i) by (9).

$$\alpha_i = \frac{\sigma_i \frac{\partial g(x_1^*, \dots, x_n^*)}{\partial x_i}}{\sqrt{\sum_{i=1}^n \left(\sigma_i \frac{\partial g(x_1^*, \dots, x_n^*)}{\partial x_i} \right)^2}} \quad (i = 1, 2, \dots, n) \quad (9)$$

3. Get the equation group about β and the current design point (\mathbf{x}^*) by (10).

$$x_i^* = \mu_i - \beta \alpha_i \sigma_i (i = 1, 2, \dots, n) \quad (10)$$

4. Put the current design point (10) into the limit state surface equation (11) to get an equation containing only β , then solve it to get β .

$$z = \gamma \sigma_y - \sigma_{equal} = g(\mathbf{x}) = g(x_1, x_2, \dots, x_n) = 0 \quad (11)$$

5. Update the value of the design point (\mathbf{x}^*) by (10).
6. Iterate step 2-5 until $|\Delta\beta| \leq \text{threshold}$ (e.g. 1e-5).
7. Calculate the failure probability by $P_f = \Phi(-\beta)$.

3.4 Monte Carlo Simulation

Since the AFOSM method is an estimation method, Monte Carlo simulation is commonly applied to verify the accuracy of the result gained by AFOSM method if possible. It is a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results. The underlying concept is to use randomness to solve problems that might be deterministic in principle (Kroese et al., 2014). In this paper, the Monte Carlo simulation is applied in the following steps.

1. Generate random variables (x_i) conforming to normal distribution $N(\mu_i, \sigma_i^2)$, based on the statistics in table 1.
2. Call the performance function (2) N_{total} (e.g. 100000) times to get the distribution of z .
3. Calculate the failure probability by $P_f = \frac{N_{fail}}{N_{total}}$, where N_{total} is the total times of the Monte Carlo tests, and N_{fail} is the times when $z \leq 0$.
4. Fit the distribution of z to get reliability index ($\beta = \frac{\mu_z}{\sigma_z}$).

3.5 Comparison of Reliability Results from Two Methods

By realizing the two methods in Python, two reliability results are gained. From table 2, it can be seen that the two results are close to each other, which can be seen as proof of the feasibility of the two methods. It can also be observed that the reliability and the index will meet the requirement or not when a different requirement is set.

Table 2. Comparison of 2 methods

	MC simulation	AFOSM	REQMT 1	REQMT 2
R	0.99455	0.99498	0.99	0.999987
β	2.5459	2.5746	2.33	4.2

4. APPLICATION: MODELING OF THE UMBILICAL CABLE IN SIEMENS NX WITH KF

4.1 Parametric Model

The steel tube umbilical cable is parameterized considering the geometric and non-geometric aspects. A list containing all the parameters can be seen in table 3. The parameters also appearing in table 1 are for reliability calculation. Regarding the definition of the geometric parameters, the umbilical is firstly decomposed into some basic shapes which can be represented with different ready classes of Siemens NX Knowledge Fusion, and then the basic features of these shapes are extracted. When it comes to the non-geometric aspect, all the variables relating to the reliability estimation are defined as input parameters.

With the help of some ready basic shape classes and some features in Siemens NX Knowledge Fusion, the geometric


```
#central tubes color depending on reliability
(Child list) body_colored_central_tubes:{
  Class, ug_body;
  Feature, {nth(child:index:, central_tubes:)};
  quantity, num_cen_tube;
  color, if (reliability_cts_real: < reliability_cts_required:)
    then ug_askClosestColor(RED)
    else if(reliability_cts_real: > reliability_cts_required:)
    then ug_askClosestColor(GREEN)
    else ug_askClosestColor(YELLOW); };
```

Fig. 4. Excerpt from Knowledge Fusion code (DFA file)

model of steel tube umbilical cable becomes possible to establish. The most frequently used shape class is the cylinder class. Plus the operation of subtraction, a tube can be easily modelled. A helix tensile armor wire can be modelled with sample line class and sweep feature. After establishing every elementary part, the array of these parts can be achieved using a child list feature. When the geometric parts are finished, the process will handle some of the non-geometric parameters according to the user's need. In this paper, the central tubes will be displayed in different colours depending on whether their reliability meets the requirement. If they meet the requirement, they will be in green, otherwise in red. (see figure 4)

Table 3. Parameters list

Type	Part	Parameter
geometric	whole model	length
geometric	central tube	diameter(mean value)
geometric	central tube	diameter(std. deviation)
geometric	central tube	thickness(mean value)
geometric	central tube	thickness(std. deviation)
geometric	central tube	number
geometric	external tube	diameter(mean value)
geometric	external tube	diameter(std. deviation)
geometric	external tube	thickness(mean value)
geometric	external tube	thickness(std. deviation)
geometric	external tube	number
geometric	inner sheath	diameter
geometric	inner sheath	thickness
geometric	inner tensile armor	diameter
geometric	inner tensile armor	number
geometric	inner tensile armor	helix angle
geometric	other unit	diameter
geometric	other unit	number
geometric	outer sheath	diameter
geometric	outer sheath	thickness
geometric	outer tensile armor	diameter
geometric	outer tensile armor	number
geometric	outer tensile armor	helix angle
non-geometric	central tube	pressure
non-geometric	whole model	tension load
non-geometric	whole model	curvature
non-geometric	all steel parts	Young's Modulus
non-geometric	central tube	material utilization
non-geometric	central tube	calculated reliability
non-geometric	central tube	required reliability

4.2 Workflow

The workflow to generate the parametric model considering reliability (stored in a file with DFA extension that contains Knowledge Fusion code) can be seen in figure 5. The whole process can be divided into 4 steps: parameters input, performance calculation (reliability in this case), the DFA file generation, model visualization. In the first

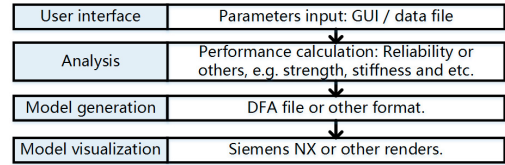


Fig. 5. Workflow to generate the parametric model

step, the user is guided to input parameters including geometric and non-geometric ones into a Python script. This end-user interface could be replaced by a graphical user interface (GUI) to improve the user-friendliness if needed. After the program receives the user input parameters, the user-cared calculation will be conducted and the result will be transferred to the next step. In this paper, we only consider reliability as an example. However, some other calculation can be added in this step according to the user concerns. With all the needed parameters prepared, the program will output the parametric model into the DFA file, which contains the geometric and non-geometric parameters. Eventually, the user can visualize the model with Siemens NX and see whether the reliability of this design meets the requirement intuitively.

If the design fails to meet the criteria, the user can input a new set of parameters and immediately get an updated DFA file. When the result is satisfying, both the 3D model and calculation have been available. Compared with the conventional design process, the data in this method can be transferred between the different phrases automatically. Consequently, the design loop can be conducted in a time-saving and intuitive way, which is a significant benefit in the customized product design.

4.3 Result Demonstration

As mentioned above, the central tubes will be in green or red depending on if the reliability meets the requirement. In this paper, different requirements are set to demonstrate this function as an example. From table 2, it can be seen that if a relatively low standard ($\beta = 2.33, R = 0.99$) is set, the central tubes can pass the criteria and will be in green (see in figure 6 (a)). Oppositely, if a relatively high standard ($\beta = 4.2, R = 0.999987$) is used, the central tubes will be displayed in red (see in figure 6 (b)). The colors are being set following the rules of the 'body colored central tube' object defined in the DFA file (see figure 4).

5. DISCUSSION

Umbilical cable is a typical customised product as its cross-section always varies depending on the specific project's requirements. By applying the KBE method to establish the parametric model instead of the conventional design method, together with the proper calculation tool, the design loop can be done in a time-saving way. Additionally, the parametric model has the potential to combine with design generation and optimization algorithm, which may lead to automatic design and optimization.

Currently, there are still some design processes difficult to parameterize and represent, e.g. the layout design of the umbilical cable. This kind of processes is usually easy to

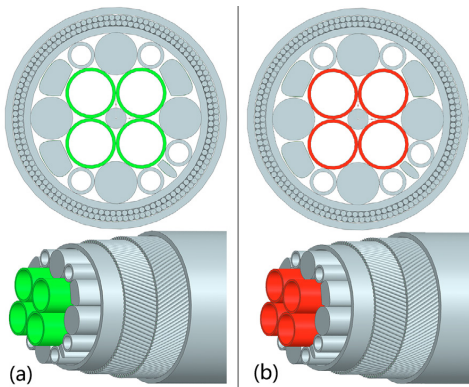


Fig. 6. Result demonstration

handle by human but difficult to be coded into a program. Since the difficulty of the program is in some cases more significant than to manipulate by hand, the KBE method is seen as a bad investment sometimes. If the product is not a customized one which needs to change the design frequently, the disadvantages are more significant.

6. CONCLUSION

A parametric model of steel tube umbilical cable considering its reliability has been established aiming at boosting the design loop of the cross-section. This work shows the potential of KBE method to consider not only the geometric parameters but also the non-geometric aspect. Compared with the conventional way, the benefits of KBE method are significantly demonstrated. Data can be transferred between upstream and downstream processes automatically, which saves significant time to get through the design loop. Additionally, this kind of parametric models lay the foundation for future automation of design and optimization.

Future investment is advised to focus on how to make KBE method easier to implement by design engineer without high requirement on programming skills, e.g. some basic and commonly used analysis code and programs which are easy to use and embed into KBE framework, especially some mathematics functions.

The current work can be extend to consider some more complex engineering process to find more obstacles to apply the KBE method in real engineering application and show the potential of KBE method to design automation, such as the above-mentioned layout automatic design and optimization.

REFERENCES

- Choi, S.K., Grandhi, R., and Canfield, R.A. (2006). *Reliability-based structural design*. Springer Science & Business Media.
- Dey, P., Walbridge, S., and Narasimhan, S. (2018). Evaluation of design provisions for pedestrian bridges using a structural reliability framework. *Journal of Bridge Engineering*, 23(2), 04017132.
- Gujarathi, G.P. and Ma, Y.S. (2010). Generative cad and cae integration using common data model. In *2010 IEEE International Conference on Automation Science and Engineering*, 586–591. IEEE.
- Hasofer, A.M. and Lind, N.C. (1974). Exact and invariant second-moment code format. *Journal of the Engineering Mechanics division*, 100(1), 111–121.
- Khan, R.A. and Ahmad, S. (2007). Dynamic response and fatigue reliability analysis of marine riser under random loads. In *International Conference on Offshore Mechanics and Arctic Engineering*, volume 42681, 183–191.
- Kroese, D.P., Brereton, T., Taimre, T., and Botev, Z.I. (2014). Why the monte carlo method is so important today. *Wiley Interdisciplinary Reviews: Computational Statistics*, 6(6), 386–392.
- Li, F.Z. and Low, Y.M. (2012). Fatigue reliability analysis of a steel catenary riser at the touchdown point incorporating soil model uncertainties. *Applied Ocean Research*, 38, 100–110.
- Lobov, A., Tran, T.A., and Oxman Prescott, S.A. (2020). Formalization of engineering knowledge for industrial robots using knowledge fusion language. *Procedia Manufacturing*, 51, 932 – 937. doi:https://doi.org/10.1016/j.promfg.2020.10.131. URL <http://www.sciencedirect.com/science/article/pii/S2351978920319880>. 30th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM2021).
- Lu, Q., Yang, Z., Yan, J., and Yue, Q. (2014). Design of cross-sectional layout of steel tube umbilical. *Journal of Offshore Mechanics and Arctic Engineering*, 136(4).
- Madsen, H.O., Krenk, S., and Lind, N.C. (2006). *Methods of structural safety*. Courier Corporation.
- Siemens website (2016). Knowledge fusion introduction. https://docs.plm.automation.siemens.com/tdoc/nx/11/nx_api. [Online; accessed Nov.-2020].
- Sobieszczanski-Sobieski, J., Morris, A., and Van Tooren, M. (2015). *Multidisciplinary design optimization supported by knowledge based engineering*. John Wiley & Sons.
- Soulat, M.E. (2012). Parametric geometry representation to support aircraft design. In *2012 IEEE Aerospace Conference*, 1–17. IEEE.
- Wang, L.y., Huang, H.h., and West, R.W. (2012). Impeller modeling and analysis based on ug nx/kf and fluent. *Journal of Central South University*, 19(12), 3430–3434.
- Yan, J., Yang, Z., Zhao, P., Lu, Q., Wu, W., and Yue, Q. (2017). Reliability optimization design of the steel tube umbilical cross section based on particle swarm algorithm. In *International Conference on Offshore Mechanics and Arctic Engineering*, volume 57694, V05AT04A014. American Society of Mechanical Engineers.
- Yong Bai, W.L.J. (2016). *Marine Structural Design (Second Edition)*. Butterworth-Heinemann.
- Zhang, Z.q., Wu, Q.m., Li, Y., Zong, C., and Zhou, C. (2008). Research on technology of variant design for main frame of tunnel boring machine based on kbe. In *2008 First International Conference on Intelligent Networks and Intelligent Systems*, 429–432. IEEE.

Paper B

An ontology-based KBE application for supply chain sustainability assessment

© 2022 Elsevier, with permission, from:

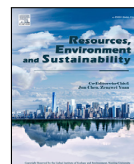
Zhang, Liang, Anna Olsen, and Andrei Lobov. "An ontology-based KBE application for supply chain sustainability assessment." *Resources, Environment and Sustainability* 10 (2022): 100086.

URL: <https://doi.org/10.1016/j.resenv.2022.100086>



Contents lists available at ScienceDirect

Resources, Environment and Sustainability

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

Research article

An ontology-based KBE application for supply chain sustainability assessment

Liang Zhang^{*}, Anna Olsen, Andrei Lobov

Norwegian University of Science and Technology, Richard Birkelands Vei 2B, Trondheim, 7034, Norway



ARTICLE INFO

Keywords:

Ontology
 Knowledge based engineering
 Product lifecycle management
 Sustainability assessment
 Supply chain

ABSTRACT

Product Lifecycle Management (PLM) plays a key role in digital transformation demanded by Industry 4.0 and life cycle assessment, including sustainability assessment. Knowledge Based Engineering (KBE) applications can support PLM by integrating heterogeneous knowledge from different stages throughout the product life. However, the integration of knowledge from different stages and teams can cause misunderstanding if not represented in a unified form. Furthermore, different forms of knowledge used by different software are neither machine-readable nor human-readable, which also sets obstacles to knowledge integration in KBE applications. Supply chain sustainability assessment is such a scenario that entails integrating knowledge from different sources. This paper firstly implements a sustainability assessment method from other scholar to calculate the supply chain sustainability performance and adapts a sustainability assessment ontology for supply chain sustainability assessment. Then, an example KBE application is developed by implementing the sustainability assessment ontology and calculation method to simulate the knowledge sharing and integration between different teams. Finally, through this example application, it is discussed that the implementation of ontology to represent knowledge in PLM application for collaborative tasks like sustainability assessment can increase the efficiency of data sharing and integration. This paper is a proof of concept for the ontology-based framework. This framework can facilitate to represent knowledge but not create new knowledge, which means it can increase the efficiency of the software development, but cannot provide a better calculation method and assessment framework for supply chain sustainability assessment.

1. Introduction

Product Lifecycle Management (PLM) is to manage the data, processes, business systems, and people in an extended enterprise, which plays a key role in digital transformation demanded by Industry 4.0 (Jaskó et al., 2020) and life cycle assessment (Joshi and Dutta, 2004). PLM software allows to manage the knowledge throughout the entire product lifecycle efficiently and cost-effectively: from ideation, design, and manufacture to service and disposal. Knowledge Based Engineering (KBE) is a technology that can support PLM software development, see details in Section 2. However, there are some challenges to share and integrate the knowledge from different stages of the product:

- Misunderstanding. Different teams have different terminology and convention, which can cause misunderstanding unless expressed in an explicit form. Standardization approach is helpful but not flexible enough due to the complexity and constant-evolving nature of the engineering (Yang et al., 2006).
- Not machine-readable: most applications in specific domain are developed without the prior intent for interoperability and integration with others (Yang and Miao, 2007). Humans need to

understand the data and write wrappers to convert it to specific format processable in the domain applications.

- Not human-readable: the data/knowledge in traditional machine-readable data format (SQL database) is not human-readable. If humans want to update or query new data, they have to write program rather than a human-friendly way.

To figure out the above challenges requires a powerful way to store and share knowledge in different phases. Ontology as a semantic way to represent knowledge is seen as a promising candidate to promote sharing and integration (Borsato, 2014). As a kind of graph database, ontology has the advantages of easy extendability and fast-speed query across different databases. Knowledge represented in this form can be queried and processed by machines in order to make data flow more easily between different stages. And it can also be shown and queried in a human-readable way for the related people to maintain the knowledge base, which makes a unified knowledge base practical. A definition of ontology is in Section 2.

Supply chain sustainability assessment (SCSA) is such a scenario that needs to share knowledge between different stages and teams.

^{*} Corresponding author.

E-mail addresses: liang.zhang@ntnu.no (L. Zhang), anna.olsen@ntnu.no (A. Olsen), andrei.lobov@ntnu.no (A. Lobov).

As the importance of sustainable development arises, the sustainability assessment of product supply chain has become an important concern in product lifecycle management. Sustainability assessment is often in relation with the three dimensions: economic growth, environmental protection and social equality. And many studies related to SCSA (Zhang et al., 2005; Kucukvar et al., 2019; Dvaipayana et al., 2021; Yani et al., 2022; Mursidah et al., 2020) show there are also many indicators to be considered in the three dimensions, (e.g. Table 2), which makes the assessment relevant with multi-disciplinary knowledge. However, few of them discuss the difficulties in data sharing and integration from different sources (data collection), which actually is very time-consuming (Kawajiri et al., 2022; Yan et al., 2021).

One of the difficulties in assessing sustainability of supply chain is that the participants of the supply chain have their own scope to assess and optimize, rather than to cooperate with the downstream and upstream. This can lead to some local optimal solutions, rather than the global optimal. There are many deep reasons resulting in this non-cooperation, e.g. technical ones, economical ones, and one of them can be the difficulty of data collection, which is above-mentioned. In the scopes of participants, the data is generated by their own sensors or software system in their own processable formats. But due to the lack of a unified, standardized format, their data can be misunderstood by each others, which means they do not know how to use the data unless they learn on purpose. Considering the data is in raw formats, it can be difficult to learn how to use the data. And even they know the usage of the data after learning, the data still need to be converted into the formats that their own systems can process, which is a time-consuming processing.

In summary, to conduct the assessment entails the exchange of knowledge between different stages and teams. There are some research using ontology in sustainability assessment for some specific domains, e.g., product design (Yang and Song, 2009), enterprise (Muñoz et al., 2013) and urban development (Kuster et al., 2020), which inspire the research question: how to use ontology-based framework to improve the efficiency of the sharing and integrating knowledge across different phases, in order to speed up the development of PLM software for supply chain sustainability assessment.

The aim of this paper is to show the potential of this ontology-based framework in helping the development of PLM software, such as the software components for supply chain sustainability assessment, by sharing and integrating knowledge across different phases of product lifecycle management. It is not to propose a better assessment calculation method and assessment framework for supply chain sustainability assessment. Section 2 introduces the Knowledge Based (KB) approach and ontology to support knowledge representation and integration. Meanwhile, some sustainability assessment research are also presented. Section 3 introduce the sustainability assessment calculation method including the data (Section 3.2) and the adapted ontology (Section 3.3) used as the example in this paper. Then, an example KBE application is developed in Section 4 by implementing the sustainability assessment ontology and calculation method to simulate the knowledge sharing between different teams. Finally, Section 5 discusses how ontology helps to share and integrate the knowledge in PLM software, together with the contribution to supply chain sustainability assessment, and also the scope of this ontology-based framework. The abbreviations used in the paper are explained in Table 1.

2. Research background

2.1. KBE and ontology

Knowledge Based Engineering System is the merger of the terms “Knowledge Based System (KBS)” and “Engineering”, while Knowledge Based Engineering (KBE) is the technology based on the use of these systems (La Rocca, 2012), which means the implementation of KBS in engineering domain. The name of KBE may be ambiguous, as it seems

Table 1
List of abbreviation used in the paper.

Abbreviation	Explanation
PLM	Product Lifecycle Management
SQL	Structured Query Language
KB	Knowledge-Based, Knowledge Base
KBS	Knowledge Based System
KBE	Knowledge Based Engineering
POP	Procedure-Oriented Programming
OOP	Object-Oriented Programming
OO	Object-Oriented
FCE	Fuzzy Comprehensive Evaluation
SPARQL	SPARQL Protocol and RDF Query Language
GUI	Graphic User Interface
SCSA	Supply Chain Sustainability Assessment

to indicate the existence of the engineering that is not based on knowledge. Actually, the term “knowledge” refers to rules, hence this name is highlighting that the Knowledge-Based (KB) approach focuses on the reuse of engineering rules (knowledge) by knowledge management techniques, e.g., capture, formalization, representation and integration. To make an analogy, the conventional approach is like Procedure-Oriented Programming (POP), which focuses more on problem-solving procedures. While KB approach is like Object-Oriented Programming (OOP), which solves problems by defining the well-described objects from the captured knowledge with reuse purpose. La Rocca (2012) gives an extended definition of KBE:

Knowledge based engineering (KBE) is a technology based on the use of dedicated software tools called KBE systems, which are able to capture and systematically reuse product and process engineering knowledge, with the final goal of reducing time and costs of product development by means of the following:

1. Automation of repetitive and non-creative design tasks;
2. Support of multidisciplinary design optimization in all the phases of the design process.

Knowledge Based System (KBS) are computer applications that use the KB approach to solve problems in a specific domain. It evolves from two types of Artificial Intelligence (AI) systems: the rule based systems (RBSs) and frame based systems (FBSs). The previous one is based on the well-known IF-THEN expert system, while the latter one is based on Object-Oriented (OO) knowledge representation, which is a closer ancestor of KBS (Negnevitsky, 2005). Compared with the non-KB approach, KBS solves problems by reasoning about facts. KBE adapts KBS towards the specific needs of the engineering design domain by enhancing the *geometry manipulation* and *data processing* (La Rocca, 2012). Parameterization is a key feature of KBE to represent the product knowledge in the OO approach. Ref. Zhang et al. (2021) demonstrates the ability of KBE to realize the geometry manipulation and data processing. This paper focuses on the data processing aspect of KBE application.

Ontology is a technology for knowledge representation, which is an important component of KBS. According to Gruber (1993) and Studer et al. (1998), an ontology is a formal, explicit specification of a shared conceptualization. The term “conceptualisation” refers to the abstract model of some phenomenon by extracting the relevant information from the real-world phenomenon containing infinite information. The term “formal” indicates that the representation should be in some sort of well understood logic to make itself machine-readable (Studer et al., 1998; Mika and Akkermans, 2003). “Explicit” refers to the fact that the type of concepts used, and the constraints on their use are explicitly defined (Studer et al., 1998), which means the relations and attributes related to the objects are pre-defined. While the term “shared” reflects that the knowledge captured in the ontology is accepted by the related community rather than private to some individuals (Studer et al., 1998;

Mika and Akkermans, 2003; Yang et al., 2019). Roughly speaking, ontology works as a more flexible data schema regulating how the data should be organized, which makes the ontology-based application easier to extend and integrate data from different sources.

2.2. Sustainability performance assessment

According to the United Nations report (Brundtland, 1987), sustainability is to meet the needs of the present without compromising the ability of future generations to meet their own needs. As introduced, the sustainability assessment of a supply chain requires the data from multi-disciplinary domains, which is a typical scenario for team collaboration. There are already some research implementing ontology to assess the sustainability performance for some specific domains. Muñoz et al. (2013) develops an ontological framework for the environmental sustainability assessment of the enterprise. In this work, the ontology, as the technology for knowledge sharing, provides an enterprise decision-making supporting tool by combining different information systems associated with the enterprise functions. Kuster et al. (2020) reconciles several domain-specific ontologies within one high-level ontology called the Urban District Sustainability Assessment (UDSA) ontology, which can support the creation of real-time urban sustainability assessment software. From these works, it can be seen that ontology is a powerful tool to share and integrate the sustainability knowledge. However, few study implements ontology-based framework in SCSA.

Regarding the supply chain sustainability assessment, Zhang et al. (2005) provide an easy-to-use calculation method based on fuzzy comprehensive evaluation (FCE) method. This method can take the uncertainty of the assess dimensions into account, expanding the amount of information and increasing the credibility of conclusion. Kucukvar et al. (2019) develop 14 macro level indicators to assess the supply chain of food consumption in the US. Dvaipayana et al. (2021) design a sustainable supply chain performance monitoring system considering 20 indicators from financial, internal business process and learning & growth perspectives. Yani et al. (2022) propose 24 indicators from economic, social, environmental and resource aspects for sustainability assessment of sugarcane agroindustry supply chain. Mursidah et al. (2020) also develop a model for SCSA of sugarcane agroindustry concerning 20 indicators using Artificial Neural Network (ANN) and Decision Tree. Meanwhile, there are some research (Yang and Song, 2009; Konys, 2018; Stark and Pfortner, 2015) providing the ontologies for sustainable product design to integrate the knowledge from different sources, which can be adapted to be used in KBE application for supply chain sustainability assessment.

3. The knowledge modeling

3.1. Overview

A KBE application can reuse the captured knowledge to conduct the supply chain sustainability assessment (SCSA) without manual intervention. An important step to develop a KBE application is to capture and formalize the knowledge for reuse purpose, and then represent it in proper format, e.g. ontology. As this paper is to demonstrate the potential of ontology-based framework in SCSA, rather than to propose a better calculation method and assessment framework for SCSA, the selection of the calculation method and the assessment framework is not a key concern in this paper.

This paper uses the ready sustainability assessment knowledge (the calculation method and the assessment framework) from other studies. As it is for demonstration purpose, the criteria to select are:

- The calculation method is simple, the input and output are clear, and better with example data.

Table 2

The factors and indexes for supply chain sustainability assessment.

Source: Adapted from Ref. Zhang et al. (2005).

Factor	Factor index
Finance value	Return on assets (ROA)
	Cash turnover ratio (CTR)
	Profit growth rate (PGR)
	Yield of net asset (YNA)
Environmental protection	Environmental protection efficiency (EPE)
	Materials utilization ratio (MUR)
	Environmental impact indicator(EII)
Information value	Information sharing ratio (ISR)
	Information flow rate (IFR)
	Information utilization ratio (IUR)
	Information inefficiency ratio (IIR)
Customer service	Customer lost rate (CLR)
	Customer satisfaction ratio (CSR)
	Customer valuable ratio (CVR)
Cost	Human resource cost (HRC)
	Materials flow cost (MFC)
	Information cost(IC)
Operation flexibility	Asset cost (AC)
	Order cycle time (OCT)
	Products flexibility (PF)
	Service response speed (SRS)
	Delivery flexibility (DF)
	Amount flexibility (AF)

- The calculation method covers various professional domains, reflecting the various data sources and the heavy workload of data conversion.
- The assessment framework is simple and compatible to the selected calculation method.

Based on the above-listed criteria, the method and data from Ref. Zhang et al. (2005) is selected. And an ontology from Ref. Yang and Song (2009) is adapted and reused to represent the sustainability assessment framework, i.e. to define the sustainability assessment options. Regarding other calculation methods and the assessment frameworks, as long as they can be expressed in an explicit form, they can be programmed as the components of the KBE application. Specifically, the calculation methods can be expressed as functions to be called by the KBE application. And the assessment framework can be expressed as ontology to be shared with other teams and to formalize their data in a unified format.

3.2. A sustainability assessment method for supply chain and the data as example

Once again, as the aim of this paper is not to study assessment method, the method and data from Ref. Zhang et al. (2005) are used for simplification purpose. The two-layer FCE method is implemented as the example method to integrate the different data sources for sustainability assessment. The short method introduction is as following:

1. Establish the assessment factors set U .

The paper uses the factors from Ref. Zhang et al. (2005): finance value, environmental protection, information value, customer service, cost, and operation flexibility. And specific indexes are chosen for each factor as the second layer (Table 2).

2. Establish the five-level assessment comments set V .

$$V = \{excellent, better, good, general, bad\}. \quad (1)$$

3. Establish the fuzzy assessment matrix R_i for each factor class.

R_1 as an example is

$$R_1 = \begin{bmatrix} 0.565 & 0.325 & 0.085 & 0.035 & 0 \\ 0.105 & 0.382 & 0.273 & 0.112 & 0.128 \\ 0 & 0.115 & 0.156 & 0.456 & 0.273 \\ 0.426 & 0.315 & 0.164 & 0.095 & 0 \end{bmatrix} \quad (2)$$

Each row in the matrix represents the membership degree distribution. The data in Eq. (2) is from Ref. Zhang et al. (2005) for simplification purpose, other matrixes (R_2 to R_6) are omitted to shorten the length. A detailed calculation introduction explains that each element is obtained by the classical ridge distribution calculation and normalization (Zhang and Feng, 2018). In some simple cases, these matrixes can be obtained by expert questionnaires.

4. Establish the weight vector for each factor and index.

The weight vector for factor is

$$W = [0.182, 0.225, 0.115, 0.165, 0.142, 0.171]. \quad (3)$$

And the weight vectors of each index set are:

$$\begin{aligned} W_1 &= [0.275, 0.225, 0.216, 0.284] \\ W_2 &= [0.235, 0.265, 0.274, 0.226] \\ W_3 &= [0.260, 0.260, 0.240, 0.240] \\ W_4 &= [0.328, 0.412, 0.260] \\ W_5 &= [0.230, 0.290, 0.250, 0.230] \\ W_6 &= [0.230, 0.175, 0.210, 0.200, 0.185] \end{aligned} \quad (4)$$

5. Calculate synthetical assessment matrix of single factor class.

Considering the weight vector for each index, the fuzzy matrix of synthetical assessment can be obtained:

$$B = \begin{bmatrix} W_1 \cdot R_1 \\ W_2 \cdot R_2 \\ W_3 \cdot R_3 \\ W_4 \cdot R_4 \\ W_5 \cdot R_5 \\ W_6 \cdot R_6 \end{bmatrix} = \begin{bmatrix} 0.299 & 0.289 & 0.165 & 0.160 & 0.087 \\ 0.045 & 0.115 & 0.248 & 0.394 & 0.198 \\ 0.118 & 0.199 & 0.394 & 0.182 & 0.107 \\ 0.152 & 0.294 & 0.250 & 0.207 & 0.097 \\ 0.160 & 0.333 & 0.283 & 0.174 & 0.050 \\ 0.164 & 0.284 & 0.298 & 0.172 & 0.082 \end{bmatrix} \quad (5)$$

6. Calculate synthetical assessment of supply chain performance.

Based on fuzzy assessment method, three fuzzy operators, i.e. $M(\wedge, \vee)$, $M(\text{power}, \wedge)$, $M(\cdot, +)$, are adopted to avoid unilateralism of the assessment. The three operators are introduced:

• $M(\wedge, \vee)$ operator: first take the minimum and then maximum.

$$W \circ B = [\vee_{i=1}^m (w_i \wedge b_{i1}), \vee_{i=1}^m (w_i \wedge b_{i2}), \dots, \vee_{i=1}^m (w_i \wedge b_{iP})], \quad (6)$$

where $\vee_{i=1}^m (w_i \wedge b_{iP}) = \max_{i=1}^m (\min(w_i, b_{iP}))$, m is the row number of the matrix B , P is the dimension of the assessment comments set V .

• $M(\text{power}, \wedge)$ operator: first take the power and then minimum.

$$W * B = [\wedge_{i=1}^m (b_{i1}^{w_i}), \wedge_{i=1}^m (b_{i2}^{w_i}), \dots, \wedge_{i=1}^m (b_{iP}^{w_i})]. \quad (7)$$

• $M(\cdot, +)$ operator: first take the product and then sum.

$$W \cdot B = \left[\sum_{i=1}^m w_i \cdot b_{i1}, \sum_{i=1}^m w_i \cdot b_{i2}, \dots, \sum_{i=1}^m w_i \cdot b_{iP} \right]. \quad (8)$$

Based on the fuzzy operators, the synthetical assessment matrix is obtained:

$$\tilde{B} = \begin{bmatrix} W \circ B \\ W * B \\ W \cdot B \end{bmatrix} = \begin{bmatrix} 0.182 & 0.182 & 0.225 & 0.225 & 0.198 \\ 0.478 & 0.615 & 0.720 & 0.716 & 0.652 \\ 0.154 & 0.246 & 0.264 & 0.227 & 0.110 \end{bmatrix}. \quad (9)$$

Taking weight vector $\tilde{W} = [1/3, 1/3, 1/3]$, the assessment result is obtained:

$$\tilde{S} = \tilde{W} \cdot \tilde{B} = [0.271, 0.348, 0.403, 0.389, 0.32]. \quad (10)$$

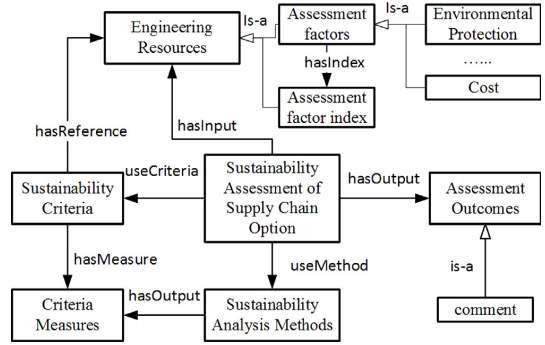


Fig. 1. The ontology about sustainability assessment framework. Source: Adapted from Ref. Yang and Song (2009).

7. Define a score vector $G = [100, 80, 60, 30, 10]$ corresponding to the assessment comment set V , and normalize the assessment result (\tilde{S}) to get the normalized assessment vector

$$\tilde{S}_n = [0.156, 0.201, 0.233, 0.225, 0.185]. \quad (11)$$

Then the score of the whole supply chain performance assessment is $S = G \cdot \tilde{S}_n = 54.54$.

3.3. A sustainability assessment ontology for supply chain

As introduced in the above, the ontology representing the sustainability assessment knowledge, as the model of data, can help to remove the ambiguity of multi-disciplinary knowledge and increase the extensibility and interoperability of the application. Therefore, an ontology adapted from Ref. Yang and Song (2009) is reused to define the sustainability assessment options, as illustrated in Fig. 1. This ontology consists of engineering resource, sustainability criteria, criteria measures, sustainability analysis methods and assessment outcomes, which means:

- Engineering resources: the factors considered in sustainability assessment, including finance, environment, logistic and etc.
- Sustainability criteria: the chosen criteria to judge from the criteria measures if the supply chain is good or not.
- Criteria measures: the score calculated by the method in Section 3.2.
- Sustainability analysis methods: the information on the calculation method, e.g., name, input, output.
- Assessment outcomes: the conclusion comment on the sustainability performance of the supply chain.

Fig. 2 elaborates an example of two sustainability assessment factors and an index with the attributes. Among the six assessment factors, “environmental_protection” and “cost” are expanded as examples, connected by the relation “hasIndex” with the corresponding indexes. And one index named “Materials_utilization_ratio_MUR” is instantiated with the example value as the attributes. In this way, all the data needed in the calculation can be stored in the ontology.

When the sustainability assessment for a supply chain is needed, the application first query and parse “sustainability assessment of supply chain option” related to this supply chain. Then the application read the “Sustainability analysis methods” object and call the calculation function pointed to by this object. This calculation function will retrieve the needed data including the membership degree, the weight vector and etc., execute the calculation, and store the result into “Criteria measures”. Then the application compares the “Criteria measures” and

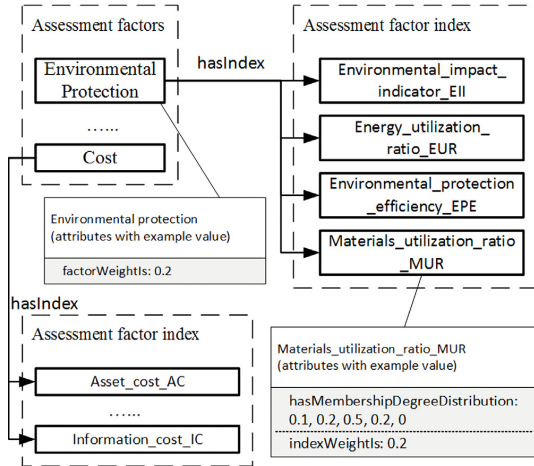


Fig. 2. The assessment factor example.

“Sustainability criteria”, and gives the “Assessment outcomes”, i.e. the “comment” in this paper.

After the assessment, all the related data will be updated to the ontology stored in the knowledge base. If other teams need the assessment comment, they can retrieve the comment through query language, and use the data in their environment. If the sustainability assessment team wants to apply a new calculation method, they can add the related information into the corresponding objects and write the new functions to work with the newly added data. And for the other teams, what they need is still the query language to retrieve the data they need, rather than to analyze and parse a new data file to extract the needed data. This is a case showing how ontology helps to promote the extensibility and interoperability of the application.

4. The KBE application

4.1. The application introduction

In order to demonstrate how to use ontology-based framework to improve the efficiency of the sharing and integrating knowledge across different phases of SCSA, this paper proposes a simple case evaluating the sustainability performance for a supply chain with KBE application. This case is to develop a KBE application based on the predefined sustainability assessment ontology for supply chain, with the FCE method as the calculation method to evaluate a supply chain.

This KBE application captures and formalizes the knowledge used in SCSA, which is shared with other teams. Thus the data from different sources can be integrated in a high-efficient way, and the supply chain sustainability assessment (SCSA) can be conducted without much manual intervention, which reduces the human labor and promotes the digitization level. Additionally, as the application is in ontology-based KBE framework, it is relatively easy to extend and interoperate with other applications, which also speeds up the development of PLM software for SCSA.

The architecture of the KBE application is illustrated in Fig. 3. The sustainability assessment team determines the assessment options and defines the knowledge in the ontology format. Then they share the ontology with other related teams and inform them to input the assessment needed knowledge of their domains into the knowledge base. Due to the benefits brought by ontology, they can understand what the sustainability assessment team needs and give the right input into the knowledge base (with their ontology_updater or in SPARQL

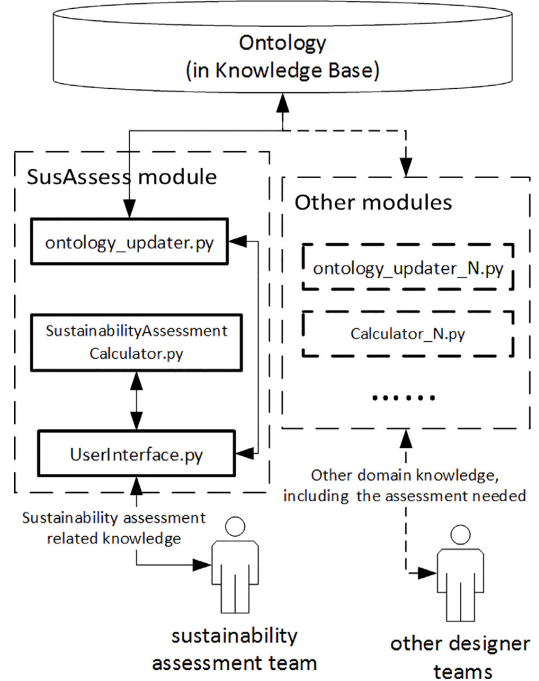


Fig. 3. The KBE application architecture.

language). After all the data (or knowledge) is collected into the knowledge base, the sustainability assessment team can type command in UserInterface.py to retrieve the needed knowledge without misunderstanding (by ontology_updater.py), and then conduct the assessment (by SustainabilityAssessmentCalculation.py) and update the result into the knowledge base (by ontology_updater.py). With this application, different teams can work together to compare several supply chains to choose the one with better sustainability performance. The modules in the architecture are described as following:

- ontology_updater.py: to query and update the knowledge in the knowledge base using SPARQL language.
- SustainabilityAssessmentCalculation.py: to calculate the sustainability performance score and give the conclusion.
- UserInterface.py: to receive the user input, call other modules and show the assessment result.

4.2. The demonstration

When the assessment needed knowledge is ready in the knowledge base, the sustainability assessment team can use the application to evaluate the supply chains automatically and update the assessment result to the unified knowledge base. They input the name of the supply chain and see the assessment result.

The process is demonstrated in Fig. 4. Fig. 4(a) gives an example how another expert team inputs their data into the knowledge base, i.e., the cost evaluation team is inputting the membership degree vector (0.565, 0.325, ..., 0) into the knowledge base in SPARQL language. Similarly, other teams, e.g. finance team, environment team and etc., can input their data in the same way. If needed, the KBE development team can also make a Graphic User Interface (GUI) for other teams to input data. Fig. 4(b) shows that the calculation module gives the

```

1 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
2 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
3 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
4 PREFIX owl: <http://www.w3.org/2002/07/owl#>
5 PREFIX susass: <http://www.semanticweb.org/lianz/ontologies/2021/10/sustainability_assessment#>
6 insert
7 { susass:Asset_cost_AC_sc1 a susass:Asset_cost_AC.
8   susass:Asset_cost_AC_sc1 a owl:NamedIndividual.
9   susass:Asset_cost_AC_sc1 susass:hasMembershipDegreeDistribution
10  "[0.565,0.325, 0.085, 0.035, 0]".}
11 WHERE {}

```

a) b)

```

1 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
2 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
3 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
4 PREFIX owl: <http://www.w3.org/2002/07/owl#>
5 PREFIX susass: <http://www.semanticweb.org/lianz/ontologies/2021/10/sustainability_assessment#>
6 select ?supply_chain ?comment
7 WHERE { ?supply_chain susass:hasComment ?comment .}

```

c) d)

supply_chain	comment
1 susass:comment_SC3	"excellent"
2 susass:comment_SC2	"general"
3 susass:comment_SC1	"good"

Fig. 4. The demonstration of the KBE application. (a) Different teams input their data into the unified knowledge base. (b) The calculation process. (c) The query to get the assessment result. (d) The assessment results of different supply chains.

assessment result after receiving all the needed data. After the calculation is done, the ontology updater module will send the result into the knowledge base, which is not shown in the figure. Fig. 4(c) illustrates that the sustainability assessment team queries the sustainability performance (the comment) of every supply chain in SPARQL language. This function can also be hard-coded into the Python script. Fig. 4(d) shows the list returned by the query, which shows the assessment outcomes for several supply chains. With this tool, the sustainability assessment team can conduct this collaborative task high-efficiently, then provide reference to support the decision-making related to supply chain optimization.

5. Discussion

The sustainability assessment ontology represents the related knowledge in an explicit form, which reduces the misunderstanding between different teams and helps to share and integrate knowledge across different stages. Furthermore, the extendability of ontology makes it practical to store knowledge in a unified database (knowledge base), which promotes the interoperability of different software applications.

Fig. 5 shows the different collaboration patterns. In conventional pattern, different teams have their own software systems generating the data files in different formats. If a team needs the data from different teams, they need to understand the data files rightly and write wrappers to parse the data and convert it into the format processable in their own systems, which is a low efficiency way. During this process, they can misunderstand the knowledge from different domains, as the data files are not designed to be shared with other software. When the system becomes large, to write wrappers for many data files can be time-consuming, not mention the chaos brought by the change of data formats.

The ontology-based knowledge base is a good way to figure this out. It provides an extendable format to represent the knowledge from different domains, which makes a unified representation possible. With this unified knowledge representation, data is explicitly explained, reducing the chance to misunderstand the multi-disciplinary data. And as no need to write the wrappers, the time in integrating the knowledge from different domains is shortened significantly.

Considering the data in Table 2, these data come from different sources, and in different formats. In conventional way, the assessment team need to communicate with other teams to study the usage of the data, and then write the wrappers to convert the data. This can be a heavy workload when there are many data sources and formats. With the help of ontology-based knowledge base, each team uploads the data in the format regulated in ontology, which is equivalent to that the data

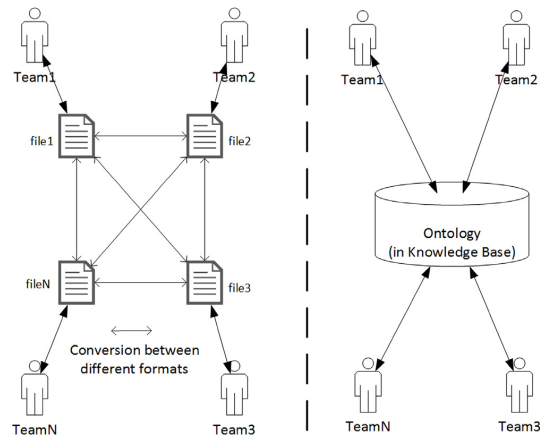


Fig. 5. The conventional v.s. ontology-based.

is converted before upload and other people do not need to convert any more.

The ontology-based knowledge representation facilitates the multi-disciplinary knowledge sharing and integration, simplifying the PLM software development, promoting the digital transformation and design automation. A high-level automation means high efficiency and less human labor needed. Less labor cost expense can increase the profit rate, which is helpful to the sustainable development of the companies. Furthermore, this can also reduce the burden of the sustainability assessment team, which can make sustainability assessment a widely used process in companies.

With the help of a high efficiency tool, the sustainability performance can be assessed automatically. The assessment result can provide reference to improve the sustainability performance of the supply chain. With a more sustainable supply chain, the companies are contributing to the environment, which is also advocated by the UN sustainable development goal (SDG) 12.2 that "By 2030, achieve the sustainable management and efficient use of natural resources" (United Nations, 2021).

Meanwhile, it is worth noticing that this paper is a proof of concept for the ontology-based framework. It shows that the ontology-based framework can facilitate to represent knowledge but not create new

knowledge. In other words, it can increase the efficiency of the software development, but cannot provide a better calculation method and assessment framework for SCSA.

6. Conclusions

This paper provides an example KBE application to assess supply chain sustainability performance, which is based on the reuse of a ready calculation and ontology-based knowledge representation. The ontology-based knowledge representation can simplify the PLM software development by facilitating the multi-disciplinary knowledge sharing and integration in collaborative tasks like sustainability assessment. From this example application, some conclusions can be drawn.

- Ontology-based knowledge representation can provide a unified and flexible data format for the teams in collaboration, which can reduce the misunderstanding caused by various data formats.
- Ontology-based knowledge representation with SPARQL query language makes the data exchange more human-readable than the conventional data exchange patterns.
- A unified and flexible data format makes a unified knowledge base possible.
- A unified knowledge base can reduce the time spent on writing data wrappers, which increases the efficiency of software development.
- The efficiency of SCSA itself can also be improved with a software, leading to a more sustainable supply chain, and also contributing to achieve the sustainable management and efficient use of natural resources.

In the future work, a larger scale of knowledge integration can be investigated to cover more stages of product lifecycle management, aiming at integrating more data sources and digitalizing more knowledge to provide more automation in PLM. Additionally, some more adaptive tools based on ontology-based knowledge representation need to be studied to manipulate the knowledge in the changeable formats, in order to improve the efficiency of capturing and formalizing the knowledge by providing more user-friendly experience.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Liang Zhang reports a relationship with Norwegian University of Science and Technology that includes: employment. Anna Olsen reports a relationship with Norwegian University of Science and Technology that includes: employment. Andrei Lobov reports a relationship with Norwegian University of Science and Technology that includes: employment.

References

Borsato, M., 2014. Bridging the gap between product lifecycle management and sustainability in manufacturing through ontology building. *Comput. Ind. 65* (2), 258–269.

Brundtland, G.H., 1987. Report of the World Commission on Environment and Development: "Our Common Future". UN.

Dvaipayana, M.A.T., Ridwan, A.Y., Santosa, B., 2021. Design of sustainable supply chain performance monitoring system for construction material management: Sustainable balanced scorecard-SCOR-ISO 14001 model. In: 2021 International Conference Advancement in Data Science, e-Learning and Information Systems (ICADEIS). IEEE, pp. 1–6.

Gruber, T.R., 1993. A translation approach to portable ontology specifications. *Knowl. Acquis. 5* (2), 199–220.

Jaskó, S., Skrop, A., Holczinger, T., Chován, T., Abonyi, J., 2020. Development of manufacturing execution systems in accordance with industry 4.0 requirements: A review of standard-and ontology-based methodologies and tools. *Comput. Ind. 123*, 103300.

Joshi, N., Dutta, D., 2004. Enhanced life cycle assessment under the PLM framework. In: *Proceeding of the International Intelligent Manufacturing Systems Forum*. pp. 17–19.

Kawajiri, K., Tahara, K., Uemiyu, S., 2022. Lifecycle assessment of critical material substitution: Indium tin oxide and aluminum zinc oxide in transparent electrodes. *Resour. Environ. Sustain. 100047*.

Konyas, A., 2018. An ontology-based knowledge modelling for a sustainability assessment domain. *Sustainability 10* (2), 300.

Kucukvar, M., Ismaen, R., Onat, N.C., Al-Hajri, A., Al-Yafay, H., Al-Darwish, A., 2019. Exploring the social, economic and environmental footprint of food consumption: a supply chain-linked sustainability assessment. In: 2019 IEEE 6th International Conference on Industrial Engineering and Applications (ICIEA). IEEE, pp. 733–742.

Kuster, C., Hippolyte, J.-L., Rezgui, Y., 2020. The UDSA ontology: An ontology to support real time urban sustainability assessment. *Adv. Eng. Softw. 140*, 102731.

La Rocca, G., 2012. Knowledge based engineering: Between AI and CAD. *Review of a language based technology to support engineering design. Adv. Eng. Inform. 26* (2), 159–179.

Mika, P., Akkermans, H., 2003. Analysis of the State-of-the-Art in Ontology-Based Knowledge Management. *Tech. Rep.*, Vrije Universiteit, Amsterdam.

Muñoz, E., Capón-García, E., Lainez, J.M., Espuña, A., Puigjaner, L., 2013. Considering environmental assessment in an ontological framework for enterprise sustainability. *J. Cleaner Prod. 47*, 149–164.

Mursidah, S., Djatna, T., Fauzi, A.M., et al., 2020. Supply chain sustainability assessment system based on supervised machine learning techniques: The case for sugarcane agroindustry. In: 2020 International Conference on Computer Science and its Application in Agriculture (ICOSICA). IEEE, pp. 1–7.

Negnevitsky, M., 2005. *Artificial Intelligence: A Guide to Intelligent Systems*. Pearson education.

Stark, R., Pfortner, A., 2015. Integrating ontology into PLM-tools to improve sustainable product development. *CIRP Ann. 64* (1), 157–160.

Studer, R., Benjamins, V.R., Fensel, D., 1998. *Knowledge engineering: Principles and methods*. *Data Knowl. Eng. 25* (1–2), 161–197.

United Nations, 2021. Sustainable development goal 12. URL: <https://sdgs.un.org/goals/goal12> [Online; accessed 29-December-2021].

Yan, K., Xu, J.-c., Gao, W., Li, M.-j., Yuan, Z.-w., Zhang, F.-s., Elser, J., 2021. Human perturbation on phosphorus cycles in one of China's most eutrophicated lakes. *Resour. Environ. Sustain. 4*, 100026.

Yang, L., Cormican, K., Yu, M., 2019. Ontology-based systems engineering: A state-of-the-art review. *Comput. Ind. 111*, 148–171.

Yang, Q., Miao, C., 2007. Semantic enhancement and ontology for interoperability of design information systems. In: 2007 IEEE Conference on Emerging Technologies and Factory Automation (EFTA 2007). IEEE, pp. 169–176.

Yang, Q., Miao, C., Zhang, Y., Gay, R., 2006. Ontology modelling and engineering for product development process description and integration. In: 2006 4th IEEE International Conference on Industrial Informatics. IEEE, pp. 85–90.

Yang, Q., Song, B., 2009. Semantic knowledge management to support sustainable product design. In: 2009 International Association of Computer Science and Information Technology-Spring Conference. IEEE, pp. 419–423.

Yani, M., Asrol, M., Hambali, E., Papilo, P., Mursidah, S., Marimin, M., et al., 2022. An adaptive fuzzy multi-criteria model for sustainability assessment of sugarcane agroindustry supply chain. *IEEE Access 10*, 5497–5517.

Zhang, L., Berisha, B., Lobov, A., 2021. A parametric model of umbilical cable with siemens NX considering its reliability. *IFAC-PapersOnLine 54* (1), 187–192.

Zhang, P., Feng, G., 2018. Application of fuzzy comprehensive evaluation to evaluate the effect of water flooding development. *J. Pet. Explor. Prod. Technol. 8* (4), 1455–1463.

Zhang, H.C., et al., 2005. Study on the performance assessment of green supply chain. In: 2005 IEEE International Conference on Systems, Man and Cybernetics, Vol. 1. IEEE, pp. 942–947.

Paper C

Extending design automation by integrating external services for product design

© 2021 IEEE. Reprinted, with permission, from:
Zhang, Liang, and Andrei Lobov. "Extending design automation by integrating external services for product design." 2021 IEEE 19th International Conference on Industrial Informatics (INDIN). IEEE, (2021).

URL: <https://doi.org/10.1109/INDIN45523.2021.9557486>

In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of NTNU's products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

Extending design automation by integrating external services for product design

Liang Zhang

Department of Mechanical and Industrial Engineering
Norwegian University of Science and Technology
Trondheim, Norway
Email: liang.zhang@ntnu.no

Andrei Lobov

Department of Mechanical and Industrial Engineering
Norwegian University of Science and Technology
Trondheim, Norway
Email: andrei.lobov@ntnu.no

Abstract—Industry 4.0 aims at the promotion of design and manufacture automation to provide customized products. Knowledge-Based Engineering (KBE) is seen as a competitive way to realise design automation. This paper first introduces the Methodology and software tools Oriented to Knowledge-Based engineering Application (MOKA) method to develop an KBE application integrated with external calculators. Then a simple case of the wood-to-wood connection with different fasteners providing the connection capacity is demonstrated, showing the advantages to establish a model in parametric method integrated with external services. Meanwhile, this case shows the shortcomings of the integration of the current tools representing knowledge in different frameworks. Then a potential better way to integrate engineering knowledge is discussed.

KBE; Knowledge integration; Parametric model; Design automation; Service-Oriented Architecture;

I. INTRODUCTION

The customized products rather than standard products have become a trend. A vision of the Fourth Industrial Revolution (or Industry 4.0) is to meet the demand for mass production of customized products. Industry 4.0 is the digital transformation of production aiming at upgrading the traditional manufacturing to a high level of automation using modern smart technology [1]. Only with a more complete design and manufacture automation, the mass production of customized products can be realised.

Smart factory is a part of Industry 4.0, aiming at manufacturing automation through the implementation of the smart sensors, Internet of Things (IoT) and other related technologies [2]. Meanwhile, design automation tools are also an important part of the mass production of customized products, providing the feasible and manufacturable design from the customer's demand automatically. Currently, more and more design tools emerge to provide the customers with the freedom to design personalized products. This kind of design tools can be seen driven by Knowledge-Based Engineering (KBE) method. Based on the parametric templates of a product, the design tools give the users the access to change some of the parameters. However, the customized products at present are relatively simple, which means no complex engineering analysis is needed. When the customized products become more and more complex, the analyses in different domains need to be conducted to give the final result whether the

design is feasible and manufacturable. Generally, it is difficult for a design software development team to understand and implement all the needed knowledge of different problem domains and integrate the analysis ability (knowledge) with the design tools. In order to provide the customers with a design tool which can generate a design solution for the following process automatically, the integration of external calculators (services) provided by other experts becomes a solution.

In this paper, the design of a wood-to-wood connection with different fasteners is chosen as an example to demonstrate the integration with the external calculators. Firstly, a practical way to develop a parametric model considering the external calculator is introduced. Secondly, the automatic design application which can generate the 3D model of wood-to-wood connection in Siemens NX Knowledge Fusion (the Knowledge-Based module) code is demonstrated to show how the external calculator is integrated. Finally, the benefit and limitation of the current integration is discussed and the advice on the future Knowledge-Based Engineering (KBE) application is given.

II. STATE OF THE ART

A. KBE

KBE is the engineering using product and process knowledge that has been captured and stored in dedicated software applications, to enable its direct exploitation and re-use in the design of new products and variants [3]. One of the main features of KBE is the parametric model. Compared with the traditional CAD method, KBE method represents a model based on parameters and rules that can be embodied in a template. Knowledge organised in such form can be re-used relatively easily. Variants of an existing product can be designed rapidly, which makes the mass production of customized products possible.

With the significant benefit to support the rapid development of products, KBE support has been introduced into many CAD software products, for example, Autodesk Intent for Autodesk Inventor and Knowledge Fusion (KF) for Siemens NX. The example in this paper is represented in Knowledge Fusion code, which is an interpreted, object-oriented language that allows adding engineering knowledge to a part by creating rules which are the basic building blocks of the language

[4]. As an interpreted language, Knowledge Fusion is platform independent compared with compiled language, which is a helpful feature when deploying the KBE application in different platforms. KBE method has been implemented in some engineering products design. Tian et al. presented a new method based on KBE that can automatically generate the software components, which saves the designers from many repetitive (manual) processes [5]. Some researchers dedicated to extend the design ability of KBE applications. Saa et al. developed an ontology database integrated with experts knowledge and railway standards to support decision making for high-performance and cost-optimized design of complex railway portal frames [6]. Lobov et al. proposed the use of Knowledge Fusion for generation of robot trajectories to support faster transition from a product information in CAD and KF till the robot code that should weld the product [7]. In summary, KBE method has become a useful tool in design application development.

B. Knowledge integration

Knowledge integration generally refers to the process of merging two or more originally unrelated knowledge structures into a single structure, which is the organic combination of the existing knowledge models of different sources [8] [9]. It is a generalized concept which appears in many fields, like database, machine learning, knowledge engineering and etc. For example, Shen et al. integrated railway signal maintenance knowledge from different sources to develop an application for maintenance knowledge push [9]. Here in this paper which is about a KBE application for product design, knowledge refers to the rules implemented in design process [10], so knowledge integration means to integrate the related knowledge from different domains used in design process into the automatic design application.

Service-Oriented Architecture (SOA) provides a good solution to integrate external service with the KBE design application. SOA is a software design method aiming to provide services to the other users through a communication protocol over a network, which is designed to be independent of platforms, products and technologies [11]. Such architecture makes each component of the application loosely coupled, which is easy to maintain and integrate new modules. This feature of SOA is helpful to dynamically meet the user's changing demands. When users propose some new demands, only the relevant new modules need to be added into the existing application without modifying other components. Some research proposed to implement SOA in KBE domain to provide customers the wanted service. Chen et al. proposed an ontology-based SOA framework to allow different legacy traditional Chinese medicine (TCM) information systems to interact, intercommunicate, and interoperate with each other [12]. Puttonen et al. proposes to implement semantic web services based on SOA to enable flexible manufacturing systems [13]. However, few research is about integrating SOA external calculators with design application based on KBE.

C. Ontology-based engineering knowledge representation

One of the most important topics of intelligent systems is to represent the knowledge in the problem domain. The trend nowadays to represent engineering knowledge is to use ontologies and rules [6]. Ontology is designed to describe and identify the concepts to be shared in the semantic web. It enables reuse of domain knowledge and allow the users to modify the model without changing the database.

Some researchers implement ontology to represent engineering knowledge. Tran et al. proposed a method to implement ontology in KBE application development which is for 3D geometry generation, to gain the freedom to incorporate development tools regardless of discipline or platform [14]. Liu et al. presented an ontology-based framework to deal with the consistency in semantic representation of exchanged product knowledge in collaborative manufacturing [15]. Sanya et al. constructed the ontology model of their KBE system in the aerospace industry, which strengthens the knowledge reuse and eliminates platform-specific approaches to developing similar KBE systems [16]. These studies show that ontology-based engineering knowledge representation, as a higher level of abstraction, is helpful to deploy KBE application in different platforms.

D. Summary

It can be seen that KBE method has been implemented in some engineering products design because the applications based on KBE method are easy to modify and integrate different knowledge domains. Meanwhile, SOA aims at providing services regardless of different platforms, which brings the convenience to integrate different design tools. This paper proposes an approach for creating KBE applications based on SOA that enables integration of heterogeneous services. Additionally, the potential benefits of implementing ontology-based engineering knowledge representation are discussed.

III. METHODS

A. Architecture

There are several KBE methodologies to develop the KBE applications, among which the Methodology and software tools Oriented to Knowledge-Based engineering Applications (MOKA methodology) is the most well-known one. This methodology divides the KBE application development into eight life-cycle steps, among which the two main focuses are "Capture" and "Formalize" [17] [18]. "Capture" is to collect and arrange the product knowledge in a structured form, which makes sure all the product parameters are obtained rightly. "Formalize" is to establish the model of product or process, in other words, to represent the knowledge using parameters and templates (rules).

The automatic design application in this paper is developed in the above-mentioned way. Aiming at easy maintenance and extension of more calculators, the Service-Oriented Architecture (SOA) is selected and illustrated in Fig. 1 and the sequence is illustrated in Fig. 2. The product knowledge is represented through data class and parametric templates stored

in knowledge base server. The design control program which can be called "designer" reads the data class to determine what to input, and then interacts with the user to receive the input parameters. After obtaining the needed input parameters, the designer program calls the external service to calculate the relevant performance and parses the returned result. Then the designer program passes all the parameters including geometric and non-geometric aspect to the DFA file generator. The generator passes the parameters into the parametric templates written in Knowledge Fusion code to generate the final model designed by the customer and store it in a file with DFA extension. Then the design of a customized product is completed in such way.

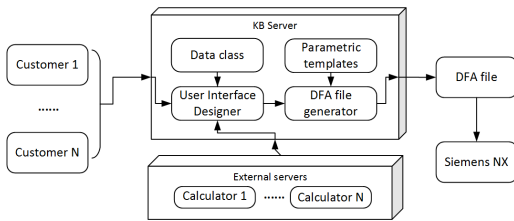


Fig. 1. The architecture of the automatic design application.

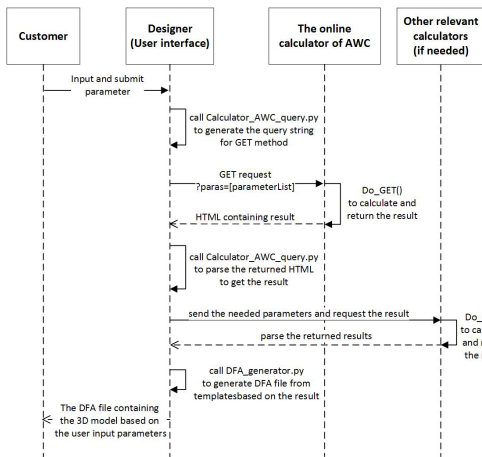


Fig. 2. The UML sequence diagram of the automatic design application.

B. The parametric model integrated with external service

To realise the above-mentioned automatic design KBE application, the parametric model and the integration with the external service are the key points. This paper proposes to develop the automatic design application in following steps. Firstly, build the initial DFA file of the product to check if all the product parameters are captured. Secondly, parameterize the connection model and convert DFA draft file into parameters plus templates considering the parameters used in the

online calculator. Thirdly, integrate the parametric model with the external service with suitable method, e.g. HTTP service. Then design the user interface to give the customer access to the templates based on the selected parameters. Finally, test and deploy the entire system for daily use.

IV. CASE STUDY

A. The case introduction

In order to demonstrate how the KBE application integrated with external calculator works in design automation, this paper proposes a simple case that two wood boards need to be connected in different ways providing sufficient lateral connection capacity. Wood-to-wood connection is widely used in furniture, which can be seen as a customized product based on the customer's demand. Different connection ways including bolt, nail, lag screw, wood screw will be used for different purposes as each provides different lateral capacity and applies in different cases. (see Fig. 3)

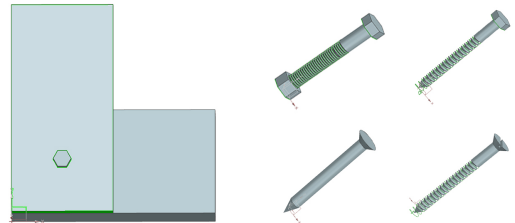


Fig. 3. Two wood boards and the four different fasteners.

Some professional organizations are providing online tools capable to calculate the connection capacity. American Wood Council (www.awc.org) provides users with a web-based approach to calculating capacities for single bolts, nails, lag screws and wood screws. Users select the connection parameters in the online user interface, e.g. the fastener type, the thickness of the main and side member, the diameter and length of different fasteners [19]. After user submission of the input parameters, the calculator will return the connection capacity, which is called the Adjusted ASD Capacity based on their professional algorithm. For simplification purpose, only part of the parameters concerned in the calculator will be considered in the model of this paper. (see Fig. 4)

For knowledge re-use purpose, an automatic design program which can provide the 3D model together with the connection capacity is developed using parametric modeling and online calculator. With this kind of tool, customers can design their wanted products by inputting the parameters of their own design and see whether they can be manufactured. Furthermore, if the automatic design program is connected with the smart factory, the customized products can be manufactured automatically after being ordered online, which is one of the visions of Industry 4.0.

Design Method	Allowable Stress Design (ASD)
Connection Type	Lateral loading
Fastener Type	Bolt
Loading Scenario	Single Shear - Wood Main Member
Submit Initial Values	

Main Member Type	Alaska Cedar
Main Member Thickness	2.5 in.
Main Member: Angle of Load to Grain	0
Side Member Type	Alaska Cedar
Side Member Thickness	0.5 in.
Side Member: Angle of Load to Grain	0
Fastener Diameter	1/2 in.
Load Duration Factor	C _D = 1.0
Wet Service Factor	C _M = 1.0
Temperature Factor	C _t = 1.0

Calculate Connection Capacity

Connection Yield Modes

Im	1641 lbs.
Is	328 lbs.
II	616 lbs.
III _m	796 lbs.
III _s	492 lbs.
IV	693 lbs.

Adjusted ASD Capacity	328 lbs.
-----------------------	----------

Fig. 4. The external online calculator to be integrated [19].

B. Development of the automatic design program

The automatic design program is developed in the above-mentioned method. Firstly, the prototype of the wood-wood connection with different fastener types are modeled in Siemens NX Knowledge Fusion code, collecting and structuring all the needed engineering knowledge (geometric and non-geometric) to represent the product. The model code in this process can represent only one specific product and inconvenient to generate the variants. As the components used in the model are standard ones, if designer needs another fastener, he needs to input all the basic parameters by hand. To summarise, the parameter set are not simplified and the model are not well formalized. To formalize the model, a well-selected parameter set are needed. A good selection helps to reduce workload in the calculation part, otherwise a plenty of conversion will be needed. Thereby, the parameters are often selected considering the following process. If a parameter is always used as an input of the calculation, it is better to make it a basic parameter rather than a deduced one. A practical mean to select the parameters is to first select all the calculation needed parameters as basic and try to deduce the others from the basic ones. Then add those which are hard to deduce into the selection set (see table I).

In this paper, the deduced parameters are obtained through two means. Some can be calculated through formula, and some parameters that must be standard values are provided using a list storing the standard values in the template (see Fig. 5).

After parameterizing the model, the external calculator can be integrated using HTTP service, which will be introduced later.

```

#! NX/KF 4.0
DefClass: twoboardtoconnect_woodScrew (ug_base_part);
(vector parameter) direction: vector(0,0,1);
(point parameter) start_point: point(45,0,0);
#the dimensions of the 2 boards
(number parameter) L_1: <mm_length>;
(number parameter) W_1: <mm_width>;
(number parameter) Th_1: <mm_thickness_text>;
(number parameter) L_2: <sm_length>;
(number parameter) W_2: <sm_width>;
(number parameter) Th_2: <sm_thickness_text>;
#the position to connect
(point parameter) connect_point: point(min(L_1, L_2:)/2,
min(W_1, W_2:)/2,Th_1 + Th_2:) + (start_point: - point(0,0,0));
#the required mechanical performance
(number parameter) connection_capacity: <connection_capacity>;
(number parameter) connection_requirement: <connection_requirement>;
(number parameter) diameter_cylinder: <diameter_wood_screw>;
(number parameter) length_wood_screw: <ls_length>;
#standard values stored in a list
(list parameter) head_thickness_list: {0.035, 0.043, 0.051,
0.059, 0.067, 0.075, 0.083, 0.091, 0.1, 0.108,
0.116, 0.132, 0.153, 0.164, 0.191, 0.196, 0.23};
(list parameter) diameter_list: {0.06, 0.073, 0.086, 0.099,
0.112, 0.125, 0.138, 0.151, 0.164, 0.177, 0.19,
0.216, 0.242, 0.268, 0.294, 0.32, 0.372};

```

Fig. 5. The excerpt of the DFA template.

TABLE I
PARAMETERS LIST.

Parameter	Remark
main member thickness	inch
main member length	inch
main member width	inch
side member thickness	inch
side member length	inch
side member width	inch
fastener type	Bolt, Lag+Screw, Wood+Screw, Nail
bolt diameter	must be standard value
lag screw wash thickness	must be standard value
lag screw diameter	must be standard value
lag screw length	must be standard value
wood screw diameter	must be standard value
wood screw length	must be standard value
nail type	common wire, sinker, box
nail size	must be standard value
connection capacity	pound
connection capacity requirement	pound

The auto-design program consists of 4 modules: user interface, data class, external service integrator, 3D model generator. Each of them is realised through a Python script:

- Designer.py: the design control program containing the user interface for parameters input.
- ConnectionClass.py: the data class containing the parameters (see table I).
- Calculator_AWC_query.py: the integrator that generates the query returning the result from the external service which is an online professional calculator, and also parses the returned result.
- DFA_generator.py: generate the 3D model (DFA file) using input parameters and result from the calculator.

The user interface (UI) module is used to guide the user to input the needed parameters which are defined in the data class and to call the other two modules to finish the design process.

For simplification purpose, the UI module in this paper is command-line style. However, it can be easily replaced by a graphical user interface (GUI) which can receive the user input in a more user-friendly way. As mentioned above, all the parameters are selected and defined in a data class, which is also beneficial for the future product upgrade, as the Object-Oriented template can be easily extended.

The external service integrator is the key to analyse the product performance. Given a professional online calculator, it is convenient for the user to get the exact connection capacity after inputting the design parameters properly. As shown in Fig. 2, this paper integrates the external calculator by implementing the “GET” method of HTTP protocol. The integrator script (Calculator_AWC_query.py) receives the user-input parameters transferred from the UI module and generates a query URL containing the parameters, then requests the data from the server by sending the generated URL. After receiving the request data containing the result, the integrator script parses it and returns the wanted connection capacity value to the designer script. In this paper, only one external calculator is integrated. However, in some complex design process, designers need to consider not only engineering performance, but also design standards, regulatory and safety codes, product cost and etc. [20]. Thus more calculators need to be integrated with the designer program.

After the user input and calculation process, all the needed data are prepared and transferred to the 3D model generator. This paper uses Siemens NX Knowledge Fusion to generate the product model (stored in a file with DFA extension that contains Knowledge Fusion code). The DFA generator fills in the parametric templates with the processed data by replacing the parameter placeholders, then generates the wanted DFA file containing the product model. User can visualise the model in Siemens NX for demonstration purpose.

C. Result

The design tool can generate the 3D model of the connection with different fasteners automatically based on the user input. Fig. 6 shows the different models given by the design tool. When the model is in green, it means this design can provide a greater connection capacity than the user requirement (user input in advance). When in red, this design fails, indicating a better design is needed. This can be useful when customers or engineers design their customized products, which contributes to the future production of customized products.

V. DISCUSSION

The automatic design tool based on KBE method and integration with external service has a plenty of advantages. The tool in this paper can quickly generate a 3D model (DFA files) according to the customers’ demands. Integrated with existing professional calculator, this tool can provide reliable calculation results. During the application development process, the software engineers are not required to understand the problem domain knowledge very well due to the knowledge re-use feature. In summary, this method gives a possibility for

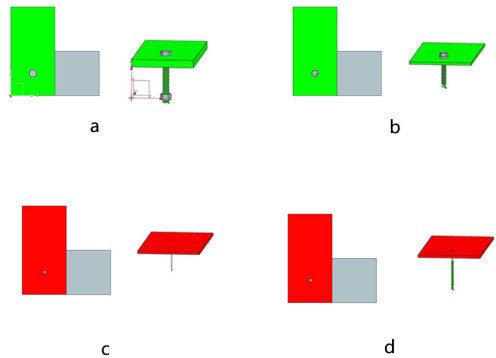


Fig. 6. Four connection designs with different fasteners: (a) bolt; (b) lag screw; (c) nail; (d) wood screw.

integrating a design tool with external services, promoting the customized product design.

However, there are also some limitations of current framework.

- There are different representing rules in different modules. For instance, at first, the local design module and the external calculator use different parameter names and parameter set to represent the connection. In order to simplify the program, the rules in local design modules are modified to keep the same with the external calculator. However, this can be difficult when a lot of external services get involved.
- As the external calculator in this paper is not designed in a SOA way, it can be time-consuming and unrobust to extract the calculation result.
- Some reusable knowledge involved in this design are only human readable, needed to convert to different schema, e.g. the dimensions of the standard components (bolts and etc.) are stored in standard files rather than a digital schema.
- The parametric model is not dynamic, in other words, it can be hard for the user to add new features into the templates, which may restrict the design freedom.
- There are still some “if-then” rules in the templates, which may limit the handling of complex products.

Ontology-based machine-readable model is seen as a potential solution to the above-mentioned difficulties. It can represent product knowledge in a dynamic form, providing the convenience to the user to extend the templates. When the product becomes complex, it provides an efficient way to query the proper rules in the knowledge base compared with the “if-then” style. Furthermore, if ontology becomes a widely used way to represent knowledge, it can help to provide the exchangeable and machine-readable knowledge and SOA style calculators.

Overall, it is envisioned, that standardization or professional

organizations, like in the case of using the online calculators from the American Wood Council [19], make it possible to dynamically integrate and extend product design with expert information provided by such organizations. As, for example, connection capacity is calculated in this paper by an online request to the AWC calculator from the CAD software getting back results that are immediately shown in the 3D model at the site of a product designer for the feasibility check.

VI. CONCLUSIONS

In this paper, an architecture allowing integration of external services (e.g. that by AWC) is outlined and demonstrated to very early product design phase, where products parameters can be updated or selected depending on performance of envisioned product. It becomes easy to integrate such knowledge and it is important to keep it with professional or standardization organization motivating those to extend and expose these types of “calculators” as online services, which are possible to integrate with CAD with the help of APIs.

In the future work the extending of the services demonstrating multidisciplinary approaches will be explored, where the “calculators” or the services can come from different expert groups.

REFERENCES

- [1] F. Yang and S. Gu, “Industry 4.0, a revolution that requires technology and national strategies,” *Complex & Intelligent Systems*, vol. 7, no. 3, pp. 1311–1325, 2021.
- [2] E. Hozdić, “Smart factory for industry 4.0: A review,” *International Journal of Modern Manufacturing Technologies*, vol. 7, no. 1, pp. 28–35, 2015.
- [3] J. Sobieszczanski-Sobieski, A. Morris, and M. Van Tooren, *Multidisciplinary design optimization supported by knowledge based engineering*. John Wiley & Sons, 2015.
- [4] Siemens website, “Knowledge fusion introduction,” https://docs.plm.automation.siemens.com/tdoc/nx/11/nx_api, 2016, [Online; accessed Nov.-2020].
- [5] F. Tian and M. Voskuijl, “Knowledge based engineering to support electric and electronic system design and automatic control software development,” in *2013 IEEE/AIAA 32nd Digital Avionics Systems Conference (DASC)*. IEEE, 2013, pp. 7A4–1.
- [6] R. Saa, A. Garcia, C. Gomez, J. Carretero, and F. Garcia-Carballeira, “An ontology-driven decision support system for high-performance and cost-optimized design of complex railway portal frames,” *Expert systems with applications*, vol. 39, no. 10, pp. 8784–8792, 2012.
- [7] A. Lobov, T. A. Tran, and S. A. O. Prescott, “Formalization of engineering knowledge for industrial robots using knowledge fusion language,” *Procedia Manufacturing*, vol. 51, pp. 932–937, 2020.
- [8] M. Schneider, *Knowledge Integration*. Boston, MA: Springer US, 2012, pp. 1684–1686. [Online]. Available: https://doi.org/10.1007/978-1-4419-1428-6_807
- [9] S. Xiao-bin, Z. Zhen-hai, and Z. Jingtian, “Method of networked knowledge integration and knowledge push for railway signal maintenance,” *2018 Chinese Automation Congress (CAC)*, pp. 4038–4043, 2018.
- [10] A. Lobov, “Smart manufacturing systems: climbing the dikw pyramid,” in *IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 2018, pp. 4730–4735.
- [11] R. Perrey and M. Lycett, “Service-oriented architecture,” in *2003 Symposium on Applications and the Internet Workshops, 2003. Proceedings*. IEEE, 2003, pp. 116–119.
- [12] S.-W. Chen, Y.-T. Tseng, and T.-Y. Lai, “The design of an ontology-based service-oriented architecture framework for traditional chinese medicine healthcare,” in *2012 IEEE 14th International Conference on e-Health Networking, Applications and Services (Healthcom)*. IEEE, 2012, pp. 353–356.
- [13] J. Puttonen, A. Lobov, and J. L. M. Lastra, “Semantics-based composition of factory automation processes encapsulated by web services,” *IEEE Transactions on industrial informatics*, vol. 9, no. 4, pp. 2349–2359, 2012.
- [14] T. A. Tran and A. Lobov, “Ontology-based model generation to support customizable kbe frameworks,” *Procedia Manufacturing*, vol. 51, pp. 1021–1026, 2020.
- [15] W. Liu, Y. Jiang, and T. Jin, “Research on integration of ontology-based product knowledge: Framework, schema and algorithm,” in *2009 IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications*. IEEE, 2009, pp. 565–569.
- [16] I. Sanya and E. Shehab, “An ontology framework for developing platform-independent knowledge-based engineering systems in the aerospace industry,” *International Journal of Production Research*, vol. 52, no. 20, pp. 6192–6215, 2014.
- [17] W. J. Verhagen, P. Bermell-Garcia, R. E. Van Dijk, and R. Curran, “A critical review of knowledge-based engineering: An identification of research challenges,” *Advanced Engineering Informatics*, vol. 26, no. 1, pp. 5–15, 2012.
- [18] R. Klein, “Knowledge modeling in design—the moka framework,” in *Artificial Intelligence in Design'00*. Springer, 2000, pp. 77–102.
- [19] American Wood Council website, “American wood council connection calculator,” <https://www.awc.org/codes-standards/calculators-software/connectioncalc>, 2015, [Online; accessed Jan.-2021].
- [20] S. Cooper, *Achieving competitive advantage through knowledge-based engineering: a best practice guide*. Prepared for the Department of Trade and Industry by Department of Enterprise Integration, Cranfield University, 1999.

Paper D

Interoperability in automating engineering tasks: An illustration with pipe routing application

© 2023 IEEE. Reprinted, with permission, from:
Zhang, Liang, and Andrei Lobov. "Interoperability in automating engineering tasks: An illustration with pipe routing application." IECON 2023-49th Annual Conference of the IEEE Industrial Electronics Society. IEEE, 2023.

URL: <https://doi.org/10.1109/IECON51785.2023.10311846>

In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of NTNU's products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

Interoperability in automating engineering tasks: An illustration with pipe routing application

1st Liang Zhang

*Dept. of Mechanical and Industrial Engineering
Norwegian University of Science and Technology
Trondheim, Norway
liang.zhang@ntnu.no*

2nd Andrei Lobov

*Dept. of Mechanical and Industrial Engineering
Norwegian University of Science and Technology
Trondheim, Norway
andrei.lobov@ntnu.no*

Abstract—Smart manufacturing solutions require well-informed decisions at the engineering stage. Such decisions become possible when the users may have the right tools at the right time. As there may be different tool preferences among engineers in different problem domains or also organizations working within the same problem domain, the interactions between the people working on a common project can be hindered as they need to agree first on the toolset. Thus, supporting different tools and formats can be a valuable asset in engineering projects. This paper presents an approach for building an interoperable solution to be able to support different engineering tools. The approach is illustrated using a pipe routing application based on genetic algorithm (GA) and A* algorithm. The output can be handled by end users in web browser, Siemens NX and AVEVA tools. The solution can also be extended to include other tools. The evaluation of the proposed approach is conducted using a qualitative metric that considers the three dimensions of the system development methodology. Through this paper, the benefit is demonstrated that the utilization of semantic data in engineering design can enhance the interoperability of design software tools.

Index Terms—design automation, interoperability, industrial informatics, semantic data, smart manufacturing

I. INTRODUCTION

Industry 4.0 demands a higher automation level in the product lifecycle including the design stage [1], which entails well-informed decisions at the engineering design stage. However, as there may be different tools preferences among engineers in different problem domains or also organizations working within the same problem domain, the interactions between the people working on a common project can be hindered as they need to agree first on the toolset. Furthermore, these software tools are typically designed for a document-centric design workflow, lacking consideration for effortless interaction with other tools. Consequently, interoperability issues arise, requiring additional efforts to establish an integrated toolchain for engineering design. Therefore, supporting various tools and formats can be a valuable asset in engineering projects.

The document-centric design workflow is built based on the software tools that can only import and export data in specific file formats. The data is exchanged through documents (files) in this kind of workflow. Fig. 1 shows a typical document-centric design workflow. The coordinator analyses the specification documents and finds similar previous designs,

then assigns the design task to the downstream team. The design engineer and analysis engineer collaborate in iteratively generating the optimized geometric model through interactions with Computer-Aided Design (CAD) and Computer-Aided Engineering (CAE) software in several loops. In this type of workflow, the information needed by downstream is contained in documents, e.g., design requirement in the specification, geometric model in CAD software files, and analysis model in CAE software files. Some ad-hoc automation solutions can be developed via parsing the needed data from data files in a specific format. However, parsing the data files is not always effortless, resulting in the interoperability issues between different software tools. Thus, the importance of semantic data has drawn increasing attention in the engineering domain. A data-centric workflow that facilitates the exchange of semantic data between different processes offers a potential solution to enhance interoperability.

Knowledge-based Engineering (KBE) originally refers to the object-oriented and parametric approaches to construct geometric model of engineering products [2], [3]. As the model built in KBE style has the potential to provide semantic data, allowing for interaction with other software tools, KBE also sometimes refers to the automation of repetitive tasks in product development based on the reuse of knowledge [4]. Despite the discussion of narrow and broad definitions of KBE, its inherent capability to provide semantic data can improve interoperability [4], enabling the establishment of a toolchain for automating engineering design tasks.

In summary, as the current tools are designed without considering interoperability, bringing the difficulties in easily forming a toolchain, it is natural to ask a research question (RQ): **How to build an interoperable solution to be able to support different engineering tools for a toolchain?** This paper contributes to this RQ by presenting a practical approach using KBE style modeling and ontology-based knowledge base (KB) to provide semantic data access. The approach is illustrated using a pipe routing application based on genetic algorithm (GA) and A* algorithm. The output can be handled by end users in web browser, Siemens NX and AVEVA tools. The solution can be also extended to include other tools.

The remainder of this paper is organized as follows. Section II introduces the theoretical background and the related works.

Section III explains the proposed approach and section IV gives an implementation example. Then a comparison between KBE solution and ad-hoc solution is discussed in section V. And finally, section VI concludes the paper. The abbreviations used in the paper are explained in TABLE I.

TABLE I
LIST OF ABBREVIATION USED IN THE PAPER

Abbreviation	Explanation
AI	Artificial Intelligence
API	Application Programming Interface
CAD	Computer-Aided Design
CAE	Computer-Aided Engineering
CommonKADS	Common Knowledge Acquisition and Design Support
CRUD	Create, Read, Update and Delete
GUI	Graphic User Interface
ICARE	Illustrations, Constraints, Activities, Rules and Entities
JSON	JavaScript Object Notation
KB	Knowledge-Based, Knowledge Base
KBS	Knowledge-Based System
KBE	Knowledge-Based Engineering
MOKA	Methodology for Knowledge-Based Engineering Applications
MBSE	Model-Based System Engineering
OO	Object-Oriented
OPC UA	Open Platform Communications Unified Architecture
OWL	Web Ontology Language
PML	Programmable Macro Language (for AVEVA)
RDF	Resource Description Framework
SPARQL	SPARQL Protocol and RDF Query Language
STEP	Standard for the Exchange of Product Data

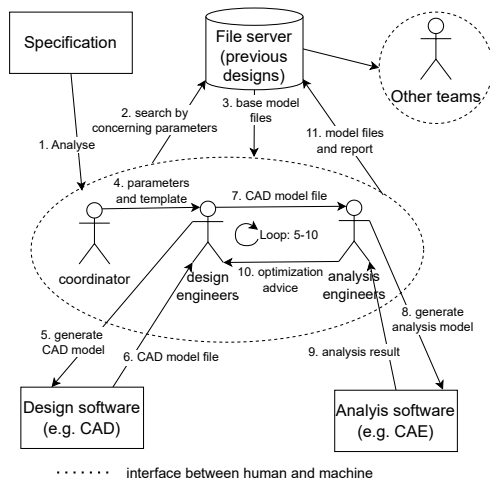


Fig. 1. A typical document-centric mechanical design workflow.

II. BACKGROUND AND RELATED WORKS

A. Semantic data and Ontology

Semantic data refers to data that is enriched with explicit semantic content. It goes beyond the relational data and

includes additional information that describes the semantics of the entities and relationships, including descriptions of connections and consistency constraints [5]. Semantic data plays a vital role in enabling effective tool integration and enhancing collaboration in various domains. For instance, in the manufacturing field, OPC UA (Open Platform Communications Unified Architecture) has emerged as a prominent standard that leverages semantic data representation [6]. OPC UA allows for seamless communication and interoperability between different manufacturing systems and devices by utilizing a standardized and semantically rich data model.

In the design domain, there has been a significant shift from document-centric approaches to data-centric design, e.g., model-based system engineering (MBSE) and Knowledge-Based Engineering (KBE) techniques, [7]–[9]. These developments emphasize the significance of semantic data representation in facilitating tool integration, enhancing collaboration. By embracing a data-centric approach and leveraging semantic data representation, software tools and systems can have better interoperability, leading to enhanced automation and decision-making processes.

Ontology, as a powerful tool, can serve as a repository for storing and organizing semantic data [10]. It provides a formal and standardized way to define concepts, relationships, and properties within a specific domain, enabling the harmonization of different data sources. Ontology elements often include classes, object properties, data properties, individuals and axioms. By defining common concepts, relationships, and properties in problem domains, ontology enhances the interoperability among diverse data sets, facilitating seamless integration and exchange of information between various systems and domains, [11]–[13].

With its ability to capture the semantics of data and establish meaningful relationships, ontology serves as a bridge between different data sources. It enables the mapping and alignment of disparate data models and vocabularies, allowing for a unified understanding and interpretation of information. By harmonizing and integrating diverse data sources through ontology, organizations can achieve improved interoperability, efficient data sharing, and better decision-making processes across different domains, [11]–[13]. There are some studies, [14], [15], using ontology to represent standardised data models to improve data interoperability, but few studies showcase how to use semantic data and ontology to form a toolchain for a product design task.

B. KBE

Literally, KBE is the implementation of KBS in the engineering domain to reduce time and costs of product development. The term “knowledge” refers to rules, hence this name is highlighting that the KB approach focuses on the reuse of engineering rules (knowledge) by knowledge management techniques, e.g., capture, formalization, representation and integration, see [2] and [16].

Some papers, for example [2] and [3], consider KBE as a narrow concept, namely the merger of AI and CAD tech-

nology. However, as the CAD model built in KBE style can support multi-disciplinary optimization (MDO) via providing the semantic data, if consider any repeatable processes as rules, e.g., a simulation to conduct performance analysis, KBE can be seen as any type of automation of repetitive tasks in product development based on the reuse of knowledge [4].

KBE style design is utilized in various engineering design tasks to construct models in a semantic manner. Ref. [17] built a parametric model for the KBE application that can estimate hypersonic vehicle weight using a physics-informed neural network. Ref. [18] proposed a framework of semantic hyper-graph-based knowledge representation to support the knowledge sharing for the product development. Ref. [19] proposed a KBE architecture definition method to build the using semantic model of user-customised designs of manufacturing systems.

The proposed approach in this paper takes advantage of KBE's ability to provide semantic data, formalises and stores the data used in engineering design tasks in a unified KB. Thus all the other applications can retrieve the needed data through a unified method, which improves the interoperability of the solution.

C. Automatic pipe routing

Pipe routing design is a crucial process in various industrial fields, including factory layout, aeroengine, shipbuilding, and large-scale integrated circuits, and has been extensively studied in these domains since the 1970s, see for example [20], [21]. There are generally two types of workflows for pipe routing: manual design workflow and ad-hoc automation design workflow.

The manual design refers to the design process that engineers interact with the GUI environment of commercial CAD software to generate the pipe routes manually. Although CAD systems are widely used in modern engineering design, in general, these commercial software only provide interactive-operation environments in the piping modules [22], not performing fully automatic pipe routing without the GUI environment as intended, e.g. Solidworks, ASD, Alias, and AVEVA [23].

The ad-hoc automation design refers to the process in which engineers utilize self-made ad-hoc pipe routers to automatically generate pipe routes. These self-made ad-hoc pipe routers often employ maze algorithms or A* algorithms to generate feasible paths, while heuristic algorithms like genetic algorithms are used as optimizers to provide optimized paths. Although these ad-hoc tools, [20], [22], [24], can complete specific pipe routing tasks, they do not consider interoperability with other software tools. For instance, the pipes generated by these ad-hoc pipe routers typically consist of a path represented as a list of elbow coordinates, rather than being represented in a semantic format that can be imported into CAD software. Moreover, obtaining input data related to equipment and existing pipes often involves parsing non-semantic data formats such as STEP files, which can also be challenging compared to working with semantic data.

III. PROPOSED APPROACH

A. The Architecture

The architecture of the proposed interoperable solution is illustrated in Fig. 2. In the proposed architecture, an intermediate layer is introduced, which includes a SPARQL query and a case-specific adaptor. This layer facilitates the harmonization of diverse data formats. By leveraging SPARQL queries, developers can retrieve relevant semantic data from the ontology-based knowledge base. Subsequently, they can develop adaptors to adjust the retrieved semantic data to meet the specific requirements of ad-hoc applications. Notably, developing case-specific adaptors is a more straightforward task compared to creating parsers for non-semantic data formats. Thus, the seamless interaction between the ad-hoc applications and software tools can be enabled with the help of the intermediate layer. A workflow can be composed based on the interaction between the tools.

Among the applications, there is a special one for the semantic data CRUD, which is responsible for the initialization of the KB. This application allows for the conversion of CAD model data into a semantic format, which can then be uploaded to the KB. This enables efficient knowledge sharing, retrieval, and reuse for other various engineering processes.

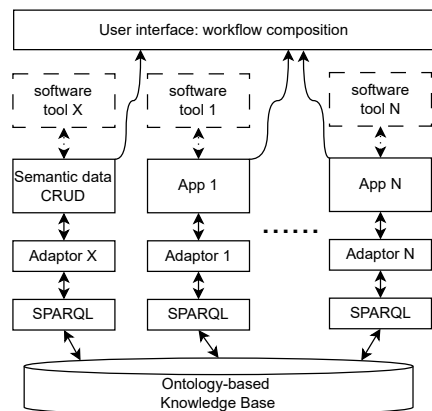


Fig. 2. The proposed architecture for automating engineering tasks powered by semantic data.

B. Semantic Modeling of Data and Processes

The geometric model and associated processes entail to be defined in semantic format. Some KBE development frameworks can be utilized for the semantic modeling, e.g., MOKA [25], CommonKADS [26].

The process typically begins with the parametric modeling of geometric objects. In cases where there is no readily available geometric model in CAD software, mechanical engineers have the option to represent the model in an informal format, such as MOKA ICARE format. Software developers then

define the ontology for the semantic representation of models, often using formats like OWL.

On the other hand, if the geometric model already exists in a CAD software, the process of defining the ontology becomes simpler. Developers can determine the necessary ontology elements (like classes, object properties, data properties) by examining the attributes of objects within the CAD software.

After the parametric modeling of geometric objects, the subsequent step involves modeling the relevant processes. The metadata, which includes the input, output, and description of the process, can be represented based on the parametric model and stored in the OWL format. This structured representation of process metadata establishes a foundation for future process discovery and management.

IV. CASE STUDY: KBE PIPE ROUTER

A. Case Description

A pipe routing case is shown to illustrate the proposed approach for building a semantic data-centric KBE solution in section III. The site used for pipe routing is from a public demonstration project (“projAPS”) of AVEVA software. AVEVA software is widely used in “Oil, Gas and Energy” industry for pipe routing design. Fig. 3 shows the site containing the equipment, existing pipes and the end pairs to be connected. In the following section, a pipe router is developed to route pipes connecting the end pairs (triangle to triangle, circle to circle). This pipe router retrieves needed data from KB via SPARQL query and an adaptor, generates the semantic data for routed pipes and then writes it to KB via SPARQL for other software tools to use. Moreover, to demonstrate the interoperability of the proposed semantic data-centric KBE solution, interactions with different software tools such as AVEVA, Siemens NX, and web browser are showcased.

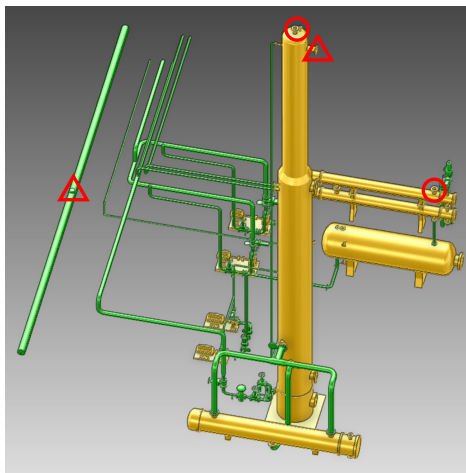


Fig. 3. Case statement: to generate 2 pipes to connect the end pairs (triangle to triangle, circle to circle).

B. The Architecture

The architecture of the semantic data-centric KBE pipe router is illustrated in Fig. 4. The semantic data CRUD module extracts and converts the model data in AVEVA into semantic format, which serves as the primary data source for other modules within the architecture. The boundary box finding module processes the semantic geometric data obtained from AVEVA, and adds the diagonal coordinates of boundary box for each element in KB. The boundary box of each element is to represent the element within the discrete space utilized by the pipe routing algorithm. Then the pipe router module can be developed to generate paths connecting end pairs and convert paths to pipes represented in semantic format. The webviewer generator module and the DFA generator module are to simulate how the KBE pipe router interacts with the existing ad-hoc applications. Based on the modules provided by developers, users can composite the workflow to form a toolchain for automating engineering design tasks. Overall, the architecture presents a comprehensive and integrated solution for pipe routing, showcasing the benefits of using semantic data in enhancing interoperability and expanding the capabilities of the system.

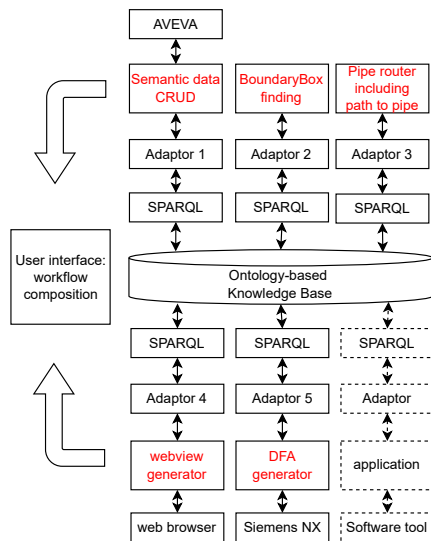


Fig. 4. The architecture of KBE pipe router including the five modules.

C. The Semantic Data

As the geometric model already exists in AVEVA software in this case, developers can determine the necessary ontology elements by examining the attributes of objects within AVEVA. Fig. 5 shows three AVEVA element instances (equipment, box and cylinder) represented in semantic format defined in ontology. Similarly, each AVEVA element type in

TABLE II has a corresponding ontology definition for its semantic representation.

TABLE II
AVEVA ELEMENTS TO CONVERT

AVEVA elements	Remarks
BOX, CYLI, NOZZ, PYRA, CONE, DISH, FLAN, GASK, ELBO, VERT, VALV, BRAN, NCYL, EXTR, RTOR, SLCY, SUBE, EQUI, PIPE, REDU, TEE, ATTA, BEND, WELD, ZONE, SITE, NBOX, POIN,	Box, Cylinder, Nozzle, Pyramid, Cone, Dish, Flange, Gasket, Elbow, Vertex, Valve, Branch, Nested Cylinder, Extrusion, Rotary Torus, Sliced Cylinder, Sub-equipment, Equipment, Pipe, Reducer, Tee, Attachment, Bend, Weld, Zone, Site, Nested Box, Point,

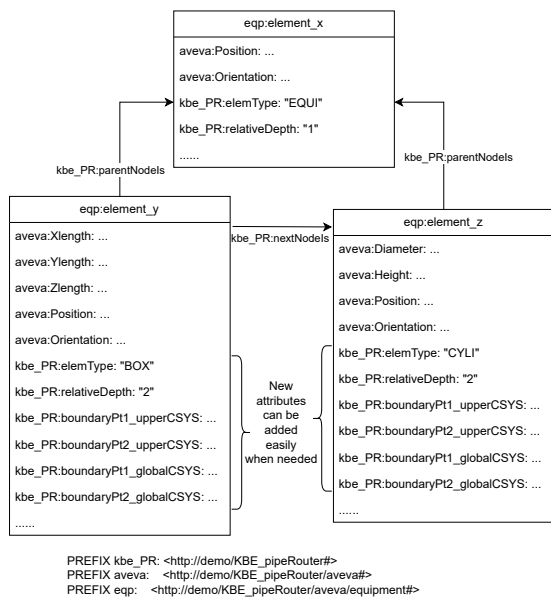


Fig. 5. Three AVEVA element instances (equipment, box and cylinder) represented in semantic format.

The pipe router employs A* algorithm as the path-finding algorithm to connect end pairs, which entails to discretize the space and so as to the equipment and pipe in the space. Thus the boundary box needs to be determined for discretizing every element in the model tree of the equipment and pipes. In this paper, the diagonal coordinates of boundary box for each element are defined and added as new attributes to the each element. See the 4 attributes related to “boundaryPt” in Fig. 5.

The DFA generator renders the equipment as cuboid and pipe as cylinder, so it needs the diagonal coordinates of boundary box for the equipment to generate the cuboids, and the elbow coordinates of each pipe to generate cylinders. The webviewer is in similar case. However, a difference is some

components of equipment will be rendered as cylinders, not only cuboids as the DFA generator and pipe router. This can be a time-consuming process for ad-hoc solutions, but luckily, the needed data about the cylinder components of the equipment are already available when the geometric model in AVEVA is converted to semantic format. See the semantic representation of cuboid and cylinder in Fig. 5.

D. The Introduction to Modules

This subsection introduces the functionality of the five modules (see them in Fig. 4) in the KBE pipe router and how they interoperate within the KBE solution. These modules include the semantic data CRUD module, the boundary finding module, the pipe routing module, as well as two visualization modules, namely the webviewer generator and the DFA generator.

The semantic data CRUD module takes advantage of the PML script language of AVEVA. The case-specific PML scripts are developed for each element type (see Table. II) in AVEVA, so that the relevant attributes of each element in the AVEVA model tree can be extracted and written into a plain text file named “modelTree.log”. Then it converts the raw data in “modelTree.log” to the semantic representation stored in a JSON file, which will be transformed to RDF triples for storage in an ontology-based KB. The selected attributes of the semantic data are carefully chosen to ensure they contain sufficient information for generating the visualization model within AVEVA, which has been depicted in subsection IV-C. This module also facilitates the manipulation of semantic data within the KB through the use of SPARQL. It offers an interface for operations such as uploading, downloading, deleting, and updating semantic data.

The boundary finding module is based on the semantic data generated by the semantic data CRUD module. It calculates the diagonal coordinates of boundary box for each element (see Table. II) in both local and global coordinate system. Thanks to the extendable semantic data format, these diagonal coordinates can be easily added as new attributes to each element for further processing within the KBE solution. Calculating the boundary box coordinates based on semantic data is often more straightforward compared to using the STEP format, which is primarily designed for machine processing. Semantic data representation provides a more human-friendly and manipulable format, allowing for a more intuitive and efficient calculation process.

The pipe routing module utilizes the A* algorithm to find paths connecting the end pairs, and the genetic algorithm (GA) for optimizing these paths. In the demonstration context, the fitness function of the GA is designed to prioritize shorter path lengths and minimize the number of bends. More details about the path generation can be found in Ref. [20], [22], [24]. Once the paths are determined, they are converted into pipes represented in a pre-defined semantic format. These pipes, along with their corresponding paths, are then uploaded to the KB for storage and retrieval. This semantic representation enables other modules within the KBE solution to understand

and interact with the pipes, ensuring interoperability and facilitating further utilization of the routed pipes.

The webviewer generator is to simulate how the KBE application interoperate with an existing ad-hoc tool. A web-based viewer is designed as a ad-hoc visualization tool. It uses cuboid and cylinder to represent equipment, and cylinder for pipes. All the necessary data for visualization can be retrieved from the knowledge base via SPARQL, making the adaptor development relatively straightforward.

The SPARQL query in Listing 1 is to retrieve all the elements meeting the following items:

- It is in equipment namespace, which means it belongs to equipment;
- It is in depth 2 of the model tree;
- It has the diagonal coordinates of boundary box. This is to ensure this is a valid element;
- It is not a cylinder;
- It does not have any cylinder as children elements.

The SPARQL query in Listing 2 is to retrieve all the cylinders meeting the following items:

- It is in equipment namespace, which means it belongs to equipment;
- It is in depth 2 or 3 of the model tree;
- It has the diagonal coordinates of boundary box. This is to ensure this is a valid element;
- It is a cylinder.

Through the above SPARQL queries, the cylinder components can be retrieved while other components can be represented as cuboids, meeting the webviewer's demand.

```

PREFIX kbe_PR: <http://demo/KBE_pipeRouter#>
PREFIX eqp: <http://demo/KBE_pipeRouter/aveva/equipment#>

SELECT ?subject ?predicate ?object
WHERE
{
  {
    ?subject ?predicate ?object.
    FILTER(strstarts(str(?subject), str(eqp))).
    ?subject kbe_PR:relativeDepth "2".
    ?subject kbe_PR:boundaryPt1_globalCSYS ?coord.
    FILTER NOT EXISTS {
      ?subject kbe_PR:elemType "CYLI".
    }
    FILTER NOT EXISTS {
      ?elem kbe_PR:elemType "CYLI".
      ?elem kbe_PR:parentNodeIs ?subject.
    }
  }
}

```

Listing 1. SPARQL to retrieve the equipment components to be rendered as cuboid.

```

PREFIX kbe_PR: <http://demo/KBE_pipeRouter#>
PREFIX eqp: <http://demo/KBE_pipeRouter/aveva/equipment#>

SELECT ?subject ?predicate ?object
WHERE
{
  ?subject ?predicate ?object.
  FILTER(strstarts(str(?subject), str(eqp))).
  ?subject kbe_PR:relativeDepth ?depth .
  FILTER(?depth IN ("2", "3"))
  ?subject kbe_PR:boundaryPt1_globalCSYS ?coord.
  ?subject kbe_PR:elemType "CYLI".
}

```

Listing 2. SPARQL to retrieve the equipment components to be rendered as cylinder.

These queries demonstrate the SPARQL queries for the semantic data retrieval are human-readable, which minimizes the workload for developers to prepare data for different tools. This is because the complex parsing task has been handled by the semantic data CRUD module and other relevant modules, allowing other modules within the KBE application to easily reuse the semantic data. After retrieving the needed data via the human-readable SPARQL query, an adaptor can be developed to read from returned data file (e.g., JSON file), which is more straightforward compared to parsing non-semantic data files.

However, in ad-hoc solutions, if certain data is not readily available, developers may need to write parsers to extract that data from raw non-semantic formats like STEP. For instance, in certain pipe routing scenarios mentioned in Ref. [20], [22], [24], all the equipment components are represented as cuboids, while the cylinders representing certain equipment components are not available. To enable the ad-hoc web-based viewer to function properly, developers would need to parse the STEP file and extract the relevant cylinder components for the equipment. This highlights the flexibility and adaptability of the KBE solution in accommodating different data sources and ensuring the interoperability of the ad-hoc tools. The DFA generator module is in a similar case.

E. The Result Demonstration

This subsection demonstrates the output results after running the KBE application. The first demonstration is to illustrate the interoperability of the KBE pipe router by visualizing the routed pipes in different tools. While the second demonstration is to show that the semantic format provides the user the freedom to manipulate the geometric data in a formal and semantic format, which is crucial for automatic workflow composition. See the two demonstration in Fig. 6 and Fig. 7.

It can be seen in Fig. 6 that the equipment and routed pipes are visualized in three software tools: AVEVA, web browser and Siemens NX. This demonstration highlights the interoperability of the KBE pipe router with various software tools with the help of semantic data. By using the semantic data representation, the KBE pipe router can effectively exchange information and interact with different software tools, facilitating a smooth workflow and enhancing collaboration among engineering teams using diverse software environments.

Fig. 7 illustrates the functionality of the pipe router in handling the addition of a new obstacle in the semantic representation. In Fig. 7 c), a code excerpt is shown, illustrating the semantic representation of the newly added obstacle. The semantic representation is designed to be human-readable, allowing users to easily understand and manipulate the data. By leveraging this semantic representation, users can generate different routing tasks and perform automatic routing for decision-making purposes. The flexibility of the semantic format enables users to adapt the routing process to accommodate changes in the environment or project requirements.

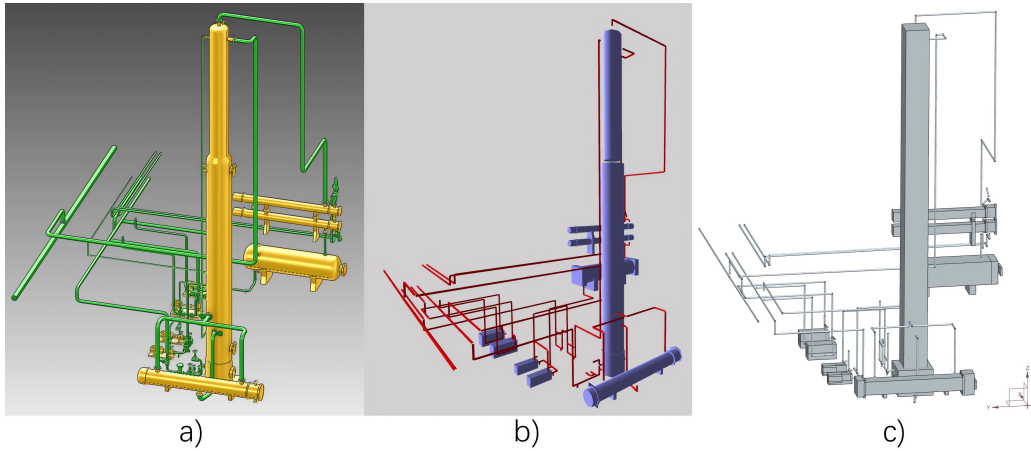


Fig. 6. The KBE pipe router interoperates with different tools. a) AVEVA. b) Web browser. c) Siemens NX.

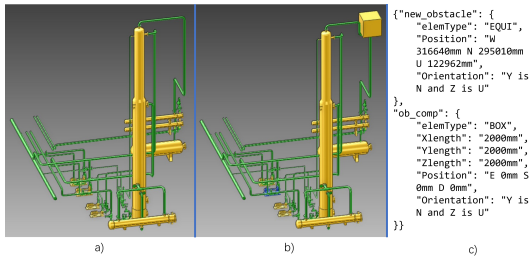


Fig. 7. The result demonstration before and after a new obstacle is added and the new obstacle in semantic representation. a) Before. b) After. c) New obstacle in semantic representation.

V. DISCUSSION

The proposed approach can help to build an interoperable solution for automating engineering design tasks, which is illustrated with a KBE pipe router case. Compared with the ad-hoc automation solution, the interoperability of the proposed KBE solution is enhanced. Considering the interaction with new software tools also imports new functions, thus the extendability often comes together. Therefore, it is reasonable to discuss both the interoperability and extendability of the proposed KBE solution.

International standard ISO/IEC/IEEE 24765:2017 [27] provides some definition of interoperability and extendability. In the standard, interoperability is defined as “3. the capability to communicate, execute programs, and transfer data among various functional units in a manner that requires the user to have little or no knowledge of the unique characteristics of those units.”, while extendability is defined as “1. the ease with which a system or component can be modified to increase its

storage or functional capacity.” Ref. [28] provides a qualitative metric to evaluate the interoperability and extendability of software based on the three dimensions of the system development methodology.

The qualitative comparison between the ad-hoc solution and proposed KBE solution is conducted based on the metrics proposed by Ref. [28], see it in TABLE III. It can be seen that the interoperability and extendability are improved by leveraging the semantic data representation. The models for the modules within the KBE solution are easier to interoperate with and extend thanks to the semantic format. And tools like SPARQL can facilitate the harmonization of different data formats. As a result, developers and users need to familiarize themselves with SPARQL as it becomes an essential technique in the proposed approach.

VI. CONCLUSIONS

This paper presents an approach for building interoperable solution to be able to support different engineering design tools. The approach is illustrated by the development of KBE pipe router. From this example, it can be seen that:

- 1) Semantic model and ontology representation can help to build interoperable solution to interact with different software tools.
- 2) The proposed KBE solution has better interoperability and extendability than the ad-hoc solutions.

In future work, it would be beneficial to explore the development of an application based on service-oriented architecture (SOA) to enhance extendability and facilitate process reuse. This can be achieved by developing low-coupling web APIs. Additionally, a low-code workflow engine could be implemented to provide users with a user-friendly platform for creating customized workflows. This engine would utilize pre-

TABLE III
COMPARISON BETWEEN KBE SOLUTION AND AD-HOC SOLUTION

	KBE solution (I for interoperability; E for extendability)	ad-hoc solution (I for interoperability; E for extendability)
Model	I: semantic data from different sources can be merged for new model as they share the same graph structure. E: add attributes is to add edges into graph data.	I: define new class to contain the children data classes. E: add new attributes into data class.
Tools	I: SPARQL and adaptor as the intermediate layer to harmonize different data formats. E: add new functions by interacting with new tools.	I: parsers are needed when data formats are different. E: add new function by writing new hard-coded functions that have to be invocable by previous programs.
Techniques	I: developers retrieve semantic data via SPARQL and then write a straightforward adaptor. E: users learn SPARQL for querying available knowledge and potentially starting to extend it. Developers learn SPARQL to retrieve and adapt semantic data for existing tools to provide new functions.	I: developers have to learn different data formats they encounter and write parsers to convert them. E: users manipulate isolated tools manually. Developers lack a unified framework to connect different tools.

defined semantic models and reusable processes to simplify the process of workflow composition.

REFERENCES

- [1] X. Wang, A. Liu, and S. Kara, "Machine learning for engineering design toward smart customization: A systematic review," *Journal of Manufacturing Systems*, vol. 65, pp. 391–405, 2022.
- [2] G. La Rocca, "Knowledge based engineering: Between ai and cad. review of a language based technology to support engineering design," *Advanced engineering informatics*, vol. 26, no. 2, pp. 159–179, 2012.
- [3] C. B. Chapman and M. Pinfold, "Design engineering—a need to rethink the solution using knowledge based engineering," *Knowledge-based systems*, vol. 12, no. 5–6, pp. 257–267, 1999.
- [4] W. J. Verhagen, P. Bermell-Garcia, R. E. Van Dijk, and R. Curran, "A critical review of knowledge-based engineering: An identification of research challenges," *Advanced Engineering Informatics*, vol. 26, no. 1, pp. 5–15, 2012.
- [5] J. Peckham and F. Maryanski, "Semantic data models," *ACM Computing Surveys (CSUR)*, vol. 20, no. 3, pp. 153–189, 1988.
- [6] "Opc unified architecture," <https://opcfoundation.org/about/opc-technologies/opc-ua/>, [Accessed 10-Jun-2023].
- [7] A. Akundi and V. Lopez, "A review on application of model based systems engineering to manufacturing and production engineering systems," *Procedia Computer Science*, vol. 185, pp. 101–108, 2021.
- [8] M.-H. Bleu-Laine, M. V. Bendarkar, J. Xie, S. I. Briceno, and D. N. Mavris, "A model-based system engineering approach to normal category airplane airworthiness certification," in *AIAA Aviation 2019 Forum*, 2019, p. 3344.
- [9] N. J. Lindsey, M. Alimardani, and L. D. Gallo, "Reliability analysis of complex nasa systems with model-based engineering," in *2020 Annual Reliability and Maintainability Symposium (RAMS)*. IEEE, 2020, pp. 1–8.
- [10] R. Studer, V. R. Benjamins, and D. Fensel, "Knowledge engineering: Principles and methods," *Data & knowledge engineering*, vol. 25, no. 1–2, pp. 161–197, 1998.
- [11] Y. Sermet and I. Demir, "Towards an information centric flood ontology for information management and communication," *Earth Science Informatics*, vol. 12, no. 4, pp. 541–551, 2019.
- [12] D. Orellana and W. Mandrick, "The ontology of systems engineering: towards a computational digital engineering semantic framework," *Procedia Computer Science*, vol. 153, pp. 268–276, 2019.
- [13] L. Yang, K. Cormican, and M. Yu, "Ontology learning for systems engineering body of knowledge," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 2, pp. 1039–1047, 2020.
- [14] P. Valluru, J. Karlapudi, K. Menzel, T. Mätäsnieni, and J. Shemeikka, "A semantic data model to represent building material data in aec collaborative workflows," in *Boosting Collaborative Networks 4.0: 21st IFIP WG 5.5 Working Conference on Virtual Enterprises, PRO-VE 2020, Valencia, Spain, November 23–25, 2020, Proceedings 21*. Springer, 2020, pp. 133–142.
- [15] J. Tutchter, "Development of semantic data models to support data interoperability in the rail industry," Ph.D. dissertation, University of Birmingham, 2016.
- [16] L. Zhang, A. Olsen, and A. Lobov, "An ontology-based kbe application for supply chain sustainability assessment," *Resources, Environment and Sustainability*, vol. 10, p. 100086, 2022.
- [17] D. Chen, Y. Li, J. Guo, and Y. Li, "Estimation of hypersonic vehicle weight using physics-informed neural network supported by knowledge based engineering," *Expert Systems with Applications*, vol. 195, p. 116609, 2022.
- [18] Z. Wu, J. Liao, W. Song, H. Mao, Z. Huang, X. Li, and H. Mao, "Semantic hyper-graph-based knowledge representation architecture for complex product development," *Computers in Industry*, vol. 100, pp. 43–56, 2018.
- [19] C. Zheng, Y. An, Z. Wang, X. Qin, B. Eynard, M. Bricogne, J. Le Duigou, and Y. Zhang, "Knowledge-based engineering approach for defining robotic manufacturing system architectures," *International Journal of Production Research*, pp. 1–19, 2022.
- [20] W. Niu, H. Sui, Y. Niu, K. Cai, and W. Gao, "Ship pipe routing design using nsga-ii and coevolutionary algorithm," *Mathematical Problems in Engineering*, vol. 2016, 2016.
- [21] X.-I. Qian, T. Ren, and C.-e. Wang, "A survey of pipe routing design," in *2008 Chinese Control and Decision Conference*. IEEE, 2008, pp. 3994–3998.
- [22] Z. Dong and X. Bian, "Ship pipe route design using improved a* algorithm and genetic algorithm," *IEEE Access*, vol. 8, pp. 153 273–153 296, 2020.
- [23] A. Asmara, "Pipe routing framework for detailed ship design," Ph.D. dissertation, TUD Technische Universiteit Delft, 2013, (Delft: VSSD)(ISBN 978-90-6592-326-3).
- [24] G. Belov, T. Czauderna, A. Dzaferovic, M. Garcia de la Banda, M. Wybrow, and M. Wallace, "An optimization model for 3d pipe routing with flexibility constraints," in *Principles and Practice of Constraint Programming: 23rd International Conference, CP 2017, Melbourne, VIC, Australia, August 28–September 1, 2017, Proceedings 23*. Springer, 2017, pp. 321–337.
- [25] CORDIS, "Methodology and software tools oriented to kbe applications," 1999, [Online; accessed Jan-2023]. [Online]. Available: <https://cordis.europa.eu/project/id/25418>
- [26] A. T. Schreiber, G. Schreiber, H. Akkermans, A. Anjewierden, N. Shadbolt, R. de Hoog, W. Van de Velde, and B. Wielinga, *Knowledge engineering and management: the CommonKADS methodology*. MIT press, 2000.
- [27] "Iso/iec/ieee international standard - systems and software engineering–vocabulary," *ISO/IEC/IEEE 24765:2017(E)*, pp. 1–541, 2017.
- [28] A. Lobov, "On extendibility and interoperability properties in knowledge-based engineering," in *AIP Conference Proceedings*, vol. 2605, no. 1. AIP Publishing, 2023.

Paper E

A Low-code KBE Solution for Engineering Design: a Pipe Routing Case Demonstration

Zhang, Liang, and Andrei Lobov. "A Low-code KBE Solution for Engineering Design: a Pipe Routing Case Demonstration.", 2024, Under review

This paper is awaiting publication and is not included in NTNU Open

Paper F

Semantic Web Rule Language-based approach for implementing Knowledge-Based Engineering systems

Zhang, Liang, and Andrei Lobov. "Semantic Web Rule Language-based approach for implementing Knowledge-Based Engineering systems.", *Advanced Engineering Informatics* (2024), Accepted.

This paper is published in *Advanced Engineering Informatics* 2024 ;Volum 62. <https://doi.org/10.1016/j.aei.2024.102587> This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

Semantic Web Rule Language-based approach for implementing Knowledge-Based Engineering systems

Liang Zhang^{a,*}, Andrei Lobov^a

^aNorwegian University of Science and Technology, Richard Birkelands Vei 2B, Trondheim, 7034, Norway

ARTICLE INFO

Keywords:

Product design
Knowledge-Based Engineering
black-box
Computer-Aided Design
Ontology
Semantic Web Rule Language

ABSTRACT

The culture of product design is shifting from case-by-case development to the Knowledge-Based Engineering (KBE) paradigm facilitating knowledge sharing and reusing among different stages and groups. The Semantic Web stack including Web Ontology Language (OWL) and Semantic Web Rule Language (SWRL) offers promising formats to represent data and rules for engineering knowledge sharing and reuse. However, many KBE applications for product design treat ontology-based knowledge bases as graph databases, often neglecting the reasoning abilities provided by the Semantic Web stack. Consequently, design rules, especially those concerning the (re)construction of geometric models, are frequently encapsulated as black-box processes within KBE systems. This type of reuse tends to result in non-cohesive solutions, where fragments of relevant knowledge, especially about the (re)construction of geometric models, are dispersed across various locations. This article demonstrates an approach to realizing the automated product design facilitated by semantically representing engineering knowledge using OWL and SWRL. This approach enables the construction of a cohesive knowledge base, leveraging the reasoning capabilities provided by the Semantic Web stack. Notably, the (re)construction of geometric models can be achieved using KBE language code snippets and the string processing capabilities of SWRL. To demonstrate this approach, a shaft design case, frequently used in research on product design, serves as a demonstrator to provide conceptual proof. The resulting geometric models are generated in KBE languages compatible with Siemens NX and AVEVA design software and can be visualized through interaction with the Computer-aided Design (CAD) kernel. This showcases the potential for seamless integration and knowledge sharing in the realm of product design through the application of the Semantic Web stack and KBE.

1. Introduction

Product design is a knowledge-intensive engineering task, particularly in light of the trend towards Multi-disciplinary Design and Optimization (MDO). The reusability of engineering knowledge is a critical concern in product design, encompassing both product data and design processes [1]. Knowledge-Based Engineering (KBE) serves as a valuable design paradigm for automatically generating product variants by leveraging explicitly represented knowledge [2]. In particular, KBE applications excel at rapidly generating Computer-Aided Design (CAD) models, representing a core outcome in the product development process [3]. Therefore, the culture of product design is shifting from case-by-case development to the KBE paradigm facilitating knowledge sharing and reusing among different stages and groups.

The representation of engineering knowledge is a pivotal aspect in the development of KBE applications. The Semantic Web has been recognized for its potential in effectively representing engineering knowledge, providing a common framework that facilitates data sharing and reuse across diverse applications, enterprises, and communities. This framework, often referred to as the Semantic Web stack, incorporates essential technologies like Web Ontology Language (OWL), Semantic Web Rule Language (SWRL), and SPARQL Protocol and RDF Query Language (SPARQL).

Several ontology-based KBE applications for product design have been developed [4, 5, 6, 7, 8], successfully harnessing ontology to enhance data interoperability and promoting the design efficiency. However, many of these applications treat the knowledge base (KB) as a graph database, without fully exploiting the reasoning capabilities provided by the Semantic Web. In most cases, design processes are executed outside the KB as black-box procedures, particularly in the (re)construction of geometric models, which is still carried out by black-box modules invoking the Application Programming Interface (API) of CAD software or using code templates of CAD models. This form of black-box reuse has long been recognized as a challenge impeding the reusability and broader application of KBE approaches [9]. A key issue contributing to this challenge is the lack of support for modification and insufficient portability, which can hinder easy reuse in practice. Unfortunately, this black-box reuse manner has remained largely unaddressed to date [10].

The KBE languages, such as Siemens NX Knowledge Fusion (KF), offer a mean to codify CAD models in a textual representation. Simultaneously, SWRL built-in operators can support string processing. This inspiration gives rise to the idea that the (re)construction of geometric models can be accomplished within the KB by embedding KBE code snippets and automatically determining variable values through reasoning based on predefined rules, such as geometric constraints. Consequently, not only can the semantic product data be reused by other applications, but also the process of CAD model (re)construction. This could be an advantage when sharing and reusing knowledge across different design

*Corresponding author


 liang.zhang@ntnu.no (L. Zhang); andrei.lobov@ntnu.no (A. Lobov)
ORCID(s): 0000-0003-4201-8535 (L. Zhang); 0000-0003-2729-489X (A. Lobov)

Table 1

List of abbreviation used in the paper

Abbreviation	Explanation
AI	Artificial Intelligence
API	Application Programming Interface
CAA	Computer-Aided Analysis
CAD	Computer-Aided Design
KB	Knowledge-Based, Knowledge Base
KBS	Knowledge-Based System
KBE	Knowledge-Based Engineering
MDO	Multi-disciplinary Design and Optimization
OO(P)	Object-Oriented (Programming)
OWL	Web Ontology Language
PML	Programmable Macro Language (for AVEVA)
RDF	Resource Description Framework
SPARQL	SPARQL Protocol and RDF Query Language
STEP	Standard for the Exchange of Product Data
SWRL	Semantic Web Rule Language

teams, as this cohesive form provides better maintainability and portability.

Based on the proposed idea, this paper addresses the following Research Question (RQ): **How to utilize the Semantic Web stack and KBE languages to construct a KBE-based design solution capable of automatically (re)constructing CAD models considering its reusability?** The contributions of this paper are as follows: (1) Classifying the design rules into two types based on their universality: universal rules and case-specific rules. Universal rules, which are assumed to have simple forms, can be represented at the semantic layer alongside product data, leveraging the reasoning capabilities offered by the Semantic Web stack. Meanwhile, case-specific design processes are implemented as black-box procedural rules, operating at the user application layer for complex tasks. (2) Utilizing textual KBE languages and the string processing capabilities of SWRL to (re)construct the geometric model. This approach encapsulates the (re)construction of geometric models in the semantic model, forming a high-cohesive knowledge base. All of these contributions aim to facilitate easy knowledge sharing and reuse, along with robust interoperability. These features bring convenience in future extending the functionality of KBE applications, which is important for supporting knowledge-intensive tasks in engineering.

The remainder of this paper is organized as follows. Section 2 introduces the theoretical background and the related works. Section 3 explains the proposed approach and section 4 gives an implementation example. Then a comparison between the proposed solution and other solutions is discussed in section 5. And finally, section 6 concludes the paper. The abbreviations used in the paper are explained in Table 1.

2. Background

2.1. Knowledge-Based Engineering

KBE is the evolution of Knowledge-Based System (KBS) towards the specific needs of the engineering domain to reduce the time and costs of product development [1]. KBS is the Artificial Intelligence (AI) system based on a general-purpose search mechanism trying to string together elementary reasoning steps to find complete solutions [11]. KBE takes advantage of the knowledge representation and reasoning competence of KBS, enhanced with geometry manipulation ability by cooperating with a CAD kernel. Additionally, the parametric modeling approach required by the declarative representation facilitates the data exchange with Computer-Aided Analysis (CAA) tools, which in turn enhances its data processing and computation ability. The key feature of KBE is the knowledge representation language to capture designers' intent so that the reasoner can automatically infer new facts representing the expected design results, especially the geometric model of the product. Thus, KBE systems can rapidly generate variant products when there are new designers' intent. This feature holds significant importance in MDO, where the toolchain often necessitates geometric models as input to carry out the optimization process.

KBE is a broad concept with varying focuses depending on the context. When referring to a "KBE system," the emphasis is on the automation of repetitive tasks in product development through knowledge reuse, rather than traditional ad-hoc programs designed for specific tasks [12]. When it comes to the term "KBE language", it typically denotes a modeling language primarily used for representing geometric models, which play a very important role in a design process. These languages are often Object-Oriented and declarative, facilitating the parametric modeling of product geometry. Moreover, geometric models represented in KBE languages can be easily converted to other neutral formats, such as the STEP (Standard for the Exchange of Product Data) format, within corresponding CAD tools. Compared to the STEP format, KBE languages offer the advantage of explicitly capturing the designers' intent, making them highly beneficial in knowledge-intensive engineering tasks. Some typical KBE languages include Siemens NX Knowledge Fusion. Additionally, certain CAD software provides script languages, such as AVEVA PML (Programmable Macro Language), which enable the automatic generation of models by interacting with the CAD kernel. Although the classification of these script languages as KBE languages may be debatable, they nonetheless provide users with geometry manipulation capabilities. These two languages are selected to represent geometric models in the case study for demonstration purposes.

Although KBE approaches have demonstrated success in rapidly generating product variants, several challenges hinder their broader application. Verhagen et al. [9] identified five key shortcomings, including: (1) case-based, ad-hoc development of KBE applications; (2) black-box reuse of

design knowledge; (3) limited knowledge re-use; (4) absence of standardized KBE success metrics; and (5) lack of an assessment framework for KBE opportunities. And Kügler et al. [10] concluded these issues remain largely unaddressed so far. Among these shortcomings, the black-box reuse manner seems an issue towards reusability and application of KBE approaches [10].

The black-box reuse manner could arise due to the limited inference capability of a reasoner, constrained by the expressiveness of the knowledge representation language. As each knowledge language is designed for a specific domain with predefined scenarios, when some newly-added processes are difficult to represent in the given knowledge representation languages, it is practical to package them as black-box to work for the reasoner in a nontransparent manner, which are also known as procedural rules or attachments [13]. This is commonly encountered when it comes to the functionality extension of an existing KBE system. A typical example could be the complex math calculations. Although in principle it is possible to codify the related math knowledge into the knowledge base, it is improper in practice. As in most cases, the users of the KBE system just want the calculation to be done by a black-box as efficiently as possible, without caring about how the result is achieved through inference. Similarly, tasks involving the invocation of external software tools are often suited to black-box processes created using programming languages. While packaging complex rules as black boxes may seem efficient for developers, it can limit knowledge reuse, as other users may struggle to comprehend and adapt these rules for their specific needs [10]. Moreover, invoking these black-box processes across different systems can pose compatibility challenges. Consequently, the question arises: when should black-box implementation be used, and when should knowledge languages be employed for implementation?

Many KBE applications in product design treat ontology-based KBs as graph databases, often neglecting the reasoning abilities provided by the Semantic Web stack. Specifically, the construction of geometric models is encapsulated as a black-box process in many cases. For instance, Gupta et al. [6] proposed a unified taxonomy for representing shape features, but the reconstruction of geometric models relies on invoking Visual Basic (VB) APIs provided by CATIA CAD software, creating a black-box procedure for other users. Similarly, Ortner-Pichler et al. [14] developed a web-based KBE application utilizing parametric modeling, but the construction of geometric models is encapsulated as a black-box process using VB APIs provided by CAD software. Likewise, Mandorli et al. [15] employed ontologies to represent geometric features and construct geometric models within the Grasshopper CAD environment, yet the reuse of this procedure is dependent on the specific environment. Tran et al. [4] instantiated new models using Siemens NX KF code templates, but data retrieval and parsing were performed by Python code, presenting another black-box process for users. Other works [5, 7, 8, 16] employed ontology to represent features extracted from CAD models

for various purposes, without mentioning SWRL rules or the reconstruction of geometric models. On the other hand, Li et al. [17] and Das et al. [18] utilized SWRL to represent logic-based rules on semantic data extracted from CAD models, but the reconstruction of CAD models was not addressed. Consequently, this paper proposes an approach to leverage OWL and SWRL to represent design knowledge including (re)constructing CAD models in KBE languages, aiming to enhance reusability and interoperability.

2.2. Reusability

Reusability is always a pursuit in software development, and KBE application is even born for this. Several factors influence the reusability of a software module, as highlighted in a literature review [19]. The top five factors identified include coupling, cohesion, complexity, inheritance, and size. Meanwhile, the author also pointed out that other factors listed in ISO 25010 System and software quality models [20] can affect reusability, such as maintainability, portability, performance efficiency, functional suitability, compatibility, reliability, and usability. The FAIR (Findability, Accessibility, Interoperability, and Reusability) principle [21] is widely adopted to enhance data reuse, and its application has been extended to improve the reuse of process descriptions and associated implementations [22].

The semantic description of CAD models has been explored for reusability, but there is limited research on the implementation of processes to construct CAD models, as discussed in Section 2.1. Ref. [22] provides a summary of three ways to facilitate the instantiation of implementation for process reuse:

1. Abstract programming language. Using domain-specific languages to represent the code generation workflow. KBE languages for CAD model construction can be considered as a form of this type of reuse.
2. Embedded source code. Embedding code snippets in manageable forms, such as the description of the implementation. The proposed method in this paper aligns with this approach.
3. Abstract functions. Decoupling the function implementation from the description, and encapsulating the implementation as a black box. Examples include web services, APIs, and most CAD construction processes discussed in Section 2.1.

2.3. Semantic Web Stack

Semantic data refers to data enriched with explicit semantic content, surpassing the scope of relational data by including additional information describing the semantics of entities and relationships, encompassing connections and consistency constraints [23]. With the aim to make the content of World Wide Web more machine-understandable by adding semantic metadata, Tim Berners-Lee coined the term "Semantic Web" in 1999 [24]. The World Wide Web Consortium (W3C) describes Semantic Web as "*The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and*

community boundaries". Consequently, the Semantic Web is often referred to as "Linked Data" or "Web of Data", aiming to transform the existing "web of documents" into a "web of data", which is a global database with machine-interpretable metadata that interconnects related information. This is also aligned with the philosophy of KBE, which emphasizes facilitating knowledge reuse through the capture and formalization of knowledge. The well-known Semantic Web stack was introduced to standardize data and explicitly establish relationships among data elements. This Semantic Web architecture incorporates underlying technologies such as ontology, SWRL (Semantic Web Rule Language), and SPARQL (SPARQL Protocol and RDF Query Language) for semantic data processing, as illustrated in Figure 1. This layered architecture provides guidelines for different implementation manners of reusable design knowledge.

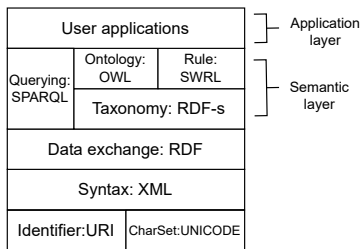


Figure 1: A simple version of the Semantic Web Stack.

The utility of the Semantic Web in improving knowledge sharing and reuse within KBE implementation has been acknowledged [9, 10]. Many studies have demonstrated how OWL enhances the reuse of product data, as discussed in Section 2.1. However, there is a limited number of studies exploring how SWRL can facilitate the reuse of design processes. Most KBE applications utilize general-purpose programming languages for all involved processes, often neglecting the reasoning abilities provided by SWRL. One reason for this might be the perceived limitations in the expressiveness of SWRL, which is constrained to logic-based rules [7]. One might wonder: what rules are not suitable for representation in SWRL? A tentative answer could include rules that involve complex mathematical calculations, large-scale data processing, processes with external tool dependencies, etc. However, SWRL can prove helpful in representing relatively simple rules [17], such as rules for derived properties in a class definition. The utility of SWRL is further demonstrated in the following sections.

3. Proposed Approach

3.1. The Architecture

The architecture of the proposed solution, integrating the Semantic Web stack and KBE, is illustrated in Figure 2. OWL and SWRL serve as the carriers of design knowledge,

encompassing class definitions of products, their relationships, and associated design rules. The design knowledge represented in this form is object-oriented (OO), enabling the integration of existing KBE languages, such as Siemens NX KF, to rapidly generate the geometric model of the product. Moreover, these two formats, based on RDF-s, can be stored together in the KB. This enables a highly cohesive form of knowledge representation, greatly enhancing easy knowledge sharing and reuse. The knowledge model including the reasoning process operates at the semantic layer of the Semantic Web stack. While the user applications, including procedural rules and user interface, are based on the knowledge model and operate at the application layer.

In a design case, some knowledge is universal and serves as global resources, such as the dimensions of standard components. It is recommended to formalize this type of knowledge as global ontologies and rules stored in a public KB. On the other hand, some knowledge is domain-specific and is intended for specific design cases within local design groups. This type of knowledge can be formalized as local ontologies and rules and stored in a local KB. Developers initially create the local ontologies using an ontology editor. The integration with KBE languages for geometric modeling is recommended to be done at this stage. Throughout the development process, global knowledge can be imported and reused. Subsequently, these ontologies, carrying the design knowledge, are uploaded to the local KB for subsequent utilization.

SPARQL serves as the interface for users to interact with the KB through CRUD (create/read/update/delete) operations. Functioning as a unified interface, SPARQL facilitates data sharing between different upper-level applications, promoting interoperability. Within this architecture, procedural rules are built to cooperate with SPARQL. These rules, written in general-purpose programming languages, are typically case-specific and too complex to be represented as SWRL rules. For example, some procedural rules can generate new assertions represented in SPARQL code through complex calculations, while others retrieve product data from the KB and interact with external tools to generate deliverables, such as CAD models. The user can manipulate these procedural rules through a user interface. As above-mentioned, the user interface and procedural rules together operate at the application layer of the Semantic Web stack.

Notably, there are two distinct roles in the architecture: developers and users. However, these roles do not need to be filled by different individuals. Design engineers or designers, possessing domain knowledge and the ability to use OWL and SWRL, can act as both developers and users. They can develop local ontologies when none are available, and if procedural rules are required, they may collaborate with programmers in their development. This is often the case in the initial stages of product design, where designers play a dual role as developers. Upon completing the development of the KBE application, designers transition to the role of users, utilizing the knowledge representation languages to define new products.

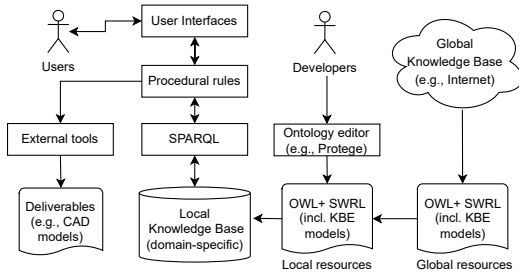


Figure 2: The proposed architecture based on KBE and the Semantic Web stack.

3.2. Considerations for Reusability

In the proposed architecture, some features are designed to enhance reusability. Although some knowledge is modeled within some KBE applications for reusability, it is actually not always reused when developing other related KBE applications [9]. The reasons could be (1) the knowledge models in the ad-hoc KBE applications are case-based; (2) the sourcing and reusing knowledge is not effectively supported. To address this, the resources of product knowledge are classified into local resources and global resources. It may be overly ambitious to expect all the knowledge models to be globally accessible and reusable, thus only the universally applicable knowledge is put in global resources, such as standard component dimensions and associated rules. The resources of these components are stored in a public location for easy access among different groups or communities.

Similarly, although SWRL can offer fine portability and reusability, there are still scenarios where black-box reuse is applicable. According to La Rocca [1], design rules can be categorized into five different types: (1) Logic rules (or conditional expressions); (2) Math rules; (3) Geometry manipulation rules; (4) Configuration selection rules (or topology rules); and (5) Communication rules with other programs. However, this classification does not provide guidance on the selection of implementation using a black-box approach and SWRL. Therefore, this paper broadly categorizes design rules into two types: universal rules and case-specific rules. By implementing universal rules in the semantic layer and case-specific rules in the application layer, a balance between reusability and development efficiency can be achieved.

Universal rules are those that apply to classes in all cases, regardless of specific instances. This paper assumes they typically have simple forms. For example, if a shaft section is paired with a bearing, then the design diameter of this section must equal the bore diameter of the bearing, regardless of the specific type of bearing. Such rules are universally applicable in all cases where a shaft section is involved, thus they can be packaged within the class definition of the shaft section, as demonstrated in the case study section. These rules primarily involve reasoning or simple arithmetic calculations, utilized to derive values of inferred attributes

and relations. Consequently, they can be effectively represented using description logic-based SWRL. Storing SWRL rules alongside assertions in OWL format facilitates a high-cohesive form of knowledge representation, greatly enhancing the ease of sharing and reuse.

In contrast, case-specific rules are those that apply only to a particular case or subset of cases, typically having more intricate forms. For instance, a design process to select a bearing can be codified as a reusable rule given its operational conditions. However, this type of rule only applies in the designers' own cases, as it is usually tailored to their specific case conditions and designers' preferences. If other designers wish to reuse it, they typically have to adapt it according to their own preferences, such as preferred bearing series, diameter range, and so on. As there are more factors to consider for specific cases, these rules often involve complex mathematical calculations, large-scale data processing, or processes with external tool dependencies. They are normally used to generate assertions as input to KB. Consequently, these rules are often implemented as black-box procedural rules, as encapsulating them into SWRL is considered uneconomical or challenging. Typically, it is sufficient to reuse the knowledge encapsulated in these case-specific rules in their black-box forms, as these rules do not universally apply to all cases.

3.3. Semantic Modeling of Data and Rules

The process of building the semantic model is depicted in Figure 3, representing a generic method for building ontologies. The core ideas of the proposed approach encompass (1) utilizing the Semantic Web stack to build the knowledge model of product design, including product data and universal design rules; and (2) codifying the geometric models in KBE languages and constructing CAD models through the string processing capabilities offered by SWRL.

The main components carrying design knowledge in the figure are TBox (Terminology Box), ABox (Assertion Box), SWRL rules, and procedural rules. The representation of product data is achieved through ontologies comprising TBox and ABox, while the implementation of design rules can take the form of SWRL or procedural rules, depending on whether they are universal or case-specific. Ontologies, SWRL rules, and SPARQL operate at the semantic layer, while procedural rules are encapsulated in the user application layer.

The TBox includes the definition of classes, attributes, and relations. Based on the concepts in the TBox, users can assert facts by instantiating these classes, thereby forming the ABox. Assertions can be generated manually or through the execution of procedural rules. When enough facts are asserted, the reasoner can execute the predefined inference rules to infer new facts, representing the expected design outcomes in a KBE application, such as the geometric models in KBE languages.

The initial step involves defining the TBox for product components. In product design, components are modeled in an object-oriented (OO) manner, utilizing classes with

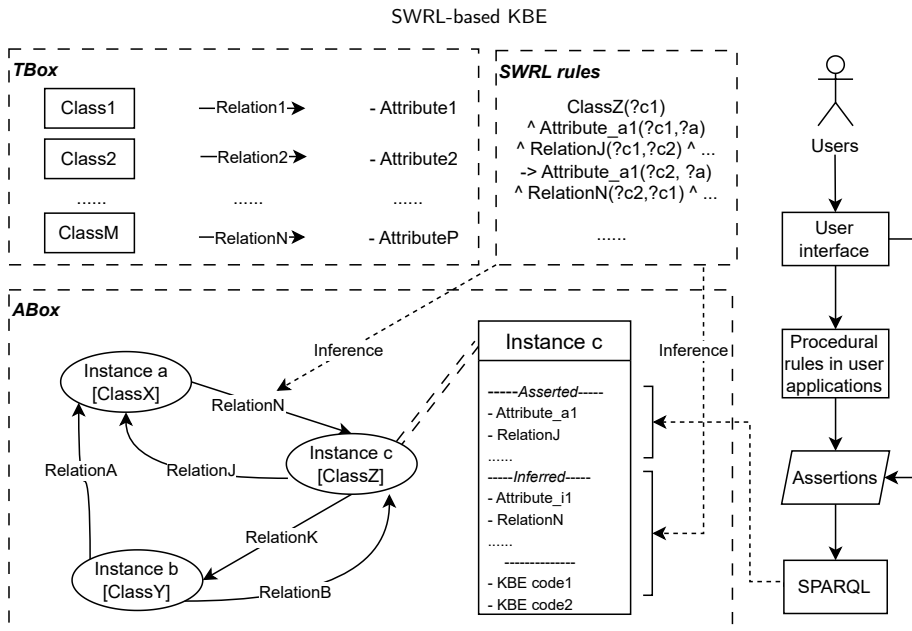


Figure 3: Schematic diagram of the semantic modeling approach.

associated attributes (referred to as data properties in OWL). Alongside attributes, there are relations between the defined classes, also termed object properties in OWL, primarily describing assembly relationships. These properties can be categorized into asserted properties, originating from user input, and inferred properties, derived from other assertions through the execution of predefined rules. Drawing an analogy with class definitions in the context of Object-Oriented Programming (OOP), asserted properties are equivalent to input properties, and inferred properties are comparable to derived or computed properties. For example, assembling relations can be asserted to represent user intent, while the KBE code representing geometric models is derived through inference by reasoners.

After defining the TBox for product representation, developers can start to build ABox. At this stage, both the asserted and inferred properties can be manually input by developers for debugging purposes. After the debugging of procedural rules and SWRL rules, some assertions can be replaced through the execution of procedural rules, and the inferred properties can be derived by the reasoner through SWRL rules.

The manner to codify design rules depends on the universality of the rules. The universal rules apply to all the instances of a class, thus it can be implemented in SWRL rules for inferring derived properties. The construction of CAD models using KBE languages belongs to this type. Conversely, the case-specific rules are not universally applicable. Encapsulating case-specific rules into the complex of

OWL and SWRL would lower their universality. Additionally, these case-specific rules often involve the preconditions describing the specific cases, making it inconvenient to implement using SWRL. Thus, it may be more economical to implement them in programming languages and encapsulate them as black-box procedural rules.

It is noteworthy that the development of the ABox and rules are iterative, which means it is not necessary to develop one after the completion of another. It is common for the debugging of SWRL rules to be based on asserted instances in the ABox, for example, the rules for the construction of geometric models using KBE code. In some cases, even the TBox can be extended for debugging SWRL rules effectively.

In practice, global resources provide universal knowledge as the TBox and SWRL rules. As they are not specific to particular use cases, the ABox can be empty. While the local ontologies serve for specific use cases, so user intent is formalized as assertions to fill in the ABox.

3.4. Summary

The proposed approach involves two roles: developers and users. Developers are responsible for building semantic models of product data and design processes, as well as developing the programs operating in the application layers. Users, on the other hand, are the product design engineers. Given that the required programming knowledge is limited, product design engineers can also assume the role of developers.

Developers can establish the proposed KBE system in the following steps:

1. Describe the design processes explicitly, including relevant components, their design variables, parameters, formulas, etc. And classify these processes into universal rules and case-specific rules.
2. Search for the resources of relevant universal components in the global KB, such as the TBox and related SWRL rules about standard components. If not available, build them locally.
3. Build the TBox and ABox for the case-specific components locally, reusing the global resources.
4. Encode the universal rules for the case-specific components in SWRL. This includes rules for geometric model generation, which involves:
 - (a) Codifying the geometric models in KBE languages.
 - (b) Constructing SWRL rules for generating textual CAD models using the string processing capabilities offered by SWRL.
5. Upload the above-formalized knowledge into KB.
6. Implement the case-specific rules as black-box procedural rules operating in the application layer. These rules can generate relevant assertions describing the product to be designed.
7. Debug and deliver to users.

Users utilize the KBE system to design variant products and obtain CAD models. The usage of this KBE system is as follows:

1. Users manipulate the user interface to provide input.
2. The assertions are generated and uploaded to the KB.
3. The KB performs automatic inferences, including generating the geometric models in textual formats.
4. The textual geometric model can be visualized after being retrieved via SPARQL.

4. Case Study: Shaft Design

4.1. Case Description

Transmission shaft is a non-standard component, entailing to be designed in different use cases. As it is relatively simple but non-standard, it is often selected as a case study in research related to CAD modeling and design knowledge representation, such as Ref. [25, 26, 27]. Therefore, this section provides an application for transmission shaft design as a case study to illustrate the proposed approach integrating KBE and the Semantic Web stack.

A transmission shaft receives the rotation power input from shaft coupling, and outputs the power through the assembled gear. For simplification purposes, this section illustrates a four-section shaft, cooperating with a case-specific gear and standard components including two bearings, two keyseats, and one shaft coupling, as shown in Figure 4. The design process (dp) for a four-section shaft can be simplified for research demonstration as follows.

- dp1** Determine power (P kW), rotation speed (n r/min).
- dp2** Calculate minimum allowed diameter (D_{min}) of section 4 (also of shaft) by $D_{min} = \sqrt[3]{\frac{9550000P}{0.2[\tau_T]n}}$, where τ_T is the torsional shear stress of the shaft material.
- dp3** Select shaft coupling according to D_{min} .
- dp4** Determine length (L_4) and diameter (D_4) of section 4 according to the length ($L_{scpBore}$) and bore diameter ($D_{scpBore}$) of selected shaft coupling.
- dp5** Select key and keyseat according to D_4 .
- dp6** Select bearings according to D_4 .
- dp7** Determine length (L_1, L_3) and diameter (D_1, D_3) of section 1 & 3 according to the bore length (L_{bBore}) and diameter (D_{bBore}) of selected bearings.
- dp8** Determine diameter (D_2) of section 2 according to D_1 and D_3 , length of section 2 (L_2) according to length ($L_{gearBore}$) of given gear.
- dp9** Select key and keyseat according to D_2 .
- dp10** Determine length (L_{sh}) and diameter (D_{sh}) of section shoulder according to D_2 .
- dp11** Generate geometric models of each part of shaft represented in KBE languages.

These steps do not consider many features such as chamfer, sleeve, retaining ring groove, etc. However, additional features can be included as long as they can be modeled in an object-oriented manner and encoded in semantic formats. Section 4.6 demonstrates how to include more features while reusing the semantically represented knowledge.

4.2. Overview of KBE Application

The architecture of the KBE application is derived from Figure 2. Protege [28] serves as the ontology editor for locally editing the model in OWL and debugging the SWRL rules. The local KB is implemented using Apache Jena Fuseki SPARQL server 4.9.0 [29], augmented with the open-source SWRL reasoner Openlet [30].

The global resources encompass knowledge applicable in a broad context. The standard components should be knowledge of this type. However, despite bearing, keyseat and shaft coupling being components standardized by international or national standards, there is a notable absence of available ontologies for them on the Internet. This paper develops the ontologies of these standard components based on standards, acting as the global resources in Figure 2. The dimensions for keyseats are sourced from ISO-2491-1974 Key and Keyway, bearings from ISO-15-2017 Rolling Bearings, and shaft couplings from GB/T 5843-2003 Flange Coupling.

The local resources are crafted for domain-specific design problems. In this case study, the formalization of knowledge related to gears and shafts is specific to the design

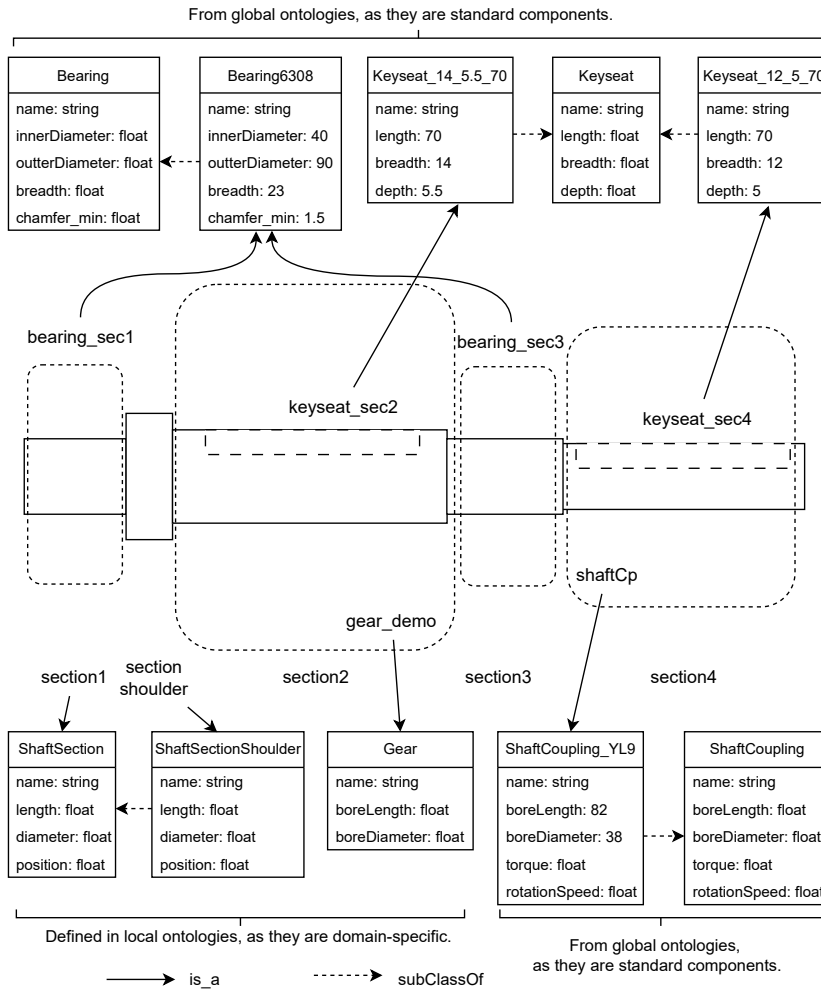


Figure 4: Schematic diagram of a transmission shaft (design case 1).

problem at hand. Nonetheless, this knowledge can be easily shared and reused if other designers accept this knowledge in their design cases. SWRL is employed to represent rules for dimension determination and KBE code generation. Siemens NX KF and AVEVA PML are selected as the languages to represent the geometric model, showing the interoperability facilitated by the semantic model. However, any language capable of interacting with a CAD kernel can be integrated into this semantic form.

The procedural rules encapsulate the complex processes that are not convenient to be represented in SWRL, such as complex math calculations, and the processes to interact with external tools. In this specific case study, the type

selection of bearings and shaft coupling based on shaft section diameter is encapsulated as procedural rules. Because the type selection from detailed standard components is case-specific. Additionally, implementing type selection in SWRL is challenging since SWRL rules primarily operate on instances rather than classes.

4.3. The Semantic Modeling

The semantic models of the components involved in this case study are established using the method outlined in Figure 3. Specifically, the modeling involves the construction of TBox, ABox, and SWRL rules.

The construction of global ontologies involves defining various product types as classes, identifying key features as attributes, and assigning values as restrictions of the class.

For instance, the class "BSeries6300" represents the bearing series 6300 in the standard ISO-15-2017 Rolling Bearings, with "Bearing6308" as a subclass. Key features of this type, such as "innerDiameter," "outerDiameter," "breadth," and "chamfer_min," are defined as data properties of the class. Meanwhile, this class "Bearing6308" is restricted by assertion "(breadth value 23.0f) and (chamfer_min value 1.5f) and (innerDiameter value 40.0f) and (outerDiameter value 90.0f)", regulating the values of each attribute. Regarding the bearings used in specific design cases, e.g., "bearing_sec1" and "bearing_sec3" in Figure 4 and 5, they are the instances of the defined classes, asserted in local ontologies.

The local ontologies for gears and shafts can be flexibly defined based on the specific design problem at hand. In this case study, the focus of the gear ontology is on the "boreLength" and "boreDiameter," while the shaft, being the primary target in this design problem, exhibits greater complexity. The relevant knowledge is formalized through classes, object properties, and data properties as depicted in Figure 6. Examples include "Shaft" and "ShaftSection" as classes (Figure 6 (a)), "hasShaftSection" and "assembled-With" as object properties (Figure 6 (b)), and "positionFrom-Leftmost" as a data property (Figure 6 (c)).

Upon establishing the foundational concepts in the TBox, an ABox specific to a shaft can be constructed. The design of a shaft involves the instantiation of relevant classes and the assertion of attributes and relations associated with them. Figure 5 showcases an instance of a transmission shaft (design case 1), which operates with a transmission power of 3.88 kW and a rotation speed of 130 r/min. Each ellipse in the figure represents an instance, with the string inside the ellipse denoting its name and the string within square brackets indicating its class. To generate this shaft instance, assertions can be created using the SPARQL update code provided in Listing 4. The shaft instance, denoted as "shaft_demo", is instantiated from the class "FourSectionShaft" and comprises four shaft sections labeled as "shaftSection1" through "shaftSection4". The "rightSideIs" relation signifies the positional relationships of each component. The shaft sections "shaftSection1" and "shaftSection3" are assembled with bearings "bearing_sec1" and "bearing_sec3", respectively. Furthermore, "shaftSection2" is assembled with the gear "gear_demo" and features a keyseat labeled "keyseat_sec2". Similarly, "shaftSection4" is assembled with a shaft coupling "shaftCp" and possesses a keyseat denoted as "keyseat_sec4". In essence, these machine-processable assertions encapsulate the designer's intent within the semantic representation.

Based on the TBox and ABox, developers can codify the design rules. The implementation manner is determined based on the universality of the rules. The design processes dp2, dp4, dp7, dp8, dp10, and dp11 can be considered as universal rules. These processes, determining the inferred dimension values of a shaft based on user input, remain constant regardless of various selection criteria in the preceding processes. For instance, if a designer asserts

"shaftSection1 assembledWith bearing_sec1", the SWRL reasoner can infer that the length of "shaftSection1" equals the breadth of "bearing_sec1", and the diameter of "shaftSection" equals the innerDiameter of "bearing_sec1". This inference is achieved through the execution of predefined SWRL rules "S3" in Listing 5. Another two predefined SWRL rules are elaborated as examples in Section 4.4.

Conversely, design processes dp2, dp3, dp5, dp6, and dp9 may not be universally applicable, as the selection criteria in different design cases are case-specific. For example, in alternative cases, the designer might prefer the series 6800 bearings, resulting in a different implementation of dp6. Encapsulating case-specific rules into the complex of OWL and SWRL would lower the universality. Nevertheless, since all series bearings are subclasses of "Bearing", dp7 remains functional. Certainly, if the ontology is tailored for a limited application scope, implementing case-specific rules in SWRL for enhanced interoperability is still feasible. In summary, considering the inconvenience of developing these case-specific rules in SWRL, it may be more economical to implement them in programming languages and encapsulate them as black-box procedural rules.

4.4. The SWRL Rules

In Section 4.3, rule "S3" is explained to illustrate how the reasoner derives the values of inferred attributes of the designed shaft from designers' assertions. This section elaborates on two additional rules "S11" and "S15" to demonstrate how the geometric models are generated by representing the components in KBE languages. See the two rules in Listing 5. Due to length restrictions, other SWRL rules are omitted from the explanation as the grammar of SWRL rules is intuitive to understand.

Rule "S11" is designed to construct geometric models of keyseats represented in AVEVA PML, aligning with the design process dp11 outlined in Section 4.1. The rule expression is divided into distinct segments called subexpressions and labeled based on different functionalities, as presented below (Listing 1 (1)-(6) are subexpression S11.1-6).

```
(1) Keyseat(?kw) ^ isKeyseatOn(?kw, ?ss)
(2) ^ length(?ss, ?h) ^ diameter(?ss, ?Di)
(3) ^ kkw:breath(?kw, ?b) ^ kkw:depth(?kw, ?d) ^ kkw:length(?kw,
    ?l)
(4) ^ swrlb:subtract(?p_tmp, ?Di, ?d) ^ swrlb:divide(?p_x, ?p_tmp,
    ?l)
(5) ^ swrlb:stringConcat(?script_kw, "NEW_NBOX_XLEN_", ?d, "_YLEN_",
    ?b, "_ZLEN_", ?l, "_Position_W_", ?p_x, "_N_0_U_0")
(6) -> AVEVA_PMLScript(?kw, ?script_kw)
```

Listing 1: SWRL rule S11 divided into S11.1-6.

The subexpression S11.1 is employed to select all instances of keyseats denoted by "?kw" and the corresponding shaft sections denoted by "?ss" on which the keyseats are located. Subsequent subexpression S11.2 retrieves the length and diameter of each shaft section instance "?ss". Similarly, the breadth, depth, and length of each keyseat instance "?kw" are obtained through subexpression S11.3. The subexpression S11.4 encapsulates a simple mathematical calculation, specifically $p_x = (D_i - d)/2$. With all

SWRL-based KBE

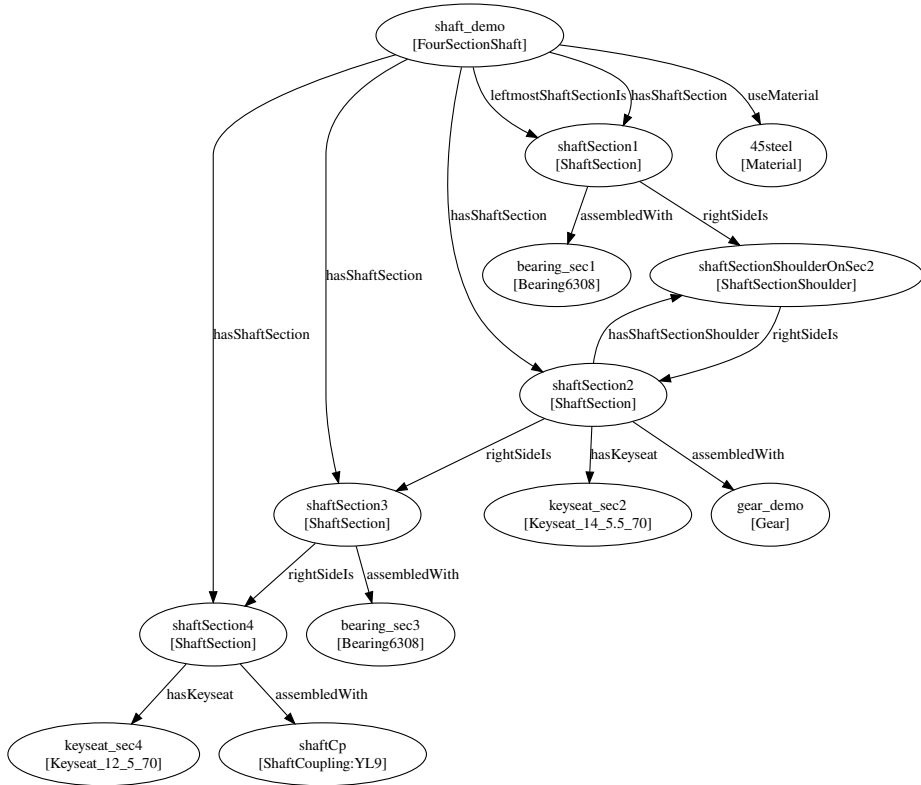


Figure 5: Semantic representation (ABox) of a transmission shaft instance (design case 1).

the operands (“?d, ?b, ?l, ?p_x”), a string is constructed in subexpression S11.5 as follows: “NEW NBOX XLEN ?d YLEN ?b ZLEN ?l Position W ?p_x N O U 0”. This string, represented in AVEVA PML, signifies a boolean subtraction operation by a box with dimensions of (?d, ?b, ?l) and positioned at (?p_x, 0, 0). Here, “W, N, U” denotes the positive directions of the coordinate system as “West, North, Up”. The consequent of S11 (S11.6) indicates the assignment of the aforementioned string (“?script_kw”) to the keyseat instance “?kw” as the value of the data property “AVEVA_PMLScript”. According to the grammar of AVEVA PML, this boolean subtraction operation will take effect on the component in front of this “NBOX” within the structure tree.

Rule “S15” is formulated to generate geometric models of shaft assembly represented in Siemens NX KF, aligning with the design process dp11 outlined in Section 4.1. Similar to S11, it can be segmented and labeled as follows (Listing 2 (1)-(4) are subexpression S15.1-4).

```
(1) FourSectionShaft(?s) ^ leftmostShaftSectionIs(?s, ?ss1) ^
    rightSideIs(?ss1, ?sssh) ^ rightSideIs(?sssh, ?ss2) ^
    rightSideIs(?ss2, ?ss3) ^ rightSideIs(?ss3, ?ss4) ^
    hasKeyseat(?ss2, ?kw2) ^ hasKeyseat(?ss4, ?kw4)
```

```
(2) ^ NX_KFCode(?ss1, ?script1) ^ NX_KFCode(?ss2, ?script2) ^
    NX_KFCode(?ss3, ?script3) ^ NX_KFCode(?ss4, ?script4) ^
    NX_KFCode(?sssh, ?script_sh) ^ NX_KFCode(?kw2, ?script_kw2)
    ^ NX_KFCode(?kw4, ?script_kw4)
(3) ^ swrlb:stringConcat(?str1, “#!_NX/KF_4.0_\n_DefClass:_
    FourSectionShaft_(ug_base_part);\n”, ?script1, “\n”) ^ swrlb
    :stringConcat(?str2, ?script_sh, “\n”, ?script2, “\n”, ?
    script_kw2, “\n”) ^ swrlb:stringConcat(?str3, ?script3, “\n”
    , ?script4, “\n”, ?script_kw4) ^ swrlb:stringConcat(?script,
    ?str1, ?str2, ?str3)
(4) -> NX_KFCode(?s, ?script)
```

Listing 2: SWRL rule S15 divided into S15.1-4.

The subexpression S15.1 is to select all instances constructing the shaft: “?s, ?ss1, ?sssh, ?ss2, ?ss3, ?ss4, ?kw2, ?kw4”. The following subexpression S15.2 retrieves the strings representing geometric models expressed in Siemens NX KF: “?script1, ?script2, ?script3, ?script4, ?script_sh, ?script_kw2, ?script_kw4”. Then subexpression S15.3 concatenates the retrieved strings into a template representing the shaft assembly and stores it as “?script”. The consequent of S15 (S15.4) signifies the assignment of the aforementioned string (“?script”) to the shaft instance “?s” as the value of the data property “NX_KFCode”.

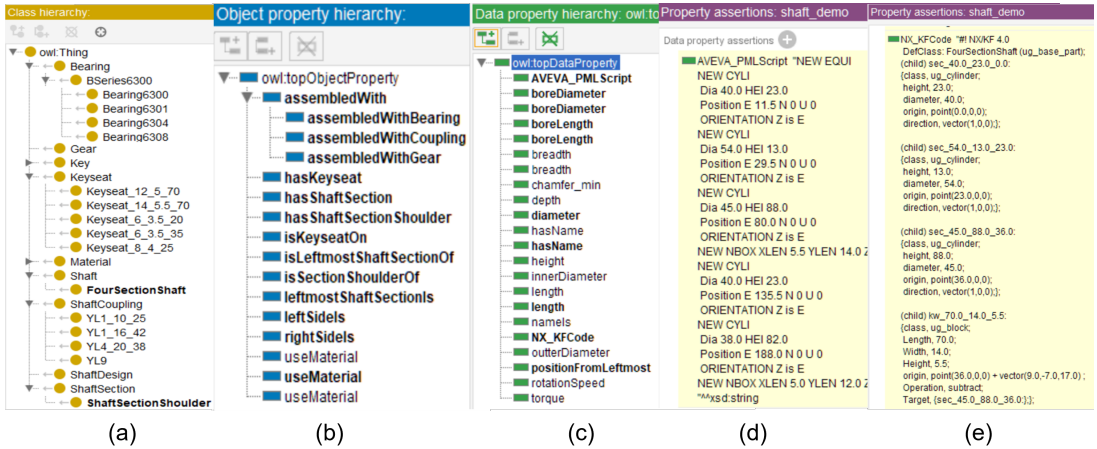


Figure 6: The development and debugging in local ontology editor Protege. (a) Class definitions; (b) Object property; (c) Data property; (d) Inferred geometric model of a shaft in AVEVA PML; (e) Inferred geometric model of a shaft in Siemens NX KF.

4.5. The Application Demonstration

This section demonstrates the KBE application and its output results after execution. Protege serves as the ontology editor for local development and debugging, as depicted in Figure 6. Figures 6 (a), (b), and (c) showcase the formalized classes, object properties, and data properties. Some properties are imported from global ontologies, while others are defined in local ontologies. The duplicated properties stem from different namespaces. After the definition of TBox, ABox for a shaft can be constructed, as shown in Figure 5.

Figures 6 (d) and (e) illustrate the inferred data properties “AVEVA_PMLScript” and “NX_KFCode”, respectively. These properties are derived by the SWRL reasoner through the execution of predefined SWRL rules based on the semantic representation of the expected shaft. The string values of these two attributes represent the geometric model of the expected shaft.

After the development of the domain-specific local ontologies, they are uploaded to the local KB. The SPARQL query from Fuseki KB is shown in Listing 3 for the retrieval of the inferred attributes representing the design results. The KB will return the result including the KBE code, as shown in Figure 7. The text version of the returned KBE code is shown in Listing 6 and 7. Based on the returned KBE code, the geometric model of design case 1 can be visualized in corresponding CAD software, as illustrated in Figure 8 (a) and (c).

As the product data is formalized in SPARQL code (Listing 4), a new shaft can be designed by simply substituting the keywords within the SPARQL code template. For instance, let us consider a specific design scenario, such as design case 2, where a small shaft is required with transmission power and rotation speed of 0.5 kW and 180 r/min, respectively. The selection of bearings, shaft coupling, and keyseat can be determined either through

procedural rules within the user application or manually. Subsequently, the SPARQL code describing the new design case can be constructed by adjusting design parameters like power and rotation speed, along with incorporating the chosen components, such as “bearing:Bearing6304”, “kkw:Keyseat_8_4_25”, and “scp:YL1_16_42”. After receiving these new assertions, the reasoner can infer new facts automatically by executing the predefined SWRL rules. Thus, design case 2 can be quickly solved and the CAD model can be automatically generated by reusing the design knowledge, as illustrated in Figure 8 (b) and (d). This case demonstrates the capability of KBE to efficiently generate variants by reusing semantically represented design knowledge.

```

PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX sft: <http://example/shaft#>
PREFIX dc1: <http://example/shaft/design_case1#>
SELECT ?subject ?aveva_pml ?nx_kf
WHERE {
  ?subject rdf:type sft:FourSectionShaft.
  FILTER(STRSTARTS(STR(?subject), STR(dc1:)))
  ?subject sft:AVEVA_PMLScript ?aveva_pml.
  ?subject sft:NX_KFCode ?nx_kf.
}
    
```

Listing 3: SPARQL query to retrieve geometric model represented in KBE languages (design case 1).

4.6. An Extended Design Case

A real-world scenario typically involves more features than the above design cases, necessitating additional design processes. For instance, in an extended case, let us consider “design case 3” to illustrate how the proposed approach can handle a broader range of features and processes. In this case, the shaft comprises five sections, requiring chamfering at both ends. Sleeves are incorporated between the bearing

```

Table Response 1 result in 0.012 seconds
sub aveva.pml nx.kf
# NX/KF 4.0 DefClass: FourSectionShaft (ug_base_part; (child) sec_40.0_23.0_0.0; (class
ion E 11.5 N 0 U 0 ORIENTATION Z is E NEW
s, ug_cylinder; height: 23.0; diameter: 40.0; origin, point(0.0,0.0); direction, vector(1,0,
CYLI Dia 34.0 HEI 13.0 Position E 29.5 N 0 U 0
0.0); (child) sec_54.0_13.0_23.0; (class, ug_cylinder; height: 13.0; diameter: 54.0; origin,
O ORIENTATION Z is E NEW CYLI Dia 45.0 H
point(23.0,0.0); direction, vector(1,0,0); (child) sec_45.0_88.0_36.0; (class, ug_cylinder;
http://ex height: 88.0; diameter: 45.0; origin, point(36.0,0.0); direction, vector(1,0,0); (child) kw,
ample/h N Z is E NEW NBOX XLEN 5.5 YLEN 14.0 ZLE
70.0, 14.0, 5.5; (class, ug_block; Length: 70.0; Width: 14.0; Height: 5.5; origin, point(36
afly/shaft N 70.0 Position W 19.75 N 0 U 0 NEW CYLI
0.0,0) + vector(9.0, -7.0,17.0); Operation, subtract; Target, {sec_45.0_88.0_36.0}; (chil
Dia 40.0 HEI 23.0 Position E 135.5 N 0 U 0 O
d); sec_40.0_23.0_124.0; (class, ug_cylinder; height: 23.0; diameter: 40.0; origin, point(1
demo RIENTATION Z is E NEW CYLI Dia 38.0 HEI 8
24.0,0.0); direction, vector(1,0,0); (child) sec_38.0_82.0_147.0; (class, ug_cylinder; heig
ht, 82.0; diameter: 38.0; origin, point(147.0,0.0); direction, vector(1,0,0); (child) kw, 70.
Z 0 Position E 188.0 N 0 U 0 ORIENTATION Z
is E NEW NBOX XLEN 5.0 YLEN 12.0 ZLEN 7
0, 12.0, 5.0; (class, ug_block; Length: 70.0; Width: 12.0; Height: 5.0; origin, point(147.0,
0.0) + vector(6.0, -6.0,14.0); Operation, subtract; Target, {sec_38.0_82.0_147.0};

```

Showing 1 to 1 of 1 entries

Figure 7: The returned KBE code for design case 1 from KB.

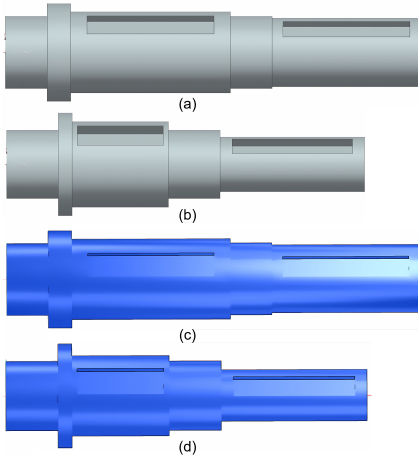


Figure 8: Two different shaft models visualized in different CAD software. (a) Design case 1 in Siemens NX; (b) Design case 2 in Siemens NX; (c) Design case 1 in AVEVA; (d) Design case 2 in AVEVA.

and shaft to mitigate wear. Additionally, adjustment in the distance between bearings is demanded, depending on the bearing positions. The third shaft section is used for this adjustment.

Based on the semantic model of the four-section shaft, much of the existing knowledge can be reused, with only the additional concepts requiring definition for the extended design case. The SPARQL code in Listing 8 illustrates the extension of the TBox to accommodate the new design features. A new class named “FiveSectionShaft” consisting of five sections, is introduced, along with object properties such as “hasChamfer” and “assembledWithSleeve”. Additionally, the classes “Chamfer” and “Sleeve” are defined, along with their corresponding properties.

Once the product data is defined, the design rules and processes can be established. The geometric generation rules for creating the geometric model are universal and can be encoded in the semantic layer. The SWRL rules for geometric model generation (S16-19) are listed in Listing 9. For demonstration purposes, only the generation of the model in Siemens NX KF is provided. On the contrary, the

dimensions of the sleeve, the distance between bearings, and the chamfer offset are determined based on the user’s specific requirements. Therefore, these case-specific rules are implemented in the application layer using programming languages and encapsulated as black-box procedural rules.

After the TBox and rules are prepared, instances for this specific shaft type can be generated. The assertions for design case 3 are listed in Listing 10. The chamfer offset, the distance between bearings (length of the third section), and the dimensions of the sleeve should be calculated by procedural rules. For demonstration purposes, the procedural rules are not provided, and these values are input manually. After inserting these assertions into the KB, the NX KF code will be inferred automatically. The geometric model in textual format can be retrieved from the KB and visualized as shown in Figure 9.



Figure 9: The five-section shaft (design case 3) visualized in Siemens NX.

5. Discussion

5.1. Qualitative Comparison

The case study showcases three design cases using the proposed approach, which integrates KBE and the Semantic Web stack, thereby validating the feasibility of this method. Additionally, the two four-section shaft design cases demonstrate its ability to rapidly generate variant products. The visualizations in different CAD software platforms underscore the interoperability facilitated by semantic modeling. Furthermore, the extended design case illustrates a five-section shaft derived from the four-section shaft knowledge, highlighting its scalability and extendability.

An alternative and commonly used approach for constructing KBE applications involves the black-box reuse of design processes, falling under type 3 reuse as discussed in Section 2.2. In this alternative approach, SWRL is not used for universal rules; instead, all reusable processes are encapsulated as black-box processes, such as libraries and web services. Comparing the two solutions across all factors related to software quality can be challenging. However, based on the discussion in Section 2.2, particularly focusing on factors related to reusability and FAIR principles, the following metrics are selected for the qualitative comparison: maintainability (cohesion, understandability, modifiability), portability (execution environment, findability), compatibility (interoperability, co-existence). A qualitative comparison between the proposed SWRL-based approach and this alternative is illustrated in Figure 10 and explained in Table 2.

SWRL-based KBE

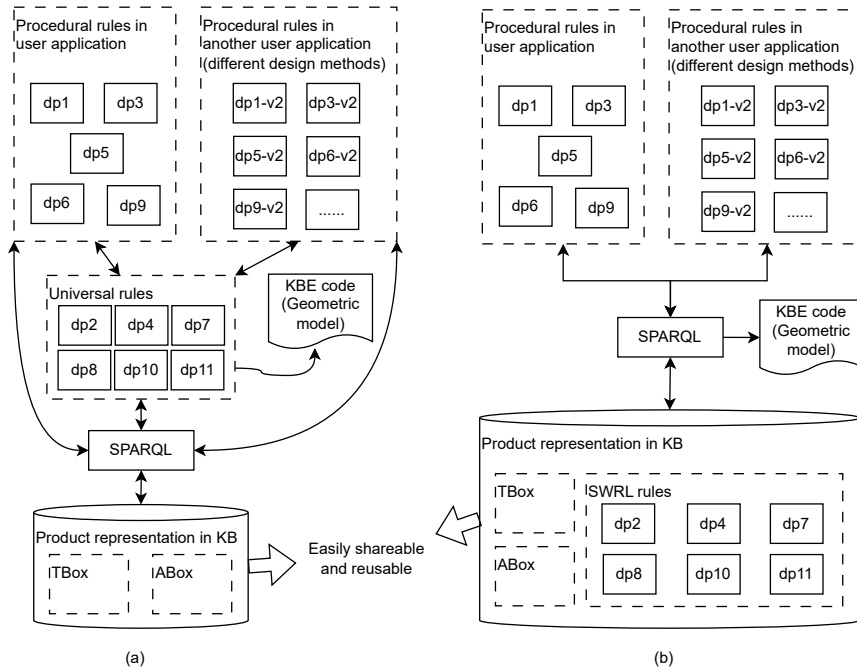


Figure 10: Comparison with the non-SWRL approach. (a) Non-SWRL approach; (b) Proposed SWRL-based approach.

Table 2

Qualitative comparison between black-box reuse and the proposed SWRL-based reuse

	Black-box Reuse	Proposed SWRL-based Reuse
Cohesion	The processes and their associated data are not packaged in the same module.	The universal rules, especially the CAD model reconstruction processes, are packaged in KB, forming a high-cohesive module.
Understandability	Users can only understand the input/output unless they have the source code of the black-box processes.	Users can see the SWRL rules, including embedded KBE code snippets, which is a human-readable format.
Modifiability	Users normally do not have access to the source code of the black-box processes.	Users can modify the SWRL rules including the embedded KBE code snippets via an ontology editor.
Execution environment	The black-box processes have to be executed in specific environments, such as servers installed with the runtime environment.	The execution environment (SWRL reasoner) is embedded in the ontology-based KB, and the rules are executed automatically.
Findability	A registration mechanism of black-box processes is needed, such as the web service repository.	The rules are stored in KB and executed automatically, no need to search.
Interoperability	The black-box processes are interacted with in a specific provided way, such as via web services or APIs.	The rules are executed automatically, and the results are returned via SPARQL, which is the same way for data retrieval.
Co-existence	The runtime environment has to be kept.	Embedded in KB, no extra burden.

In Figure 10, both approaches encapsulate the case-specific rules (dp1, dp3, dp5, dp6, dp9) as black-box processes for knowledge sharing and reusing. The differences lie in the implementation of the universal rules (dp2, dp4, dp7, dp8, dp10, dp11). The non-SWRL approach implements the universal rules, especially the CAD model (re)construction,

also as black-box processes. Comparatively, the proposed SWRL-based approach implements them as SWRL rules stored in the ontology-based KB. The scenario depicted in the figure involves another design group attempting to reuse these universal rules in their own application. For instance, the case study in this paper involves the utilization of the

series 6300 bearing in the standard ISO-15-2017 Rolling Bearing, as codified in the design process dp6. Suppose these designers from the other group try to integrate different design rules (e.g., dp6-version2) to create various types of shafts assembled with different bearings, such as the series 6800 bearings. In the non-SWRL approach, these designers have to consider how to invoke universal rules, as it cannot be assumed that they know how the previous application invokes these black-box universal rules. In contrast, in the SWRL-based approach, these universal rules are well-packaged in a high-cohesive module and automatically executed by the reasoner. These designers only need to retrieve the results from the KB through SPARQL, which is the same way for product data retrieval. Therefore, the universal rules become easily shareable and reusable, similar to the product data, due to the central position of the ontology-based KB.

Table 2 summarizes the benefits of the SWRL-based approach in terms of the selected factors. Because the universal rules are formalized in SWRL, they can be stored in the ontology-based KB, forming a high-cohesive module. Since the ontology-based KB can be integrated with a SWRL reasoner, users do not need to consider the execution environment. For example, the query in Listing 3 can easily retrieve the geometric model in KBE languages, as shown in Figure 7. This easy access to product data facilitates knowledge sharing within different design teams. Unlike the black-box reuse, the runtime environment has to be kept or even integrated into the new applications. When the number of universal rules becomes large, a registration mechanism of black-box processes is required, but it is not a problem for the SWRL rules stored in KB. Additionally, as SWRL is a human-readable rule language based on description logic, it is intuitive to understand, and users can modify them according to their own requirements. Based on these features, an application with fine interoperability can be developed, allowing for convenient functionality extension, especially the integration with different CAD tools. For instance, two different KBE languages used in different software, AVEVA PML and Siemens NX KF, can be easily integrated into the semantic product model, as shown in Figure 6 (d) and (e). Furthermore, this integration framework can accommodate additional KBE languages in a similar manner, facilitating comprehensive support for diverse engineering environments if required. The demonstrated good interoperability and extendability are crucial in the future development of applications for multi-disciplinary design optimization. In these contexts, where designs involve diverse disciplines and evolving requirements, applications must be adaptable and open to incorporating new capabilities.

5.2. Limitations

Considering this paper serves as a proof-of-concept to propose an approach for semantically representing product design knowledge, it is important to note that the shaft's geometry in the case study may be simplified for some cases to give a focus to the construction and application of semantic rules. With the complexity of production models

increasing, a significantly greater number of features and processes can be involved. However, they can still be captured and supported with semantic modeling in a similar manner as demonstrated above. In particular, the design case 3 in Section 4.6 provides an example of extending the existing model to include additional features. Through this case, it's conceivable that more features and rules can be modeled in an object-oriented manner and encoded in semantic formats. Certainly, it is beneficial to explore more complex design cases containing a wider range of features to further assess the effectiveness of the proposed approach.

Moreover, the effectiveness of the proposed approach might encounter limitations due to the expressiveness of SWRL and KBE languages. Currently, it is more practical to implement simple rules in SWRL, while complex rules are still best suited for implementation as black-box procedural rules. Additionally, the geometric modeling tasks requiring precise edge and vertex selection can be inconvenient when using textual KBE languages. This difficulty arises from the necessity for underlying information about geometric entities and sometimes relies on information within all involved geometric entities.

6. Conclusions

This paper introduces an approach that utilizes KBE and the Semantic Web stack to build KBE applications for product design. This approach demonstrates fine performance in terms of reusability and interoperability, which are essential in developing applications to support knowledge-intensive engineering tasks. To validate the feasibility of the proposed approach, a case study involving the design of a four-section shaft is conducted as conceptual proof. The findings can be summarized as follows:

1. Design rules can be broadly categorized into two types: universal rules and case-specific rules.
2. The implementation manner for rules can be determined based on their universality, with SWRL suitable for universal rules and black-box functions for case-specific rules.
3. The proposed approach allows the construction of a cohesive knowledge base that encapsulates universal rules, leveraging the reasoning capabilities provided by the Semantic Web stack.
4. The (re)construction of geometric models can be achieved using KBE language code snippets and the string processing capabilities of SWRL. The resulting CAD models can be generated in KBE languages and visualized through interaction with the CAD kernel.
5. This high-cohesive knowledge representation form can offer easy sharing and reuse together with robust interoperability, facilitating future functionality extension.

In future research, in addition to addressing the limitations mentioned in Section 5.2, it is essential to tackle certain inconveniences encountered during the development

of semantic models. Firstly, the difficulty in finding ontologies for standard components, with only human-readable standard documents available, highlights the need for digital transformation of this knowledge for improved data interoperability. The creation of a tool capable of converting human-readable standard documents into interoperable ontologies would be valuable in this regard. Secondly, the development of a user-friendly tool to convert mathematical expressions into SWRL could significantly enhance convenience when writing rules involving mathematical calculations. A similar demand also applies to other operations, such as string processing. Additionally, addressing the need for a faster and more robust reasoner that can be integrated with ontology-based KB is essential for improving reasoning performance.

A. Appendix

A shaft can be designed by creating assertions using the SPARQL update code, e.g. the code for design case 1 in Listing 4.

```

prefix : <http://example/shaft#>
prefix bearing: <http://example/bearing#>
prefix gear: <http://example/gear#>
prefix kkw: <http://example/key_keyway#>
prefix mat: <http://example/material#>
prefix owl: <http://www.w3.org/2002/07/owl#>
prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
prefix scp: <http://example/shaftCoupling#>
prefix xsd: <http://www.w3.org/2001/XMLSchema#>
prefix xml: <http://www.w3.org/XML/1998/namespace>
prefix dcl: <http://example/shaft/design_case1#>

```

INSERT DATA

```

{
  # create new instances
  dcl:bearing_sec1 rdf:type owl:NamedIndividual, bearing:Bearing6308
  .
  dcl:bearing_sec3 rdf:type owl:NamedIndividual, bearing:Bearing6308
  .
  dcl:45steelForShaft rdf:type owl:NamedIndividual, mat:45steel.
  dcl:keyseat_sec2 rdf:type owl:NamedIndividual, kkw:Keyseat_14_5_5_70.
  dcl:keyseat_sec4 rdf:type owl:NamedIndividual, kkw:Keyseat_12_5_70
  .
  dcl:shaftCp rdf:type owl:NamedIndividual, scp:YL9.
  dcl:gear_demo rdf:type owl:NamedIndividual, gear:Gear;
    gear:boreLength "90.0"^^xsd:float;
    gear:boreDiameter "45.0"^^xsd:float.
  dcl:shaftSection1 rdf:type owl:NamedIndividual, :ShaftSection.
  dcl:shaftSection2 rdf:type owl:NamedIndividual, :ShaftSection.
  dcl:shaftSection3 rdf:type owl:NamedIndividual, :ShaftSection.
  dcl:shaftSection4 rdf:type owl:NamedIndividual, :ShaftSection.
  dcl:shaftSectionShoulderOnSec2 rdf:type owl:NamedIndividual, :
    ShaftSectionShoulder.

  # assert the relations between instances
  dcl:shaftSection1 :rightSideIs dcl:shaftSectionShoulderOnSec2;
    :assembledWith dcl:bearing_sec1.
  dcl:shaftSectionShoulderOnSec2 :rightSideIs dcl:shaftSection2.
  dcl:shaftSection2 :rightSideIs dcl:shaftSection3;
    :hasShaftSectionShoulder dcl:shaftSectionShoulderOnSec2;
    :assembledWith dcl:gear_demo;
    :hasKeyseat dcl:keyseat_sec2.
  dcl:shaftSection3 :rightSideIs dcl:shaftSection4;
    :assembledWith dcl:bearing_sec3.
  dcl:shaftSection4 :assembledWith dcl:shaftCp;

```

```

    :hasKeyseat dcl:keyseat_sec4.
  dcl:shaft_demo rdf:type owl:NamedIndividual, :FourSectionShaft;
    :hasShaftSection dcl:shaftSection1,
      dcl:shaftSection2,
      dcl:shaftSection3,
      dcl:shaftSection4;
    :leftmostShaftSectionIs dcl:shaftSection1;
    :power "3.88"^^xsd:float ;
    scp:rotationSpeed "130.0"^^xsd:float ;
    :useMaterial dcl:45steelForShaft.

  # assert the instances are different from each other to speed up
  reasoning
  [] rdf:type owl:AllDifferent ;
  owl:distinctMembers (
    dcl:45steelForShaft
    dcl:bearing_sec1
    dcl:bearing_sec3
    dcl:shaftCp
    dcl:gear_demo
    dcl:keyseat_sec2
    dcl:keyseat_sec4
    dcl:shaftSection1
    dcl:shaftSection2
    dcl:shaftSection3
    dcl:shaftSection4
    dcl:shaftSectionShoulderOnSec2
    dcl:shaft_demo
  ) .
}

```

Listing 4: The SPARQL update code for assertions of a shaft.

The SWRL rules (S1-15) used in the semantic model of the design are listed in Listing 5.

```

@prefix : <http://example/shaft#> .
@prefix kkw: <http://example/key_keyway#> .
@prefix mat: <http://example/material#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix scp: <http://example/shaftCoupling#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix gear: <http://example/gear#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix swrlb: <http://www.w3.org/2003/11/swrlb#> .
@prefix bearing: <http://example/bearing#> .
@base <http://example/shaft#> .

#S1: (dp2) Calculate minimum allowed diameter of section 4 (also of
shaft);
FourSectionShaft(?s) ^ power(?s, ?p) ^ scp:rotationSpeed(?s, ?n) ^
useMaterial(?s, ?m) ^ mat:torsionalShearStress(?m, ?tau) ^ swrlb
:multiply(?numerator, 9550000, ?p) ^ swrlb:multiply(?denominator
, 0.2, ?tau, ?n) ^ swrlb:divide(?temp, ?numerator, ?denominator)
^ swrlb:pow(?D_min, ?temp, 0.3333) -> diameter_min(?s, ?D_min)

#S2: (dp4) Determine length and diameter of section 4 according to
assembled shaft coupling;
ShaftSection(?ss) ^ scp:ShaftCoupling(?cp) ^ assembledWith(?ss, ?cp) ^
scp:boreDiameter(?cp, ?Di) ^ scp:boreLength(?cp, ?bb) -> length
(?ss, ?bb) ^ diameter(?ss, ?Di)

#S3: (dp7) Determine length and diameter of section 1 & 3 according to
assembled bearings;
ShaftSection(?ss) ^ bearing: Bearing(?bearing) ^ assembledWith(?ss, ?
bearing) ^ bearing:innerDiameter(?bearing, ?Di) ^ bearing:
breadth(?bearing, ?bb) -> length(?ss, ?bb) ^ diameter(?ss, ?Di)

#S4: (dp8) Determine diameter of section 2 according to diameter of
section 1 & 3, length of section 2 according to given gear;

```

SWRL-based KBE

```

ShaftSection(?ss) ^ gear:Gear(?gear) ^ assembledWith(?ss, ?gear) ^
gear:boreDiameter(?gear, ?Di) ^ gear:boreLength(?gear, ?bb) ^
swrlb:subtract(?bb2, ?bb, 2) ^ swrlb:round(?bb3, ?bb2) -> length
(?ss, ?bb3) ^ diameter(?ss, ?Di)

#S5: (dp10) Determine length and diameter of section shoulder
according to section 2;
ShaftSectionShoulder(?sss) ^ isSectionShoulderOf(?sss, ?sec) ^
diameter(?sec, ?d) ^ length(?sec, ?l) ^ swrlb:multiply(?d2, ?d,
1.2) ^ swrlb:round(?d3, ?d2) ^ swrlb:multiply(?l2, ?l, 0.15) ^
swrlb:round(?l3, ?l2) -> diameter(?sss, ?d3) ^ length(?sss, ?l3)

#S6: Define the position of 1st section as 0 from leftmost;
Shaft(?s) ^ leftmostShaftSectionIs(?s, ?ss1) -> positionFromLeftmost(?
ss1, "0.0"^^xsd:float)

#S7: Calculate the position of other sections from leftmost;
ShaftSection(?ss) ^ leftSideIs(?ss, ?left) ^ length(?left, ?l) ^
positionFromLeftmost(?left, ?p1) ^ swrlb:add(?p, ?l, ?p1) ->
positionFromLeftmost(?ss, ?p)

#S8: Give a name to each shaft section according to the design
variables;
ShaftSection(?ss) ^ diameter(?ss, ?d) ^ swrlb:stringConcat(?n, "sec_"
^^rdf:PlainLiteral, ?d, "^^rdf:PlainLiteral, ?l, "^^rdf:
PlainLiteral, ?p) ^ positionFromLeftmost(?ss, ?p) ^ length(?ss,
?l) -> hasName(?ss, ?n)

#S9: Give a name to each keyseat according to the design variables;
kkw:Keyseat(?kw) ^ kkw:depth(?kw, ?b) ^ kkw:length(?kw, ?l) ^ kkw:
breadth(?kw, ?b) ^ swrlb:stringConcat(?name, "kw_", ?l, "_", ?b,
"_", ?h) -> hasName(?kw, ?name)

#S10: (dp11) Generate geometric models of shaft section represented in
AVEVA PML;
ShaftSection(?ss) ^ length(?ss, ?h) ^ diameter(?ss, ?Di) ^
positionFromLeftmost(?ss, ?p) ^ swrlb:divide(?halfH, ?h, 2) ^
swrlb:add(?p_cen, ?halfH, ?p) ^ swrlb:stringConcat(?script, "NEW
_CYLIN_Dia_", ?Di, "_HEI_", ?h, "_N_Position_E_", ?p_cen, "_N_0
_U_0\N_ORIENTATION_Z_is_E") -> AVEVA_PMLScript(?ss, ?script)

#S11: (dp11) Generate geometric models of keyseat represented in AVEVA
PML;
kkw:Keyseat(?kw) ^ diameter(?ss, ?Di) ^ length(?ss, ?h) ^ kkw:depth(?
kw, ?d) ^ kkw:length(?kw, ?l) ^ kkw:isKeyseatOn(?kw, ?ss) ^
swrlb:subtract(?p_tmp, ?Di, ?d) ^ swrlb:divide(?p_x, ?p_tmp, 2)
^ swrlb:stringConcat(?script_kw, "NEW_NBOX_XLEN_", ?d, "_YLEN_",
?b, "_ZLEN_", ?l, "_Position_W_", ?p_x, "_N_0_U_0") ^ kkw:
breadth(?kw, ?b) -> AVEVA_PMLScript(?kw, ?script_kw)

#S12: (dp11) Generate geometric models of shaft assembly represented
in AVEVA PML;
FourSectionShaft(?s) ^ leftmostShaftSectionIs(?s, ?ss1) ^ rightSideIs
(?ss1, ?sssh) ^ rightSideIs(?sssh, ?ss2) ^ rightSideIs(?ss2, ?ss3
) ^ rightSideIs(?ss3, ?ss4) ^ hasKeyseat(?ss2, ?kw2) ^ hasKeyseat
(?ss4, ?kw4) ^ AVEVA_PMLScript(?ss1, ?script1) ^ AVEVA_PMLScript
(?ss2, ?script2) ^ AVEVA_PMLScript(?ss3, ?script3) ^
AVEVA_PMLScript(?ss4, ?script4) ^ AVEVA_PMLScript(?sssh, ?
script_sh) ^ AVEVA_PMLScript(?kw2, ?script_kw2) ^
AVEVA_PMLScript(?kw4, ?script_kw4)
^ swrlb:stringConcat(?str1, "NEW EQUI\N", ?script1, "\N", ?
script_sh, "\N", ?script2, "\N")
^ swrlb:stringConcat(?str2, ?script_kw2, "\N", ?script3, "\N", ?
script4, "\N", ?script_kw4, "\N")
^ swrlb:stringConcat(?script, ?str1, ?str2)
-> AVEVA_PMLScript(?s, ?script)
^ swrlb:stringConcat(?str1, "NEW EQUI\N", ?script1, "\N", ?
script_sh, "\N", ?script2, "\N")
^ swrlb:stringConcat(?str2, ?script_kw2, "\N", ?script3, "\N", ?
script4, "\N", ?script_kw4, "\N")
^ swrlb:stringConcat(?script, ?str1, ?str2)
-> AVEVA_PMLScript(?s, ?script)

#S13: (dp11) Generate geometric models of shaft section represented in
Siemens NX Knowledge Fusion;
ShaftSection(?ss) ^ :hasName(?ss, ?name) ^ :length(?ss, ?bb) ^ :
diameter(?ss, ?Di) ^ :positionFromLeftmost(?ss, ?p) ^ swrlb:
stringConcat(?script, "(child)", ?name, ":\N(class, ug_cylinder;\N
n_height, ", ?bb, ";\N_diameter, ", ?Di, ";\N_origin, _point(", ?p, "
, 0, 0);\N_direction, _vector(1, 0, 0));\N") -> :NX_KFCode(?ss, ?
script)

#S14: (dp11) Generate geometric models of keyseat represented in
Siemens NX Knowledge Fusion;
kkw:Keyseat(?kw) ^ kkw:isKeyseatOn(?kw, ?ss) ^ :hasName(?ss, ?name_s) ^
length(?ss, ?h) ^ diameter(?ss, ?Di) ^ :positionFromLeftmost(?
ss, ?p) ^ hasName(?kw, ?name) ^ kkw:width(?kw, ?b) ^ kkw:depth
(?kw, ?d) ^ kkw:length(?kw, ?l) ^ swrlb:subtract(?v_tmp1, ?h, ?l
) ^ swrlb:divide(?v_x, ?v_tmp1, 2) ^ swrlb:divide(?v_y, ?b, -2) ^
swrlb:divide(?v_tmp3, ?Di, 2) ^ swrlb:subtract(?v_z, ?v_tmp3, ?d
)
^ swrlb:stringConcat(?vec, "vector(", ?v_x, ", ", ?v_y, ", ", ?v_z, ", ")")
^ swrlb:stringConcat(?str1, "(child)", ?name, ":\N(class, ug_block;\N
n Length, ", ?l, ", "\N Width, ", ?b, ", "\N Height, ", ?d)
^ swrlb:stringConcat(?str2, "\N origin, point(", ?p, ", 0, 0) + ", ?
vec)
^ swrlb:stringConcat(?str3, "\N Operation, subtract;\N Target, {", ?
name_s, "};\N")
^ swrlb:stringConcat(?script_kw, ?str1, ?str2, ?str3)
-> NX_KFCode(?kw, ?script_kw)

#S15: (dp11) Generate geometric models of shaft assembly represented
in Siemens NX Knowledge Fusion;
FourSectionShaft(?s) ^ leftmostShaftSectionIs(?s, ?ss1) ^ rightSideIs
(?ss1, ?sssh) ^ rightSideIs(?sssh, ?ss2) ^ rightSideIs(?ss2, ?ss3
) ^ rightSideIs(?ss3, ?ss4) ^ hasKeyseat(?ss2, ?kw2) ^ hasKeyseat
(?ss4, ?kw4)
^ NX_KFCode(?ss1, ?script1) ^ NX_KFCode(?ss2, ?script2) ^ NX_KFCode(?
ss3, ?script3) ^ NX_KFCode(?ss4, ?script4) ^ NX_KFCode(?sssh, ?
script_sh) ^ NX_KFCode(?kw2, ?script_kw2) ^ NX_KFCode(?kw4, ?
script_kw4)
^ swrlb:stringConcat(?str1, "##! NX/KF 4.0 \N DefClass:
FourSectionShaft (ug_base_part);\N", ?script1, "\N")
^ swrlb:stringConcat(?str2, ?script_sh, "\N", ?script2, "\N", ?
script_kw2, "\N")
^ swrlb:stringConcat(?str3, ?script3, "\N", ?script4, "\N", ?
script_kw4)
^ swrlb:stringConcat(?script, ?str1, ?str2, ?str3)
-> NX_KFCode(?s, ?script)

```

Listing 5: SWRL rules for semantic model of shaft design.

The geometric models of the designed shaft (design case 1) are automatically constructed by the reasoner and represented in AVEVA PML and Siemens NX Knowledge Fusion, as shown in Lists 6 and 7.

```

NEW EQUI
NEW CYLI
Dia 40.0 HEI 23.0
Position E 11.5 N 0 U 0
ORIENTATION Z is E
NEW CYLI
Dia 54.0 HEI 13.0
Position E 29.5 N 0 U 0
ORIENTATION Z is E
NEW CYLI
Dia 45.0 HEI 88.0
Position E 80.0 N 0 U 0
ORIENTATION Z is E
NEW NBOX XLEN 5.5 YLEN 14.0 ZLEN 70.0 Position W 19.75 N 0 U 0
NEW CYLI
Dia 40.0 HEI 23.0

```

```
Position E 135.5 N 0 U 0
ORIENTATION Z is E
NEW CYLI
Dia 38.0 HEI 82.0
Position E 188.0 N 0 U 0
ORIENTATION Z is E
NEW NBOX XLEN 5.0 YLEN 12.0 ZLEN 70.0 Position W 16.5 N 0 U 0
```

Listing 6: Geometric model of a shaft represented in AVEVA PML.

```
#! NX/KF 4.0
DefClass: FourSectionShaft (ug_base_part);
(child) sec_40.0_23.0_0.0:
{class, ug_cylinder;
 height, 23.0;
 diameter, 40.0;
 origin, point(0.0,0,0);
 direction, vector(1,0,0);};

(child) sec_54.0_13.0_23.0:
{class, ug_cylinder;
 height, 13.0;
 diameter, 54.0;
 origin, point(23.0,0,0);
 direction, vector(1,0,0);};

(child) sec_45.0_88.0_36.0:
{class, ug_cylinder;
 height, 88.0;
 diameter, 45.0;
 origin, point(36.0,0,0);
 direction, vector(1,0,0);};

(child) kw_70.0_14.0_5.5:
{class, ug_block;
 Length, 70.0;
 Width, 14.0;
 Height, 5.5;
 origin, point(36.0,0,0) + vector(9.0,-7.0,17.0) ;
 Operation, subtract;
 Target, {sec_45.0_88.0_36.0};};

(child) sec_40.0_23.0_124.0:
{class, ug_cylinder;
 height, 23.0;
 diameter, 40.0;
 origin, point(124.0,0,0);
 direction, vector(1,0,0);};

(child) sec_38.0_82.0_147.0:
{class, ug_cylinder;
 height, 82.0;
 diameter, 38.0;
 origin, point(147.0,0,0);
 direction, vector(1,0,0);};

(child) kw_70.0_12.0_5.0:
{class, ug_block;
 Length, 70.0;
 Width, 12.0;
 Height, 5.0;
 origin, point(147.0,0,0) + vector(6.0,-6.0,14.0) ;
 Operation, subtract;
 Target, {sec_38.0_82.0_147.0};};
```

Listing 7: Geometric model of a shaft represented in Siemens NX Knowledge Fusion.

The new class definitions and related properties can be inserted to the KB using the code in Listing 8.

```
prefix : <http://example/shaft#>
prefix bearing: <http://example/bearing#>
prefix gear: <http://example/gear#>
prefix kkw: <http://example/key\_keyway#>
prefix mat: <http://example/material#>
prefix owl: <http://www.w3.org/2002/07/owl#>
prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
prefix scp: <http://example/shaftCoupling#>
prefix xsd: <http://www.w3.org/2001/XMLSchema#>
prefix xml: <http://www.w3.org/XML/1998/namespace>
prefix slv: <http://example/sleeve#>
```

```
INSERT DATA
{
  # define new classes, DatatypeProperty and ObjectProperty.
  :FiveSectionShaft rdf:type owl:Class ;
    rdfs:subClassOf :Shaft ,
    [ rdf:type owl:Restriction ;
      owl:onProperty :hasShaftSection ;
      owl:qualifiedCardinality "5"^^xsd:nonNegativeInteger ;
      owl:onClass :ShaftSection ] .

  slv:Sleeve rdf:type owl:Class .
  slv:breathd rdf:type owl:DatatypeProperty ;
    rdfs:subPropertyOf owl:topDataProperty ;
    rdfs:range xsd:float .
  slv:innerDiameter rdf:type owl:DatatypeProperty ;
    rdfs:subPropertyOf owl:topDataProperty ;
    rdfs:range xsd:float .
  slv:outerDiameter rdf:type owl:DatatypeProperty ;
    rdfs:subPropertyOf owl:topDataProperty ;
    rdfs:range xsd:float .
  :assembledWithSleeve rdf:type owl:ObjectProperty ;
    rdfs:subPropertyOf :assembledWith ;
    rdf:type owl:FunctionalProperty ;
    rdfs:domain :ShaftSection ;
    rdfs:range slv:Sleeve .

  :Chamfer rdf:type owl:Class .
  :chamferOffset rdf:type owl:DatatypeProperty ;
    rdfs:subPropertyOf owl:topDataProperty ;
    rdfs:range xsd:float .
  :chamferPosition rdf:type owl:DatatypeProperty ;
    rdfs:subPropertyOf owl:topDataProperty ;
    rdfs:range xsd:float .
  :chamferOn rdf:type owl:ObjectProperty ;
    rdfs:subPropertyOf owl:topObjectProperty ;
    rdfs:domain :Chamfer ;
    rdfs:range :ShaftSection .
  :hasChamfer rdf:type owl:ObjectProperty ;
    rdfs:subPropertyOf owl:topObjectProperty ;
    owl:inverseOf :chamferOn .
}
```

Listing 8: SPARQL code to insert the extended definitions for a five-section shaft.

The SWRL rules for the five-section shaft are listed in Listing 9. Only the rules for generating geometric models in Siemens NX KF are listed.

```
# S-16 (For extended case) Determine length and diameter of
section 1 & 3 according to assembled sleeves;
ShaftSection(?ss) ^ slv:Sleeve(?sleeve) ^ assembledWith(?ss, ?
sleeve)
^ slv:innerDiameter(?sleeve, ?Di) ^ slv:breathd(?sleeve, ?b)
-> length(?ss, ?b) ^ diameter(?ss, ?Di)

# S-17 (For extended case) Give a name to each chamfer according
to the design variables;
```

SWRL-based KBE

```

Chamfer(?c) ^ chamferOn(?c, ?ss) ^ :hasName(?ss, ?name_s) ^ :
  chamferOffset(?c, ?offset) ^ swrl:stringConcat(?name, "
  chmf_", ?name_s, "_", ?offset) -> :hasName(?c, ?name)

# S-18 (For extended case) Generate geometric models of chamfer
  represented in Siemens NX Knowledge Fusion;
Chamfer(?c) ^ :hasName(?c, ?name_c) ^ :chamferOffset(?c, ?offset) ^
  :chamferPosition(?c, ?cpo)
^ :chamferOn(?c, ?ss) ^ :hasName(?ss, ?name_s) ^ length(?ss, ?h) ^
  diameter(?ss, ?d) ^ :positionFromLeftmost(?ss, ?p)
^ swrl:multiply(?r, ?d, 0.5)
^ swrl:stringConcat(?cp_g, "point(", ?p, "_", ?r, ", 0)_+__", ?cpo, "*"
  vector(", ?h, "_", ?_0, "_0)")
^ swrl:stringConcat(?str1, "(child_", ?name_c, ":\n(class, _
  ug_single_offset_chamfer;\n")
^ swrl:stringConcat(?str2, "References, _
  ug_feature_askEdgeClosestToPoint(", ?name_s, ":", ?cp_g, ")
  ;\n")
^ swrl:stringConcat(?str3, "Offset1, "_", ?offset, "};\n")
^ swrl:stringConcat(?script, ?str1, ?str2, ?str3)
-> NX_KFCode(?c, ?script)

# S-19 (For extended case) Generate geometric models of the five-
  section shaft assembly represented in Siemens NX Knowledge
  Fusion, considering chamfers, distance between bearings,
  sleeves;
FiveSectionShaft(?s) ^ leftmostShaftSectionIs(?s, ?ss1) ^
  rightSideIs(?ss1, ?sssh) ^ rightSideIs(?sssh, ?ss2) ^
  rightSideIs(?ss2, ?ss3) ^ rightSideIs(?ss3, ?ss4) ^
  rightSideIs(?ss4, ?ss5)
^ hasKeyseat(?ss2, ?kw2) ^ hasKeyseat(?ss5, ?kw5) ^ chamferOn(?
  chmf1, ?ss1) ^ chamferOn(?chmf2, ?ss5)
^ NX_KFCode(?ss1, ?script1) ^ NX_KFCode(?ss2, ?script2) ^
  NX_KFCode(?ss3, ?script3) ^ NX_KFCode(?ss4, ?script4) ^
  NX_KFCode(?ss5, ?script5) ^ NX_KFCode(?sssh, ?script_sh)
^ NX_KFCode(?kw2, ?script_kw2) ^ NX_KFCode(?kw5, ?script_kw5)
  NX_KFCode(?chmf1, ?script_chmf1) NX_KFCode(?chmf2, ?
  script_chmf2)
^ swrl:stringConcat(?str1, "#!_NX/KF_4.0_\n_DefClass:_
  FiveSectionShaft_(ug_base_part);\n", ?script1, "\n", ?
  script_chmf1, "\n")
^ swrl:stringConcat(?str2, ?script_sh, "\n", ?script2, "\n", ?
  script_kw2, "\n", ?script3, "\n")
^ swrl:stringConcat(?str3, ?script4, "\n", ?script5, "\n", ?
  script_kw5, "\n", ?script_chmf2, "\n")
^ swrl:stringConcat(?script, ?str1, ?str2, ?str3)
-> NX_KFCode(?s, ?script)

```

Listing 9: The SWRL rules (S16-19) for generating NX KF code for a five-section shaft.

An instance of a five-section shaft (design case 3) can be instantiated using code in Listing 10.

```

prefix : <http://example/shaft#>
prefix bearing: <http://example/bearing#>
prefix gear: <http://example/gear#>
prefix kkw: <http://example/key_keyway#>
prefix mat: <http://example/material#>
prefix owl: <http://www.w3.org/2002/07/owl#>
prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
prefix scp: <http://example/shaftCoupling#>
prefix xsd: <http://www.w3.org/2001/XMLSchema#>
prefix xml: <http://www.w3.org/XML/1998/namespace>
prefix slv: <http://example/sleeve#>

INSERT DATA
{
  # create new instances
  dc:sleeve_sec1 rdf:type owl:NamedIndividual, slv:Sleeve;
  slv:breathd "15.0"^^xsd:float;

```

```

  slv:innerDiameter "20.0"^^xsd:float.
  dc:sleeve_sec4 rdf:type owl:NamedIndividual, slv:Sleeve;
  slv:breathd "15.0"^^xsd:float;
  slv:innerDiameter "20.0"^^xsd:float.
  dc:45steelForShaft rdf:type owl:NamedIndividual, mat:45steel.
  dc:keyseat_sec2 rdf:type owl:NamedIndividual, kkw:Keyseat_8_4_25.
  dc:keyseat_sec5 rdf:type owl:NamedIndividual, kkw:Keyseat_6_3_5
    _35.
  dc:shaftCp rdf:type owl:NamedIndividual, scp:YL1_16_42.
  dc:gear_demo rdf:type owl:NamedIndividual, gear:Gear;
  gear:boreLength "30.0"^^xsd:float;
  gear:boreDiameter "25.0"^^xsd:float.
  dc:shaftSection1 rdf:type owl:NamedIndividual, :ShaftSection.
  dc:shaftSection2 rdf:type owl:NamedIndividual, :ShaftSection.
  dc:shaftSection3 rdf:type owl:NamedIndividual, :ShaftSection;
  :length "28.0"^^xsd:float;
  :diameter "25.0"^^xsd:float.
  dc:shaftSection4 rdf:type owl:NamedIndividual, :ShaftSection.
  dc:shaftSection5 rdf:type owl:NamedIndividual, :ShaftSection.
  dc:shaftSectionShoulderOnSec2 rdf:type owl:NamedIndividual, :
    ShaftSectionShoulder.
  dc:chamfer_sec1 rdf:type owl:NamedIndividual, :Chamfer;
  :chamferPosition "0.0"^^xsd:float;
  :chamferOffset "2.0"^^xsd:float.
  dc:chamfer_sec5 rdf:type owl:NamedIndividual, :Chamfer;
  :chamferPosition "1.0"^^xsd:float;
  :chamferOffset "2.0"^^xsd:float.

# assert the relations between instances
dc:shaftSection1 :rightSideIs dc:shaftSectionShoulderOnSec2;
  :assembledWith dc:sleeve_sec1.
dc:shaftSectionShoulderOnSec2 :rightSideIs dc:shaftSection2.
dc:shaftSection2 :rightSideIs dc:shaftSection3;
  :hasShaftSectionShoulder dc:shaftSectionShoulderOnSec2;
  :assembledWith dc:gear_demo;
  :hasKeyseat dc:keyseat_sec2.
dc:shaftSection3 :rightSideIs dc:shaftSection4.
dc:shaftSection4 :assembledWith dc:sleeve_sec4;
  :rightSideIs dc:shaftSection5.
dc:shaftSection5 :assembledWith dc:shaftCp;
  :hasKeyseat dc:keyseat_sec5.
dc:chamfer_sec1 :chamferOn dc:shaftSection1.
dc:chamfer_sec5 :chamferOn dc:shaftSection5.

dc:shaft_demo rdf:type owl:NamedIndividual, :FiveSectionShaft;
  :hasShaftSection dc:shaftSection1,
  dc:shaftSection2,
  dc:shaftSection3,
  dc:shaftSection4,
  dc:shaftSection5;
  :leftmostShaftSectionIs dc:shaftSection1;
  :power "0.5"^^xsd:float ;
  scp:rotationSpeed "180.0"^^xsd:float ;
  :useMaterial dc:45steelForShaft.

```

assert the instances are different from each other to speed up reasoning

```

[] rdf:type owl:AllDifferent ;
  owl:distinctMembers (
  dc:45steelForShaft
  dc:sleeve_sec1
  dc:sleeve_sec4
  dc:shaftCp
  dc:gear_demo
  dc:keyseat_sec2
  dc:keyseat_sec5
  dc:shaftSection1
  dc:shaftSection2
  dc:shaftSection3
  dc:shaftSection4
  dc:shaftSection5
  dc:shaftSectionShoulderOnSec2

```

```

dc3:shaft_demo
dc3:chamfer_sec1
dc3:chamfer_sec5
) .
}

```

Listing 10: SPARQL code to insert a five-section shaft instance (design case 3).

CRediT authorship contribution statement

Liang Zhang: Conceptualization, Investigation, Methodology, Software, Validation, Writing – original draft. Andrei Lobov: Conceptualization, Funding acquisition, Project administration, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgment

This research project is part of the PhD studies at Norwegian University of Science and Technology in Trondheim, Norway.

References

- [1] G. La Rocca, Knowledge based engineering: Between ai and cad. review of a language based technology to support engineering design, *Advanced engineering informatics* 26 (2012) 159–179.
- [2] J. Sobieszcanski-Sobieski, A. Morris, M. Van Tooren, Multidisciplinary design optimization supported by knowledge based engineering, John Wiley & Sons, 2015.
- [3] S. Abdul-Ghafour, P. Ghodous, B. Shariat, E. Perna, F. Khosrowshahi, Semantic interoperability of knowledge in feature-based cad models, *Computer-Aided Design* 56 (2014) 45–57.
- [4] T. A. Tran, A. Lobov, Ontology-based model generation to support customizable kbe frameworks, *Procedia Manufacturing* 51 (2020) 1021–1026.
- [5] S. Dai, G. Zhao, Y. Yu, P. Zheng, Q. Bao, W. Wang, Ontology-based information modeling method for digital twin creation of as-fabricated machining parts, *Robotics and Computer-Integrated Manufacturing* 72 (2021) 102173.
- [6] R. K. Gupta, B. Gurumoorthy, Feature-based ontological framework for semantic interoperability in product development, *Advanced Engineering Informatics* 48 (2021) 101260.
- [7] R. Arista, X. Zheng, J. Lu, F. Mas, An ontology-based engineering system to support aircraft manufacturing system design, *Journal of Manufacturing Systems* 68 (2023) 270–288.
- [8] A. Raju Kulkarni, M. Hoogreef, G. La Rocca, Combining semantic web technologies and kbe to solve industrial mdo problems, in: 18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, 2017, p. 3823.
- [9] W. J. Verhagen, P. Bermell-Garcia, R. E. Van Dijk, R. Curran, A critical review of knowledge-based engineering: An identification of research challenges, *Advanced Engineering Informatics* 26 (2012) 5–15.
- [10] P. Kügler, F. Dworschak, B. Schleich, S. Wartzack, The evolution of knowledge-based engineering from a design research perspective: Literature review 2012–2021, *Advanced Engineering Informatics* 55 (2023) 101892.
- [11] S. Russell, P. Norvig, *Artificial intelligence: a modern approach*, 4th ed, University of California, Berkeley (2021).
- [12] E. Rigger, K. Shea, T. Stankovic, Task categorisation for identification of design automation opportunities, *Journal of Engineering Design* 29 (2018) 131–159.
- [13] L. Steels, Procedural attachment, *Proceedings of the Artificial Intelligence Memo* 543 (1979).
- [14] A. Ortner-Pichler, C. Landschützer, Integration of parametric modelling in web-based knowledge-based engineering applications, *Advanced Engineering Informatics* 51 (2022) 101492.
- [15] F. Mandorli, S. Borgo, P. Wiejak, From form features to semantic features in existing mcad: An ontological approach, *Advanced Engineering Informatics* 44 (2020) 101088.
- [16] C. Favi, F. Campi, M. Germani, M. Mandolini, Engineering knowledge formalization and proposition for informatics development towards a cad-integrated dfx system for product design, *Advanced Engineering Informatics* 51 (2022) 101537.
- [17] Z. Li, X. Zhou, W. Wang, G. Huang, Z. Tian, S. Huang, An ontology-based product design framework for manufacturability verification and knowledge reuse, *The International Journal of Advanced Manufacturing Technology* 99 (2018) 2121–2135.
- [18] S. K. Das, A. K. Swain, An ontology-based framework for decision support in assembly variant design, *Journal of Computing and Information Science in Engineering* 21 (2021) 021007.
- [19] B. Mehboob, C. Y. Chong, S. P. Lee, J. M. Y. Lim, Reusability affecting factors and software metrics for reusability: A systematic literature review, *Software: Practice and Experience* 51 (2021) 1416–1458.
- [20] ISO, *Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models*, Standard, International Organization for Standardization, Geneva, CH, 2011.
- [21] M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L. B. da Silva Santos, P. E. Bourne, et al., The fair guiding principles for scientific data management and stewardship, *Scientific data* 3 (2016) 1–9.
- [22] B. De Meester, T. Seymoens, A. Dimou, R. Verborgh, Implementation-independent function reuse, *Future Generation Computer Systems* 110 (2020) 946–959.
- [23] J. Peckham, F. Maryanski, Semantic data models, *ACM Computing Surveys (CSUR)* 20 (1988) 153–189.
- [24] T. Berners-Lee, J. Hendler, O. Lassila, The semantic web, *Scientific american* 284 (2001) 34–43.
- [25] J. Pokojski, Ł. Woźnicki, Concept of product complexity modeling in the development of a machine shaft, *Management and Production Engineering Review* (2017).
- [26] Y. Wen, H. Zhang, F. Li, J. Sun, Identifying and constructing elemental parts of shafts based on conditional random fields model, *Computer-Aided Design* 62 (2015) 10–19.
- [27] Q. Yang, C. Reidsema, Design information handling in a knowledge based intelligent design system, *Cybernetics and systems: an international journal* 38 (2007) 549–573.
- [28] M. A. Musen, The protégé project: a look back and a look forward, *AI matters* 1 (2015) 4–12.
- [29] Apache.org, Apache Jena Fuseki SPARQL server, <https://jena.apache.org/documentation/fuseki2/>, Accessed Nov-2023.
- [30] Miguel2617@Github.com, Integrate Openlet SWRL reasoner to Apache Jena Fuseki: Step by step guide, <https://gist.github.com/Miguel2617/eea8c79e209727233026ea47b72d2d65>, Accessed Nov-2023.

ISBN 978-82-326-8016-0 (printed ver.)
ISBN 978-82-326-8015-3 (electronic ver.)
ISSN 1503-8181 (printed ver.)
ISSN 2703-8084 (online ver.)



NTNU

Norwegian University of
Science and Technology