Aalto-yliopisto
Aalto-universitetet
Aalto University

Master’s Programme in Building Technology

KNOWLEDGE MANAGEMENT FOR BRIDGE DESIGN PROCESS WITH BUILDING BLOCK CONCEPT

Bao Gia Luong

Master’s thesis
2023

Aalto-yliopisto
Aalto-universitetet
Aalto University

Author Bao Gia Luong

Title of thesis Building Block Concept: Knowledge Management for Bridge Design Process

Programme MSc. Building Technology

Major Building Technology

Thesis supervisor Prof. Weiwei Lin

Thesis advisor(s) Sean LeCoultre

Collaborative partner Ramboll Finland Oy

Date 10.07.2023 **Number of pages** 118 **Language** English

Abstract

This thesis explores the concept of building blocks in the context of structural design within the Architecture, Engineering, And Construction (AEC) industry. The building block concept, developed by the Common prOject Delivery platform (CODA) team, refers to the pieces of design knowledge from previous projects in the form of digital files that can be assembled and reused in future structural bridge projects. These building blocks consist of multiple types of file: codes and standards, drawings, calculation reports, templates, tools, scripts, etc. The goal of the building block is to utilize the accumulated knowledge created by structural design engineers to increase productivity and design quality. With that in mind, the aims and objectives of the thesis include establishing the foundational theories of the building block concept in the knowledge management field, understanding the knowledge involved in the structural design process, developing a conceptual framework, and creating a proof-of-concept system.

Literature in knowledge base development and knowledge reuse in the context of engineering design showed that reusing design knowledge requires not only the knowledge content but also the context surrounding the knowledge. In other words, engineering knowledge is context-dependent and it would have no value without the context. Context can be measured through the knowledge dimensions of abstraction and granularity. Both dimensions determine how relevant and reusable knowledge is to different users. Despite the most reusable engineering knowledge being one with high abstraction and granularity, the building block concept requires documentation of knowledge at different levels of context and use case. In other words, no single level of context is adequate to cover all engineering knowledge. Therefore, multi-granularity on a case-by-case basis with a minimum level was chosen in the knowledge base.

Not only that, the contextual information must be retained through the retrieval method of knowledge. Such considerations must be taken into account when constructing the knowledge base to facilitate the reuse and retrieval of design knowledge.

The implementation of the building block concept was through the development of an ontological Expert System, capable of modeling the complex knowledge in structural design. The development focused on 2 stages: creating an informal model and translating that model to a machine-readable ontology. The informal model used the Icam DEFinition for Function Modeling (IDEF0) modeling method to represent the information requirements of the bridge design process. The building blocks are the Inputs, Controls, Outputs, and Mechanisms (ICOM) of a design process. Each design process has ICOMs that could be shared with other design processes, creating connections and relationships between them. This connection becomes the basis for the bridge design process assembly. The IDEF0 is then translated into the OWL language using Protégé. The proof-of-concept demonstrated the extraction of building blocks associated with design processes through Protégé's inferencing and querying capability with additional user inputs.

Keywords AEC, Bridge Design, Design Process, Building Block, Knowledge Management, Ontology, Expert System, Semantic Network, IDEF0, Protégé

Preface and acknowledgments

I want to thank Professor Weiwei Lin and my thesis advisor Sean LeCoultre for their good advice and guidance.

I also want to thank many friends and colleagues, too many to list, for their input, advice, and answers to my questions regarding data science and computer science. They made my life much easier and gave me the foundational knowledge required for this thesis.

And gratitude to my parents and my partner who supported me along the way.

Otaniemi, 29 March 2023

Bao Gia Luong

Table of Contents

Table of Figure

Abbreviations

AEC	Architecture, Engineering, And Construction
AI	Artificial Intelligence
BIM	Building Information Modeling
CBR	Case-Based Reasoning
CODA	Common prOject Delivery platform
CD	Computational Design
CAD	Computer-Aided Design
ES	Expert System
IDEF0	Icam DEFinition for Function Modeling
IFC	Industry Foundation Classes
IT	Information Technology
IPO	Input-Process-Output
ICOM	Inputs, Controls, Outputs, and Mechanisms
KB	Knowledge Base
KBS	Knowledge-based System
KM	Knowledge Management
KR	Knowledge Representation
LPG	Labeled Property Graph
OWA	Open World Assumption
RDF	Resource Description Framework
SWRL	Semantic Web Rule Language
SPOT	Single Point Of Truth
SPARQL	SPARQL Protocol And Rdf Query Language
OWL	Web Ontology Language
W3C	World Wide Web Consortium

1 Introduction

1.1 Background

The global market today requires companies to change rapidly and they struggle to maintain any competitive advantage. In this age of knowledge, the ability to acquire information and data and turn them into usable assets is the key to power. Knowledge is recognized as the most significant economic resource [1] and the most important competitive advantage for companies [2]. Intellectual capital makes the difference between competitions and it is not the exception in the AEC (Architecture, Engineering, And Construction) industry. The AEC industry, particularly the design area, is highly reliant on knowledge-intensive processes [3], meaning the services and products are heavily dependent on the delivery processes knowledge.

Design, being a highly knowledge-driven process [4], has impacts throughout the lifecycle of an AEC project. In the construction project, design encompasses various interconnected disciplines, including architectural, electrical, and structural design [5]. Of these, the structural design process is arguably the most important one, demanding extensive knowledge to design a secure and durable structure. Throughout different projects with changing scopes and specifications, the only things that stay relatively consistent are the engineering knowledge of the design processes.

A new structural design project is commonly built upon design knowledge from previous projects. There is an old saying that 80% of designs are redesigns [6]. It is hard to pin down the amount of design being reused since repetitiveness varies depending on many factors: disciplines, phase, nature of tasks, etc. However, there are a few studies to back up this number. For example, IDM packages were created to standardize product information exchange requirements in building design projects [7]. It was found that 69% of design packages were reused between the first and second case studies. In another research by [8], the ADePT modeling tool has incorporated over 90% of the necessary activities for defining a project's design. This aligns with the views of the report [9] about the repetitive nature of construction projects and the importance of standardizing processes and products. Consequently, the practice of redesigning previous designs to accommodate changing functional and performance requirements has become prevalent in structural design.

In structural design, reused design knowledge includes valuable insights derived from previously designed and constructed facilities, subsystems, or components along with the embedded knowledge and expertise acquired from these past projects [10]. This knowledge manifests in repeatable tasks, similar workflow patterns, and defined roles and responsibilities [11]. It serves as the foundation that makes one design better and more innovative than others, necessitating knowledge availability and accessibility to designers when needed. Leveraging reused knowledge enables more comprehensive design processes that can effectively guide future projects, minimizing the risk of repetitive errors. Without awareness of existing knowledge within the organization, designers may waste significant time searching for information that already exists, do a lot of re-inventing of the wheel, and repeat past mistakes.

Despite recognizing the importance of reusing knowledge, reuse often fails due to knowledge not being captured or the captured knowledge being decontextualized, rendering it non-reusable [10]. Unlike in the mechanical industry where old designs and even designing processes are configured into new designs as a normal practice, most design processes from the AEC industry are project-oriented, context-specific, and experiential [12]. The objectives, specifications, and the nature of constraints of each project make them unique, temporary, non-routine, and non-repetitive, and hard to reuse previous knowledge with the existing retrieval methods. Moreover, project knowledge is intricately tied to its creator and the specific context in which it was generated. Consequently, although construction companies accumulate and store design knowledge, the lack of formal mechanisms, both from information technology (IT) and organizational perspective, to discover, explore, and retrieve reusable knowledge poses challenges for others seeking to harness this valuable asset. It is within this context that the concept of the building block emerged.

1.2 Building Block Concept

1.2.1 Existing Framework

The need to reuse knowledge in structural design has led to the conception of the building block concept, which was conceived by the CODA (Common prOject Delivery plAtform) team from Ramboll Finland, specifically for bridge projects. The building block, as they defined it, is the smallest viable size of a data item. It refers to common design values, folder structures, tools, scripts, etc. (see Figure 1) used for the creation of a SPOT (Single Point Of Truth) database, a database containing information about all project-related matters located in one place and automating all information to design software.

The goal is to create building blocks made of files containing previous or standardized design processes and knowledge. They are connected, can be retrieved and assembled from a repository to be reused, and built into a centralized complete workflow for a new design project, similar to a set of Lego blocks being assembled into much bigger models.

There is also a requirement for information to be standardized so that a seamless connection between all tools can be done (i.e. the input data and the output data are understood by all the tools). This is a problem addressed using the SPOT database. It acts as a project-level database that stores all relevant project information in a human and machine-readable format in a single location. It is assumed that building blocks from a central database are extracted and assembled to create a SPOT at the beginning of a new project. This is especially useful at the initial design stage by reducing the amount of work and errors in transferring information throughout the project and handling data for computation design. Since the SPOT database would handle the interoperability of building blocks, this requirement was not addressed in the context of this thesis.

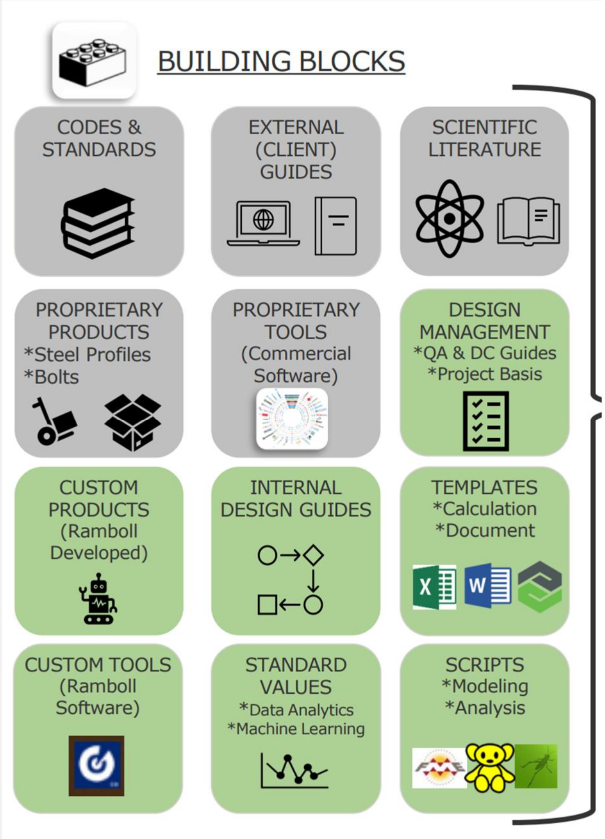
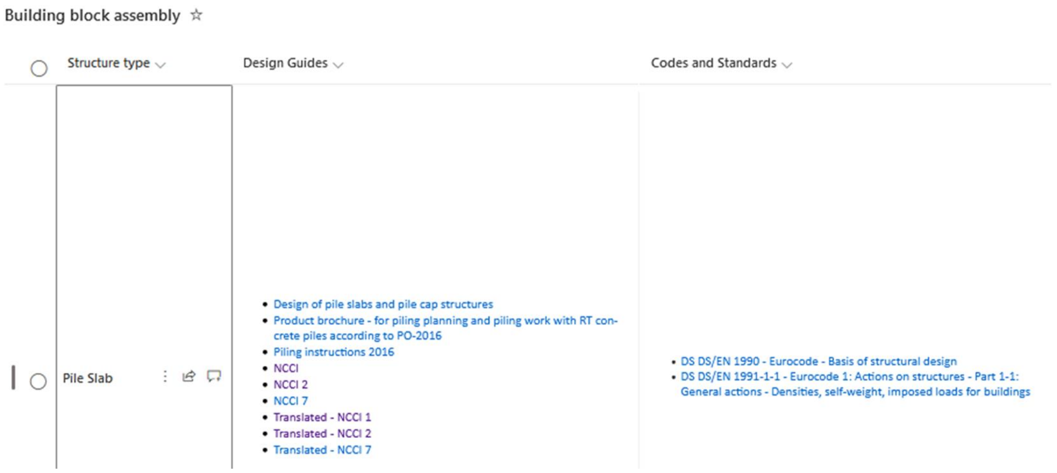

BUILDING BLOCKS
CODES & STANDARDS
EXTERNAL (CLIENT) GUIDES
SCIENTIFIC LITERATURE
PROPRIETARY PRODUCTS *Steel Profiles *Bolts
PROPRIETARY TOOLS (Commercial Software)
DESIGN MANAGEMENT *QA & DC Guides *Project Basis
CUSTOM PRODUCTS (Ramboll Developed)
INTERNAL DESIGN GUIDES
TEMPLATES *Calculation *Document
CUSTOM TOOLS (Ramboll Software)
STANDARD VALUES *Data Analytics *Machine Learning
SCRIPTS *Modeling *Analysis

Figure 1. File types that could be transformed into building blocks

For now, all files, documents, tools, etc. that could be considered a building block are stored in a central repository. The building blocks are assembled as lists of recommended documents, drawings, templates, scripts, links, etc. that have been used in previous projects (see Figure 2). They are categorized by structural type and document purpose. Structural type groups files by the type of structural system that the files are used in (e.g. pile slab, noise barrier, bridge type, etc.). And document purpose separates files by their use context: design guides, codes and standards, technical data, calculation template, document template, cook book (project-specific guide and knowledge), typical drawings, archive (previous projects), grasshopper scripts (automation scripts for geometry, analysis, verification), SOFiSTiK scripts. Each design process of structural elements has different lists with download links of relevant files.



Building block assembly ☆

○	Structure type	Design Guides	Codes and Standards
○	Pile Slab	• Design of pile slabs and pile cap structures • Product brochure - for piling planning and piling work with RT concrete piles according to PO-2016 • Piling instructions 2016 • NCCI • NCCI 2 • NCCI 7 • Translated - NCCI 1 • Translated - NCCI 2 • Translated - NCCI 7	• DS DS/EN 1990 - Eurocode - Basis of structural design • DS DS/EN 1991-1-1 - Eurocode 1: Actions on structures - Part 1-1: General actions - Densities, self-weight, imposed loads for buildings

Figure 2. Snippet of building block lists for Pile Slab design

1.2.2 Vision

What a building block represents is a design process or design task, not structural objects and elements, comprising the design process. Through the context of the design process, one building block is connected to another. The files are geometrical and non-geometrical knowledge that act as different aspects of the design activities such as inputs, constraints, rules, tools, and content of the process.

The building block workflow is envisioned as follows: At the beginning of a new project, existing building blocks created from previous projects and experience are stored in a central repository. All files of building blocks have metadata attached to them, and the selection of building blocks happens via filtering. Relevant blocks are extracted from the central repository to be assembled and saved in the new project folder and the new SPOT database. Data stored in the SPOT is appended and queried throughout the project. Changes and updates to the building blocks are made as necessary, or new blocks can be created. New building blocks that weren't available in the central repository are built on a project-by-project basis. At the end of the project, new and updated building blocks can be brought back into the central repository after being refined.

It is important to be aware that the building block concept is still very much in an immature stage, this is only the attempt of the thesis author to pin down the unrefined ideas and vague assumptions from CODA into something more concrete to work on. With the nature of the concept and the requirement checklist established, the objective now is to explore how to build the building block concept.

1.3 Aims and Objectives

The aim of this thesis was to lay the foundation for the building block concept through exploration and to put the concept in the appropriate academic context. With this, future contributors to the building block can develop a structured approach for managing and sharing bridge design knowledge in the context of knowledge management. At the same time, effort was made to create a proof-of-concept product using the laid theoretical foundation. This will facilitate the future development of the knowledge management (KM) system through lessons learned from the created system. The following objectives were determined:

- Establishing the basic foundations of the building block concept, its definitions, and existing methodology.
- Understand the Structural and Bridge design process specifically and identify the knowledge involved in it and the existing approaches for managing this knowledge.
- To review concepts related to KM and to identify the supporting tools that can be applied to the building block.
- To explore the potential of KM for structural design knowledge
- To develop a conceptual framework for the building blocks and to support its use by detailed methodologies for managing structural design knowledge.
- To capture partial aspects of the developed methodologies into a proof-of-concept system.

1.4 Thesis Structure

With the goals determined, this thesis would be structured around the realization of the building block concept and the end target of creating a proof-of-concept. The thesis would explore the building block concepts by attempting to meet the requirements of and come closer to the vision for it. The work would be divided into 8 chapters:

- Chapter 1 begins with an introduction to the building block concepts.

- Chapter 2 and 3 laid down the foundational concepts required to understand the nature of building blocks as knowledge in the bridge structure design process.

- Chapter 4 explores the insights and requirements that lead to the successful reuse of design knowledge.

- Chapter 5 outlines the method adopted in this thesis for the building block concept and explains the rationale of this method.

- Chapter 6 and 7 is the implementation of the chosen method, showcasing the potential and implications of the chosen method for the building block concept.

- Chapter 8 summarizes the results and major findings, and the extended future works required to realize the full building block concept.

2 Structural Designing Concepts

2.1 Productivity Issue of The AEC Industry

The global AEC industry has long struggled with low productivity compared to the economy development and other sectors. For example, retail witnessed a shift half a century ago from small, family-owned businesses to large-scale modern retailers with global supply chains, digitalized distribution systems and intelligent customer data gathering [13]. Likewise, the manufacturing sector has been completely changed through the adoption of lean principles and extensive automation. In comparison, the construction sector innovates at a glacial pace. This is evidently shown in many ways: being one of the least digitized industries worldwide or averaging merely 1 percent in annual labor-productivity growth compared to 3.6 percent growth in manufacturing. The productivity issue is multifaceted, complex and should be tackled from several areas such as management, construction, designing, etc. [14] In the context of this thesis, the building block concept was directly related to issues of the design process and digitalization and thus, they were the main perspectives considered.

According to [14], three productivity issues are closely related to design process and digitalization. First is the highly fragmented nature of the industry. Large and complex projects are performed by many stakeholders, such as architect, engineer, and contractor, with different requirements and interests but driven toward a single goal. Moreover, the competitiveness of the project procurement, where the lowest bids are frequently chosen, does not foster a knowledge transfer culture among companies in the construction industry.

The second reason is the construction of predominantly unique and custom-built products, in contrast to the mass production model employed in manufacturing [15]. This significantly limits the optimization from repeatable work on the product. However, the engineering processes in the AEC industry are still repetitive and thus, can still be changed to improve productivity as it did in manufacturing.

Thirdly, the AEC system as a whole is extremely complex, not only the product but also the processes and organizational aspects [16], [17]. The design processes are almost entirely wicked problems, characterized by their intricate and interconnected nature that limit attempts to find a single optimal, risk-free solution [18]. Assumptions and irrational information flows happen regularly as a natural part of such a complex system compared to relatively stable and repetitive industries like manufacturing [19], [20].

2.2 Structural Design Characteristics

2.2.1 Structural Design and Bridge Design

To understand why the building block concept is needed, it is important to understand structural design and its inner working process. Structural design, a specialized part of civil engineering, is a complex process that demands extensive expertise [21]. The objective is to create the "bones and muscles" for a structure, designing a structure that can endure environmental stresses and pressures while maintaining safety, stability, and security throughout its lifespan [22]. The fundamental approach is searching for a way to transfer loads within a given space to a support or foundation [21]. This involves making decisions regarding the shape and size of the structure and its components, aiming to ensure that it is suitable for its intended purpose and meets the functional specifications outlined by design rules and criteria [23], [24].

Structural design is the one of most important factors in deciding the end product quality. In typical construction projects, owners communicate their engineering requirements to design companies, who then establish construction requirements [25]. The construction company then erects the project according to the specified construction requirements from the design company. It is clear that the competence and productivity of structural design engineers play a pivotal role in delivering a desired product required by the owners.

In the context of this thesis, it was specifically the designing of bridge, one of the two most prominent products of civil engineering along with building, that was focused on. Bridge design shares many fundamental similarities with building design, similar enough that many conclusions can be applied to for one another. Both bridge design and building design follow the same fundamental understanding of structural design: analyzing load paths, designing to standards and codes, and checking for design feasibility. Both follow typical design phases: Prior to the design process, concept structure, and alternatives are developed, possibly with a scaled model and 3D simulation models. Parameters such as structural performance and estimated costs are considered, leading to discussions with clients, managing agencies, and local government bodies. Once a consensus on costs and the concept is reached, the following stages of preliminary design, detail design, and construction design can commence.

However, distinct differences exist when designing bridge versus designing building due to shape, scale, and detailing level. Unlike vertical architecture of buildings, bridges are linear horizontal structures spanning long distance and thus, requires heavy methods of construction [26]. The layout of a bridge may appear basic and straightforward but deeper attention to detailing is needed compared to building design. In fact, buildings often have more repetitive designs and standards and codes to follow, while bridge design demands bespoke and unique solutions. Although both products involve multiple disciplines, bridge relies heavily on the topography and it introduces many variations into the surrounding landscape and therefore, requires expertise in hydraulics, geotechnics, landscaping, economics, and even sociopolitical aspects [27]. Finally, the structural design of bridges is led by structural engineering resulting in easier coordination, whereas, in buildings, architects take the lead [5].

2.2.2 Structural Design Process

Structural design can be described from two distinct perspectives [28]:

- Rational activity: It assumes that design work is a rational, methodical, problem-solving activity, often involving information processing.
- Interactive inquiry: It sees design work as a creative, artistic, and interactive inquiry deeply rooted in social, cultural, and psychological considerations.

In other words, designers cannot simply follow predefined rules or procedures but are encouraged to be thoughtful and examine their own work to adapt the design in response to the complexity and nuances of the design context. While the rational activity perspective oversimplifies the complexities and messiness of the creative side of the design process, the interactive inquiry perspective can become chaotic and difficult to comprehend and manage. There needs to be a balanced understanding of both views in structural design for creating the most value for the client.

Indeed, design work can be investigative, creative, rational, or decision-oriented in nature [29]. This is well-reflected in the steps performed in structural design. [19] proposes a design process comprising three iterative steps: analysis, synthesis, and evaluation. Analysis and synthesis steps are similar to the procedure of system engineering, while the evaluation step is important for learning and improvements to be incorporated in the next iterative cycle. The first two basic steps can be further expanded into 5 essential steps to design any structure (1) modeling, (2) load analysis, (3) structural analysis, (4) structural design and (5) detailing, each of which encompasses various activities: decision-making, fact-finding, and information gathering, topic distillation, etc.

Due to the repetitive cycles of the above steps, structural design is called a process of progressive refinement, through which the design solution reaches a level of detail and maturity that aligns with the established design requirements and constraints [30]. These requirements and constraints are intertwined by various uncertainties, including changes in client preferences, environmental factors, assumed live and wind loads, limited investigation of soil conditions, and unpredictable seismic events [5]. Additional influence factors include human intelligence factors, such as experience and engineering judgment [23], [24], as well as regulatory guidelines such as codes of practice and design standards. This ill-defined and complex problem is present within the product, processes, and organization of the structural design process and the interdependencies between each of them [16]. Avoiding complexity is not possible. Due to this wicked nature of the structural design process, incremental progress is made after each design cycle to arrive at a finalized design that meets the accepted criteria [31]. The complex nature of structural design process is demonstrated in more detail in Chapter 6.

2.2.3 Design Phase and Activity

In order to effectively navigate the design environment, some form of structure is necessary [28]. This often manifests in the form of the phase model currently adopted in practice. Structural design comprises many phases, comprising a multitude of interconnected activities [24], [32]. [33] divided bridge design into 4 phases: conceptual design, preliminary design, detailed design, and construction design. In the beginning, the conceptual design phase produces multiple feasible bridge concepts and one or more concepts are selected for further evaluation. Next, in the preliminary design phase, the most suitable concept from the proposed ones is chosen by assessing its feasibility and refining the cost estimates. The detailed design phase focuses on finalizing the details of the chosen bridge concept to create a comprehensive document that can be used for tendering and construction. Finally, the construction design phase aims to provide a procedural guide for the actual construction process of the bridge. Not all four phases exist in every bridge project as it depends on the scope of the design contract. For example, it is common for the bridge design to start at the preliminary or detailed design since a concept has already been chosen by the municipality.

Creativity and rational viewpoint can be used as a spectrum to gauge the activities of the design phases. At one end is the concept design relying heavily on the experience and expertise of engineers, known as "tacit knowledge", and the other end is the detailed design relying on the application of mathematical formulas and codes of practice, referred to as "explicit knowledge". More detail about tacit and explicit knowledge is discussed in Section 3.1.2. This does not mean that concept design is exclusively creative work without problem-solving activity. It is simply useful to categorize design processes based on the dominant types of design activities so it can be appropriately addressed in this thesis.

As stated in [5], the concept design stage is considered the most creative phase [34]because feasible structural concepts are created based on rules of thumb and experience [32], [35]. The designers attempt to meet the requirements of the proposed structure and thus, the decisions that are made here greatly influence the remaining phases [23], [24], [32]. While there is no one approach to concept design, there are common activities [24], [35], [36]:

- Generating design alternatives: Designers rely on experience in previous projects to brainstorm design ideas. The alternative bridge designs can vary in structural configurations, component sizes, and material selections e.g. steel, concrete, timber, etc.

- Evaluating the alternatives' feasibility: Designers evaluate with simple calculations and past experiences. The evaluation is based on both internal (imposed by the designer's choice or by working with particular materials or technologies) and external (clients' needs, technology, and the construction process) constraints

- Selecting the ideal solution: Designers select the best solution that meets the project's requirements based on engineering judgment and foresight. The selected solution will be used in the detailed design.

In subsequent stages, such as detailed design, the goal is to fulfill the constraints set by the concept design. Detailed design activities are typically handled by less experienced engineers, whereas concept design responsibilities are often handled by senior engineers. Detailed design is a dynamic and iterative process that typically involves multiple cycles, influenced by various factors, including the degree of uncertainties in the project and the availability of information and expertise. The common activities in detailed design are [24], [35], [36]:

- Detailed analysis: Actions and loads (deflections, vibrations, shearing stresses, bending moments, etc.) are calculated and the design controlling loads are identified.

- Sizing and proportioning: Designers decide the dimensions of structural elements using design codes and standards. This includes the detailing such as rebars of reinforcement concrete.

- Design check: Designers check whether the whole structure and its elements meet the constraints (requirements for safety, engineering and physical laws, or other local constraints)

2.3 Computational Design

2.3.1 Definition

A motivation in the building block concept creation is the adoption of the Computational Design (CD) method, which has hugely affected the AEC industry as a whole in recent years. In traditional structural design, designers rely on computer-aided design (CAD) programs, created by a third party [37]. Consequently, the designer's creativity is partially influenced by the software creator's opinion and the design workflow is constrained by the chosen software's limitations and thus, the number of design alternatives is restricted [38]. A true master of design, who possesses a deep understanding of design, should be allowed to create customized engineering tools according to their desire. The method has its origin in computer science, mathematics, and geometry, and has been adapted to solve design problems. That is where CD comes into play.

CD is a design method that combines algorithms, parameters, and computational thinking [39] to produce design solutions using the power of computer processing. When the design process is broken down into measurable steps, it becomes the framework for algorithms to design automatically. In other words, it digitalizes and formulizes engineering knowledge into scripts and applications that automate design tasks or even a whole process. The end goal is data-driven design, meaning design decisions are driven by strong input data in combination with intuition and personal preference to produce reliable results [37].

CD is also referred to by the various name of its categories: Algorithmic design, Generative design, and Parametric design [37], [40]. Parametric design is a design approach based on the use of parameters and rules to create a design solution that is easily modified. Generative design is a design approach that uses algorithms to generate multiple design alternatives for evaluation. Algorithmic design is a Generative design approach where the algorithm is directed and limited by specific desired outcomes.

The CD process involves decomposition of a design, pattern recognition, abstraction, and the utilization of algorithms [41]. Designers are required to systematically break down their design process into quantifiable steps to implement in CD [37]. Next, similarities, trends, and patterns within the design are identified. These are then abstracted, focusing on the essential elements and discarding the unnecessary, and then translated into computer language. The algorithm takes project-specific parameters to solve design problems, such as creation, fabrication, interaction, and analysis, at a faster pace with a wider range of alternatives. Once the initial programming is completed, the design process becomes dynamic and repeatable, fostering efficiency and flexibility.

It needs to be emphasized the difference between CAD and CD [42]. CAD refers to the use of computer as a technique (computer-aided or digitalized) while CD refers to a thinking approach to design. In CAD, conventional design processes are assisted by computational tools, making the traditional existing process faster. In CD, the entire process must be changed to reflect the way of thinking. CD surpasses mere automation of traditional design practices or repetitive tasks; it involves a fundamental shift from design thinking to computational thinking [39] in designer and design methodologies.

Both traditional design and CD require the utilization of multiple software applications to effectively convey the multimedia aspects of a design. For example, designers often rely on various analysis programs offered by competing software developers [43]. After analysis, the designs are documented in CAD drafting software to generate drawings that serve as the basis for contractors' plans during the construction phase. Additional software tools to aid in the review process are also employed, which enable error detection and ensure accurate data transfer. With the addition of CD, the reliance on multiple software becomes even more crucial. As demonstrated in the Virginia Tech Innovation Campus project in Alexandria, VA., a diverse range of software was employed, including Grasshopper for parametric modeling, Ladybug and Honeybee plugins for exporting into analysis software Energy Plus and Radiance for, Colibri plugin for simulating design permutations, a customized database environment for storing simulation results, PowerBi for data exploration and visualization, and machine learning algorithms to power real-time performance predictions [44]. The result is a custom-made application that can adapt to real-time decisions and unforeseen inquiries. Another benefit is the potential to reuse these models in future projects at no additional costs, as long as they address common design challenges rather than specific options. This is one of the reasons why the building block concept is created.

2.3.2 Effects and Application

The AEC industry and the structural design field are transforming thanks to the rise of computational design. This shift has been accelerated in part by the widespread adoption of user-friendly visual programming tools like Grasshopper for Rhinoceros and Tekla, and Dynamo for Revit [43]. These visual coding platforms liberate companies from the constraints of structural modeling software, enabling them to tailor their software to their unique workflows instead of conforming to predefined limitations. Therefore, a plethora of innovative design techniques have been introduced, empowering users to create powerful computer scripts that can streamline workflows or realize entirely new modes of design integration [44]. Engineering firms worldwide are figuring out the implications and impacts of these advancements on typical design practices to effectively incorporate them into their design processes.

CD has proven to be highly valuable for early design exploration in the conceptual design phase, where design options are constantly evolving [45]. The exploration of design alternatives can be done at a rapid pace within a defined solution space [44]. By utilizing parametric modeling and iterative over a range of variables, such as volume, cost, and performance, CD facilitates the identification of optimal solutions for a given problem. For instance, CD can be employed to optimize the arrangement of bridge trusses, reducing material usage and cost. Even the previously slow analysis and optimization process is now feasible during the conceptual design phase. By incorporating automation and data analysis techniques early in the design process, structural engineers now have access to timely and meaningful data. The analysis data enables informed decision-making during conceptual design, similar to an experienced expert drawing upon years of knowledge and expertise.

Although CD is the most useful in the conceptual design phase, designers should not go back to traditional workflows in other phases. CD offers an opportunity for the integration of computational methods and the development of entirely new workflows [46]. It can automate calculation, repetitive and manual work, streamline traditional design processes, and facilitate the reuse of previous design solutions. Through CD, the iteration of structure and components can be done swiftly no matter the phase. The parametric models allow for generating volumes, analyzing them, receiving graphical feedback, and helping the users make more informed modifications through parameters and controlling mechanisms in any way they wish. This ensures on-time completion even after numerous changes [37]. Another usage is in custom-developed CD scripts which allow for checking at each design step [43]. This means QAQC is repeatedly done without diverting more resources. Furthermore, the combination of visual programming tools and artificial intelligence further empowers designers to surpass human capabilities and test designs under multiple scenarios [37].

3 Knowledge Management Concepts

3.1 Knowledge

3.1.1 Definition

It is difficult to determine what sort of system the building block can be built on without some insight into its nature. The key lies in the connection between the existing form of building blocks and their purpose. These documents and files direct the user throughout the design process to get the desirable deliverable from the inputting data and information. They are procedures, rules, experience and lessons learned, custom tools and documents, standards, and technical understanding created by humans to be used in the design process, also known as knowledge. A collection of data, information, and knowledge organized and stored in a way that makes it easily accessible and retrievable is called a Knowledge Base (KB) [47]. The KB is used as a repository of resources and information to support humans in decision-making, problem-solving, and learning. More details about KB technology are discussed in Section 3.3.2.

In order to move forward to the other concepts such as KB and KM, the definition of knowledge needed to be laid down as the basis of this thesis. Knowledge is defined by two perception groups, which revolve around the relation of knowledge to information and data. The first group sees knowledge as an evolved stage of information and data in the information hierarchy, see Figure 3. According to [48], data is “unorganized and unprocessed facts”, while information refers to “an aggregation of processed data” and knowledge is defined as “evaluated and organized information”. Another definition by [49] describes “Knowledge is the {ability, skill, expertise} to {manipulate, transform, create} {data, information, ideas} to {perform skillfully, make decisions, solve problems}”. This is to show that if the building blocks are engineering knowledge, it is not enough to provide them in a raw and unprocessed form. The building block must incorporate the frameworks, perspective, usage understanding, conclusion drawn, connections, etc. by the creator of a building block in relation to relevant design knowledge. Only then the building block stop being just files containing data and information start to become knowledge. In contrast, the second group defines data and information as the product of knowledge. As said by [50], knowledge should not be treated as a separate entity from the knower. They criticized the tendency to equate knowledge with information and view it as a static resource rather than a dynamic process. A quote from [51] explains this view concisely: “Information is converted to knowledge once it is processed in the mind of individuals and knowledge becomes information once it is articulated”. Despite contradicting views, common ground can be established using the definition of [52]: knowledge is a set of “insights, experiences, and procedures” that are “correct and true” to control the “thoughts, behaviors, and communications of people”. This is also supported by the Ancient Greek philosopher Plato’s theory on knowledge: knowledge is justified true belief, that is to say, knowledge can be justified and is believed to be true.

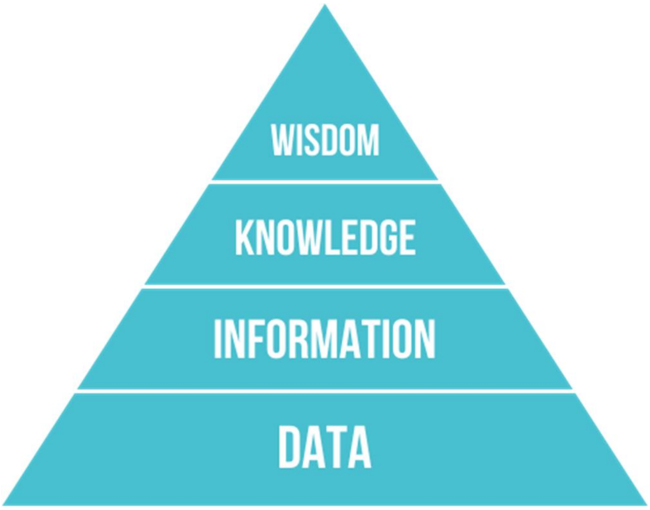


Figure 3. Each level above adds value on top of the level below allowing more complicated inquiries. Source: [53]

3.1.2 Knowledge Type

In this thesis, the concept of knowledge as defined by [54] was examined. They divided knowledge between explicit and tacit knowledge. Explicit knowledge refers to information that has been codified (articulated in a record or document) in a physical or digital form, allowing for dissemination and sharing among users. On the other hand, tacit knowledge encompasses the personal knowledge (including experience and know-how) that individuals possess but have not been explicitly articulated. To effectively share tacit knowledge, it must be converted into explicit knowledge. Numerous studies have suggested that approximately 80 percent of important knowledge is tacit knowledge [55]. Failure to share the tacit knowledge will result in the loss of project insights, leading to organizational amnesia [56]

However, [57] agrees with the second perspective of knowledge and challenges the notion of tacit and explicit knowledge, asserting that knowledge cannot be codified since it can only reside within individuals and codified knowledge should be seen as information. And yet, codified knowledge may be considered as knowledge and not information by individuals with sufficient tacit knowledge in the respective field. For example, a construction drawing is considered as codified knowledge on how to build a structure to civil engineers and as information to non-engineers. This perspective aligns with theories that view companies as knowledge-based organizations, where members consciously or unconsciously interpret and apply information based on their accumulated experience and skills [54], [55]. Therefore, for its relevance to structural design activities and the context-specific nature of the domain, this thesis chose to align with the first view of knowledge and referred to codified knowledge as knowledge.

Knowledge can be transformed from one type to another through four modes of knowledge conversion [54], see Figure 4. Tacit knowledge possessed by one individual can be converted into another person's tacit knowledge through socialization, and then turn into explicit knowledge through externalization. Socialization occurs during face-to-face interactions, while externalization involves the codification of knowledge. Capturing tacit knowledge through externalization is the most significant challenge in KM. Conversely, explicit knowledge can be converted into other forms of explicit knowledge through combination, and then turn into tacit knowledge through internalization. Combination synthesizes two sources of codified knowledge to generate new knowledge and Internalization is performed by individuals when they read and comprehend codified written knowledge. Managing explicit knowledge, which is already codified in various forms, is easier than tacit knowledge and requires fewer resources.

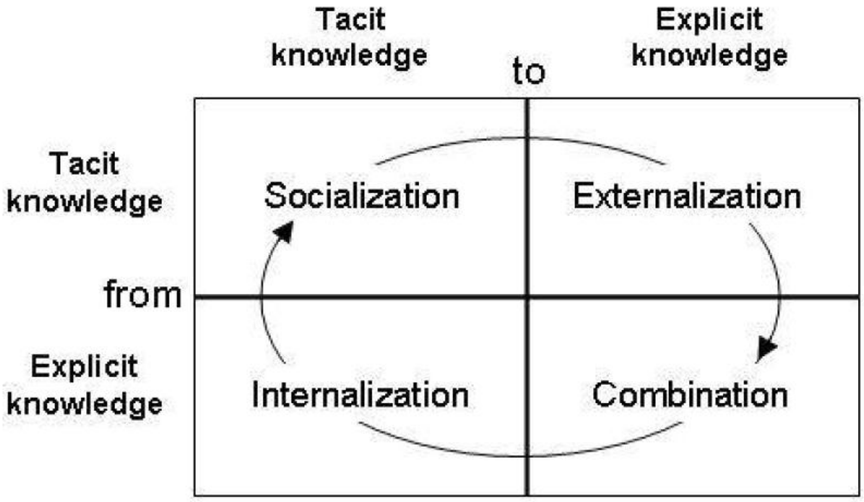


Figure 4. Modes of knowledge conversion. Source: [54]

As expressed by [58]: “we know more than we can tell”, it is important to acknowledge that not all project knowledge can be codified, as certain aspects of tacit knowledge remain difficult to articulate explicitly and must be transferred to other individuals through socialization. This aligns with two distinct approaches to knowledge reuse [59]. The first approach emphasizes capturing and storing knowledge in an external repository for widespread sharing and reuse among employees. The second, more common approach recognizes the significance of valuable tacit and contextual knowledge inside individuals within the organization. Companies adopting this approach use technology to facilitate communication and cultivate knowledge transfer through social networks. This thesis only focused on the capturing, managing, and retrieving explicit knowledge since the building blocks were codified knowledge.

3.2 Knowledge Management

3.2.1 Benefits of KM

This brings us to the following question: why is it necessary to manage knowledge and how does it differ from information or data management? While data and information can be easily stored and obtained from company repositories in this era of communication, the generation of knowledge requires both time and extensive proficiency in the domain of interest. The process of knowledge generation often takes place within an individual's mind and is not externalized. Consequently, as valuable as knowledge is to an organization, it is challenging to identify when new knowledge is generated and subsequently documented [60]. KM is crucial as it ensures that relevant knowledge is accessible when needed, which prevents the loss of valuable knowledge. The promised benefits and successful implementations of KM increase the interest in it [5].

Furthermore, sharing knowledge increases its values and impacts on organization's success [61]. According to [62], In the modern economy, a firm's competitive advantage lies in its ability to effectively harness its knowledge resources and can criticality determine the business success and even business survival. In contrast, organizations failing to implement KM strategies risk falling behind competitors who embrace the value of knowledge, ultimately endangering business longevity [61].

According to [63], there are many compelling reasons for companies to need KM. First of all, it facilitates the acceleration and accessibility of information and expertise, and supports locating resources and individuals possessing the required knowledge without wasting time. This improves the overall efficiency and productivity of company's workforce [64]. Additionally, KM minimizes trial and error, enabling organizations to learn from past experiences and exploit existing knowledge assets. By reusing knowledge capital in different areas and leveraging it for improvement and innovation, such as using lessons learn from previous bridge projects to design a new bridge, organizations can derive significant benefits.

Next, KM improves the decision-making process by enhancing the quality of information obtained from employees and expediting the speed of decision-making. Collaboration platforms also enable diverse perspectives and experiences, hence improving the quality of decisions and desired outcomes. Additionally, KM fosters the development of essential competencies and skills while eliminating outdated knowledge, ensuring the preservation of key knowledge and competencies within the company [65].

Finally, KM drives innovation and cultural transformation in organizations [66]. In today's dynamic business landscape, innovation has become vital for adapting to change effectively. The rapid pace of innovation, fueled by advancing technology, shorter product lifecycles, and increased new product development, has reshaped global economic growth. Knowledge is at the core of the innovation process, as it enables the transformation of general knowledge into specific knowledge, leading to the creation of valuable products, services, and processes [67]. This is shown the clearest in the inception of the building block concept and its parent project SPOT. Organizations recognize the potential economic benefits and therefore show a growing interest in KM [68].

3.2.2 KM Process

A thorough examination of the existing literature reveals the absence of a universally accepted definition for KM. Different interpretations, even conflicting ones, of KM exist due to variations in individuals' experiences, backgrounds, and organizational contexts [69]. In this thesis, four major processes of KM were creation, capture, sharing, and retrieval. Finding the appropriate knowledge capture and retrieval methods directly fulfilled the main objectives. The available methods are explored and selected in Section 4.4, considerations of the chosen method are addressed in Section 4.5 and the proof-of-concept product based on the considerations is described in Chapter 7.

- Creation: In the context of this thesis, knowledge is created during the knowledge-intense structural design process [59]. Created knowledge can be completely new content or a replacement of existing content, encompassing both explicit and tacit knowledge [70]. [54] emphasize that the generation of new knowledge, whether implicit or explicit, arises through the dynamic interaction between individuals, groups, and organizations.

- Capture: It is the generation of new valuable knowledge from within an individual's mind that needs to be captured and codified. The selection of appropriate storing methods must also be taken into account. The building blocks are the direct result of the knowledge capture process. Only a fraction of generated knowledge finds its way into project documentation and the failure to capture knowledge results in knowledge loss [71]. Knowledge capture is not a one-time project but rather an ongoing program that invests in the company’s future [3].

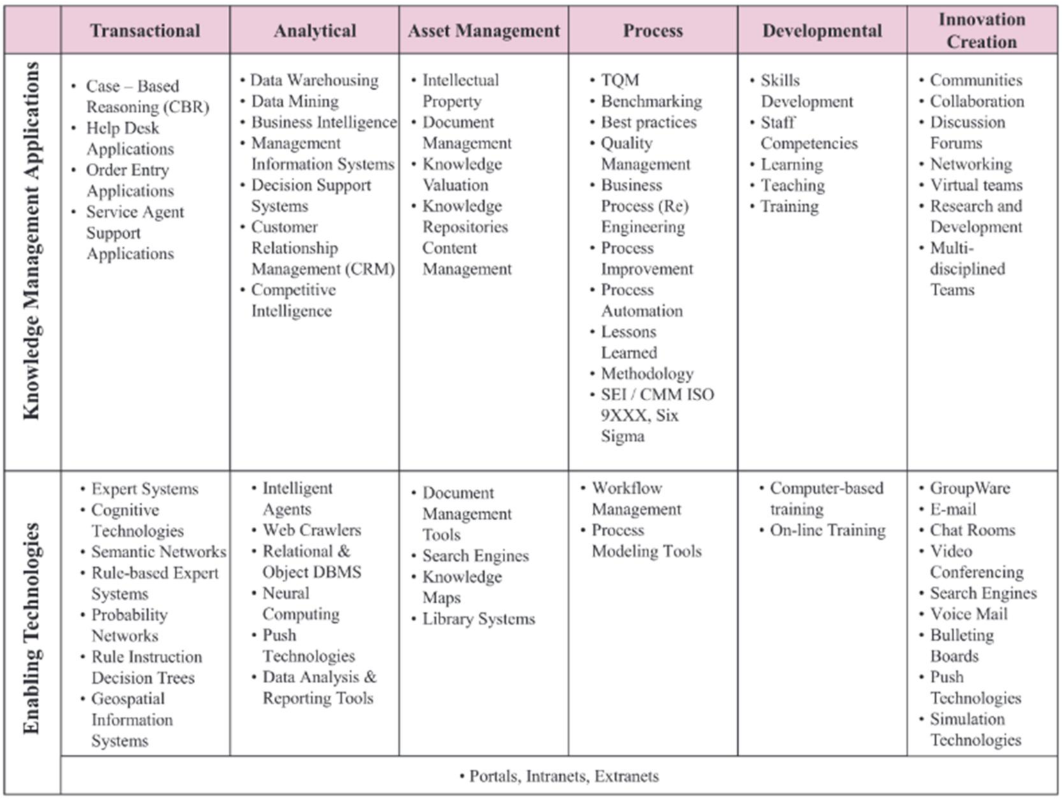
- Dissemination and Sharing: Knowledge dissemination is to distribute or to diffuse any information or knowledge to make them available to the relevant individual at the right time. Knowledge sharing is when the individual created new knowledge and is incentivized to transfer it through information networks with other people. Both processes are integrated and work simultaneously with each other [72].

- Retrieval and Apply: Knowledge retrieval allows members to access and apply stored knowledge during projects. An effective retrieval method facilitates the understanding of knowledge assets and enables quick and efficient access to the desired information. However, simply finding the knowledge is not enough; it must be refined and processed for specific contexts before it can be used by engineers [55].

3.3 Relevants KM technologies

Traditionally, the approach to KM is either from the management or technological perspective [55]. A hybrid solution that incorporates both people and technology is recognized as an effective KM strategy [73]. The technology aspect serves as a crucial enabler in the form of KM systems, which leverage technology to accelerate and streamline corporate learning processes through codified knowledge [74]. Thus, the technological solution is the focus of the author since aligns more closely with the building block concept.

KM technologies are closely intertwined with IT [5]. The rapid growth in IT-related applications over recent years has presented a wide range of solutions, see Figure 5, from various providers that are capable of fulfilling different tasks. However, the abundance of options has made it extremely challenging to determine the most suitable applications for our KM implementation. 6 different technologies were identified as relevant to the building block concept.



	Transactional	Analytical	Asset Management	Process	Developmental	Innovation Creation
Knowledge Management Applications	• Case – Based Reasoning (CBR) • Help Desk Applications • Order Entry Applications • Service Agent Support Applications	• Data Warehousing • Data Mining • Business Intelligence • Management Information Systems • Decision Support Systems • Customer Relationship Management (CRM) • Competitive Intelligence	• Intellectual Property • Document Management • Knowledge Valuation • Knowledge Repositories Content Management	• TQM • Benchmarking • Best practices • Quality Management • Business Process (Re) Engineering • Process Improvement • Process Automation • Lessons Learned • Methodology • SEI / CMM ISO 9XXX, Six Sigma	• Skills Development • Staff Competencies • Learning • Teaching • Training	• Communities • Collaboration • Discussion Forums • Networking • Virtual teams • Research and Development • Multi-disciplined Teams
Enabling Technologies	• Expert Systems • Cognitive Technologies • Semantic Networks • Rule-based Expert Systems • Probability Networks • Rule Instruction Decision Trees • Geospatial Information Systems	• Intelligent Agents • Web Crawlers • Relational & Object DBMS • Neural Computing • Push Technologies • Data Analysis & Reporting Tools	• Document Management Tools • Search Engines • Knowledge Maps • Library Systems	• Workflow Management • Process Modeling Tools	• Computer-based training • On-line Training	• GroupWare • E-mail • Chat Rooms • Video Conferencing • Search Engines • Voice Mail • Bulleting Boards • Push Technologies • Simulation Technologies
	• Portals, Intranets, Extranets					

Figure 5. Technology and applications for KM. Partially correlate to the determined KM processes. Sources: [75], [76]

3.3.1 Knowledge base

A KB serves as a machine-readable repository that organizes and stores knowledge on a specific topic in a concise manner. It encompasses factual information found in various sources such as books, websites, and human knowledge [5]. Knowledge is also used in the context of a component of an expert system (ES), which uses the knowledge in the KB to make decisions.

The primary objective of developing a KB is to be able to reuse explicit knowledge and possibly interact with and transform it. This is motivated by the fact that capturing and reusing knowledge is less costly than recreating it [77]. Currently, in many AEC firms, knowledge capture and reuse are limited to traditional paper archives or digital archives consisting of electronic folders and files, which are difficult to explore and navigate [55], [77]. In the context of AEC project, the challenge is developing a KB for storing knowledge generated throughout the project and effectively distributing it to project members who can benefit from it [78]. Thus, an effective KB must enable the four aforementioned KM processes.

A topic this thesis deemed relevant to address the difference between a database and a KB. Both KB and databases share similarities and fundamental distinctions (see Figure 6). While a database is a collection of organized data and information that requires further analysis and processing before application, a KB serves as a repository for knowledge that acts as ready-to-use solutions or answers [79]. In other words, a database serves as a repository for information, while a KB includes the meaning of that information [80]. A KB captures knowledge about a specific topic using an appropriate notation, while a database stores large amounts of shared data. In databases, there is only a rudimentary and largely intuitive interpretation of the stored information, while KB must commit to specific interpretations of the information. Therefore, database is only expected to meet performance standards related to response time, robustness, reliability, security, etc., while KB also needs an associated theory to effectively interpret the information they contain. Ontology can be used to add meaning and theory to existing data and information [14].

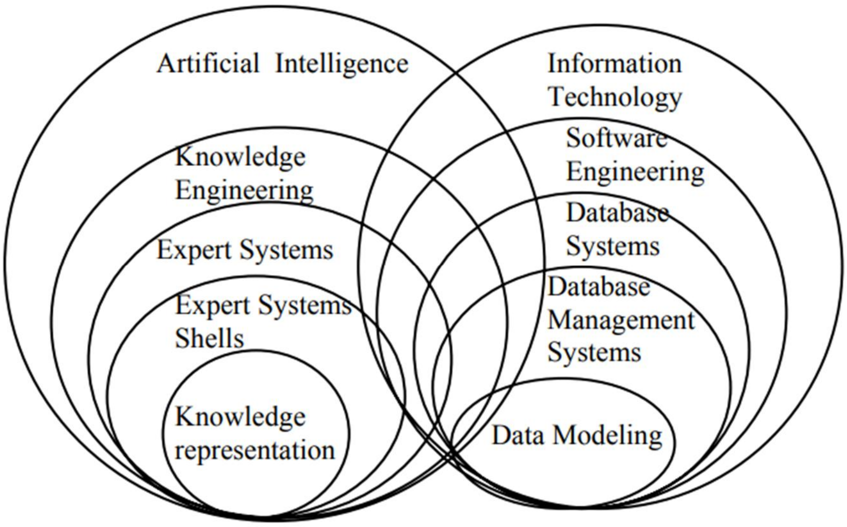


Figure 6. Database and KB in parallel. Source: [81]

For example, when representing a bridge design [79], a database is populated with data about bridge dimensions, bridge types, project names, construction years, main materials, etc. On the other hand, a KB begins with a substantial amount of data (probably from a database). The emphasis then shifts to enriching the existing information for ontology aggregation, such as pylons and cables are classified as suspension bridge parts, and establishing connections, such as the preference for choices of prestressed concrete or suspension bridge in relation to long span length.

3.3.2 Knowledge-based System and Expert system

If the KB is in machine-readable form, a system can use the KB to do intelligent things. Such a system is called a Knowledge-based System (KBS). A KBS simulates human intelligence in a specific domain and thus, it is a form of Artificial Intelligent (AI). The types of KBS are numerous as are their usage purpose: ES for decision-making, neural networks for pattern recognition tasks, case-based reasoning (CBR) for problem-solving using previous experience, etc. [47] The most interesting KBS to the topic at hand is the ES.

An ES is a computer-based representation of expert knowledge, capable of providing intelligent advice or making informed decisions about a specific knowledge domain. It is also desirable for an ES to explain its reasoning in a way that is understandable to the user. This is achieved through rule-based knowledge representation (KR) but other forms of representation also exist. In the context of construction projects, one of the primary objectives of KM is to support engineers in their decision-making process [82]. ESs have been developed and effectively utilized in the construction industry to fulfill this purpose. These computer-aided inference systems rely on knowledge acquired from domain experts or various other sources. However, the amount of available data has exponentially increased in today's information age, making capturing expert knowledge a resource-intensive, time-consuming, and costly endeavor.

3.3.3 Taxonomy and Ontology

Ontologies and taxonomies are in many ways similar – they are both systems of classification and are arranged in a hierarchical structure. A taxonomy classifies data into categories and sub-categories, providing a unified perspective and providing common terminology and semantics for multiple systems. An established taxonomy is static [83]. Ontology, on the other hand, defines terms and relationships of data in a more sophisticated manner. They are formal and explicit specifications of a shared conceptualization [84], serving as graphical representations to describe domain knowledge. Nodes in ontologies represent classes and instances that depict domain entities, and edges specify attributes and connections, not limited to hierarchical relations, between entities [85]. Taxonomy and ontology are often mistaken one for another since ontologies may encompass taxonomies, and taxonomies can be enriched to resemble ontologies [86]. As information and KM converge, the overlapping of taxonomies and ontologies has become more prevalent. While some may use the terms interchangeably, they are not identical, although they are increasingly integrated. The same software might support the creation of both.

3.3.4 Metadata

Metadata is data about data. It serves as a fundamental component in understanding and organizing multimedia documents [87]. Acting as an intermediate representation of the documents, it encapsulates valuable information about the content and the external context in a compact form. Metadata can take various forms, ranging from manually generated textual annotations to specific attributes extracted through content analysis. With the structure, nature, and relations of document contents described by metadata, users can search, index, and retrieve relevant documents.

There are two methods of information retrieving using metadata [88]. The first is a basic method, operating similarly to a library system where each document is associated with predefined attributes such as author, title, and ID number. The second is an intelligent method, where information is derived from a semantic network, allowing the system to generate alternative answers to queries that yield no answers.

The metadata-based systems are highly detailed since they can cover a wide range of attributes [88]. Since a document can have an unlimited number of attributes, classification based on these attributes can achieve an unlimited fine level of granularity. Thus, searching in this type of system yields very good results. Additionally, intelligent metadata-based systems with predefined semantics can offer alternative options when searches yield empty results. However, a drawback of these systems is the huge amount of manual work in reading and understanding these documents prior to classification. That means the metadata model is subjective and reliant on the classifier's interpretation. The externalization process becomes the biggest obstacle and the biggest cost to efficiency for metadata-based systems.

3.3.5 AI Applications

KBS as an intelligent computer program fits into the larger context of artificial intelligence (AI). The widespread integration of AI is expected to significantly impact various aspects of organizational functioning, including KM [89]. KM processes can greatly benefit from various implementations of AI technologies. In this section, some AI applications are briefly discussed since they can bring potential benefits to the building block concept but have not necessarily been fully explored or implemented.

Neural network and deep learning are two prominent AI applications. This application is more relevant when AI is mentioned nowadays. Similar to how ES automated inferences based on descriptive knowledge, neural networks can achieve the same for procedural knowledge [90]. Tasks that require expert knowledge to perform procedures are ideal for automation using deep learning. While acquiring knowledge still necessitates the involvement of domain experts, constructing a machine learning dataset simply requires them to conduct their expertise rather than deconstruct it. Moreover, engineers find it easier to represent this knowledge in a dataset compared to devising methods for encoding and manipulating symbolic knowledge. As datasets grow and evolve over time, updating and enhancing models can be done seamlessly. However, neural network systems might not be appropriate for the building block concept as they cannot provide details on which variable was manipulated and the rationale behind their conclusions, which is essential in cases of dispute [91]. Although neural network can learn, it requires a huge amount of data for training and development.

A domain of AI deals with the concept of KR. In the context of structural engineering, the challenge lies in effectively quantifying and representing the heterogeneous forms of data-driven design [92]. The forms can include text, graphics, and tables, all of which are essential for conveying comprehensive design knowledge. Textual information alone is not enough to convey a design meaning holistically, particularly when documents were created to be viewed all at once. A solution from the KR domain for structural engineering knowledge is to leverage logic and ontology to construct computable domain-specific models that can solve complex problems [93]. This has the most relevance to the building block concept and will be explored in Section 7.1.

Finally, CBR is an AI approach that combines problem-solving and learning. CBR is extensively discussed in the literature as a technology supporting KM. It operates on the idea that similar problems tend to recur, allowing past solutions to be applied to current situations [94]. The system stores solved cases in a case base, which consists of problems and their corresponding solutions. When a new problem arises, the CBR system retrieves the most similar cases from the case base and reuses their solutions to compose a solution for the query, with possible adaptations, and then the revised case is added back into the case base. This iterative process follows a cycle of retrieve, reuse, revise, and retain, which mirrors knowledge cycles [95], allowing for continuous improvement. Case-based systems are particularly useful when domain knowledge is difficult to elicit, requires constant maintenance and records of previous solutions exist. However, the performance of CBR tools can deteriorate when dealing with large amounts of data, and the externalization of cases can be costly and inefficient [88].

4 Knowledge Base Development

4.1 Considerations

Numerous repositories for design knowledge have been established in various fields under different names and different forms. Extensive research has been conducted across diverse domains such as medicine, biology, transportation, agriculture, and economy. Additionally, the trend in recent years is exploring the utilization of Semantic Web technologies for ontological KB [96]. In this thesis, ontological KBs specifically related to construction and structural design were studied, which serve as valuable references for our KB in Chapter 7.

The manufacturing KB warrants closer examination since they use the KB for product design. According to [97-100], The KBs in manufacturing engineering have evolved far from simple storages of knowledge into intelligent design repositories integrated with design artifact modeling systems. These repositories consist of heterogeneous product design information that facilitates the representation, capture, sharing, and reuse of corporate design knowledge. The structure of the design KB is a crucial component of these systems, as it allows for the hierarchical decomposition of designs into smaller and simpler entities, making them easier to adapt to new design cases and support intelligent design environments. This is discussed at length in Section 6.3.2. By storing designs in a computable format, design repositories enable the inference engine of intelligent design support environments to make more informed decisions based on data from previous designs. In essence, design repositories serve as valuable sources of information that can be converted into knowledge through data, similar to how a library provides accessible, recognizable, and comprehensive information.

Despite possessing more advanced capabilities, the design repositories are still inadequate to handle complex project knowledge, as observed by research evidence by the high amount of time spent looking for information and the low percentage of documented knowledge for reuse [60]. Some of the issues limiting the effectiveness of the design repository and KB are related to capturing human knowledge, choosing types of KR, and targeting the right user

The first issue, as pointed out by [101], is that knowledge repositories can store explicit knowledge, typically as documents, allowing for successful capture and utilization of a portion of the domain project knowledge. However, a significant amount of valuable knowledge is tacit knowledge and resides within individuals, resulting in the loss of essential contextual, informal, and inductive reasoning underlying decision-making processes. A part of tacit knowledge is delivered through contextual information, which is discussed in Section 4.3.1.1. Another part of tacit knowledge can be conveyed using multimedia files to enhance the recording and subsequent reuse of informal information.

Multimedia files play a vital role in bridge design, encompassing various programs that address different aspects of the design process, such as modeling and analysis, integrated design, component design, substructure design, etc. [102] However, the second issue arises in effectively representing the knowledge contained within these multimedia files while maintaining their relationships in the KB. The complex and multi-perspective nature of design knowledge in multimedia files necessitates analysis and transformation into a format compatible with a KBS. Typically, algorithms or analyzers are used to extract knowledge and convert it into metadata, which serves as an intermediate representation facilitating easier manipulation and processing using information retrieval methods [87]. Consequently, this transformation introduces inherent uncertainty, as no analysis and transformation are error-free or not based on human interpretation. Additionally, the unstructured original data requires analysis tools to extract the underlying structure and knowledge and store them in formats that enable convenient access and manipulation. It was beyond the scope of this thesis to delve into the extraction and transformation of information from multimedia files. However, the selection of KR and the role of metadata and properties will be discussed in Section 7.2.

The third issue is scale and knowledge retrieval. KB accumulates vast amounts of unnoticed and unutilized information within the KB. [103] discovered that unless knowledge is possessed by individuals who find it relevant, its development, utilization, and maintenance will not be optimized. This is due to the lack of awareness from the user on the existence of the knowledge of interest in a large KB. Another challenge lies in the fact that users frequently struggle to precisely articulate their information requirements [104]. The arises when users are not fully aware of their own needs, leading to a potential mismatch between the information obtained and their expectations. In short, the knowledge retrieval methods are often inadequate and fail to accurately target the intended audience [105]. Not only does this necessitate spending more time on searching, but it also generates feelings of dissatisfaction. This issue of knowledge retrieval will be addressed in Section 4.4.

4.2 Knowledge Taxonomy

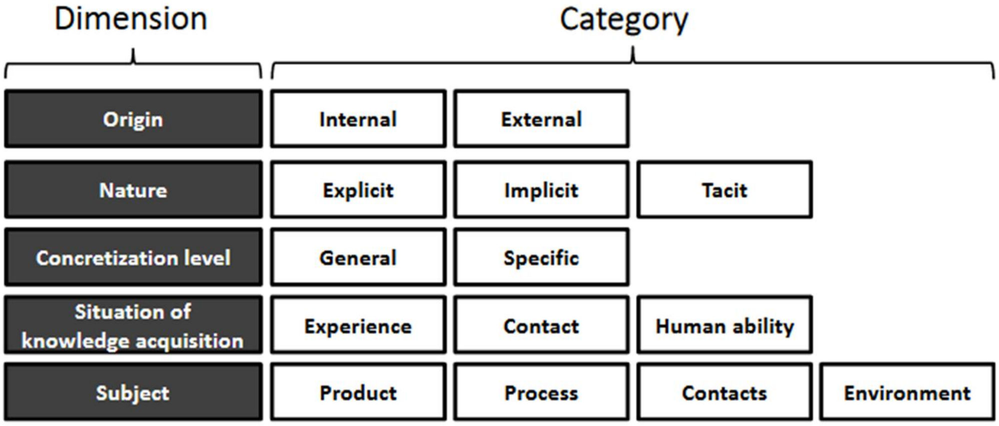
The effectiveness of a KB is also determined by the design of its taxonomy, which provides the domain specifications through a shared vocabulary. Taxonomy plays a crucial role in developing a structure for KBs. It defines the categories of a KB at a high level and the specific components at a low level

[106]. Since knowledge taxonomy is a hierarchical relationship, it is closely related to the concept of granularity as discussed in Section 4.3.1.2.

The way to classify knowledge in the KB is a topic that lacks consensus. Various studies have highlighted different approaches to classifying knowledge repositories. Generally, knowledge can be categorized into four types: know-what, know-why, know-how, and know-who [107], [108]. However, [109] demonstrated that most repositories are typically classified by subject areas, such as procurement, cost control, and risk management. On the other hand, [110] proposed an alternative classification, where project knowledge is categorized based on activities. [111] focuses on the design process phases and attempts to determine the appropriate knowledge for each phase and its purpose such as Problem definition, Preliminary design, Design communication, Final design, etc. In the context of construction projects, [112] identified three types of project knowledge: technical knowledge, concerning the product, its components, and technologies; procedural knowledge, concerning the production and utilization of the product in a project; and organizational knowledge, concerning communication and collaboration.

By drawing upon the various knowledge classification approaches, [113] developed a comprehensive taxonomy that encompasses diverse types of knowledge essential to design engineering, illustrated in Figure 7. This taxonomy took the different classifications and requirements for knowledge and eliminated the ones considered irrelevant to the design engineering field. The taxonomy consists of five knowledge dimensions, with each dimension representing a unique aspect of knowledge:

- Origin: The source of knowledge within an organization, categorized as either internal or external.
- Nature: the way knowledge is expressed, captured, and shared. It is classified into Explicit, Implicit, and Tacit.
- Concretization level: Knowledge's Level of detail, categorized as either General or Specific.
- Situation of knowledge acquisition: the context in which the knowledge was acquired. It is classified into Experience, Contact, and Human.
- Subject: the perspective or the types of knowledge field, including Product knowledge, Process knowledge, Supplier of knowledge, and Environment.



Dimension	Category			
Origin	Internal	External		
Nature	Explicit	Implicit	Tacit	
Concretization level	General	Specific		
Situation of knowledge acquisition	Experience	Contact	Human ability	
Subject	Product	Process	Contacts	Environment

Figure 7. Design Engineering Knowledge Taxonomy. Source: [113]

The five design knowledge dimensions allow the taxonomy to have a dual purpose: classification of knowledge elements and acts as guidance for engineers on the various types of knowledge required when encountering novel design challenges [60]. Thus, the taxonomy would be under constant consideration throughout the development of this thesis's KB, ensuring comprehensive coverage of generated knowledge during the bridge design process and its subsequent representation in the implementation phase.

4.3 Knowledge Reuse:

4.3.1 Reusing Engineering Design Knowledge

According to an ethnographic study of design knowledge reuse in AEC industry [59], designers, like architects and structural engineers, place a strong emphasis on reusing knowledge from previous projects in their work. Reuse of knowledge commonly takes the form of standard details. These are small design elements that remain consistent across different projects, such as bolt connections. The standard details usually appear on design output drawings. In the interview, the designers stressed the importance of fully comprehending a detail before using it in a new project. They highlighted that the process of structural design encompasses far more than simply assembling standardized components. Designers, especially structural engineers, often reuse design tools such as spreadsheets and structural analysis models. This practice is extremely common as the study mentioned personal accumulation of tools collection and incorporating spreadsheets into the company standards repository. Another common practice is to consult or verify assumptions made in previous projects similar to the current project. The reason is twofold, one is the iterative nature of design work and second is the continuous improvement of design engineers. By going through iterative cycles, the accuracy of information is refined with each iteration [114]. And the designers who engage in this activity gain new insights and perspectives that may challenge their initial assumptions. In the long run, those who tap into the experience from previous iterations or similar projects can push the boundaries of their designs and deliver better solutions.

As mentioned in Section 4.1, knowledge reuse in design engineering knowledge repositories poses significant challenges. [115] gave two characteristics present in structural design but not in the general document repositories that cause these challenges: the unique structure of construction content and the distinct information needs and searching habits of engineers. The first characteristic directly affects the reusability of knowledge and thus, it is discussed here while the second characteristic is related to knowledge retrieval and is elaborated in Section 4.4.

[115] continue to argue that context and granularity are fundamental concepts of structural design knowledge. They are considered when evaluating the relevance of the retrieved knowledge to the users. In other words, the effectiveness of a civil engineering KB is determined by these 2 factors.

4.3.1.1 Context

The concept of context is of great importance in cognitive psychology, linguistics, and computer science [116]. It is defined as not a static state but rather a dynamic part of a process involving the interacting human. Contextual information can be any available elements during an interaction, such as identity, spatial and temporal details, environmental factors, social circumstances, nearby resources, physiological measurements, and emotional states.

[117] establishes several connections between knowledge and context: context and knowledge are the same since knowledge cannot be understood isolated from context. The structure and level of detail of context change and evolve with the ever-changing focal point. [77] also agrees as they introduced the notion of knowledge in context. The knowledge in context is rich, detailed, and contextual when it is embedded within a designer's memory.

Indeed, the knowledge of structural design process is highly context-specific. It requires a comprehensive understanding of the applying situation for the knowledge to be valid. For example, a holistic knowledge of a composite deck is the evolution of the design process from initial sketches and rough calculations to intricate 3D BIM (Building Information Modeling) models, analysis, simulations, design rationale, and the relationships of cross-disciplinary perspectives. Moreover, in project-based environments of structural design, having access to the appropriate knowledge plays a crucial role in preventing duplicated knowledge and the repetition of mistakes across projects [115].

Different approaches exist to define and quantify context, including text-based analysis, hierarchical structures, and the utilization of metadata. Metadata, in particular, serves a dual function by condensing and organizing knowledge for subsequent reuse, establishing mutual relationships within an ontology [118]. That is, metadata provides context at different levels of granularity of an ontology. For example, each level of a project's hierarchy (Discipline or Component) included metadata that described its content, such as name, cost, design data, and maintenance data [115].

The study by [59] found that senior engineers explain design knowledge to novice engineers by exploring two dimensions of context: the project context and the evolutionary history. Specifically, the exploration comprises a total of six degrees of freedom, up, down, and sideways for each of the two contextual dimensions.

- Project context:

This dimension refers to the relationships between items. Conceptually, it can be visualized by positioning an item in a 2D space of depth and breadth. Depth is moving up and down, going from component to subassembly and vice versa, respectively. For example, a structural component's depth would involve its parent Discipline and grandparent Project, its children subassemblies, and grandchildren sub-subassemblies. Breadth is moving sideways from one item to related items. For example, a structural component's breadth would involve other similar components from the repository, regardless of the project.

The significance of project context in knowledge reuse becomes evident as the senior engineer considers the tradeoffs made during the design of standard details. A decision must be made between knowledge-rich details that are highly specific and generic details that are widely applicable. The tradeoff is very difficult since when a typical detail is extracted from its project context and standardized, it loses much of its value. This also leads to a lack of effectiveness in reusing unfamiliar systems in which the design knowledge they offer is decontextualized. The essential contextual information can only be provided by the senior engineer, who either created or worked with the system originally.

- Evolution history:

Equally important is the evolutionary history. Evolutionary history is the progression of design, from a set of requirements to a fully designed physical entity. Similar to project context, up and down movements represents going from detailed to conceptual design and vice versa, respectively, while sideways movements involve the exploration of alternative design options.

Evolution is demonstrated when a novice designer reuses an old component, senior engineers always explain the facts behind the design and throughout its evolution. The novice engineer might even question the missing information to understand the purpose and underlying reasoning behind a design. Such information is significant for determining how the component should be modified before it could be effectively reused.

4.3.1.2 Granularity

Granularity can be defined as the position of an item within the hierarchy. For example: a project, represented by a bridge, is at a coarser level compared to a bolt plate that constitutes a tiny level of granularity [115]. Granularity and context are both basic structuring mechanisms in various reasoning tasks performed by humans. It enables us to structure a continuous reality into a more handleable hierarchical structure of finite domains [116]. Moreover, contextual information can be derived from granularity [119]. Items at lower granularity levels can be contextualized by parent items. Similarly, information at higher levels of granularity, such as a steel girder, can be enriched and clarified by the details of its subparts, such as flange and web.

According to [120], The level of granularity in a model is influenced by factors such as the scope and purpose, user requirements, and available resources. The choice of granularity has significant implications for various stages of the KB construction process, from design and analysis to reuse. Selecting an appropriate level of granularity involves considering multiple factors and striking a balance. The chosen level should be sufficiently detailed to support accurate reuse of design tasks in complex bridge design while avoiding excessive detail that may hinder user navigation through the design process [121]. This requires defining the scope or boundaries, adopting a modeling approach, and making informed decisions on how to represent specific real-world entities [122].

Typically, the focus of granularity lies within the product domain and is not commonly addressed in the design of process modeling. Despite that, the significance of granularity in the process domain should not be overlooked since it can significantly impact the behavior of process models. But no matter the domain, defining granularity is a complex task as it relies on subjective concepts and lacks a clear theoretical foundation. It can hardly be measured and since anything can be treated as a complicated object, it usually falls on human subjectivity to determine the granularity level [123]. One notable distinction between process model and product model is that it does not have physical elements that could be used as a baseline. This makes process model harder to break down without affecting its performance. For example, coarser models may combine multiple tasks into one, which becomes problematic if it masks the iteration nature of the processes. Thus, it is crucial for process models to capture the relevant process structure in sufficient detail without excessively complicating the model [120].

4.3.1.3 Context–Granularity Interaction

As [59] observed, reusing elements from previous designs in new designs can increase productivity but may limit creativity in the design process. While reusing small pieces of previous design may not significantly enhance a designer's productivity, it is less likely to compromise the creativity of the overall design. As mentioned, in the AEC industry, it is common practice to reuse standard details from one project in a completely different project. These standard details, being small parts of design in nature, can be applied in various design scenarios without compromising the creativity of the new design. Conversely, larger design pieces contain more knowledge but are also less reusable. This was exemplified in the study when specific details for an elevator pit, which was designed for the original project, have to be stripped down of its features and components to focus on the essential subcomponents of an elevator pit, making it more generic and applicable across different projects.

This behavior can be understood by considering the interactions between two factors: the level of granularity, which ranges from the entire structure to its smaller sub-components, and the level of abstraction (precision), which spans from abstract ideas to precisely defined elements. These two factors are then conceptualized as dimensions of a two-dimensional knowledge space [59], as depicted in Figure 8.

At the upper right corner of the knowledge space, the reused knowledge involves precise, whole structural elements, such as reusing a bridge design. In this scenario, a lot of reuse occurs (reusing all parts of the structure) with a trade-off in creativity since the entire element is reused without modification or alternatives. In contrast, at the lower left corner of the knowledge space, the reused knowledge involves generic and smaller parts of the elements, such as reusing the idea of putting rebars around opening. In this case, reuse takes place without compromising the originality of the solution.

Finally, [77] concluded that the reusability of a design item depends on its level of granularity and precision. These characteristics play a significant role in determining the suitability of the item for reuse. They suggested that items with finer granularity and convey abstract concepts have higher levels of reusability. This aligns with the modularity concept in Section 6.3.3 and user's information retrieval habits in Section 4.4.

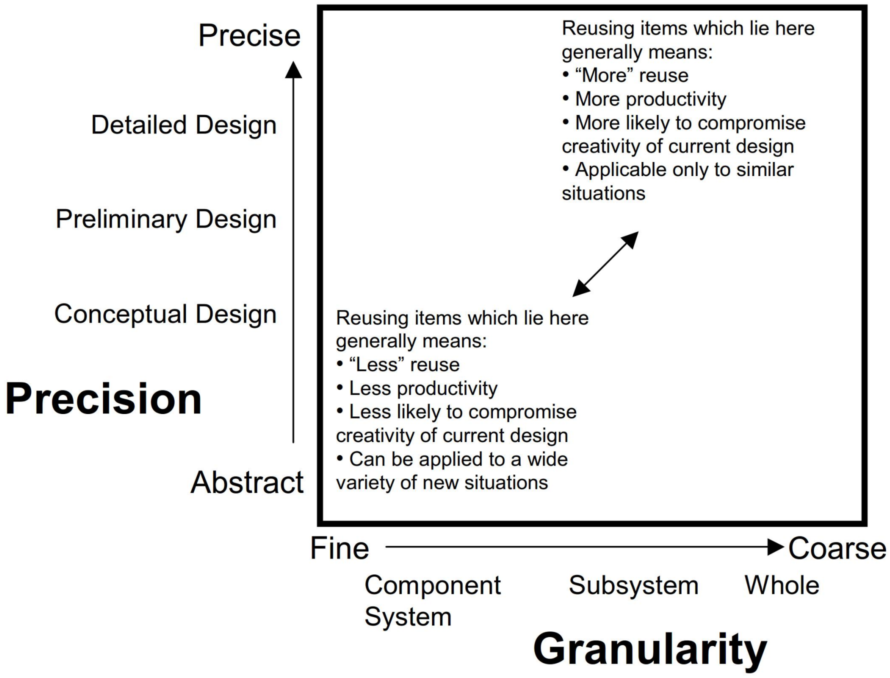

Precise
Detailed Design
Preliminary Design
Conceptual Design
Precision
Abstract
Reusing items which lie here generally means:
• "More" reuse
• More productivity
• More likely to compromise creativity of current design
• Applicable only to similar situations
Reusing items which lie here generally means:
• "Less" reuse
• Less productivity
• Less likely to compromise creativity of current design
• Can be applied to a wide variety of new situations
Fine
Coarse
Component System
Subsystem
Whole
Granularity

Figure 8. Precision and granularity represent the balancing act between productivity and creativity. Source: [59]

4.3.2 Reaching reusability

Despite the suggestion of fine granularity and fine abstraction for optimal knowledge reuse, not all design knowledge and process can be of the same granularity level. Relying on a single granularity results in a limited and inflexible representation of building block functions and the process flow, making it challenging to fully comprehend the diverse range of knowledge required in the process [124]. Examples of multi-granularity needs for different building blocks are: a Grasshopper script for rebar modeling on a bridge deck is a lower granularity than the encompassing level of the Eurocode 2: Design of concrete structures, and higher granularity than the tendon profiles used in the Tekla model. Multi-granularity structure should be used to organize design knowledge, enabling a more comprehensive understanding of the internal relationships within the knowledge.

When choosing the granularity to structure the design KB, we must be aware of a conflict in reusability, illustrated in Figure 9. While there tends to be a preference for decomposing design elements toward fine granularity levels, it is not always beneficial [123]. In fact, coarse-grained elements offer high reuse efficiency, due to their significant contribution to the system, but low reusability, due to highly specific problem-solving capabilities. On the other hand, fine-grained elements are required to achieve maximum flexibility and reusability for the opposite reasons. This behavior aligns with the context-granularity dynamic in Section 4.3.1.2.

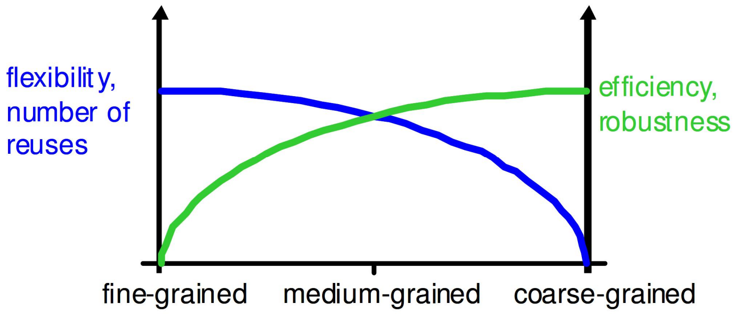


Figure 9. Conflicts in reusability when granularity changes. Source: [125]

In addition to its impact on reusability, it has been observed that granularity is inversely related to managerial objectives such as cost-effectiveness, customization, and maintainability [126]. When granularity is coarser, the maintainability increases and cost decreases due to a reduced number of components and interactions needed, thereby enhancing efficiency and robustness. In contrast, fine granularity facilitates flexibility in component assembling in exchange for higher cost and harder maintenance.

Therefore, the selection of a granularity level requires careful balancing of cost and benefits from reusability. The effort in constructing the model for KB depends on the granularity level [120]. For example, reaching an agreement on abstract models can be time-consuming, while developing highly detailed models demands specific skill sets and computational resources. Managing, deploying, and maintaining fine-grain models require substantial effort. Additionally, fine-grained models may pose challenges in eliciting the required information since such information is not readily available in many scenarios.

Given the cost-benefit factors, it is still necessary to find the minimum level of granularity in for the simplest workable model [127]. The question to ask is whether adding more detail serves to enhance model fidelity or not. If not, it can even lead to a model that is excessively complex and challenging to calibrate, use, and interpret [128]. Asking the opposite question is also valid: can the decomposed elements accommodate requirement changes through configuration? If not, then its granularity should be increased [129].

In addition to cost-benefit, and model fidelity considerations, decisions regarding the granularity should also take into the factors specific to the nature of the engineering design process. Engineering design involves both product and process modeling, which are often treated as separate entities but can also be integrated [130]. It is important to recognize that the granularity choices made in one domain can have an impact on the other. For instance, the level of granularity in a product model can affect design process sequencing [131]. Therefore, when multiple models are present in the final product, maintaining the harmony of different levels of granularity across the models is crucial to avoid inconsistencies [120].

Another consideration affecting granularity is that engineering design processes are iterative. It requires selecting a modeling approach and making individual decisions on how to represent specific real-world behavior [122]. One example is the modeling of two closely related tasks: they can either be combined into one, simplifying the model but obscuring the iterative behavior, or treated as separate tasks, capturing the iteration loop [120]. The second choice is briefly discussed through the clustering concept in Section 6.3.3. Either choice is driven by the choice of the modeler, supported by clearly articulated requirements [132].

As discussed since the beginning of this section, achieving the optimal level of granularity was not a straightforward task in practice and no clear answer could be given in this thesis. However, a constant is that the knowledge element must accommodate design knowledge reuse. According to [6], the key to achieving design reusability is ensuring that the design's functionality fulfills a specific need for the user. This means that either the entire design or certain components of it are duplicated or commonly used across multiple applications. This requirement can be satisfied in three ways. Firstly, the design part implements a common basic function. For example, a calculation template is reused for load combination check. Secondly, designers reuse a component in their designs to adhere to standards without needing an in-depth understanding of the standard itself or its implementation details. For example, steel beam profiles are reused to adhere to industrial or national standards. Lastly, designs are reused inherently as part of the design evolution. Since a large part of a design remains unchanged as more functionalities are added, it automatically serves as a solid foundation for subsequent design revisions.

4.4 Knowledge Retrieving

Knowledge reuse requires an effective knowledge retrieval method. Various retrieval tools have been developed, including search engine techniques, the automatic clustering or classification of documents, and user or expertise profiles [133]. Although, two fundamental methods that are the most widely used, especially in the AEC industry are browsing and keyword searching [55]. Browsing: users navigate through the KB based on the index until the desired information is located. The effectiveness of browsing depends on the formatting, structuring, and classification of knowledge. In the context of a construction company, project knowledge is organized in a taxonomy of processes and activities. This allows users to retrieve specific knowledge directly related to the relevant activity in the KB. Keyword searching: A method for locating digital data by pattern-matching a keyword or phrase. Keyword search works most effectively when an area of interest is determined. The approach is commonly employed in conventional web search engines. This retrieval method is used extensively by construction companies as well. A third method worth mentioning is querying. Query searches typically operate under the assumption of existing relations or structures within the KB. In essence, query searches are regulated by stringent syntax rules akin to command languages, utilizing keywords or positional parameters [60].

The methods of retrieval align with two distinct approaches to interacting with a knowledge repository [134]. The first approach is the retrieval mode, where users have a specific information need that they express through queries. The system then takes these queries as input and generates a ranked set of items as output. The second approach is the exploration mode, which occurs when users have a general information need and engage in browsing the repository. In this mode, the process of exploration holds significance alongside the actual retrieval of items, as it contributes to the user's understanding of the content. Ranking the relevance of items in the repository is essential for both interaction modes, whether it is to retrieve the most relevant items or to guide the user's exploration.

As [135] observed, the reuse process of AEC professionals consists of two essential steps: engineers locate the reusable items and then understand these items within their created context. This is because design and construction knowledge is context-specific, requiring knowledge retrievers to possess a certain level of expertise and experience or be provided with them by the knowledge system, to effectively interpret the codified knowledge [55]. The contextual information ranges from structural hierarchy to the meaning and purposes behind the retrieved knowledge, such as legislation or maintenance costs. Therefore, an effective tool for supporting reuse must enable identifying reusable items and aid in their comprehension [136]. Such a task is extremely difficult due to engineers usually have distinct information needs. The difficulty arises when engineers are unsure of what they are searching for, except that it should be a relevant standard for their current design task. [137] notes that users frequently fail to utilize reuse systems because they are unaware of the system's relevant content or cannot formulate effective queries. To address this, a reuse system should be capable of assessing the relevance between the current task of interest and the items in the repository. Such implicit queries can greatly assist novices who are unfamiliar with the repository's contents. On the other hand, designers with more experience or those in search of specific items, which they may have previously worked on, should be able to formulate explicit queries [59].

To summarize, the context-specific of engineering content and the unique information requirements of civil engineers render traditional retrieval techniques and visualization methods less effective. As a result, a customized solution that is tailored to the specialized nature of civil engineering design knowledge is necessary. This solution needs to offer a balance between granularity levels was crucial, as too fine-grained components lacked context and were deemed less relevant by users. The retrieved contents should be the specific sections of a document rather than the entire document, but metadata about context and general details of the whole document is required to understand the retrieved sections [136], [138-140]. Complex tasks, particularly decision-making tasks, benefitted more from contextual information compared to simple fact-finding tasks, which could be addressed with narrower contextual information [141]. Finally, user should receive the result in an order ranked by relevance

4.5 Requirements of a KB

At this stage, we have examined numerous factors that influence KBs. It is now essential to address other technical aspects and outline the requirements for establishing a successful KB. In doing so, we must consider the needs of both knowledge users and knowledge contributors. In essence, a KB entails two primary information flows: the dissemination of information and its extraction or utilization.

The study conducted by [142] addresses the three essential requirements of a KB, namely the ability to manipulate knowledge in various forms, low effort and ease of interaction with knowledge, and the ability to update and actualize knowledge. Building upon these requirements, [49] emphasizes the significance of catering to the needs and expectations of diverse end-users. Thus, he stresses the importance of a clear and structured KB to facilitate documenting, searching, and reusing knowledge. Furthermore, [143] suggested that offering various ways of visualization, such as filtering and contextual view, can enhance information retrieval and understanding of the retrieved knowledge. This aspect becomes particularly useful for teaching purposes and when integrating new designers into a company.

The doctoral thesis by xx provides 2 core values that frame the requirements for any KB. Lauer proposes two hypotheses:

- By having well-defined and descriptive parameters shared by both processes and documents, it is possible to automatically make connections between processes and documents and thus, draw conclusions about their relevance,

- As a consequence of the first point, when processes and documents are connected, users experience tangible benefits of combined values of the knowledge and the context surrounding it.

In light of the above, the following eight requirements rose.

Based on the aforementioned considerations, the building block concept necessitated the following nine requirements:

- Reuse is the main goal of the building block concept
- Building blocks are tools and documents from different granularity levels in context. A KB should be able to represent the knowledge through every level of complexity and abstraction, showing the whole picture of the contained knowledge. This is in the form of general details provided by metadata and multimedia KR.
- The documents and tools are contained in a building block
- Building blocks are process-oriented. As mentioned before, having a connection between processes and documents facilitates understanding of reused knowledge, particularly for inexperienced engineers.
- Building block's value must be intuitive and easily recognized.
- Building blocks can be viewed from different perspectives. Different people coming from different areas or sectors will contribute to the building block. They have different information needs and ways to access information.
- Building blocks can be extracted. The search function can find the knowledge associated with user-provided parameters.
- The extracted results are adjusted to demand. The KB provides the right amount and relevant knowledge only, hiding meaningless knowledge from the user.
- Building blocks can be updated and new Building blocks can be added. Must come with a version control function.

5 Knowledge-based System Development

5.1 Knowledge-based System Choice

Having established the theoretical background and usage context, it is time to develop the KBS supporting the building block concept. The type of KBS that matches our vision is the Expert System (ES). ES is designed to imitate the abilities of a human expert in a specific domain to solve problems or make decisions. With the automation of problem-solving (designing tasks automation) can be achieved with CD already, what we want to focus on here is the ability to make decisions of ES.

The ES consists of many elements (Figure 10). Each one plays a role in achieving its main function of making intelligent decisions and accordingly [47], meeting the requirements of the building block concept:

- Knowledge Base: a collection of where the knowledge is stored. In the building block concept, the knowledge is the design tasks comprising a design process, also known as the building blocks.

- Knowledge acquisition module: helps when building up new KB

- Inference engine: the “brain” of the ES. The engine interprets the user's input and uses KB to generate the appropriate output, which can be newly inferred knowledge or knowledge consistency check results [144].

- Explanatory interface: it explains how the conclusion was reached by the ES.

- User interface: allow humans to use and interact with the ES. Through this, we input the problem that the ES provides answer for.

Typically, the development of an ES involves the collaboration of three key parties: the domain expert, the knowledge engineer, and the software engineer [146]:

- The domain expert holds crucial importance in the ES development team, possessing extensive knowledge and skills in a specific domain. This expert is responsible for effectively communicating their expertise and investing a significant amount of time in the system's development process.

- The knowledge engineer works alongside the expert and takes on the role of designing, constructing, and testing the ES. They engage with the expert to determine appropriate reasoning methods and make decisions regarding the representation of knowledge within the system.

- The programmer is tasked with the actual programming work, translating the domain knowledge into a computer-understandable format. Nowadays, modern applications implement ES concepts through basic coding without being classified as part of AI [147]. In some cases, a programmer is not needed when utilizing ES shells, which provide pre-built components and an empty KB. Consequently, the knowledge engineer's primary responsibility becomes populating the KB [148].

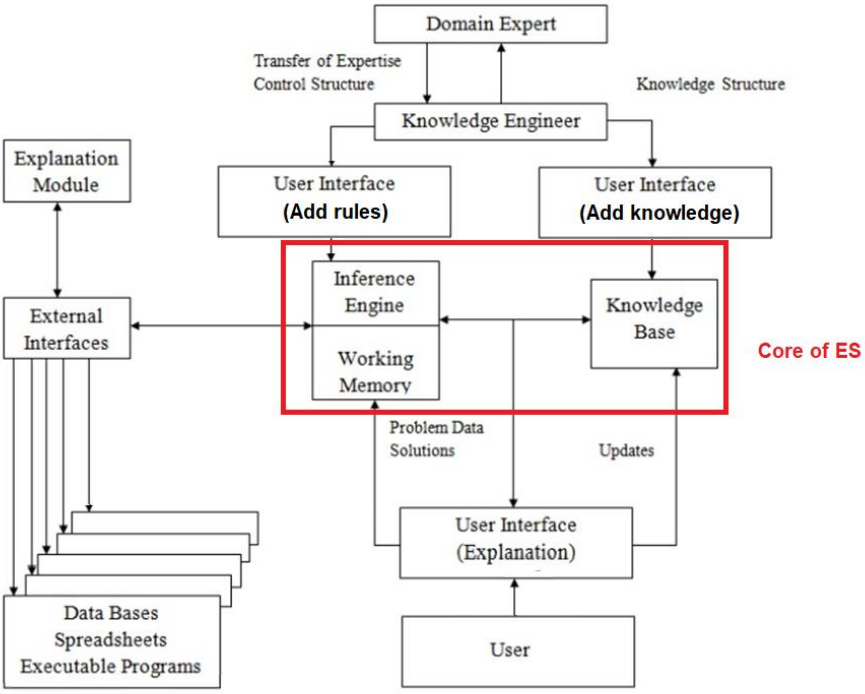


Figure 10. Expert system elements. Adapted: [145]

5.2 Methodology

According to [149], the development of design KBS applications for the AEC industry has been relatively limited, and even less information regarding the developing methods of such applications. This hinders the ability to build upon previous research findings and a KBS methodology for the construction industry is still missing [150]. However, alternative methodologies from other fields exist, such as CommonKADS, which was created at the University of Amsterdam and finds more extensive use in computer sciences [151]. Another example is MOKA (Methodology and tools Oriented to Knowledge-based engineering Applications), a more general methodology that does not readily suit the needs of the construction industry [152].

Considering the necessity and constraints associated with this thesis, a more generalized framework (Figure 11) for knowledge acquisition had been selected over other comprehensive models. The knowledge acquisition process entails a four-step cycle [153]: Elicitation, Representation, Implementation, and Validation. Elicitation involves identifying and categorizing the data used by field experts and defining vocabularies, taxonomies, and rules. Representation focuses on representing knowledge in a formal language that aligns closely with the implementation phase. This step involves developing domain ontologies and outlining strategies for problem-solving. Implementation is the stage where expertise is transformed into a knowledge-based program. Finally, Validation serves as the last step, wherein experts test and verify the accuracy, completeness, and correctness of system data and rules.

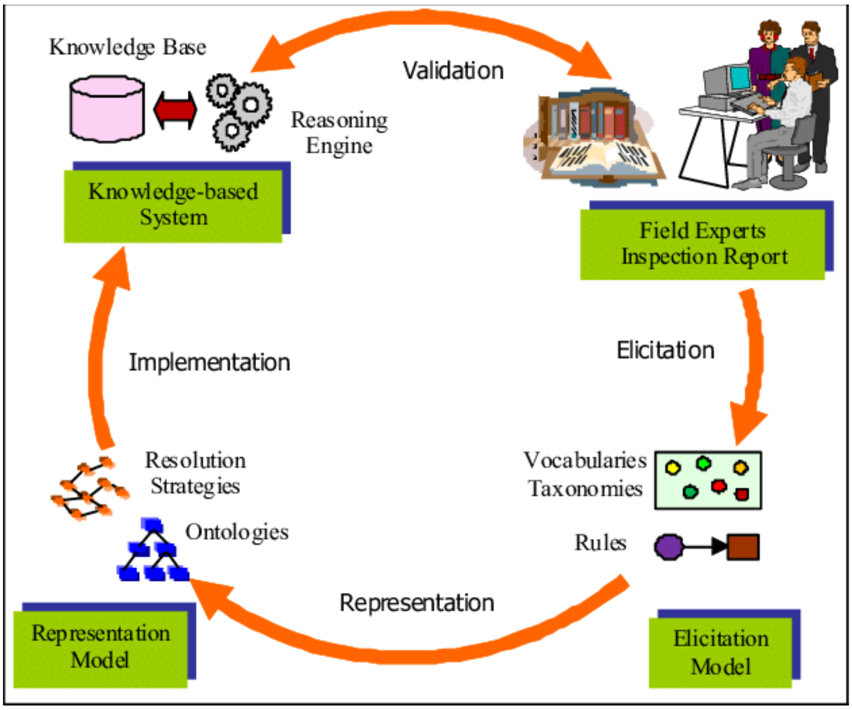


Figure 11. KBS development generalized framework. Source: [153]

Scope and assumptions

The focus of this Master's thesis revolves around the development of an ES and its specific emphasis on two crucial steps within the design process: Elicitation and Representation. These steps hold significant importance in addressing the fundamental questions outlined in Section 1.2.2 regarding the concept of building blocks. With this in mind, the ES aims to fulfill the following objectives:

In our perception, building blocks are synonymous with the associated files linked to design tasks. These design tasks, or rather the functions they perform, can be viewed as fragments of knowledge within a dynamic design workflow. The ES harnesses the potential of these design tasks to make informed decisions when selecting the relevant files for user queries. It is essential to note that the design tasks serve to provide the context in which the building blocks (design tasks) and their corresponding files can be utilized. It offers a recommended framework of the design workflow and possibly be extended into a step-by-step instructional guide to the design but by no mean an established rigid design sequence that must be followed.

To streamline the knowledge requirement and present a representative design process, our focus will be on a specific type of bridge: the concrete girder bridge. We have intentionally excluded the conceptual and preliminary design phases from the scope of this study. This decision was driven by the fact that effectively executing these tasks would entail grappling with tacit and heuristic knowledge, necessitating either its explicit conversion or an exceedingly sophisticated reasoning system. Given the constraints of time and resources within this thesis, such an undertaking was deemed impractical. Consequently, we opted to concentrate on the detail design phase, which offerred a lot of explicit information, making it more easier to modeling and representation.

Considering the aims and objectives outlined in Section 1.3, the scope of our work is bounded by the goal of developing a proof-of-concept product. This approach enables us to evaluate the feasibility and efficacy of the proposed solution in real-world scenarios before embarking on the actual development. Hence, our modeling process will not encompass a comprehensive representation but rather focus on acquiring sufficient knowledge to depict typical design flows and behaviors, serving as a foundation for validating the proposed idea.

Certain assumptions were made during the modeling of the design process to keep it manageable. Firstly, it was assumed that there would be no cross-discipline activities or information exchange. This means that the Structural Design team is the only discipline designing the bridge system, eliminating the need for a discipline hierarchy. Also, there would be no managing work in the design process since they can be considered as 2 different functions [154], [155]. The managerial work is project-specific and happens in parallel to the design work [156]. The author deems it not relevant to the topic at hand.

Additionally, it was assumed that all the information in the model would be determined information. This assumption negated the necessity to model questions, queries, and information requests. Similar to how a designer would require a well-defined brief before designing a project, the ES could only consider goals that had been explicitly instructed [157].

The last assumption was that all information transfer would occur through records of digital mediums, making it possible to document and record all exchanges. This ensured that information was well-documented and accessible as explicit knowledge.

6 Knowledge Elicitation

6.1 Elicitation methodology

Knowledge elicitation is the crucial first step in our development process. It is not only challenging but also time-consuming in nature, which tends to impose a significant bottleneck on the overall construction of knowledge-based systems[158]. The extent of detail attainable during the elicitation process is typically bounded by the available knowledge, the system's intended purpose, and the specific type of explanation sought by the end user. Two main methods were used in this thesis to tackle this endeavor effectively: expert interviews and literature reviews.

The first method, and also the most common one, is to explore the availability of experts in bridge design who would be inclined to participate in our knowledge elicitation process. In complex and diverse domains like structural design, it is common for experts to possess proficiency limited to a specific subset within the domain. Even in a seemingly unified domain, there is a multitude of distinct expertise at closer look [159] [160]. By gaining multiple perspectives from different experts, the knowledge model is grounded on a shared knowledge foundation within the domain. However, this thesis failed to perform this first method in a significant extend.

Many factors constrained this thesis from acquiring the necessary expert sources and limiting the level of detail of the models. Firstly, the knowledge elicitation process was constrained by the willingness and limited amount of time available for the expert. Experts are busy people with their own tasks, and it is difficult to ask them to dedicate significant time to the iterative development of the ES. It also didn’t help that the theoretical basis of the building block concept was still evolving and lacked clarity at the time of implementation. This made it harder to determine the scope of information required from experts. As a result, due to the time constraints of the thesis, the elicitation process had to proceed with the ambiguous collected information. The finished form of the building block in the future should be clear and validated from real-life experience of multiple experts. However, it is possible that even with multiple experts’ participation a comprehensive and unified KR would not emerge as contradictions and conflicts that may turn up when eliciting knowledge from multiple experts[161]

Due to the mentioned challenges, a decision was made to expedite the creation of a working prototype within a short timeframe using the expertise of a single individual, who is also my Advisor. This approach can facilitate more effective communication of the research objective as it develops. Focusing on the prototype can also demonstrate the potential of the proof-of-concept product to potential future collaborators who could contribute during the model-building stage. Despite this, it is important to acknowledge that the chosen expert was significantly occupied and could only provide feedback and recommendations for our model. In the absence of a comprehensive interview, knowledge acquisition primarily relied on personal experience and other elicitation methods. As a student with limited experience in bridge designing, the model may contain inaccuracies and omissions. However, the process model still captures the essential characteristics of a typical bridge design process.

Paralleled to the expert interviewing process, an extensive literature review was conducted. The purpose is to enhance the understanding of the relevant concepts utilized in the model as well as to incorporate insights from relevant scientific studies without starting from scratch. The AEC sector has a plethora of ontology-related studies, primarily about vertical buildings and bridge inspection and evaluation. Notably, these studies can be categorized into three groups: general knowledge modeling, information extraction and sharing, and reasoning and conformance checking [162]

The first group focus on KM through the creation of general domain KBs. These KBs are not specific to any particular application context and serve as a foundation for developing specific ontologies. Various general ontologies have been developed to formalize knowledge related to building and infrastructure construction, encompassing important concepts such as products, stakeholders, and processes [163], [164]. Other efforts shift the target to BIM, where various ontologies have been created to enhance information sharing and collaboration. One notable example is the IFC (Industry Foundation Classes) developed by BuildingSMART, which serves as a widely used platform for seamless exchange of construction data [165]. Additionally, the e-COGNOS project introduced a prototype ontology for the construction domain, enabling semantic KM and supporting activities such as semantic indexing, information retrieval, and ontology-based collaboration.[166]

Both the second and third group is context-dependent. For the information extraction and sharing group, studies concentrate on establishing semantic connections between various sources of information and utilizing queries to retrieve relevant data[85]. This means two searching techniques are supported, keywords match and browsing semantically related information. Not limited to semantic information, file link containing the established relation can be added, eliminating the need to contact the designer. For example, [167] linked building design elements with semantic information to manage design constraints which allows collaboration between multiple users on a project.

Ontologies in the second group can be further subcategorized into object-oriented and process-oriented. The former organizes based on physical objects, such as building elements like walls and windows, into taxonomies, while the latter focuses on sequencing and constraining tasks [168]. Object-oriented ontologies are more prominent due to the availability of various physical elements taxonomies, such as UNIFORMA for building elements[169]. Furthermore, they are primarily used to store extracted static information from digital sources, such as building engineering systems [170], damage [171], and structural health [172]. On the other hand, simplified process-oriented ontologies were developed to store progress-related information, such as tracking and monitoring project progress using qualitative metrics [168], masonry work procedures to facilitate risk information retrieval [173], and energy plant projects constraint information [174]

For the third and final group, reasoning work is performed using rules and reasoners to facilitate complex decision-making processes and ensure compliance. This approach allows for the identification of implicit knowledge relating to specific attributes or relationships between entities. Examples of compliance checks include: embedded safety rules in the ontology that detect safety regulations violations in building elements and areas [173] and condition evaluation rules that assess the condition of individual components and the overall bridge[171]. Examples of decision-making include: PV-TONS ontology to enhance selection of photovoltaic systems [175], rule-based decision-making for manufacturers in the Chinese steel industry [176], and the design of complex railway portal frames [177].

Specific application of KBS and ES in the field of bridge design has been relatively limited, mainly focusing on conceptual and preliminary design stages, as well as evaluation and rating processes [102]. Particularly, the purposes of bridge KBS has been for rating, system selection, damage assessment, planning, evaluation, and management. Some tools used in conceptual and preliminary design utilize a database of past bridge projects to provide solutions for new problems through information matching or more advanced methods like Fuzzy Logic, Artificial Neural Networks, Genetic Algorithms, etc. Existing bridge engineering ontologies and KBS were not built to be design repositories and there is a lack of necessary domain-specific knowledge to effectively manage bridge design information, such as constraints and tasks [85].

6.2 Informal Model

6.2.1 Choice of Representation

At the end of knowledge elicitation, an elicitation model is produced. It acts as an intermediate model, or informal model, to analyze the acquired knowledge effectively alongside the prototype development. The purpose was not to create a comprehensive and formal paper model of expertise but rather to construct an informal model that could serve as a valuable tool for organizing, structuring, and analyzing specific subsets of domain knowledge. This informal model could be adapted into a formal model in the following Section 7.2. Also, as noted by [178], when using human language and graphics to display interconnected semantic patterns, the informal model is easy to read and understand. It can act as a communication tool utilized during the elicitation and validation processes, as well as to store knowledge for future reuse[178]. Through the informal model, fundamental aspects of building block concepts that support knowledge reuse, including their form (choice of representation), size (granularity and abstraction), and connections (attributes and properties), are answered. It also attempted to meet the requirements of a reusable KB from Section 4.5

Various types of graphical models can be used to represent the structural design process [179] but IDEF0 (Icam DEFinition for Function Modeling) has been recognized as the most suitable technique [7], [179], [180]. IDEF0 is a type of flowchart model that allows the representation of a process from an information perspective rather than focusing on the temporal one.

According to [156], the IDEF0 technique possesses three key characteristics that make it a suitable tool for modeling systems of interconnected processes or tasks at different levels of granularity and with encapsulation:

Firstly, IDEF0 adopts a data-centric perspective, showing the transformation of information as it travels through different paths and coordination in a system. Unlike other approaches, IDEF0 does not impose or document any temporal control over the information flow, thus it does not represent a predetermined sequence of task execution. This flexibility also allows for modeling and understanding of iterative tasks by providing the mechanisms of information transfer between the coupled tasks in the iterative cycle.

The second and third characteristics show that IDEF0 offers modeling options based on the availability of building blocks and the level of detail required. It does not mandate the modeling of individual design tasks. Instead, each design task can be treated as an encapsulated entity, with a focus on studying the input and output of information. This enables the analysis of information flows and transformations without the need for detailed modeling of every design task.

On the other hand, IDEF0 also allows detailed modeling of a design task by breaking it down into subtasks. The top level of the model provides a high-level overview of the design process, while the lower levels offer increasing levels of detail. This hierarchical structure allows users to obtain an overall understanding of the system by examining the higher regions of the model while accessing more detailed information from lower levels as necessary. More detail on this is in Section 6.3.2. IDEF0 can be constructed in a top-down manner, subdividing tasks, or in a bottom-up manner, aggregating tasks. Often, a combination of both approaches is appropriate.

Moreover, the IDEF0 model served as an informal model, as mentioned above, to store elicited knowledge, facilitating efficient communication of the knowledge gathered to the expert. This allowed for a thorough review before implementation into the machine. In our case, IDEF0's simplicity and ease to understand were crucial factors, particularly when rapid modifications were required during the proof-of-concept development and in the future when building blocks are added or changed.

6.2.2 IDEF0 Components

This section will cover the basic techniques used in IDEF0 that are relevant to our topic at hand. Additional modeling instructions of the IDEF0 methodology can be found in [181]. It is noted that in this thesis, the term "function" was used because the process was modeled with information requirements and not as a time-based sequence, so "function" was more apt than "process" or "activity". Nonetheless, the terms were used interchangeably throughout the rest of the thesis.

In the IDEF0 methodology, the main elements of the modeling process are functions (activities, actions, processes, operations…), depicted as boxes on a diagram. The data and objects that connect and interact with these functions are represented by arrows. The collection of the 4 different types of arrows (Inputs, Controls, Outputs, and Mechanisms) can be called ICOM. Additionally, IDEF0 utilizes a combination of natural language and visual representations to effectively communicate the essence of a given process. More detail can be seen in Figure 12.

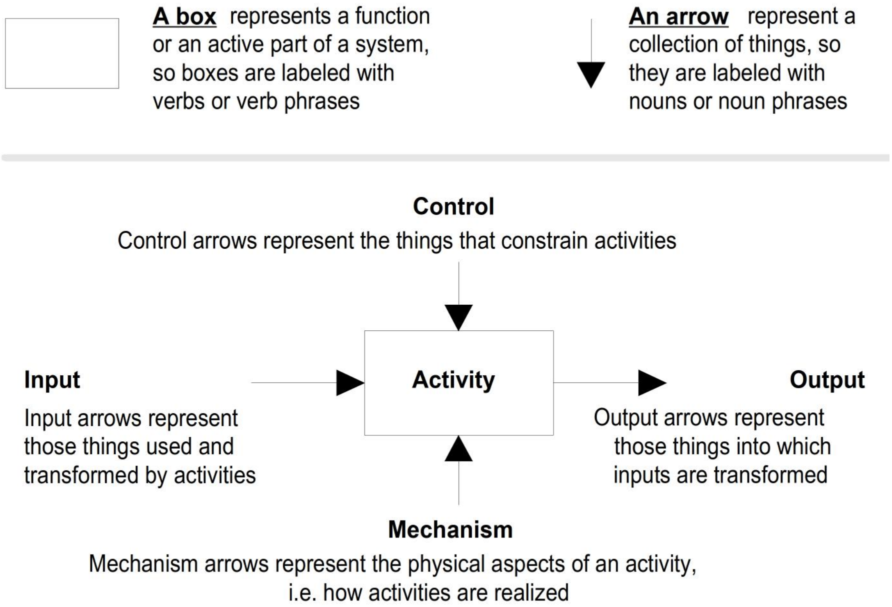

A box represents a function or an active part of a system, so boxes are labeled with verbs or verb phrases
An arrow represent a collection of things, so they are labeled with nouns or noun phrases
Control
Control arrows represent the things that constrain activities
Input
Input arrows represent those things used and transformed by activities
Activity
Output
Output arrows represent those things into which inputs are transformed
Mechanism
Mechanism arrows represent the physical aspects of an activity, i.e. how activities are realized

Figure 12. The basic components of IDEF0. Each side of the box gives different roles to the arrows. Source: [114]

In principle, IDEF0 models the input-process-output (IPO) pattern with additional information requirements from control and mechanism. The design activity utilizes available constraints and resources to convert input into output. The output generated by one function can subsequently serve as input or impose constraints and mechanisms on another function, establishing a sequence of information dependency. Thus, by connecting multiple function boxes through ICOM arrows, the IDEF0 model becomes a coordinated set of diagrams. Furthermore, IDEF0 models aggregates and arrange diagrams into a hierarchical structure, where the diagrams at the higher level offer a lower level of detail compared to those at the lower level. This is illustrated by the linked diagram of a decomposed model in Figure 13.

The building block items can be categorized into the 4 information requirements. The categories are not static but rather depend on the project context [182]. For instance, the output generated from one activity can serve as an input or control for another activity.

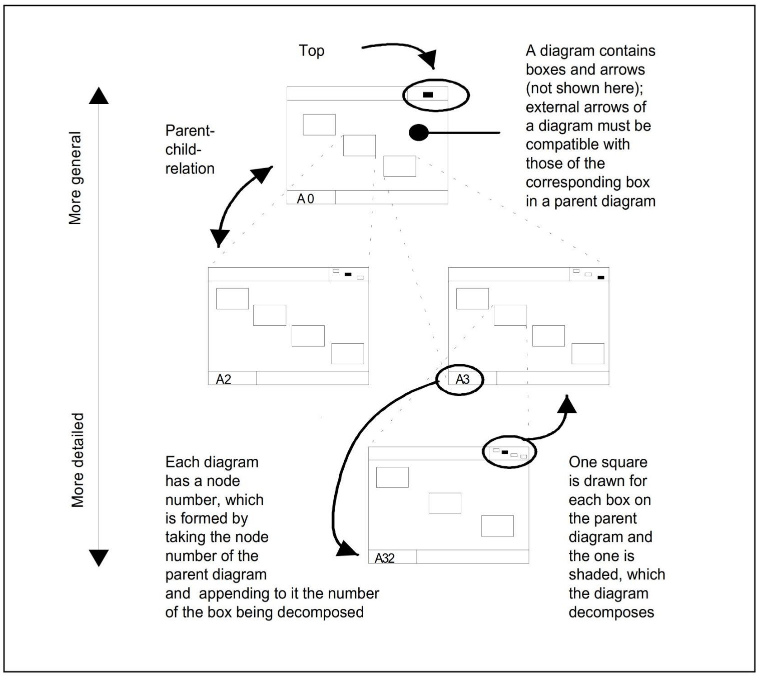

Top
A diagram contains boxes and arrows (not shown here); external arrows of a diagram must be compatible with those of the corresponding box in a parent diagram
More general
Parent-child-relation
A0
A2
A3
More detailed
Each diagram has a node number, which is formed by taking the node number of the parent diagram and appending to it the number of the box being decomposed
A32
One square is drawn for each box on the parent diagram and the one is shaded, which the diagram decomposes

Figure 13. Hierarchical structure of IDEF0 model using parent-child diagrams. Source: [114]

For the informal model, this thesis used the IDEF0 method developed for the IDM package as described in [7] with some minor changes to suit the building block concept. The authors of that paper took the traditional IDEF0 method and refined it to make it more appropriate to model design activities and information requirements. The methodology attempts to standardize and modularize the AEC process flows and their information contents while acknowledging that the nature of design projects is unique, dynamic, iterative, and interdependent. The adapted IDEF0 notations based on their refinements are illustrated in Figure 14.

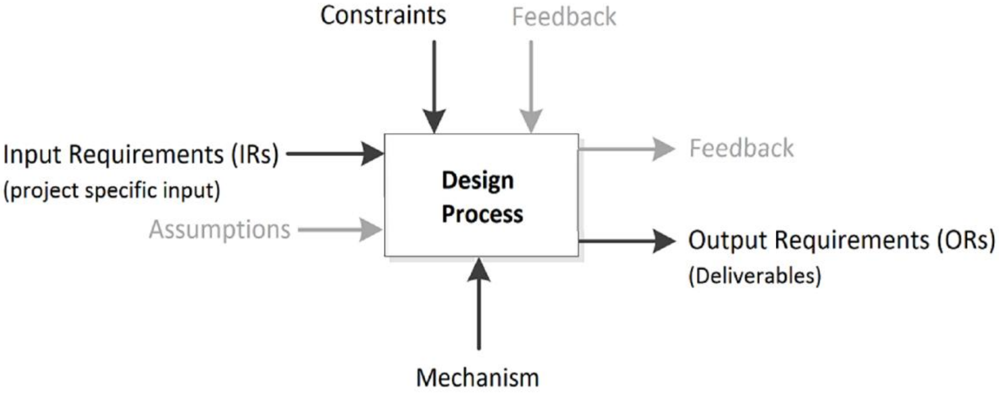


Figure 14. The adapted components of IDEF0. The greyed information is not formalized. Adapted: [7]

Mechanism encompasses various means or resources through which the design process is performed to achieve accurate outcomes. It is represented by specific steps, actions, and tools. While mechanisms typically involve human actors, our focus here lies solely on digital knowledge and tools. Thus, mechanisms can be either automated or manual, with examples including software, GH scripts, BIM and analysis models, Excel calculation tables, and standardized beam profiles, see Figure 15.

Controls be categorized into two aspects: constraints and feedback. Constraints refer to external requirements that define the quality requirements or desired properties of the structure or its components. Controls can take various forms, such as national or local standards, technical specifications, design guidelines, client demands, and specific logical or physical criteria... Alternatively, feedback is a part of the iterative design process, providing valuable information on how input can be transformed to enhance the outcomes of another design process. Feedback is a type of output and will be discussed more below. This division of control aims to distinguish between information from external sources, readily available to designers, and internal sources, where timely information delivery is required. In the current model, only constraints are modeled since they are determined as part of the possible multimedia building blocks, see Figure 15. Feedback is not modeled because of time and complexity. But it is important in the future semantic network, especially to infer connections between building blocks.

There are two types of inputs. The first type is Assumption, reflecting the necessity for designers to set domain-specific conditions to be true. Due to the fast pace of the structural design process, not all information is available in time and thus, assumptions are made based on engineering experience to initiate the design process. Therefore, assumptions are required as part of the engineer's decision-making process since the chosen assumption can influence which constraints or mechanisms are applicable in the design activity. Assumptions must be reasonable within the scope of the design constraints or rather, constraints set the boundaries of the available assumptions. These assumptions tend to vary depending on the project and client. However, this thesis assumed that the informal model would have all the prerequisite information. Therefore, despite being potentially valuable to reasoning and inference in the KR stage, Assumption was not part of the informal model. It might serve as a valuable resource for querying and inferencing and should be implemented in the future.

The second type is project-specific input. It is limited to the minimum and essential information required for the smooth functioning of the process, for example, deck length and depth, main material of bridge, and other relevant information. Non-essential elements and assumptions should not be included since many of those are likely to be unavailable, especially at the beginning of the project. These points directly correspond to the first type of output, Solution.

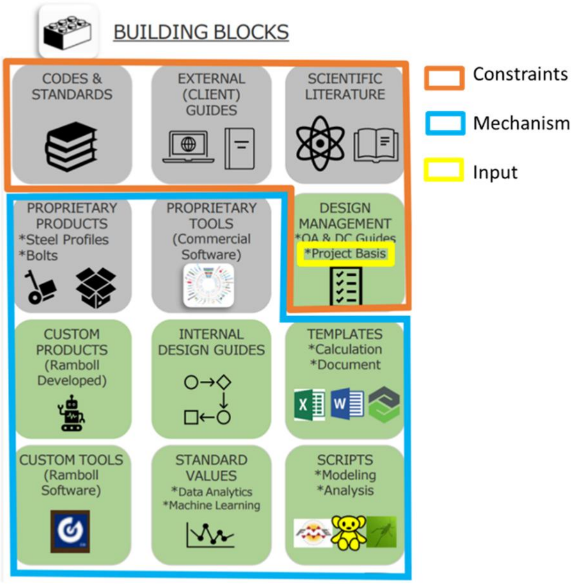

BUILDING BLOCKS
CODES & STANDARDS
EXTERNAL (CLIENT) GUIDES
SCIENTIFIC LITERATURE
PROPRIETARY PRODUCTS *Steel Profiles *Bolts
PROPRIETARY TOOLS (Commercial Software)
DESIGN MANAGEMENT *QA & DC Guides *Project Basis
CUSTOM PRODUCTS (Ramboll Developed)
INTERNAL DESIGN GUIDES
TEMPLATES *Calculation *Document
CUSTOM TOOLS (Ramboll Software)
STANDARD VALUES *Data Analytics *Machine Learning
SCRIPTS *Modeling *Analysis
Constraints
Mechanism
Input

Figure 15. Possible building blocks categorized into ICOM types.

The first output type, Output requirement (Solution), involves transforming inputs into models, drawings, and documents to represent the outcomes of the current design process. These outputs serve as the solutions that keep the information flow moving forward and are not part of the iteration loop, a tangible representation of the project's progress and results. This is shown in Figure 16

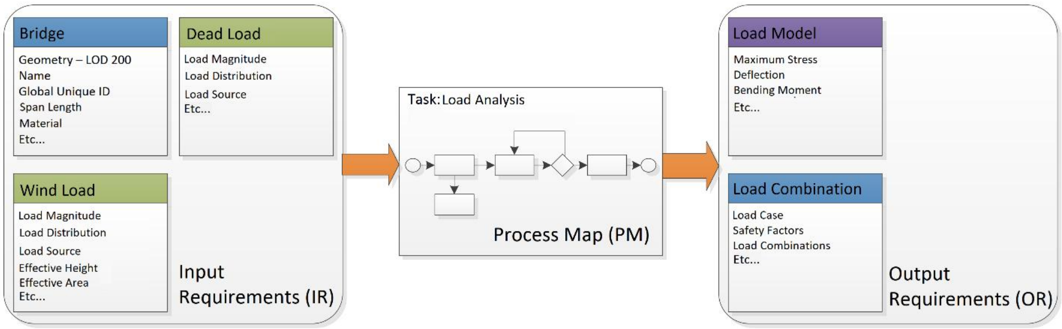


Figure 16. Example of Input requirements and the transformation into Output requirements. Adapted: [7]

The second type of output is Feedback. As mentioned, it plays a crucial role in the iterative design process as any type of information from the Output of one process can be the Control of another. Although the IDEF0 flow diagram appears linear, most processes or clusters of processes (children processes of the parent process) are iterated multiple times to achieve the desired design. The Feedback output can become a previous activity’s control, affecting its output solution and resulting in a design loop. Design loops allow for continuous improvement in the project. More detail about this behavior is discussed in Section 6.3.1

6.3 Informal Model Construction

With the IDEF0 components explained, we can model the 3 types of relationships in a design process: dependency, hierarchy, and substitution. The effective utilization of building blocks with the IDEF0 model requires the consideration of all three relationships. Dependency provides insights into information exchange requirements and the positioning of building blocks within the design sequence. Hierarchy, on the other hand, determines the spatial and level of detail decomposition of building blocks. Lastly, alternative design options facilitate specific project derivations.

It is noted that the information on structural design can be categorized into product information and process information [183], [184]. To capture and organize such information, a product model and a process model are employed [185]. The product model serves as a host for various information about structural systems, such as bridge type or bridge components, while the process model outlines the interconnections, sequence, and structure of activities within a process. Only the Hierarchy and Substitution relationship exist in the product model while all three types of relationships exist in the process model. Furthermore, since the information requirements of one model can influence or even determine the information requirements of the other, it is crucial to maintain a similar granularity level between the two models.

6.3.1 Dependency Relationship

In understanding the challenges of a design process, it is crucial to grasp the foundational dependency of information, which creates a sequential relationship [186]. Dependency reveals the related tasks and the design information they require and produce. However, it does not dictate a specific order in which tasks should be carried out; its purpose is solely to map the path of information through the design process. A look into this type of relationship reveals the nature of task connectivity, distinguishing between parallel or pooled (independent), sequential (dependent), and coupled or reciprocal (interdependent) [187], illustrated in Figure 17.

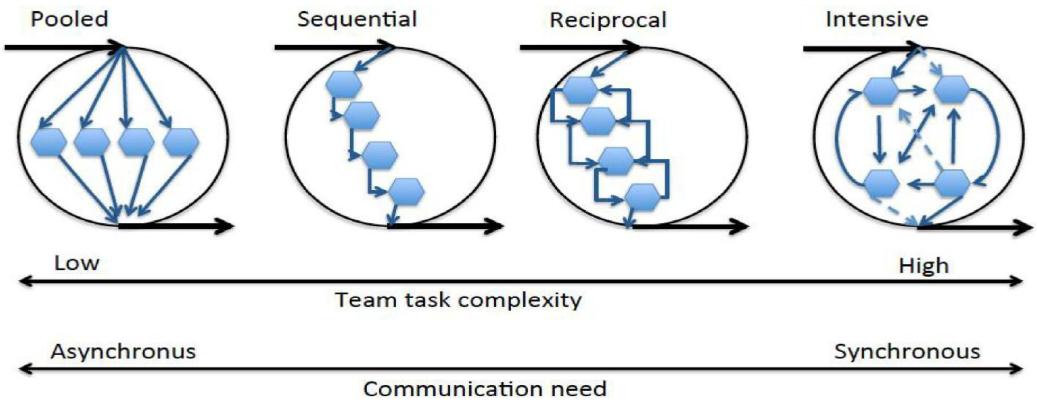


Figure 17. Different types of dependency are sorted by complexity and collaboration level. Source: [14], [186], [187]

According to [1] and [2], pooled and sequential dependencies naturally occur in designing well-defined problems. In pooled dependency, individual team members perform separate work and activities that are later combined to create the final product. In the sequential arrangement, work and activities flow in a unidirectional manner from one team member to another. In order to proceed, a decision must be made regarding an element or structure. Otherwise, the process will either come to a halt or won't even begin. A concrete decision in favor of a solution may initiate a sequential process, whereas a decision against a solution may trigger a new reciprocal process. The reflective logic of reciprocal and intensive dependencies is needed in wicked or ill-defined design problems. Reciprocal is characterized by work and activities flowing back and forth between team members one by one over time. Finally, the most interdependent arrangement, intensive, requires team members to simultaneously collaborate as one single body to accomplish their task.

There are two main types of iteration in design, illustrated in Figure 18: 1) iterations between a set of coupled design processes executed simultaneously as mentioned above and 2) iterations that occur within a sequential design process, repeated in design loops. Type 2 of iteration can happen, for instance, the architect creates a design proposal, which would then be analyzed by engineers and followed by simulations that provide feedback to adjust the initial proposal. The type of iteration involved influences the decision to cluster tasks into activities or keep them separate. Coupled tasks are typically combined, while iterations within a sequential loop are better separated to demonstrate the dependency based on the required assumptions to initiate the loop.

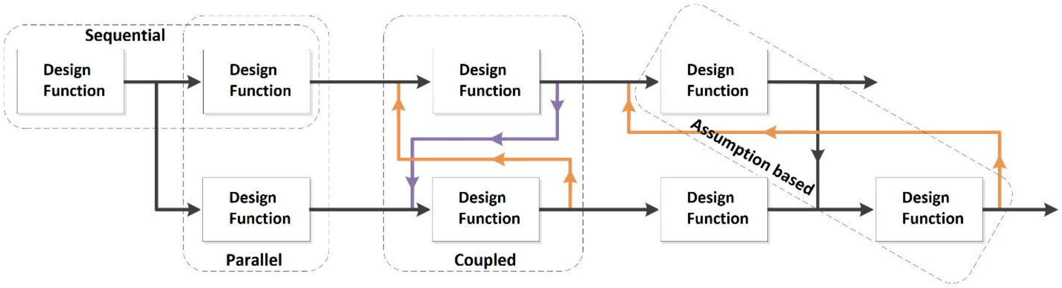


Figure 18. Different types of dependency in a design workflow. Adapted: [7]

The majority of structural design deals with reciprocal tasks, particularly in the initial stages. During the design phase, teams contend with a multitude of interdependencies and complex tasks as they seek optimal solutions. It is common to see the design criteria and the potential solutions are intertwined, making it challenging to have a clear understanding of the logical predecessors ahead of time [14]. For instance, Task A relies on input from Task B to progress, while Task B requires input from Task A to deliver its output. The design proceeds by going through many design loops as additional knowledge is acquired after each loop. This nature makes the design process complex to model, as it involves a reflective logic that encompasses reciprocal, iterative, and intensive procedures. However, as design problems are resolved, interdependencies decrease, and the tasks become more simple and linear in nature (e.g. drawing completion) [186].

6.3.2 Hierarchy Relationship

Another important relationship in the structural process model is the hierarchy relationship. Each activity within the model involves the IPO pattern without explicitly modeling the internal transformation procedure. When the need to expand an activity and show the procedure, IDEF0 allows the decomposition of an activity and represents its finer details on separate child diagrams. This approach ensures clarity and prevents overwhelming a single diagram, making them too complex.

The most prominent kind of hierarchical relationship is the decomposition of a bridge into its structural parts. It is part of the building block concept's goal to develop a system that can relate design processes to the designed bridge element. This allows the process model to be integrated into a larger, more intricate system. Thus, it is important to develop a complementary product model that acts as a base to break down the process model. This product model also provides additional properties of the structural elements and relationships to other models, such as material or construction methods, to the associated design process. The granularity of the process model is also influenced by this model of decomposed bridge parts. The IFC bridge parts acted as the basis of this thesis' product model [189]

The hierarchical structuring of a design process model does not stop at structural decomposition but also process decomposition, breaking down parent processes into child tasks. This is crucial for two main reasons: to simplify the modeling and enable the reuse of processes [190]. By decomposing activities into more detailed levels, sub-processes and tasks can be documented comprehensively, allowing for a higher-level design process to be effectively represented. According to [191-194], the levels of design process can be categorized and defined as follow:

- Project: The highest level. It can be considered as an end-to-end process, involving a comprehensive process that guides a system from inception to completion, delivering a fully functional solution that caters to the customer's requirements (also known as project deliverables). This is broken down into various design activities, including processes, sub-processes, and tasks, all aimed at fulfilling the customer's needs and ensuring a successful outcome.

- Process: A collection of interconnected Sub-Processes that collaborate to accomplish a specific objective that generates value (e.g. Phase deliverable). It starts with work acquisition and concludes with the delivery of a product or service of value to a customer. With this in mind, a design process of a project can be seen as a division of design phases

- Sub-Process: The line between Process and Sub-Process is blurry. It represents the typical IPO pattern that transforms input, adds values to them, and provides output. In this sense, one can associate each deliverable item with a corresponding sub-process, which may be linked to a specific structural element. Process and sub-process are likely to mirror the division of parts in the product model.

- Task: A broken-down component of a process or sub-process, representing an action performed by a role, system, or organization with specific ICOMs. It can be seen as a small milestone within a larger process, where it produces meaningful results, although not directly the final deliverable. Essentially, all activities that significantly contribute to achieving the desired outcome are integral parts of a process, e.g. calculating wind force and finding critical bending moment. Tasks, being the smallest elements of value in a process or sub-process, should not involve contain decision-making within and thus should not be divided further.

- Procedure: a set of steps, represents the atomic activity (it cannot be decomposed further). In simpler terms, a procedure can be understood as a step-by-step instruction given to someone to complete a task. It does not constitute meaningful work, e.g. calculating coefficient and finding characteristic strength. Procedures restrict the task to an order of steps to be accomplished. Therefore, the task divided into procedures is at risk of bringing restrictions and limiting the creativity of engineers. The only exception is in cases where a task necessitates specific instructions or descriptions for its execution.

This hierarchy is the basis to organize textual or non-textual building blocks into ICOMs of various functions in the IDEF0 model. ICOMs of parent activity are split and matched to the appropriate child activities, i.e., the ICOMs of child activities are aggregated when moving up the hierarchy. This is shown in Figure 19.

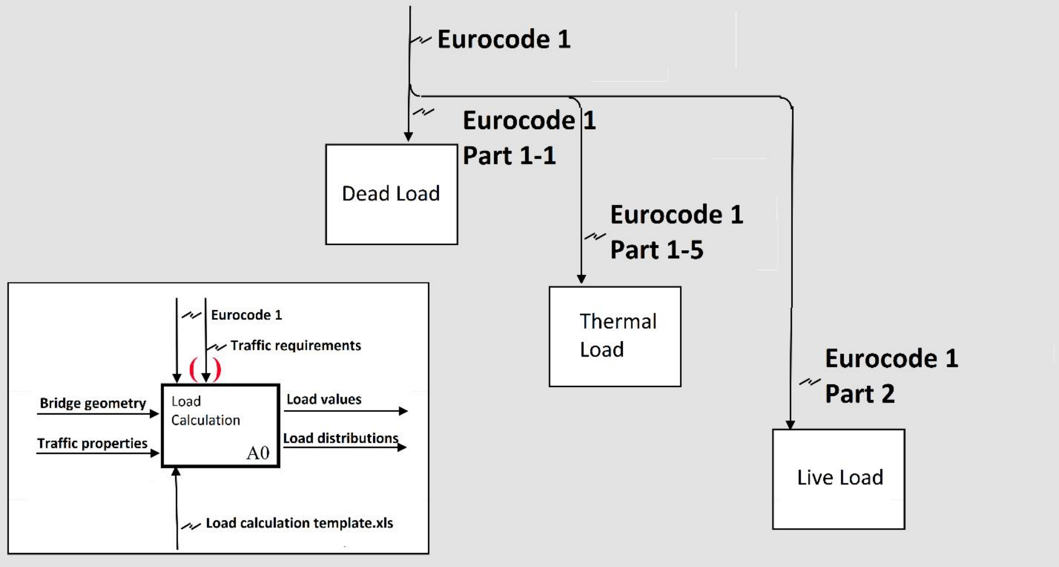


Figure 19. Bundling behavior of controls. Adapted: [195]

Typically, the decomposition preference generally lies in favor of maximum depth. Within computer programming, functional decomposition aims to modularize processes to the highest possible extent. In systems engineering, functional decomposition is a method for analyzing engineered systems. The purpose is to divide a system in a manner that allows each block within the diagram to be described independently, without the inclusion of "and" or "or" statements. This way, pure functions for each component of the system are achieved and enable reusability and replaceability. The reason is that simplified and generic interfaces between modules emerge as a significant byproduct of this decomposition, facilitating the substitution of related or similar pure functions[196].

Despite that, it was deemed inappropriate in our specific application to break down all the processes to the task level. Such an approach would result in an overwhelming number of activities, potentially providing irrelevant information and hindering clarity [197]. The depth of functional models should be adapted based on the specific requirements they need to fulfill, in our case different types of textual and non-textual documents and CD scripts.

This concept, known as multi-granularity, recognizes the necessity of both high and low levels of detail. Low-level granularity is essential for large-scale and complex design tasks that require substantial tacit knowledge, such as the automated reinforcement design for deck slabs. Such generative design process typically involves multiple stages [198], represented by sizable Grasshopper scripts, ultimately producing specific structural components, and in our example is the complete reinforcement layout of a deck. On the other hand, high-level granularity is employed for individual practical design tasks, such as automating documentation processes like generating 2D drawings and populating them with tags for names and numbering. Additionally, small code snippets can be utilized for minor tasks like text replacement, orienting cross-sections, center lining mesh pipes… As the KB system evolves and building blocks are populated, more granular building blocks are expected to be more dominant in number, catering to various specific design scenarios.

6.3.3 Substitution relationship:

The aim of the IDEF0 is to provide a generalized overview of the design process, encompassing common systems and elements[180], [199]. Such a model represents a typical but not actual process, offering suggestive examples of various aspects such as dependency, methods, and the roles of individuals [190]. It can be called a reference model or the term "intermediate version" is also used to convey that the model should be adaptable to specific cases without significant compromises [114]. Reference models main purpose is to facilitate the reuse of knowledge in design process, providing suggestive examples that must be customized to suit each unique scenario.

As mentioned, reference models require certain adjustments when applying for specific projects. At this stage, the model becomes a specific model, capturing information about the actual design process and reflecting the perception of the design engineer [190]. This is shown in Figure 20. However, there exists a significant disagreement among domain experts regarding the level of uniqueness, repeatability, and best practices of design processes. Even similar use cases have varied design processes across AEC companies and bridge engineers, with certain variations proving more efficient than others. [7] compared these design processes from their documentation and concluded that all use cases demonstrate potential for optimization. Therefore, to maximize the effective reuse of existing knowledge, the objective is to develop a generic representation model that includes a diverse set of design options that are customizable for different contexts, such as construction methods, site conditions, and client requirements [180].

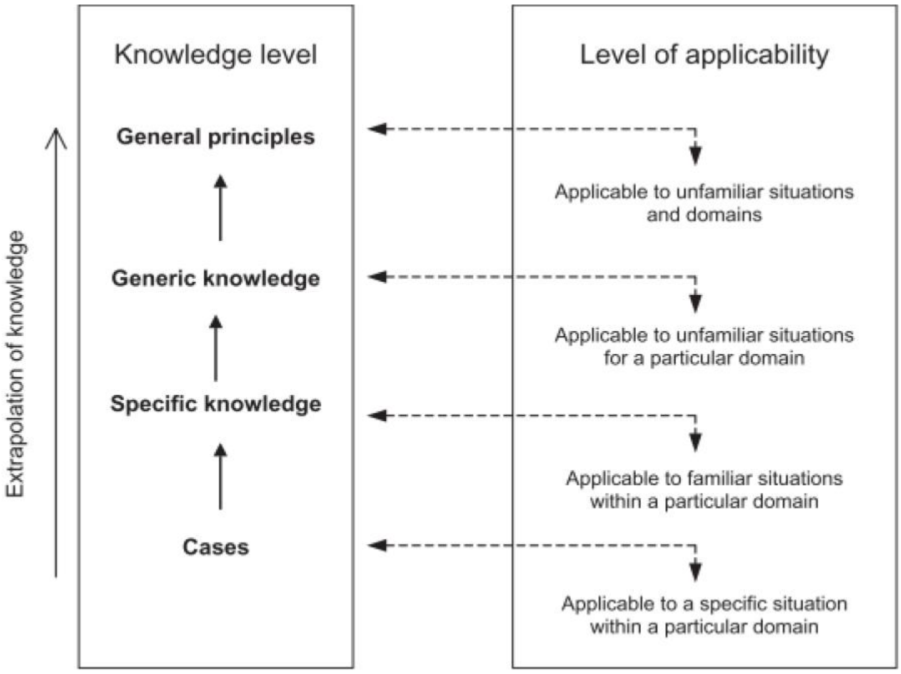

Knowledge level
Level of applicability
Extrapolation of knowledge
General principles
Generic knowledge
Specific knowledge
Cases
Applicable to unfamiliar situations and domains
Applicable to unfamiliar situations for a particular domain
Applicable to familiar situations within a particular domain
Applicable to a specific situation within a particular domain

Figure 20. Knowledge levels and states of applicability. Source: [200]

In order to achieve this, [156] recommended that the reference model serves as a framework, complemented by submodels that represent design alternatives. The aim is to create a single model capable of encompassing the design of a "general" bridge. It should comprise a typical structural process with components that can be swapped with smaller, distinct submodels. Within IDEF0, this concept can be represented in the same manner as parent-child activities, or in this case, generic activity and alternative options. The difference is that the alternative options have no dependency on each other [199], illustrated in Figure 21. For instance, the foundation of a bridge could be designed as spread footing or pile slab, which entail distinct processes and information dependencies and thus, they necessitate separate submodels. The alternative options can be decomposed into corresponding submodels. By selecting the appropriate variety of submodels combination and integrating them into the base “skeleton”, specific models for different projects can be constructed.

To ensure the seamless interaction of substituted activities with other types of relationship, especially dependency relationship, it is crucial to standardize the input and output of functions through encapsulation. Encapsulation, a technique employed in computer programming, minimizes inter-dependencies among separate modules by defining strict external interfaces. By establishing a clear contact between a module and its dependency, encapsulation allows for internal modifications of the module without affecting the external functionality relied upon by other functions [201]. To put it simply, modularity is achieved by highly cohesive and loosely coupled activity clusters. This approach also enables alternative options, as it operates based on abstract data types inheritance (Liskov substitution principle). Substitution lies at the core of this inheritance type, where one module can exhibit the same behavior and characteristics as another [202]. It also explains why alternative options can be modeled using parent-child boxes since inheritance is a hierarchical feature.

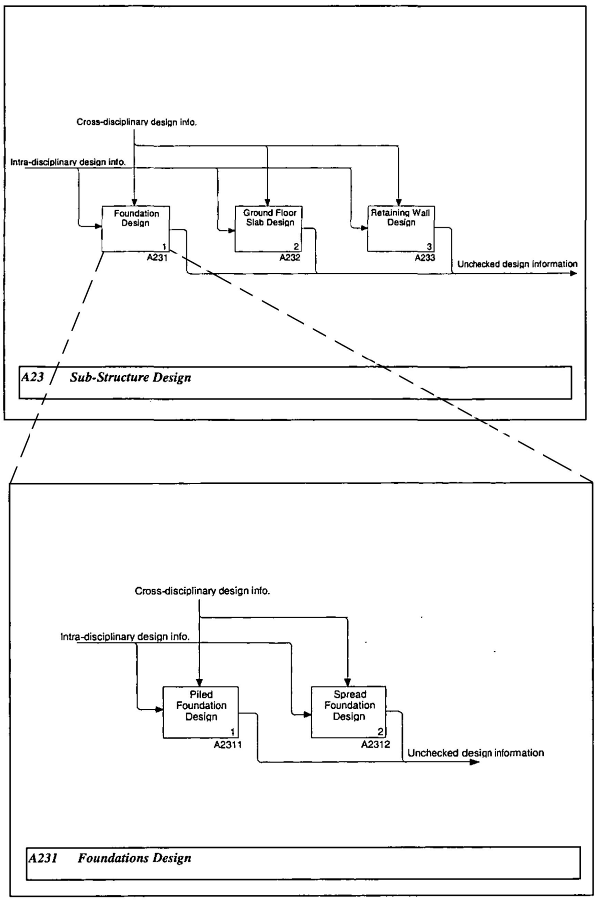


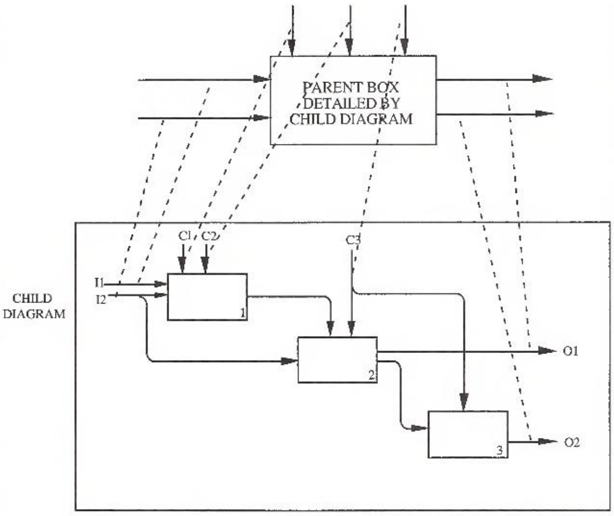
Figure 21. Generic activity and alternative options. Source: [199]

Clustering and encapsulation are closely related concepts when dealing with iteration loops. Clustering involves grouping interconnected elements of a model into clusters while minimizing connectivity outside of those clusters [203]. Clustering forms groups based on relationships between design processes, such as task sequences or shared responsibilities of engineers. It is commonly used to modularize mechanical products or software functions, where tightly coupled components are grouped into modules that are loosely coupled with the rest of the system [204]. Well-designed functions only have a number of necessary parameters that are closely aligned with the functionality, reducing the likelihood of unintended side effects and improving reusability and comprehension. The goal in applying this is to achieve highly cohesive functions that perform a well-defined set of tasks with clear input-output relationships, enhancing clarity, understandability, and modularity [205].

However, encapsulation brings attention to some ICOM behavior complications:

- ICOM arrows of an activity create the boundaries of its corresponding decomposition diagrams. They should match the boundary arrows (arrow with one end not connected to any box) in the decomposition of that activity, see Figure 22.

- Controls and mechanisms can be inherited from a generic activity but can also be modified. This is similar to the business constraint of IDM packages, where business rules can be used to vary the result of an exchange requirement without changing the requirement itself [7]. Therefore, modifying the controls and mechanisms can affect the input and output values, but not the input and output requirements themselves, as the controls and mechanisms do not change the position of the block in the model.



NOTE: The dashed lines show how the ICOMs on the child diagram relate boundary arrows on the child to the arrows of its parent box.

Figure 22. Boundary arrows correspondence. An ICOM for a parent box may be ICOMs for one or more of its child boxes. Adapted: [181]

It is worth noting that alternative options may have different numbers of ICOM compared to the generic function. The design option can inherit the information requirements of the generic function while adding additional requirements specific to that option. The generic function should only provide the baseline ICOM to be inherited instead of having all the ICOM of its variations. This is a similar problem to the IDM packages where they increased the number of packages instead of adding additional requirements from different use cases in the same package and making it too complex [7]. However, changes in design options' ICOM must be done with consideration of the level of difference in functionality. Since such changes might turn an alternative option into a completely new function. For instance, removing input and output requirements in an option can change its position in the model and relationship with established dependent activity, making it a new function completely.

6.3.4 Modeling strategy

According to [199], The development of the generic model for detailed structural design involved two distinct stages. In the first stage, we identified the activities comprising the overall design process and established a hierarchical structure for them:

- First level: the process was divided according to the professional disciplines involved, such as architecture, mechanical engineering, structural engineering, etc. This level is based on the existing division of labor among the parties involved in the design process. However, this level is irrelevant to our thesis.

- Second level: By using the Product model as a baseline, we organized each discipline into structural systems, subsystems (joints at the ends of a girder), and sub-subcomponents (bolt groups within the joints).

- Finally, we reached the level of individual design activities, referred to as leaf tasks, such as calculation, drawing, and specifying. At this level, sequence-based thinking was adopted. It facilitated a clear understanding of the subsequent steps in the design process. By organizing interrelated activities together instead of solely grouping on similar concepts, it became easier to discern the logical progression of tasks [114].

The proposed hierarchical structure incorporates two aspects of context in the model formation: granularity and abstraction. The higher levels primarily employ the low-granularity/ high-abstraction, while the choice of design options at lower levels is guided by the high-granularity/ low-abstraction [156].

In the second stage, the information requirements (ICOMs) were identified for each leaf task at the lowest level of the hierarchy. These information requirements were then connected to the corresponding leaf task as appropriate ICOMs. ICOMs were grouped together at the higher levels of the model to maintain clarity and manageability.

Throughout the modeling process, many rules and tips were followed to ensure the model consistency. Some of these rules were derived from business process construction practices [206], which exhibited similarities in modeling approaches to this thesis. The purpose of these rules was to simplify the overall model or specific activities by removing or grouping elements that lack significance.

- Activity Elimination: It identifies activities conducted by an internal process to achieve immediate objectives and provide insignificant value to the deliverable. The activity also lacks an associated building block and has a maximum cardinality of 2 (no hierarchy)

- Coupled Activity Abstraction: Iteration can be incorporated within the process by grouping tasks that form a loop under an abstracted activity. By doing so, the relationship between the loop and preceding or succeeding tasks remains unaffected [199].

- Sequence Activity Abstraction: Aggregating a group of activities when they are executed by the same individuals, occur in a sequential manner, have a maximum of two sequence flows (in and out), and either lack any information requirements or share the same information requirements connected to all of them.

- Significant Activity: Activities involving information exchange between different individuals or requiring decision-making are deemed significant and should not be abstracted or eliminated,

- Avoid Big Cluster Abstraction: The point is to avoid excessive abstraction, especially in loops with sequences that have more than six activities. The big cluster should be minimized by breaking it into smaller ones. This could be a topic by itself and would not be covered in this thesis. Additional information on the assessment and resolving techniques can be found here: [203], [207]

[7] provided some general tips that can be adopted when iteratively constructing the functions:

1. Identify industry best practices and use them as a basis for constructing a function.

2. Describe function in a generic manner that allows for internal optimization by users, without requiring change to external interfaces.

3. Adopting a bottom-up approach for considering the information exchange by prioritizing defining leaf tasks.

4. Accept the need for iterations when defining the ICOM of activity, as they must be harmonized with the surrounding design processes.

5. Design activity primarily serves to define building blocks, so ICOM should be the modeling priority and not the activity.

    a. Implementing a model that prioritizes "activity nature" would result in a challenging and impractical abstract model, requiring significant effort, expertise, and advanced computerized tools to effectively utilize.[114]

Despite the described methodology, the intended IDEF0 diagram was not created. Instead, simplified flowcharts were employed as the main representation scheme, see Figure 23, to capture the 3 types of relationships but without the ICOM. This is due to the fact that IDEF0 was discovered late in the theory refining process, after the creation of the existing flowchart model, and time restrictions prevents translating it into IDEF0. The flowcharts were created in Microsoft Visio, which provides the necessary tools and hyperlink functionality for easy navigation between process and subprocess. Despite this, enough similarities between the main characteristics of an IDEF0 and the flowchart were shared or can be extrapolated from the existing model to proceed to the KR section. Additionally, limited validation was done by the expert advisor due to conflicting schedules. The thesis author deems the absence of this step acceptable since it only requires an adequate level of accuracy from the typical design process to create a proof-of-concept.

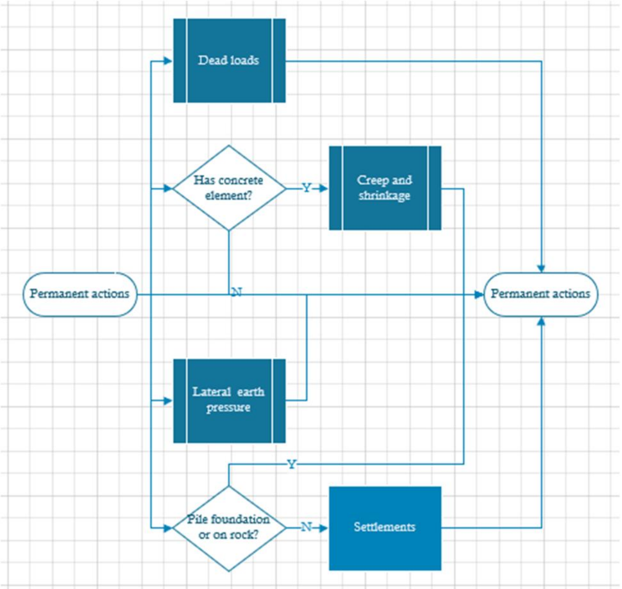


Figure 23. Snippet of a design activity decomposition in the informal model

7 Knowledge Representation

7.1 Methodology

One of the main purposes of the thesis was to implement a proof-of-concept product attending to the requirements and challenges explained in the previous sections. The ES that can do reasoning over a KB was chosen as the solution for this thesis. The knowledge in the ES can be represented in various ways and the representation choice would dictate how the system can be developed. Generally, there are 3 types of representation: rule, frame, and semantic network [208]

7.1.1 Types of Knowledge Representation

7.1.1.1 Rule

In a rule-based ES, knowledge is represented through a collection of IF-THEN rules, with the system utilizing both domain knowledge and current facts to make decisions. The main advantage of this approach is its simplicity and self-documentation nature, as the rules are easily comprehensible without the need to translate. Consequently, it can be developed incrementally, allowing for quick prototyping and new rules are incorporated to enhance performance and efficiency. The transparency of rules also enables the tracing of rule sequences for reasoning explanations. However, there are certain limitations related to scaling. A rule-based system is not efficient for real-time applications as the system often goes through exhaustive search of rules. They also lack hierarchical structure which hinders the understanding of logical interdependencies in a large set of rules. The last drawback of not only rule-based but also general ES is it lacks the ability to learn from experience, placing the responsibility on knowledge engineers to revise and maintain the system. [146]

7.1.1.2 Frame / Object-oriented

In the frame-based ES, a frame is a data structure that represents knowledge about a specific object or concept. By consolidating all the structural and behavioral knowledge within a single object, frames effectively apply principles from object-oriented programming to ES. Each frame consists of a name and a collection of attribute-value pairs, referred to as slots, which can be filled with specific values, rules, or methods for acquiring slot values, or references to other frames. [209]

Object-oriented and frame-based models share a close relationship historically and structurally [210]. Despite that, they are still distinct concepts and should be separated. The most fundamental distinction is object-oriented embraced encapsulation as a fundamental requirement while frame lacks this property. Thus, the system can look for useful information by accessing frames at any level of a hierarchy [210]. In other words, frames are objects without encapsulation which are referred to as "object-relational" [211]. Additionally, frame representation emphasizes multiple inheritances to mirror human conceptualizations. In contrast, many object-oriented languages lean towards single inheritance to uphold encapsulation and modularity[212].

Frame-based KR offers several advantages over rule-based one [47], [209]. Firstly, frames provide a richer representation, allowing for the modeling of complex knowledge domains. Secondly, frames support default values for attributes, reducing redundancy and improving efficiency in specifying attribute values for instances. Thirdly, frame-based systems can represent and reason about procedural knowledge, facilitating the execution of more complex behaviors. Finally, frame-based representations allow for inheritance and subsumption, enabling hierarchical organization and classification of knowledge.

However, frame-based representation has its drawbacks too. As the KB grows larger and more interconnected, frame-based representations can become less scalable and more complex to manage. The rigid structure of frames may also limit their flexibility in representing diverse and evolving relationships. Furthermore, frame-based systems may lack robust built-in support for advanced semantic reasoning and inference capabilities, which rules and knowledge graphs can provide. Finally, there are interoperability issues due to their specific implementation and lack of standardized exchange formats.

7.1.1.3 Semantic Network

One of the most ancient and easily comprehensible methods for representing knowledge is the Semantic Net [47]. It employs a graphical framework to depict objects and their interrelationships. This approach is also referred to as ontology, knowledge graph, or graph database, which is essentially a type of graph model [213]. Knowledge graph emphasizes storing data at both the class and individual levels, represented using nodes and links. Nodes symbolize objects or concepts, while links represent the connections between nodes. These links have directions and are labeled. The nodes that are connected to one another give the network its structure and defined its meaning [214].

Semantic networks possess significant advantages that make them a valuable method of KR [214]. Their flexibility allows for the integration of diverse data structures and sources, accommodating evolving schemas and data over time. The graphical nature of Semantic networks makes them intuitive and easy to comprehend, aligning with human information processing.

Moreover, their expressiveness surpasses rule-based formalisms, allowing properties to be overridden and creating clusters of related elements.

Nevertheless, Semantic networks also come with limitations [214]. They do not have a standardized semantic or an agreed-upon notion of links and nodes, leading user's responsibility to interpret the definitions. Similarly, their flexibility can be a double-edged sword as it can have a wrong inheritance of properties. For example: it's correct to say Dumbo is an Elephant, Elephant is a Mammal, and so Dumbo is a Mammal. But it's wrong to say Dumbo is an Elephant, Elephant can't Fly, and so Dumbo can't Fly. Therefore, it heavily relies on the creator's understanding of the meaning of links to create the appropriate boundaries. Nevertheless, advanced ontology tools like Protégé can address some of these issues.

7.1.2 Choice of Knowledge Representation

With a clear understanding of the strengths and weaknesses of each representation type, we can choose one that is the best fit for the building block concept. Rule-based representation is considered inadequate for complex design tasks due to several limitations [215]. While it is possible to formalize knowledge in the form of rules, it becomes challenging when dealing with a vast domain space and identifying all feasible rule combinations. The maintenance work, including creating a large number of rules for each design activity or building block, understanding the interactions among rules, debugging them, and controlling their behavior, becomes increasingly difficult as the KB expands. Moreover, much of the design knowledge is declarative, requiring a hierarchical composition of objects and attributes linked to them. Frame-based system or semantic network is more expressive to structure this type of knowledge and can handle complex design scenarios [216].

That left us with frames and semantic networks, of which the distinction in the literature tends is blurred [217]. This is because ontology construction shares similarity in their fundamental task of classification and thus, involve the concepts of classes, attributes, properties, objects, and individuals [218]. In fact, it is a straightforward task to translate between semantic networks and frame-based representations, wherein nodes in the semantic network correspond to objects in the frame system, links are represented as slots, and the node at the other end of the link serves as the slot value [219]. Nevertheless, there are still certain edges each system has over each other. Frame systems are more commonly associated with structures that are complex and exhibit greater organization, such as the need for computable node labels rather than assigned ones [218]. In contrast, in semantic networks, intricate class hierarchy relations can be inferred based on the class definition itself while frame systems require explicit assertion of subclass relations [220].

Our primary objective is to develop an ES reusable KB that is flexible and adaptable to changes and updates based on context-specific requirements. The emphasis of our thesis so far has been on the relationships of design activities with building blocks and not the content of the activities themselves. The mentioned characteristics can be achieved with a knowledge graph, which offers significant advantages in three key areas: effective modeling of data with numerous relationships. flexible expansion of new data and data relationships and real-time querying of data relationships [221]. Moreover, in recent years, knowledge graphs have emerged as highly efficient and effective methods for integrating knowledge, making them particularly relevant in the construction domain, where they are regarded as advanced KM technology [213]. Therefore, a semantic network or knowledge graph is the most suitable choice for our purposes.

7.1.3 Choice of Software Implementation

Currently, there are two primary approaches utilized in graph database solutions: RDF (Resource Description Framework) graphs and LPG (Labeled Property Graph) graphs [222], represented by the most popular management systems: Protégé and Neo4j, respectively. RDF stores, also known as triple stores, possess two crucial characteristics. Firstly, RDF represents, stores, and queries data in the form of a graph. Secondly, they are semantic in nature, allowing for the storage of not only data but also explicit descriptions of the data's meaning. The explicit descriptions are what referred to a whole as ontologies or a schema [223]. However, it is worth noting that LPG graphs offer similar functionality to RDF. Both are languages for defining graphs and have specific query languages tailored for themselves (Cypher and SPARQL). The primary distinction between property graphs and RDF lies in their ability to incorporate attributes and relations on properties [224].

In the context of this thesis, the choice of representing the formal model with OWL (Web Ontology Language), which is based on RDF, and Protégé as the system of graph model creation was made for several reasons [224]. Firstly, it provides formal semantics, enabling automated reasoners to validate ontology consistency and infer relationships. Secondly, it grants access to a vast array of structured and freely available W3C-compliant (World Wide Web Consortium) knowledge graphs accessible on the Internet. Moreover, OWL and RDF leverage industry and technical vocabularies, providing standardized ontologies for diverse domains. In contrast, Neo4j lacks standardization, rendering it vendor-dependent, while the Semantic Web allows for seamless integration and migration across various vendors and open-source platforms adhering to W3C standards, ensuring the crucial flexibility needed to future-proof IT infrastructure. Furthermore, due to programming capabilities and time constraints, the selected software must be complete in its functionality and still simple to use. Protégé as the most popular open-source ontology editor outshines Neo4j in these points.

Implementation with Protégé requires 4 components to work effectively:

1. Knowledge base: This includes the translated informal model in the form of an ontology model with SWRL (Semantic Web Rule Language) rules. This is the most important component. More details in Section 7.2
2. Ontology management tool: The open-source software Protégé 5.6.1 was used in this thesis. It facilitates the editing KB through machine-understandable language and enables querying and retrieving results. Additionally, Protégé includes various plugins that enhance user experience, including OntoGraf for graph visualization, SWRLTab for rule creation, and Snap SQWRL to query inferred knowledge.
3. Built-in reasoner: Pellet was chosen in this thesis. It plays a vital role in reading rules, inferring implicit knowledge, and ensuring the validity of the ontology model by identifying syntactic errors. Whenever the KB was modified, the Pellet was run to guarantee model consistency and was ready for further modifications.
4. Query interface: this is used to interact with the KM system, e.g., finding building blocks that match chosen design activities. More details in Section 7.3.2

7.2 Ontology Model

The ontology model includes a bridge structure ontology and a bridge design process ontology. Each ontology model generally consists of 4 components: class, instance, relation, and axiom [225]:

Concept (also called classes) represents distinct sets of entities or categories within the domain. For example: bridge technical terms such as ”BridgeType”, ”Material”, ”Process”. There are two types of concepts:

Primitive concepts are defined as classes with properties that fulfill the necessary conditions. Fulfilling the necessary does not guarantee an entity’s association with a class. For instance: a bridge must be a structure with load-bearing capacity, but a load-bearing structure is not necessarily a bridge

Defined concepts are defined as classes with properties that fulfill both necessary and sufficient conditions. For example, a Girder Bridge must have at least 1 girder is the necessary condition and not a sufficient condition because a bridge having girders does not make it a girder bridge.

Instance (also called individuals) represents specific objects of concepts. For example: “Kruunuvuorensilta” is an instance of class “Bridge”. The presence of concepts forms an ontology and in conjunction with instances, ontology forms a KB. Determining whether something qualifies as a concept or an instance can be challenging and often relies on the modeler’s sound judgments.

Relation represents the relationships or properties among concepts, instances, and between concepts and instances themselves. For example, the relation "hasIndividual" links the concept of "Bridge" to the specific instance of "Kruunuvuorensilta". Similar to concepts, relations can be organized into taxonomies, and they possess properties that provide additional information about the connections between concepts, which is elaborated further in Section 7.2.1.2. In OWL ontology, relationships and properties between concepts are called object properties, while relationships and properties between instances are called datatype properties.

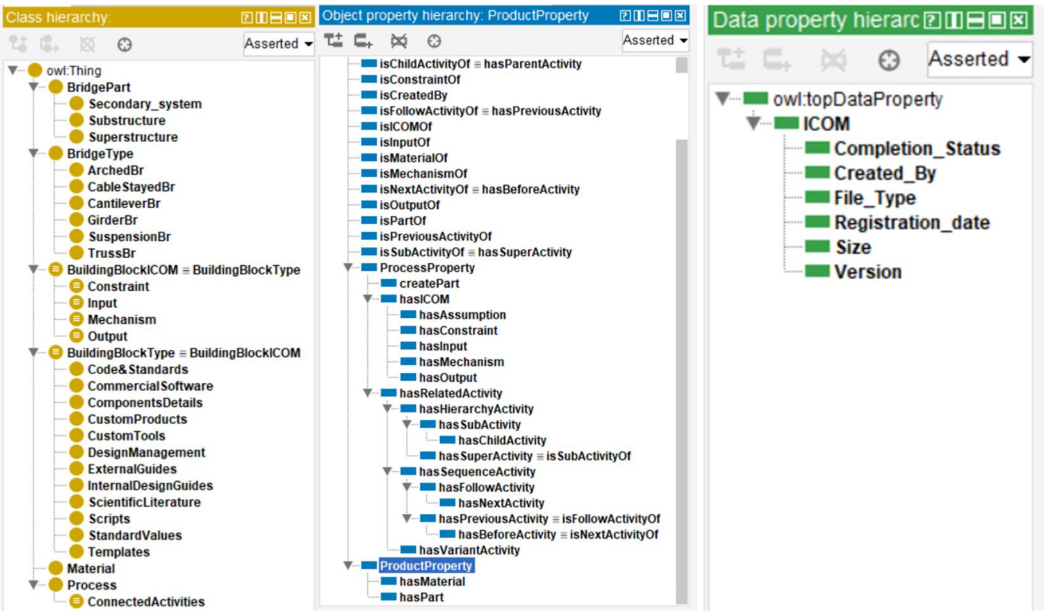
Axiom serves as “statement”, stating the facts and knowledge in the domain. Axiom imposes rules and constraints on classes, instances, and relations and is an integral part of concept and property construction. For example, "Kruunuvuorensilta" has datatype property "length", which must be a numerical value.

7.2.1 Ontology Construction Methodology

7.2.1.1 Concepts

In the field of product modeling exists the concepts of generic product definition (product with unassigned property values), specific product definition (product variation with assigned property values), and occurrence definition (a specific product with location) [226]. This example provides a clearer look into the class-subclass-instance relationships that form the foundation of any domain KR. A class represents a general category, a subclass specified that category with some additional properties, and instances embody specific entities within those categories. It is essential for the properties of instances to align with the definitions of their respective classes. The creation of the class hierarchy is use case-specific, taking into account the complexity and scale of the application case, and should be undertaken during knowledge implementation or validation[85]. Despite being the foundation of any domain knowledge, designing a class hierarchy is less of a science and more of an art.

According to [227], determining whether a concept should be classified as a class or an individual instance within an ontology depends on the design intention. The boundary between classes and individual instances is established by identifying the lowest level of granularity in the representation, which is determined by the ontology's potential application. In other words, what specific items are to be represented in the KB? When considering the concept of a building block, the focus is in the design knowledge of different structural bridge types rather than a specific instance from a project. Therefore, the instances should focus on representing reusable knowledge applicable to various projects, rather than project-specific instances like for example, Deck geometry creation script of Kruunuvuorensilta. Another consideration, at least in Protégé, is that class and instance can only have subsumption relations to their own kind. For that reason, instance choice in the product model will have an impact on the instance choice in the process model. Despite that, there might be a case for including project-specific instances to represent building blocks under development, which are not ready for general use. It is unclear how this scenario fits within the building block concept without transforming the KB into a project archive. Hence, this aspect will be excluded from the thesis. The complete hierarchy of the proof-of-concept and the final choices are shown in Figure 24.



```
owl:Thing
    BridgePart
        Secondary_system
        Substructure
        Superstructure
    BridgeType
        ArchedBr
        CableStayedBr
        CantileverBr
        GirderBr
        SuspensionBr
        TrussBr
    BuildingBlockICOM ≡ BuildingBlockType
        Constraint
        Input
        Mechanism
        Output
    BuildingBlockType ≡ BuildingBlockICOM
        Code&Standards
        CommercialSoftware
        ComponentsDetails
        CustomProducts
        CustomTools
        DesignManagement
        ExternalGuides
        InternalDesignGuides
        ScientificLiterature
        Scripts
        StandardValues
        Templates
    Material
    Process
        ConnectedActivities
```

```
        isChildActivityOf ≡ hasParentActivity
        isConstraintOf
        isCreatedBy
        isFollowActivityOf ≡ hasPreviousActivity
        isICOMOf
        isInputOf
        isMaterialOf
        isMechanismOf
        isNextActivityOf ≡ hasBeforeActivity
        isOutputOf
        isPartOf
        isPreviousActivityOf
        isSubActivityOf ≡ hasSuperActivity
    ProcessProperty
        createPart
        hasICOM
            hasAssumption
            hasConstraint
            hasInput
            hasMechanism
            hasOutput
        hasRelatedActivity
            hasHierarchyActivity
                hasSubActivity
                    hasChildActivity
                hasSuperActivity ≡ isSubActivityOf
            hasSequenceActivity
                hasFollowActivity
                    hasNextActivity
                hasPreviousActivity ≡ isFollowActivityOf
                    hasBeforeActivity ≡ isNextActivityOf
            hasVariantActivity
    ProductProperty
        hasMaterial
        hasPart
```

```
owl:topDataProperty
    ICOM
        Completion_Status
        Created_By
        File_Type
        Registration_date
        Size
        Version
```

Figure 24. Snippet of class and property hierarchy in the proof-of-concept

7.2.1.2 Relations

In Protégé, properties play a crucial role in establishing connections between classes and their attributes, creating subject-property-object triples. There are three distinct types of properties: object, datatype, and annotation properties.

- Object properties serve as semantic relations between classes, such as the “createPart” relation connecting a design process and a bridge part.

- Datatype properties represent quantitative or qualitative attributes of classes. For example, the instances of class “BuildingBlock” has a

“Registration_date” property that shows the date the building block was registered to the KB.

- Annotation properties are used to provide explanations for entities and properties, ensuring readability. It also has the potential to be assigned as attributes to other properties [85].

Complex properties can be organized in a hierarchical structure, with property and sub-property. This hierarchical arrangement helps to further categorize and define the relationships between properties within the ontology. For example: “hasBeforeActivity” and “hasNextActivity” are sub-properties of “hasSequentialActivity”. It is noted that the naming convention of both properties and classes is not well-established. Detail on possible naming conventions in ontology construction can be found here [222]. In this thesis, camel case was used due to its popularity in other ontology models.

Facets are a type of axiom used to enhance property in many aspects, encompassing cardinality restriction, characteristic settings, and domain and range restrictions.

- Cardinality restriction establishes the number of values a property can have; for example, the property “Registration_date” possesses a single cardinality due to building block versions can be registered only the first time.

- Characteristics indicate different behavior of properties such as functional, inverse, transitive, symmetric/ asymmetric, and reflexive/ irreflexive. Further information can be found here [228].

- Domain and range restrict the datatypes or classes of the subject and object in a property triple. For instance, the domain of “createPart” is ”Process” and the range is ”BridgePart”. Properties and facets are defined at the class level, and instances of the class inherit these settings to determine their attributes and behaviors.

As with class hierarchy choice, determining whether to categorize data as a property or concept presents a challenging decision. Data modeling is both art and science, lacking definitive right or wrong answers, but rather focusing on what suits the specific use case effectively [229]. As [227] put it simply, within a domain, if an object holds significance distinction and the distinct values make us regard the objects as different types, it makes sense to create a new class for that distinction. For example: a bridge supported by a pylon and cables is regarded as a different type of bridge from one with only girders. Therefore, there are subclasses of bridge types in the product model. A more specific sign to help with the decision is if a property value of a concept affects the property values in other classes, it is preferable to establish a new class for that property value. Otherwise, the value can remain a property. Finally, the best solution is to ask if the instances of a particular class change often. If the answer is yes, the class should turn into a property.

With the previous discussion in mind, a summary of data modeling is given [229]:

- Class excels when dealing with variables that have a low number of options (cardinality) and overlapping categories (multi-inheritance). It often serves as a means of partitioning a graph into sections.

- Properties can accommodate high-cardinality variables or when variables change quickly. However, property is a less optimal choice when the values are frequently checked for overlap with other objects.

- Instances come into play when searching for objects with a shared property value or when handling high-cardinality situations. Additionally, instances are required to capture additional metadata about the object. Ideally, instances should have a significantly lower number of connections to other entities. Otherwise, it should be a class.

7.2.1.3 Datatype Properties

As mentioned above, datatype properties were assigned to instances only. In this thesis, it was used to model metadata of the building block (ICOM). The list of datatype properties for ICOM can be seen in Figure 24. Name and Description were not declared since they are represented by the name of instances in Protégé or can be assigned through the annotation property feature. The rest are fundamental metadata commonly found in any KB with self-describing and required no further explanation. These parameters serve the purpose of identifying, locating, and describing a specific building block. It is important to note that the author does not believe this is a complete list of required metadata, but simply a starting point for future development.

7.2.2 Product and Process Model

The bridge structure ontology model (product model) has been constructed to support the process model and thus, it only has the minimum level of detail required for its supporting role in creating the proof of concept. The classes of this model include: BridgeType, BridgePart, and Material. Each class presents a different perspective on the design product.

BridgeType is divided into different bridge type subclasses such as girder bridge or cantilever bridge, see Figure 25. In the figure, the arrow directions indicate relationships, e.g. Blue arrow points from BridgePart to GirderBr, meaning BridgePart has subclass GirderBr. Specific bridge projects can be added as instances of BridgeType and its subclasses. BridgeType is connected to BridgePart by property hasPart to show that a bridge is comprised of multiple structural elements. It is divided into 3 subclasses: Superstructure, Substructure, and SecondarySystem. As a result of the class hierarchy strategy above, the structural elements within the BridgePart are treated as instances rather than subclasses Structural element instances were classified according to the 3 subclasses. The BridgeType class has been given a necessary condition of hasPart at least 1 Superstructure and Substructure. Finally, Material is connected to both BridgeType and BridgePart by property hasMaterial. This class represents the main materials associated with certain a certain bridge or bridge part. It has no subclass as the materials are instances of the Material class.

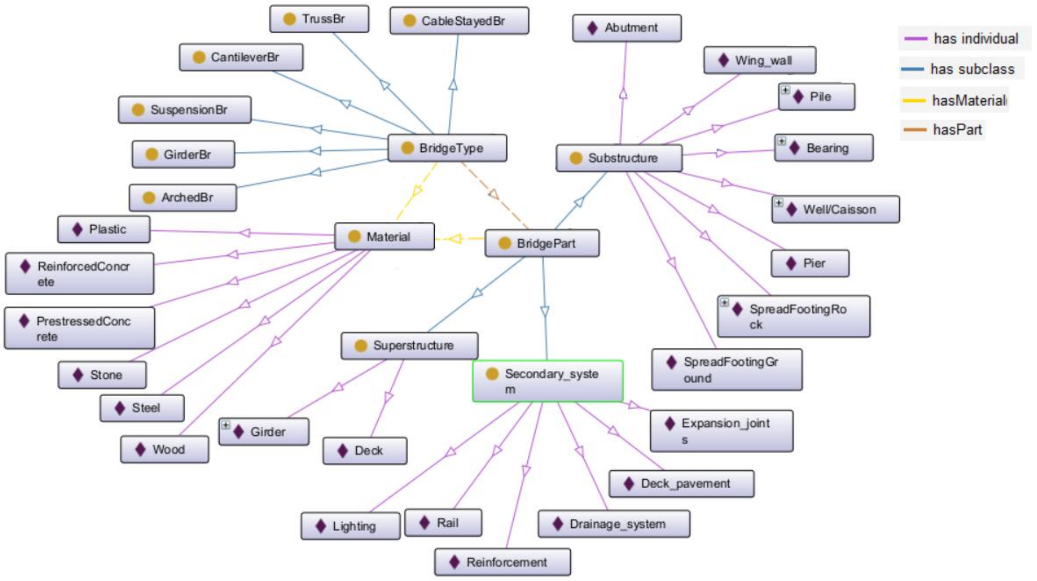


Figure 25. Snippet of the product model.

The process model consists of activities from the informal model and the building block associated with those activities. Classes of this model include: “Process” and “BuildingBlockICOM”. All activities are treated as instances of the class “Process” rather than being grouped into subclasses. This is because it is difficult to definitively distinguish which processes should be classified as classes and which as instances. For example, we cannot describe "LoadCalculation” as a class for instance "WindLoadCalculation". Although the relationship may resemble a hierarchy, it does not imply the "is a" relationship, which can be defined by applying the phrase “is a” between two entities (e.g., Girder Bridge is a Bridge), in class-subclass connections but rather a "part of" or "comprise of" relationship. In another example, the “FoundationDesign” is thought to be the class for “PileFoundationDesign”, “SpreadFootingDesign” and “WellFoundationDesign” due to meeting the requirement of a “is a” relationship and is related by substitution. However, turning “FoundationDesign” into a class will restrict it to only have relationships with other classes, since no other relationship besides "is a" can be made between a class and an instance. Besides, turning "FoundationDesign" into a class seems arbitrary when comparing it to other activities.

Similar to the product model, various object properties were used to represent the 5 relationships: Dependency, Hierarchy, Substitution, Product model relation, and BuildingBlockICOM relation. An overview can be seen in Figure 26

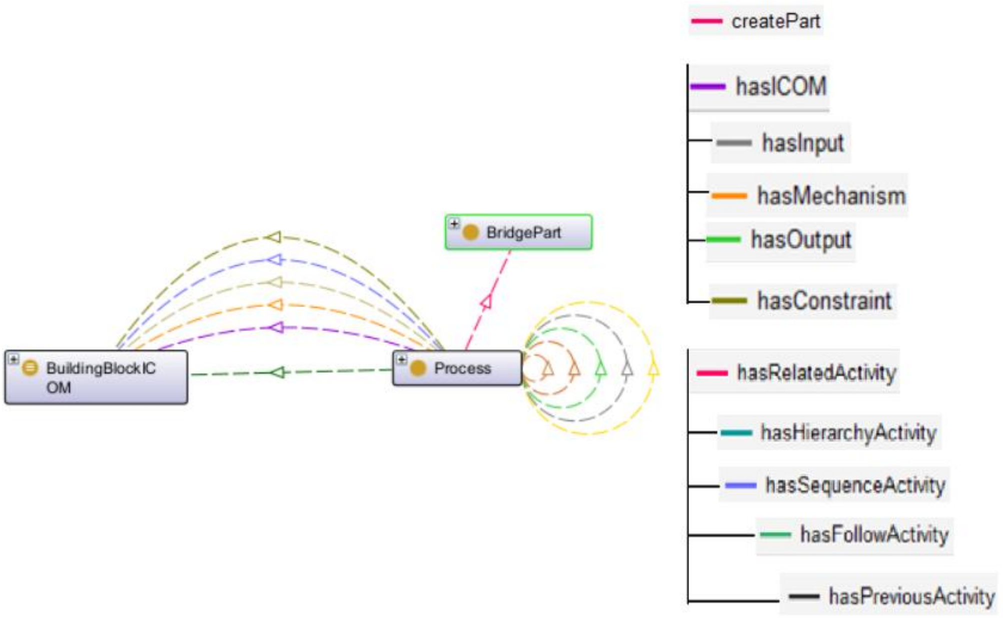


Figure 26. Snippet of the process model. Some properties have sub-properties.

- Dependency: The assignment of precedence or succession relationships between activities is crucial. These relationships are defined mainly with "hasNextActivity" and "hasBeforeActivity". The latter is an inverse property of the former. Both are irreflexive (an activity cannot follow by itself) and asymmetric (an activity cannot follow and precede the same activity). These properties are sub-property of "hasFollowActivity" (connecting all activities that come after the current one) and "hasPrecedeActivity" (connecting all activities that come before the current one), respectively. Both are in turn sub-property of "hasSequenceActivity", connecting all activities that are part of a sequence on the same level of hierarchy. All properties of parent activities are transitive, inheriting all relations of their sub-properties.

- Hierarchy: Any process can be decomposed into subprocesses and tasks. This relationship is defined by "hasChildActivity", which has irreflexive and asymmetric characteristics. A parent activity aggregates all constraints, mechanisms, and boundary inputs and outputs of its children. It has the transitive properties "hasSubActivity" and "hasSuperActivity" as the parent property. These properties give the position of activity in a hierarchy, showing the all ancestors and descendants of an activity. Finally, both are sub-properties of "hasHierarchyActivity", which is transitive as well to display all the activities in the hierarchy branch.

- Substitution: "hasVariant" property states that an activity may have other activities as variants. This property is symmetric (A has variant B also means B has variant A) and irreflexive. The variant of activities must have the same inputs and outputs requirements from the first variant's definition and therefore, can be inherited. Properties of Dependency, Hierarchy, and Substitution relationships share the same class "Process" for domain and range, see Figure 26.

- BuildingBlockICOM relation: ICOM is represented by the properties "hasInput", "hasOutput", "hasMechanism", "hasConstraint" and those properties have parent property "hasICOM", see Figure 24. These are the most important properties since they connect the design activities with the information requirements and affect other relationships as well. For example, a shared ICOM makes the activities related to it depend on each other. This is shown in Figure 27.

- Product model relation: The property "createPart" indicates which structural bridge element is designed by the process.

The aforementioned attributes and characteristics can be declared explicitly as axioms of the object properties or inferred dynamically from the model structure or SWRL rules.

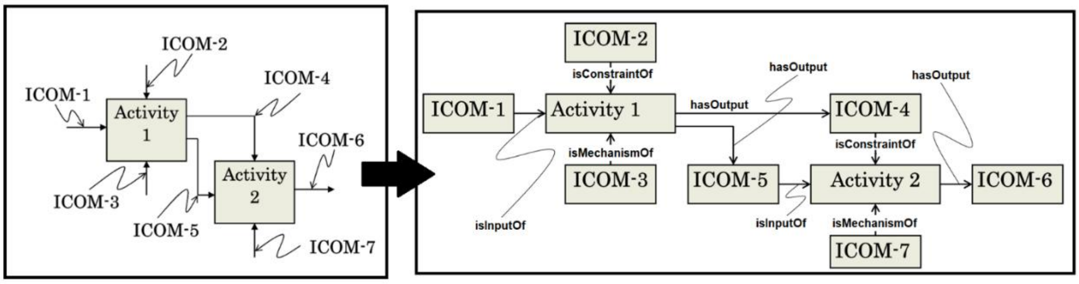


Figure 27. Adapting ICOM and activity relationship for Knowledge Graph representation. Adapted: [230]

7.3 Process Assembling

7.3.1 Inferencing

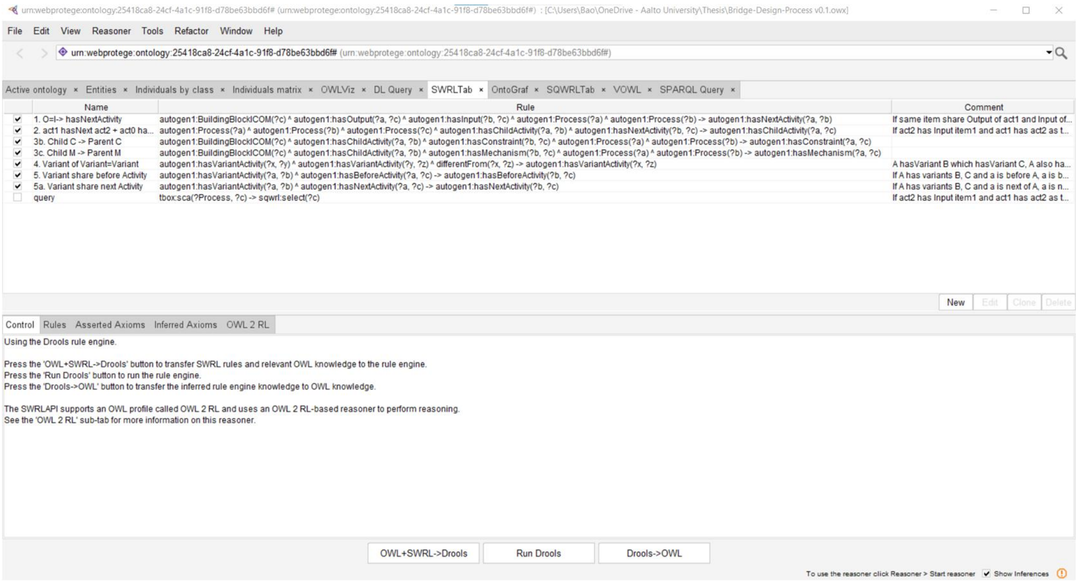
One of the key features of Protégé is the ability to use reasoner for two main purposes: consistency checking and new knowledge inference. According to [228], in consistency checking, the reasoner identifies the appropriate placement of concepts within definitions, based on provided statements about classes and properties. This ensures the overall coherence and compatibility of statements and definitions within the ontology. Thus, the reasoner helps in maintaining the class hierarchy, especially in scenarios involving multi-inheritance. Moreover, the reasoner can deduce additional information. While we can manually assert all property combinations representing the relationships in the informal model ourselves, this approach is time-consuming and inefficient. By leveraging the power of the reasoner, we can automatically infer the missing combinations. For instance, if two properties are inverses, the user only needs to assert one value, and the reasoner will automatically infer the inverse value.

However, reasoning with OWL is limited to structural inference, such as subsumption and identity [231]. The informal requires other relationships such as dependency, substitution, etc. which require more advanced deductive reasoning capabilities. To address this, SWRL rules were used to express and infer more dynamic and intricate KRs that cannot be adequately represented using the syntax of OWL. [232] identifies five key scenarios where rules can be effectively utilized, 2 of which are implemented in this KB:

- Standard rules for chaining ontology properties, facilitating the transfer of properties from parts to wholes. For example: Act1 has output Block1, Block1 is input of Act2-> Act1 has next activity Act2

- Meta-rules for supporting ontology engineering tasks such as acquisition, validation, and maintenance. For example: Act2 has ICOM block1-2-3-4, Act1 has parent of Act2-> Act1 has ICOM block1-2-3-4 inherited from Act2

In Protégé, the built-in SWRLTab provides a convenient interface for editing and saving various SWRL rules. The ontology and SWRL rules defined through Protégé and SWRLTab are stored in a KB. The full rule list for the proof-of-concept can be seen in Figure 28. The rule reasoner executes the SWRL rules in the order in the ruleset and generates new facts within the ontology management system. These rules can be repeatedly applied to all instances or executed to generate new facts.


urn:webprotege:ontology:25418ca8-24cf-4a1c-91f8-d78be63bbd6f# (urn:webprotege:ontology:25418ca8-24cf-4a1c-91f8-d78be63bbd6f#) : [C:\Users\Bao\OneDrive - Aalto University\Thesis\Bridge-Design-Process v0.1.owx]
File
Edit
View
Reasoner
Tools
Refactor
Window
Help
urn:webprotege:ontology:25418ca8-24cf-4a1c-91f8-d78be63bbd6f# (urn:webprotege:ontology:25418ca8-24cf-4a1c-91f8-d78be63bbd6f#)
Active ontology
Entities
Individuals by class
Individuals matrix
OWLViz
DL Query
SWRLTab
OntoGraf
SQWRLTab
VOWL
SPARQL Query
New
Edit
Clone
Delete
Control
Rules
Asserted Axioms
Inferred Axioms
OWL 2 RL
OWL+SWRL->Drools
Run Drools
Drools->OWL
To use the reasoner click Reasoner > Start reasoner
☑ Show Inferences

	Name	Rule	Comment
☑	1. O=I-> hasNextActivity	autogen1:BuildingBlockICOM(?c) ^ autogen1:hasOutput(?a, ?c) ^ autogen1:hasInput(?b, ?c) ^ autogen1:Process(?a) ^ autogen1:Process(?b) -> autogen1:hasNextActivity(?a, ?b)	If same item share Output of act1 and Input of...
☑	2. act1 hasNext act2 + act0 ha...	autogen1:Process(?a) ^ autogen1:Process(?b) ^ autogen1:Process(?c) ^ autogen1:hasChildActivity(?a, ?b) ^ autogen1:hasNextActivity(?b, ?c) -> autogen1:hasChildActivity(?a, ?c)	If act2 has Input item1 and act1 has act2 as t...
☑	3b. Child C -> Parent C	autogen1:BuildingBlockICOM(?c) ^ autogen1:hasChildActivity(?a, ?b) ^ autogen1:hasConstraint(?b, ?c) ^ autogen1:Process(?a) ^ autogen1:Process(?b) -> autogen1:hasConstraint(?a, ?c)	
☑	3c. Child M -> Parent M	autogen1:BuildingBlockICOM(?c) ^ autogen1:hasChildActivity(?a, ?b) ^ autogen1:hasMechanism(?b, ?c) ^ autogen1:Process(?a) ^ autogen1:Process(?b) -> autogen1:hasMechanism(?a, ?c)	
☑	4. Variant of Variant=Variant	autogen1:hasVariantActivity(?x, ?y) ^ autogen1:hasVariantActivity(?y, ?z) ^ differentFrom(?x, ?z) -> autogen1:hasVariantActivity(?x, ?z)	A hasVariant B which hasVariant C, A also ha...
☑	5. Variant share before Activity	autogen1:hasVariantActivity(?a, ?b) ^ autogen1:hasBeforeActivity(?a, ?c) -> autogen1:hasBeforeActivity(?b, ?c)	If A has variants B, C and a is before A, a is b...
☑	5a. Variant share next Activity	autogen1:hasVariantActivity(?a, ?b) ^ autogen1:hasNextActivity(?a, ?c) -> autogen1:hasNextActivity(?b, ?c)	If A has variants B, C and a is next of A, a is n...
□	query	tbox:sca(?Process, ?c) -> sqwrl:select(?c)	If act2 has Input item1 and act1 has act2 as t...

Using the Drools rule engine.

Press the 'OWL+SWRL->Drools' button to transfer SWRL rules and relevant OWL knowledge to the rule engine.
Press the 'Run Drools' button to run the rule engine.
Press the 'Drools->OWL' button to transfer the inferred rule engine knowledge to OWL knowledge.

The SWRLAPI supports an OWL profile called OWL 2 RL and uses an OWL 2 RL-based reasoner to perform reasoning.
See the 'OWL 2 RL' sub-tab for more information on this reasoner.

Figure 28. Interface of SWRLTab with created rules in the knowledge base

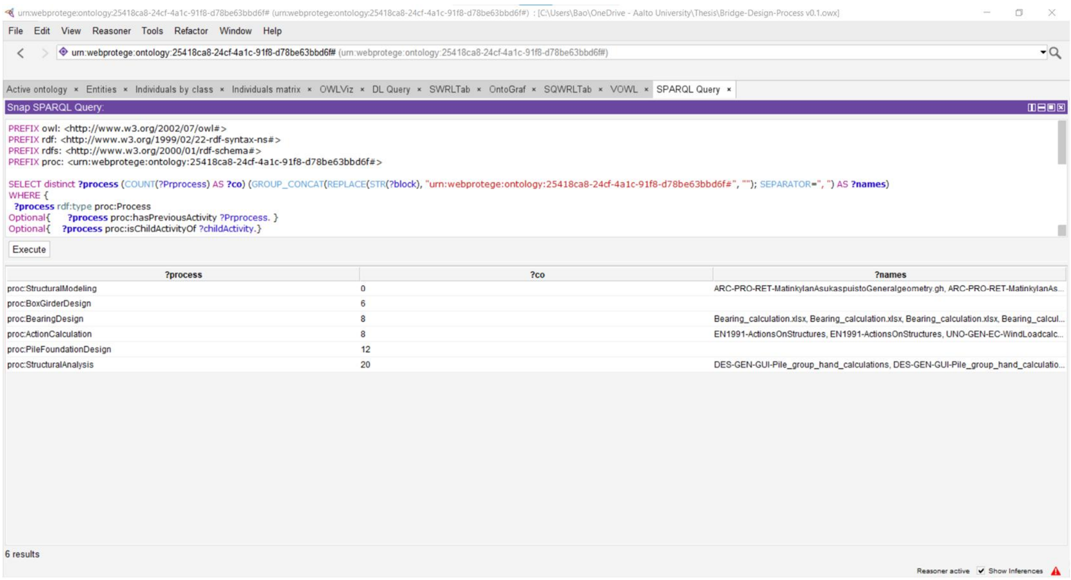
7.3.2 Query and Workflow Vision

SQWRL (Semantic Query-enhanced Web Rule Language; pronounced squirrel) is an extension that integrates queries within SWRL rules, allowing for simultaneous reasoning and querying [233]. While SWRL and SQWRL demonstrate efficiency in inferring and searching for information in an ontology, they have certain limitations inherent to the RDF/OWL syntax. As stated by [228], OWL and SWRL feature the Open World Assumption (OWA) and the monotonic reasoning, which can be frustrating for new users. On one hand, OWA assumes the absence of information does not imply that a statement is false and thus, everything can be true unless a negative statement was declared explicitly (close the open world). A related consequence of OWA is it does not have the Unique Name Assumption, meaning the reasoner may infer that different names refer to the same individual. On the other hand, monotonic reasoning can only do inferences based on existing facts and adding new facts during the reason cannot change the conclusion. Also, new information can only be added to existing knowledge without altering or retracting previously derived conclusions. Particularly, negation and counting are not supported, limiting many functions usually seen in normal programming. Additionally, updating the attributes of entities dynamically using newly inferred information can lead to infinite loops.

To address these problems, we employ SPARQL (SPARQL Protocol And Rdf Query Language) as our query solution. SPARQL is a powerful tool for extracting information from the network and inferring new knowledge. It is currently the most widely accepted standard for querying RDF and OWL data. It allows us to formulate queries by matching patterns in the graph and binding variables as solutions are found [233]. It is better than SQWRL since

SPARQL is not constrained by monotonic reasoning or the OWA. It grants us the ability to perform non-monotonic reasoning and make use of the more prevalent Closed World Assumption. For instance, query all processes having no building blocks.

Protégé offers built-in SPARQL Query functionality but can only perform queries using the asserted axioms and not an inferred one. The inferred knowledge must be exported into and new ontology and be queried from there. To allow quick testing, the plugin Snap-SPARQL was used to allow querying directly with the inferred results, see Figure 29. The full queries can be found in the Appendix. However, it is not a replacement for the normal SPARQL Query since Snap-SPARQL is not built to the latest SPARQL 1.1 version and thus, it is limited in some functions such as aggregation formula or subquery [234].



urn:webprotege:ontology:25418ca8-24cf-4a1c-91f8-d78be63bbd6f# (urn:webprotege:ontology:25418ca8-24cf-4a1c-91f8-d78be63bbd6f#) : [C:\Users\Bao\OneDrive - Aalto University\Thesis\Bridge-Design-Process v0.1.owx]
File Edit View Reasoner Tools Refactor Window Help
urn:webprotege:ontology:25418ca8-24cf-4a1c-91f8-d78be63bbd6f# (urn:webprotege:ontology:25418ca8-24cf-4a1c-91f8-d78be63bbd6f#)
Active ontology × Entities × Individuals by class × Individuals matrix × OWLViz × DL Query × SWRLTab × OntoGraf × SQWRLTab × VOWL × SPARQL Query ×
Snap SPARQL Query:
Execute
6 results
Reasoner active
Show Inferences

```
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 proc: <urn:webprotege:ontology:25418ca8-24cf-4a1c-91f8-d78be63bbd6f#>

SELECT distinct ?process (COUNT(?Prprocess) AS ?co) (GROUP_CONCAT(REPLACE(STR(?block), "urn:webprotege:ontology:25418ca8-24cf-4a1c-91f8-d78be63bbd6f#", ""); SEPARATOR=", ") AS ?names)
WHERE {
 ?process rdf:type proc:Process
Optional{     ?process proc:hasPreviousActivity ?Prprocess. }
Optional{    ?process proc:isChildActivityOf ?childActivity.}
```

?process	?co	?names
proc:StructuralModeling	0	ARC-PRO-RET-MatinkylanAsukaspuistoGeneralgeometry.gh, ARC-PRO-RET-MatinkylanAs...
proc:BoxGirderDesign	6	
proc:BearingDesign	8	Bearing_calculation.xlsx, Bearing_calculation.xlsx, Bearing_calculation.xlsx, Bearing_calcul...
proc:ActionCalculation	8	EN1991-ActionsOnStructures, EN1991-ActionsOnStructures, UNO-GEN-EC-WindLoadcalc...
proc:PileFoundationDesign	12	
proc:StructuralAnalysis	20	DES-GEN-GUI-Pile_group_hand_calculations, DES-GEN-GUI-Pile_group_hand_calculatio...

Figure 29. Snap-SPAQRL query result showing all processes and its building blocks, filtered by design parts

This is the end of the proof-of-concept implementation. The previous features were built with a software usage flow in mind: To create new building blocks, firstly, the user creates a new design process at any level of hierarchy or adds variations to existing processes. Dependency links are then attached to these processes in the KB. The newly created process requires assigning building blocks, which can be either newly added by the user or existing ones in the KB, to its ICOM. These building blocks in the KB only serve as links to the real file in separate file storage. These building blocks possess their own metadata that gives information that cannot be described using ontology relationships.

To extract building blocks, the user performs a design process assembling with the query interface. The querying process starts with a selection screen where the user makes decisions related to the design project, such as material choice and the type of bridge to be designed. The software provides suggestions for possible design choices, which the user verifies. A process map is displayed, illustrating the assembly of the design process as the user makes decisions, providing a contextual understanding of how each decision impacts the overall design and guiding the user through a sequence of decision steps. The tree-based heuristic decision method forms the basis for subsequent pruning of decisions, with information from previous steps influencing the decision-making process. User decisions lead to the omission of certain activities, such as removing options for foundation design, adding activities specific to the chosen design systems, or reproduction of activities when multiple systems similar to an existing bridge are incorporated. Lastly, the configured design process undergoes a review before the associated building block is extracted.

8 Conclusion and future works

In this thesis, the building blocks concept was grounded in the KM field. Fundamental theories of structural design and KM were established to understand the nature of the building blocks. Building blocks embody the knowledge contained inside complex structural design processes, broken down into simpler, more manageable units, allowing for easier understanding, organization, and reusing of knowledge. The building block is represented in the form of digital files and documents contained within contextual design process, giving it connectivity to other building blocks. The method of implementation would greatly influence the effectiveness of reusing building blocks. A KB system built on ontology model technology was chosen as the method of implementation. The proof-of-concept was built to accommodate many factors; such as knowledge's granularity, abstraction, modularity, and repository's scalability, flexibility and retrievability; all of which determine the reusability of the building blocks. Ultimately, the ontological solution presented here provides a solid foundation for the future development of the building block concept.

Despite that, the building block ontology has not been completed and there is still a lot of work required to develop the proof-of-concept into a complete product. Additionally, throughout the course of this thesis, several areas of interest emerged, warranting further exploration. These recommendations are closely aligned with the building block's objectives and methodology employed. It is advised to conduct further research in the following areas:

The building block KB needs to be populated with more design processes and design files. If possible, a proper knowledge-gathering effort of the building block files and their associated design process should be done to have a clear understanding of the entire design workflow. The number of processes required is not known but it is certainly a massive scale. For example, in another ontology for building design, there are 150 diagrams, 600 design tasks, and 4,000 information requirements [180]. At the moment, only design processes and information of detailed design were considered. It is unknown how difficult it is for those of conceptual design to be represented in the current ontological model. Future work can look into this matter and use [215], [235] as references. Since creating a system like this means attempting to standardize unstructured knowledge, many choices have to be made regarding the balance between maximizing reusability, flexibility, and retaining the specificity of user choice. This is an ongoing task and hopefully, as more content is added to the KB, a clearer view of the abstraction level can be found.

Another important aspect of ontology modeling is the relationships and properties of populated information requirements and design processes need to be established to provide context. More contextual information should be added since a bridge and its building block can be categorized in many ways: bridge size, bridge type, bridge location, primary material, structural system, construction methodology, activity types, etc. [102]. The data needed for relationships and properties often requires analyzing textual building documentation, such as structural bridge records [236]. These records contain vital information about materials used, location, structural health, modifications, and administrative details. However, this is not an easy task due to the semi-structured nature of the documents and the limitations of traditional programming approaches not guaranteeing effective extraction for future versions of the document structure. It would either require a lot of manual work or use machine learning for automatic data extraction.

The wicked relationship will be a special concern in modeling, inferencing, and querying. It should be a priority to look into the methods to reduce the interrelationship between design clusters. It could be done by making important assumptions to tear the dependency between clusters, but that would require for user to give inputs at many steps of the design workflow. Perhaps fuzzy logic, which handles incomplete information, might be beneficial to predict the design values and reduce the number of inputs needed by users.

The building block retrieval system requires interfaces that can measure the relevance of files to the user with different knowledge needs. The CoMem project [136] should be used as a reference. The project focused on reusing design content from archives of CAD building models, with the measure of relevance to the designer's current task indicates by color. Projects, discipline subsystems, and individual components are represented as rectangles nested within each other [135]. It has two user interfaces: The Project Context Explorer helps users explore project context, while the Evolution History Explorer allows for version history exploration. A combined method of standard information retrieval techniques (through text and semantics) and enhance relevance measurement (by considering contextual building model elements) were employed to measure relevance. The study revealed that even with sparse texts in building models, good retrieval results were achievable.

Progress of the design process might be possible to measure through design task interdependencies level since it decreases as the design process moves closer to completion. Design tasks become simpler and more linear. This progress tracking and the completed ontology should be tested thoroughly in a validation check, a compulsory step in ontology building that was skipped due to resource limitations.

To expedite development, a combination of informal and formal models is a wanted feature. For example, [237] eliminates the need for manual translation of the completed formal model into a specific KBE language by a software engineer. Instead, BPMN and DMN were used to enable the visualization of process sequences in the informal model and directly develop the formal model. This approach facilitates seamless progress during the development of the formal model and subsequent KBE application, as it eliminates the need for constant switching between different presentation formats.

AI is likely an area that needs to be explored since it has many complementary aspects to the semantic network. First of all, the machine learning and CBR methods that were mentioned in Section 3.3.5. The author sees some integration possibilities between semantic network and neural network. Other useful AI applications are semantic search-based KM system, which aims to process fragmented and diverse design documentation in order to create a structured KB [92], and granularity computation, a significant area of research in artificial intelligence, plays a crucial role in various domains such as knowledge discovery, image compression, and semantic Web services [238].

There is a big concern over the entire building block concept that future researchers need to have in mind. The tools are there, but integrating and utilizing them might be more costly than the resources available to the CODA team. There is simply too much work in collecting and modifying the information required for each tool for one development team to handle. Therefore, the developing goals should be geared towards creating a community platform, moving the responsibility of populating the building blocks to the users. The job of the developer should be providing the framework that the building block is based on while the development and maintenance of building blocks are done by the users as bridge projects progress. Therefore, it is very important for the developing product to emphasize and encourage user contributions.

References

[1] P. F. Drucker, “The Information Executives Truly Need,” *Harvard Business Review*, Jan. 01, 1995. Accessed: Jun. 24, 2023. [Online]. Available: https://hbr.org/1995/01/the-information-executives-truly-need

[2] T. Davenport, D. Long, and M. Beers, “Successful Knowledge Management Projects,” *Sloan Management Review*, vol. 2, p. 43, Dec. 1998.

[3] S. Hari, C. Egbu, and B. Kumar, “A knowledge capture awareness tool: An empirical study on small and medium enterprises in the construction industry,” *Engineering, Construction and Architectural Management*, vol. 12, no. 6, pp. 533–567, Jan. 2005, doi: 10.1108/09699980510634128.

[4] I. C. Smith and J. H. G. Jr, “Guest Editors&#x27; Introduction: AI Applications in Structural/Construction Engineering,” *IEEE Intelligent Systems*, vol. 11, no. 03, pp. 20–22, May 1996, doi: 10.1109/MEX.1996.506750.

[5] A. M. Al-Ghassani, “Improving the structural design process: a knowledge management approach,” thesis, Loughborough University, 2003. Accessed: Jun. 19, 2023. [Online]. Available: https://repository.lboro.ac.uk/articles/thesis/Improving_the_structural_design_process_a_knowledge_management_approach/9462509/1

[6] E. Girczyc and S. Carlson, “Increasing Design Quality and Engineering Productivity through Design Reuse,” in *30th ACM/IEEE Design Automation Conference*, Jun. 1993, pp. 48–53. doi: 10.1145/157485.164565.

[7] Niels Treldal and Jan Karlshøj, “Information Flow Management in Building Design based on IDM packages,” *Journal of Computing in Civil Engineering*, vol. In review, 2017.

[8] S. Austin, A. Baldwin, B. Li, and P. Waskett, “Integrating Design in the Project Process,” *Proc ICE: Civil Eng*, vol. 138, Jan. 2000, doi: 10.1680/cien.2000.138.4.177.

[9] T. and the Regions. C. T. F. Department of the Environment and J. Egan, *Rethinking construction*. DETR, 1998.

[10] R. Fruchter, K. Saxena, and P. Demian, “Discovery of Re-usable Architecture/Engineering Construction Knowledge Through Exploration of a Corporate Memory,” Nov. 2004.

[11] M. Zanni, R. Soetanto, and K. Ruikar, “Towards a BIM-enabled sustainable building design process: roles, responsibilities, and requirements,” *Architectural Engineering and Design Management*, vol. 13, pp. 1–29, Aug. 2016, doi: 10.1080/17452007.2016.1213153.

[12] M. Park, K. Lee, H.-S. Lee, P. Jiayi, and J. yu, “Ontology-based construction knowledge retrieval system,” *KSCE Journal of Civil Engineering*, vol. 17, Nov. 2013, doi: 10.1007/s12205-013-1155-6.

[13] F. Barbosa *et al.*, “Reinventing construction through a productivity revolution,” *McKinsey Global Institute*, 2017.

[14] N. Treldal, *Digitalization as Driver for Standardized Specification and Design of Buildings: In Search of an Efficient Building Design Management Methodology*. in B Y G D T U. Rapport. Technical University of Denmark, Department of Civil Engineering, 2017.

[15] J. M. K. ANUMBA CHIMAY J., “Introduction to Concurrent Engineering in construction,” in *Concurrent Engineering in Construction Projects*, Routledge, 2006.

[16] E. Pikas, L. Koskela, M. Thalfeldt, B. Dave, and J. Kurnitski, “COMPLEXITY IN DESIGNING ENERGY EFFICIENT BUILDINGS: TOWARDS UNDERSTANDING DECISION NETWORKS IN DESIGN,” Jul. 2015. Accessed: Jun. 20, 2023. [Online]. Available: https://www.semanticscholar.org/paper/COMPLEXITY-IN-DESIGNING-ENERGY-EFFICIENT-BUILDINGS%3A-Pikas-Koskela/df857a9c38e75d08b2637c9aac20932d41626dd1

[17] S. Bertelsen and Sven, “Construction as a Complex System,” *Proceedings for the 11th Annual Conference of the International Group for Lean Construction*, Aug. 2003.

[18] Euphemia Wong, “What Are Wicked Problems and How Might We Solve Them?,” *The Interaction Design Foundation*, Feb. 10, 2022. https://www.interaction-design.org/literature/article/wicked-problems-5-steps-to-help-you-tackle-wicked-problems-by-combining-systems-thinking-with-agile-methodology (accessed Jun. 20, 2023).

[19] B. Lawson, *How Designers Think: The Design Process Demystified*, 4th edition. Amsterdam Heidelberg: Routledge, 2005.

[20] S. Bertelsen, “COMPLEXITY – CONSTRUCTION IN A NEW PERSPECTIVE,” 2003. Accessed: Jun. 20, 2023. [Online]. Available: https://www.semanticscholar.org/paper/COMPLEXITY-%E2%80%93-CONSTRUCTION-IN-A-NEW-PERSPECTIVE-Bertelsen/65c588fc2a38347a939c98a37c459bcb87b8daa0

[21] M. L. Maher, “Expert Systems for Structural Design,” *J. Comput. Civ. Eng.*, vol. 1, no. 4, pp. 270–283, Oct. 1987, doi: 10.1061/(ASCE)0887-3801(1987)1:4(270).

[22] Dean McClements, “What are the Subdivisions of Civil Engineering?,” *NewEngineer*, 2021. https://newengineer.com/advice/what-are-the-subdivisions-of-civil-engineering-1306598 (accessed Jun. 20, 2023).

[23] G. Pahl, W. Beitz, J. Feldhusen, and K.-H. Grote, *Engineering Design: A Systematic Approach*. Springer Science & Business Media, 2006.

[24] T. Bell and R. Plank, *Microcomputers in Civil Engineering*. John Wiley & Sons, Incorporated, 1986.

[25] J. Wacker, J. Hershauer, K. D. Walsh, and C. Sheu, “Estimating professional service productivity: theoretical model, empirical estimates and external validity,” *International Journal of Production Research*, vol. 52, no. 2, pp. 482–495, Jan. 2014, doi: 10.1080/00207543.2013.836611.

[26] “Bridge Information Modeling (BrIM) brings bridge engineering to the modern era.” https://www.tekla.com/resources/articles/bridge-

information-modeling-brim-brings-bridge-engineering-to-the-modern-era-2 (accessed Jun. 21, 2023).

[27] A. Pipinato, Ed., *Innovative Bridge Design Handbook: Construction, Rehabilitation and Maintenance*, 2nd edition. Oxford, United Kingdom Cambridge, MA, United States: Butterworth-Heinemann, 2021.

[28] S. E. Ruikar Kirti, *Collaborative Design Management*. London: Routledge, 2013. doi: 10.4324/9780203819128.

[29] A. Ertas, *Transdisciplinary Engineering Design Process*, 1st edition. Hoboken, NJ: Wiley, 2018.

[30] M. Obergriesser and A. Borrmann, "Infrastructural BIM standards – Development of an Information Delivery Manual for the geotechnical infrastructural design and analysis process," pp. 581–587, Aug. 2012, doi: 10.1201/b12516-93.

[31] D. Grierson and G. Cameron, "An expert system for structural steel design," *Artificial Intelligence in Engineering: Design, edited by JS Gero (New York: Elsevier)*, pp. 279–294, 1988.

[32] S. Austin, A. Newton, J. Steele, and P. Waskett, "Modelling and managing project complexity," *International Journal of Project Management*, vol. 20, no. 3, pp. 191–198, Apr. 2002, doi: 10.1016/S0263-7863(01)00068-0.

[33] M.-C. Tang, "Conceptual design of bridges," *Structure and Infrastructure Engineering*, vol. 13, no. 4, pp. 418–427, Apr. 2017, doi: 10.1080/15732479.2016.1164723.

[34] N. Harty and M. Danaher, "A knowledge-based approach to preliminary design of buildings.," *Proceedings of the Institution of Civil Engineers - Structures and Buildings*, vol. 104, no. 2, pp. 135–144, May 1994, doi: 10.1680/istbu.1994.26324.

[35] B. Kumar and B. H. V. Topping, "Knowledge-based processing for structural design. (includes appendices.)," *Proceedings of the Institution of Civil Engineers*, vol. 90, no. 2, pp. 421–426, Apr. 1991, doi: 10.1680/iicep.1991.14038.

[36] B. Kumar and B. Topping, "Issues in the development of a knowledge-based system for the detailed design of structures," *in Artificial Intelligence in Engineering Design*, pp. 295–314, 1988.

[37] "What Is Computational Design?," Apr. 21, 2022. https://constructible.trimble.com/construction-industry/what-is-computational-design (accessed Apr. 04, 2023).

[38] Jose Luis Garcia del Castillo Lopez, "Introduction to Computational Design," Harvard University Graduate School of Design, 2020. Accessed: Jun. 23, 2023. [Online]. Available: https://www.gsd.harvard.edu/course/introduction-to-computational-design-fall-2020/

[39] N. Kelly and J. S. Gero, "Design thinking and computational thinking: a dual process model for addressing design problems," *Design Science*, vol. 7, p. e8, Jan. 2021, doi: 10.1017/dsj.2021.7.

[40] I. Caetano, L. Santos, and A. Leitão, "Computational design in architecture: Defining parametric, generative, and algorithmic design,"

Frontiers of Architectural Research, vol. 9, no. 2, pp. 287–300, Jun. 2020, doi: 10.1016/j.foar.2019.12.008.

[41] "Ultimate Guide To Computational Design," *Applied Software, GRAITEC Group*, Feb. 21, 2023. https://asti.com/ultimate-guide-to-computational-design/ (accessed Jun. 23, 2023).

[42] M. Bolpagni, R. Gavina, and D. Ribeiro, *Industry 4.0 for the Built Environment: Methodologies, Technologies and Skills*, vol. 20. Springer Nature, 2022.

[43] Phillip Bellis and Steve Reichwein, "Computational Embrace," *STRUCTURE magazine*, 2021. https://www.structuremag.org/?p=18847 (accessed Apr. 04, 2023).

[44] Brittany Fiema and Leland Curtis, "How computational design, modeling are changing engineering," *Consulting - Specifying Engineer*, Jul. 24, 2020. https://www.csemag.com/articles/how-computational-design-modeling-are-changing-engineering/ (accessed Apr. 04, 2023).

[45] M. Maierhofer and A. Menges, *Towards integrative design processes and computational design tools for the design space exploration of adaptive architectural structures*. 2019.

[46] R. Mikaelsson, *Computational Design in the AEC industry : Applications and Limitations*. 2022. Accessed: Jun. 08, 2023. [Online]. Available: https://urn.kb.se/resolve?urn=urn:nbn:se:ltu:diva-94855

[47] Simon Kendal and Malcolm Creen, *An Introduction to Knowledge Engineering*. London: Springer, 2007. doi: 10.1007/978-1-84628-667-4.

[48] F. Ameri and D. Dutta, "Product Lifecycle Management: Closing the Knowledge Loops," *Computer-Aided Design and Applications*, vol. 2, pp. 577–590, Aug. 2013, doi: 10.1080/16864360.2005.10738322.

[49] N. Milton, "Knowledge Acquisition in Practice: A Step-by-step Guide," Jan. 2007.

[50] L. Fahey and L. Prusak, "The Eleven Deadliest Sins of Knowledge Management," *California Management Review*, vol. 40, pp. 265–276, Apr. 1998, doi: 10.2307/41165954.

[51] M. Alavi and D. E. Leidner, "Review: Knowledge Management and Knowledge Management Systems: Conceptual Foundations and Research Issues," *MIS Quarterly*, vol. 25, no. 1, pp. 107–136, 2001, doi: 10.2307/3250961.

[52] R. van der Spek and A. Spijkervet, *Knowledge Management: Dealing Intelligently With Knowledge*. Knowledge Management Network, 1997.

[53] "DIKW pyramid," *Wikipedia*. May 21, 2023. Accessed: Jun. 19, 2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=DIKW_pyramid&oldid=1156200339

[54] I. Nonaka and H. Takeuchi, *The Knowledge-creating Company: How Japanese Companies Create the Dynamics of Innovation*. Oxford University Press, 1995.

[55] I. I. Falqi, "Knowledge Capture and Retrieval in Construction Projects," 2011, doi: 10.13140/RG.2.2.20566.65600.

[56] R. Weber, D. W. Aha, and I. Becerra-Fernandez, “Intelligent lessons learned systems,” *Expert Systems with Applications*, vol. 20, no. 1, pp. 17–34, Jan. 2001, doi: 10.1016/S0957-4174(00)00046-4.

[57] P. F. Drucker, *Post-Capitalist Society*, Reprint edition. New York, NY: Harper Business, 1994.

[58] M. Polanyi, *The Tacit Dimension*. Chicago, IL: University of Chicago Press, 2009. Accessed: Jun. 19, 2023. [Online]. Available: https://press.uchicago.edu/ucp/books/book/chicago/T/bo6035368.html

[59] P. Demian and R. Fruchter, “An ethnographic study of design knowledge reuse in the architecture, engineering and construction industry,” *Research in Engineering Design*, vol. 16, Apr. 2006, doi: 10.1007/s00163-006-0010-x.

[60] Paula María Montesa Rausell, “Developement of a knowledge base for knowledge reuse in design projects. Mini’s interior design as a case of study,” Universitat Politècnica de València, 2016.

[61] A. Tiwana, *The Knowledge Management Toolkit: Orchestrating It, Strategy, and Knowledge Platforms*, 2nd ed. edition. Upper Saddle River, N.J: Prentice Hall, 2002.

[62] T. Davenport and L. Prusak, *Working Knowledge: How Organizations Manage What They Know*, vol. 1. 1998. doi: 10.1145/348772.348775.

[63] M. Anshari, M. Syafrudin, A. Tan, N. L. Fitriyani, and Y. Alas, “Optimisation of Knowledge Management (KM) with Machine Learning (ML) Enabled,” *Information*, vol. 14, no. 1, p. 35, Jan. 2023, doi: 10.3390/info14010035.

[64] J. Abbas, Q. Zhang, I. Hussain, S. Akram, A. Afaq, and M. A. Shad, “Sustainable innovation in small medium enterprises: the impact of knowledge management on organizational innovation through a mediation analysis by using SEM approach,” *Sustainability*, vol. 12, no. 6, p. 2407, 2020.

[65] E. Hajric, “Knowledge Management System and Practices-A Theoretical and Practical Guide for Knowledge Management in Your Organization,” *Jacksonville, Florida, USA: Helpjuice*, 2018.

[66] N. AlQershi, S. S. M. Mokhtar, and Z. B. Abas, “Innovative CRM and performance of SMEs: The moderating role of relational capital,” *Journal of Open Innovation: Technology, Market, and Complexity*, vol. 6, no. 4, p. 155, 2020.

[67] C. López-Nicolás and Á. L. Meroño-Cerdán, “Strategic knowledge management, innovation and performance,” *International journal of information management*, vol. 31, no. 6, pp. 502–509, 2011.

[68] J. Darroch, “Knowledge management, innovation and firm performance,” *Journal of knowledge management*, vol. 9, no. 3, pp. 101–115, 2005.

[69] E. Tsui, “Tracking the Role and Evolution of Commercial Knowledge Management Software,” in *Handbook on Knowledge Management: Knowledge Directions*, C. W. Holsapple, Ed., in International

Handbooks on Information Systems. Berlin, Heidelberg: Springer, 2003, pp. 5–27. doi: 10.1007/978-3-540-24748-7_1.

[70] B. T. Pentland, “Information Systems and Organizational Learning The social epistemology of organizational knowledge systems,” in *Strategic Information Management*, 3rd ed.Routledge, 2002.

[71] R. Fruchter, “Metaphors for knowledge capture, sharing and reuse,” presented at the Proc of eWork and eBusiness in Architecture, Engineering and Construction, Turk & Scherer (eds.), ECPPM Conference, 2002, pp. 17–26.

[72] N. Mohamed, A. A. Rahman, and W. Z. Abidin, “KNOWLEDGE DISSEMINATION TOOLS,” 2009.

[73] A. Abecker, A. Bernardi, K. Hinkelmann, O. Kuhn, and M. Sintek, “Toward a technology for organizational memories,” *Intelligent Systems and their Applications, IEEE*, vol. 13, pp. 40–48, Jun. 1998, doi: 10.1109/5254.683209.

[74] L. Damodaran and W. Olphert, “Barriers and facilitators to the use of knowledge management systems,” *Behaviour & Information Technology*, vol. 19, pp. 405–413, Nov. 2000, doi: 10.1080/014492900750052660.

[75] D. Binney, “The knowledge management spectrum – understanding the KM landscape,” *Journal of Knowledge Management*, vol. 5, no. 1, pp. 33–42, Jan. 2001, doi: 10.1108/13673270110384383.

[76] M. Lytras and N. Pouloudi, “Towards the development of a novel taxonomy of knowledge management systems from a learning perspective: An integrated approach to learning and knowledge infrastructures,” *J. Knowledge Management*, vol. 10, pp. 64–80, Nov. 2006, doi: 10.1108/13673270610709224.

[77] R. Fruchter and P. Demian, “CoMem: Designing an interaction experience for reuse of rich contextual knowledge from a corporate memory,” *AI EDAM*, vol. 16, pp. 127–147, Jun. 2002, doi: 10.1017/S0890060402163025.

[78] R. Grant, “Shifts in the World Economy: The Drivers of Knowledge Management,” 2000, pp. 27–53. doi: 10.1016/B978-0-7506-7247-4.50005-7.

[79] S. Garfield, “What is the difference between a database and knowledge base?,” *Medium*, Sep. 09, 2019. https://stangarfield.medium.com/what-is-the-difference-between-a-database-and-knowledge-base-9c8144981f37 (accessed Jun. 19, 2023).

[80] M. L. Brodie and J. Mylopoulos, “Knowledge Bases vs Databases,” in *On Knowledge Base Management Systems: Integrating Artificial Intelligence and Database Technologies*, M. L. Brodie and J. Mylopoulos, Eds., in Topics in Information Systems. New York, NY: Springer, 1986, pp. 83–86. doi: 10.1007/978-1-4612-4980-1_9.

[81] J. Fong and S.-M. Huang, “A frame model approach for expert and database system integration,” *Int. J. Soft. Eng. Knowl. Eng.*, vol. 09, no. 03, pp. 369–396, Jun. 1999, doi: 10.1142/S021819409900022X.

[82] J. Szelka and Z. Wrona, “Knowledge discovery in data in construction projects,” *Archives of Civil Engineering*, vol. 62, no. 2, 2016.

[83] “Innodata — AI Taxonomies, Ontologies, Schemas, and Knowledge Graphs,” *Innodata Inc.*, Feb. 13, 2020. https://innodata.com/understanding-the-role-of-taxonomies-ontologies-schemas-and-knowledge-graphs/ (accessed Jun. 20, 2023).

[84] 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.

[85] C. Wu, P. Wu, J. Wang, R. Jiang, M. Chen, and X. Wang, “Ontological knowledge base for concrete bridge rehabilitation project management,” *Automation in Construction*, vol. 121, p. 103428, Jan. 2021, doi: 10.1016/j.autcon.2020.103428.

[86] H. Hedden, “The Accidental Taxonomist: Taxonomies vs. Ontologies,” *The Accidental Taxonomist*, Jan. 31, 2023. http://accidental-taxonomist.blogspot.com/2023/01/taxonomies-vs-ontologies.html (accessed Apr. 07, 2023).

[87] P. J. Moreno, J.-M. Van Thong, B. Logan, and G. J. F. Jones, “From multimedia retrieval to knowledge management,” *Computer*, vol. 35, no. 4, pp. 58–66, Mar. 2002, doi: 10.1109/MC.2002.993772.

[88] B. Schmid, K. Stanoevska-Slabeva, S. Handschuh, and A. Hombrecher, “Efficient Information Retrieval: Tools for Knowledge Management,” *University of St.Gallen*, Jan. 1998.

[89] M. H. Jarrahi, D. Askay, A. Eshraghi, and P. Smith, “Artificial intelligence and knowledge management: A partnership between human and AI,” *Business Horizons*, vol. 66, no. 1, pp. 87–99, Jan. 2023, doi: 10.1016/j.bushor.2022.03.002.

[90] R. Heberer, “Expert Systems 2.0,” *Medium*, Aug. 31, 2020. https://towardsdatascience.com/expert-systems-2-0-c8c552f6b2d8 (accessed Jun. 20, 2023).

[91] I. Motawa and A. Almarshad, “A knowledge-based BIM system for building maintenance,” *Automation in Construction*, vol. 29, pp. 173–182, Jan. 2013, doi: 10.1016/j.autcon.2012.09.008.

[92] J. Gammack, H. Akay, C. Ceylan, and S.-G. Kim, “Semantic knowledge management system for design documentation with heterogeneous data using machine learning,” *Procedia CIRP*, vol. 109, pp. 95–100, Jan. 2022, doi: 10.1016/j.procir.2022.05.220.

[93] C. Ghannoum, M. Mabsout, and M.-A. Abdul-Malak, “Knowledge Management Systems for Structural Engineering Design in the Era of Emerging Technological Advancements,” pp. 398–407, Nov. 2020, doi: 10.1061/9780784482889.042.

[94] B. Chakraborty, D. Ghosh, Ranjan, S. Garnaik, and N. Debnath, “Knowledge Management with Case-Based Reasoning applied on Fire Emergency Handling,” in *2010 8th IEEE International Conference on Industrial Informatics*, Jul. 2010, pp. 708–713. doi: 10.1109/INDIN.2010.5549654.

[95] K.-D. Althoff and R. Weber, “Knowledge management in case-based reasoning,” *The Knowledge Engineering Review*, vol. 20, Sep. 2005, doi: 10.1017/S0269888906000543.

[96] S. Hou, H. Li, and Y. Rezgui, “Ontology-based approach for structural design considering low embodied energy and carbon,” *Energy and Buildings*, vol. 102, pp. 75–90, Sep. 2015, doi: 10.1016/j.enbuild.2015.04.051.

[97] S. Szykman, R. Sriram, C. Bochenek, J. W. Racz, and J. Senfaute, “Design repositories: Engineering design’s new knowledge base,” *Intelligent Systems and their Applications, IEEE*, vol. 15, pp. 48–55, Jun. 2000.

[98] M. R. Bohm and R. B. Stone, “Product Design Support: Exploring a Design Repository System,” in *Computers and Information in Engineering*, Anaheim, California, USA: ASMEDC, Jan. 2004, pp. 55–65. doi: 10.1115/IMECE2004-61746.

[99] A. B. Mikes, “Data Mining a Design Repository for Automating and Validating Functional Modeling,” Oregon State University, 2020.

[100] Michael Cook, “A Methodology to Breakdown Bridge Designs into a Hierarchal Decomposition Assembly of Design Entities,” University of Evansville, 2005. Accessed: Jun. 14, 2023. [Online]. Available: http://zeiny.net/Publications/NCUR2005Paper.htm

[101] M. Firdaus, H. Wang, H. Qin, and Y. Liu, “DIGITAL REPOSITORY FOR DESIGN KNOWLEDGE REUSE,” *DS 80-10 Proceedings of the 20th International Conference on Engineering Design (ICED 15) Vol 10: Design Information and Knowledge Management Milan, Italy, 27-30.07.15*, pp. 315–324, 2015.

[102] N. Anwar, *The Impact and Future Role of Computations and Software in Bridge Modeling, Analysis and Design*. 2007.

[103] E. Wenger, “Communities of Practice: The Structure of Knowledge Stewarding,” Elsevier, 2000, pp. 205–224. doi: 10.1016/B978-0-7506-7247-4.50013-6.

[104] S. Ahmed and K. M. Wallace, “Understanding the knowledge needs of novice designers in the aerospace industry,” *Design Studies*, vol. 25, no. 2, pp. 155–173, Mar. 2004, doi: 10.1016/j.destud.2003.10.006.

[105] W. Lauer, *Integrative Dokumenten- und Prozessbeschreibung in dynamischen Produktentwicklungsprozessen*. München: Dr. Hut, 2010.

[106] D. E. O’Leary, “Using AI in knowledge management: knowledge bases and ontologies,” *IEEE Intell. Syst.*, vol. 13, no. 3, pp. 34–39, May 1998, doi: 10.1109/5254.683180.

[107] D. F. Lundvall Bengt-Åke, “The Knowledge-Based Economy: From the Economics of Knowledge to the Learning Economy,” in *The Economic Impact of Knowledge*, Routledge, 1998.

[108] C. Egbu and H. Robinson, “Construction as a Knowledge-Based Industry,” 2008, pp. 31–49. doi: 10.1002/9780470759554.ch3.

[109] M. Millie Kwan and P. Balasubramanian, “KnowledgeScope: managing knowledge in context,” *Decision Support Systems*, vol. 35, no. 4, pp. 467–486, Jul. 2003, doi: 10.1016/S0167-9236(02)00126-4.

[110] H. Ping Tserng and Y.-C. Lin, "Developing an activity-based knowledge management system for contractors," *Automation in Construction*, vol. 13, no. 6, pp. 781–802, Nov. 2004, doi: 10.1016/j.autcon.2004.05.003.

[111] Zuhal Erden, "Phases of Engineering Design," presented at the MECE 202 Principles of Mechatronics design, Department of Mechatronics Engineering, 2011.

[112] J. J. J. Kasvi, M. Vartiainen, and M. Hailikari, "Managing knowledge and knowledge competences in projects and project organisations," *International Journal of Project Management*, vol. 21, no. 8, pp. 571–582, Nov. 2003, doi: 10.1016/S0263-7863(02)00057-1.

[113] C. Carro Saavedra, T. Villodres, and U. Lindemann, *Review and Classification of Knowledge in Engineering Design*. 2017, p. 627. doi: 10.1007/978-981-10-3521-0_53.

[114] V. Karhu, M. Keitilä, and P. Lahdenperä, *Construction process model: Generic present-state systematisation by IDEF0*. in VTT Tiedotteita - Meddelanden - Research Notes. Espoo: VTT Technical Research Centre of Finland, 1997.

[115] P. Demian and P. Balatsoukas, "Information Retrieval from Civil Engineering Repositories: Importance of Context and Granularity," *Journal of Computing in Civil Engineering*, vol. 26, pp. 727–740, Nov. 2012, doi: 10.1061/(ASCE)CP.1943-5487.0000229.

[116] M. Bazire and P. Brézillon, "Understanding Context Before Using It," in *Modeling and Using Context*, A. Dey, B. Kokinov, D. Leake, and R. Turner, Eds., in Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2005, pp. 29–40. doi: 10.1007/11508373_3.

[117] P. Brezillon, "Focusing on context in human-centered computing," *IEEE Intelligent Systems*, vol. 18, no. 3, pp. 62–66, May 2003, doi: 10.1109/MIS.2003.1200731.

[118] S. B. Yoo and Y. Kim, "Web-based knowledge management for sharing product data in virtual enterprises," *International Journal of Production Economics*, vol. 75, no. 1, pp. 173–183, Jan. 2002, doi: 10.1016/S0925-5273(01)00190-6.

[119] P. Balatsoukas and P. Demian, "Effects of granularity of search results on the relevance judgment behavior of engineers: Building systems for retrieval and understanding of context," *Journal of the American Society for Information Science and Technology*, vol. 61, no. 3, pp. 453–467, 2010, doi: 10.1002/asi.21268.

[120] J. F. Maier, C. M. Eckert, and P. J. Clarkson, "Model granularity in engineering design – concepts and framework," *Design Science*, vol. 3, p. e1, ed 2017, doi: 10.1017/dsj.2016.16.

[121] J. F. Maier, C. M. Eckert, and P. J. Clarkson, "MODEL GRANULARITY AND RELATED CONCEPTS," *DS 84: Proceedings of the DESIGN 2016 14th International Design Conference*, pp. 1327–1336, 2016.

[122] C. M. Eckert and M. K. Stacey, "What is a Process Model? Reflections on the Epistemology of Design Process Models," in *Modelling and Management of Engineering Processes*, P. Heisig, P. J. Clarkson, and

S. Vajna, Eds., London: Springer, 2010, pp. 3–14. doi: 10.1007/978-1-84996-199-8_1.

[123] R. Haesen, M. Snoeck, W. Lemahieu, and S. Poelmans, *On the Definition of Service Granularity and Its Architectural Impact*. 2008. doi: 10.1007/978-3-540-69534-9_29.

[124] Y. Fan, C. Liu, and J. Wang, "Integrating multi-granularity model and similarity measurement for transforming process data into different granularity knowledge," *Advanced Engineering Informatics*, vol. 37, pp. 88–102, Aug. 2018, doi: 10.1016/j.aei.2018.04.012.

[125] A. Szdzuy, "Aspects of granularity for components," presented at the Workshop on Component-based Software Development, Technical University Berlin, 2002.

[126] P. Vitharana, H. Jain, and F. Zahedi, "Strategy-based design of reusable business components," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 34, no. 4, pp. 460–474, Nov. 2004, doi: 10.1109/TSMCC.2004.829258.

[127] S. Robinson, "Conceptual modelling for simulation Part I: Definition and requirements," *Journal of the Operational Research Society*, vol. 59, pp. 278–290, Mar. 2008, doi: 10.1057/palgrave.jors.2602368.

[128] M. Pidd, "Just Modeling through: A Rough Guide to Modeling," *Interfaces*, vol. 29, no. 2, pp. 118–132, 1999.

[129] Z.-J. Wang, D.-C. Zhan, and X.-F. Xu, "STCIM: a dynamic granularity oriented and stability based component identification method," *ACM SIGSOFT Software Engineering Notes*, vol. 31, no. 3, pp. 1–14, 2006.

[130] C. Eckert *et al.*, *Integrated Product and Process Models: Towards an Integrated Framework and Review*. 2015.

[131] J. F. Maier, D. C. Wynn, W. Biedermann, U. Lindemann, and P. J. Clarkson, "Simulating progressive iteration, rework and change propagation to prioritise design tasks," *Res Eng Design*, vol. 25, no. 4, pp. 283–307, Oct. 2014, doi: 10.1007/s00163-014-0174-8.

[132] T. R. Browning, E. Fricke, and H. Negele, "Key concepts in modeling product development processes," *Systems engineering*, vol. 9, no. 2, pp. 104–128, 2006.

[133] Y. Geo and A. Kokossis, "A knowledge management platform development for technological monitoring in chemical industry," presented at the Proceedings of the 5th European Conference on Knowledge Management, 2004, pp. 349–356.

[134] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*, First Edition. New York : Harlow, England: Addison Wesley, 1999.

[135] P. Demian and R. Fruchter, "Measuring Relevance in Support of Design Reuse from Archives of Building Product Models," *Journal of Computing in Civil Engineering - J COMPUT CIVIL ENG*, vol. 19, Apr. 2005, doi: 10.1061/(ASCE)0887-3801(2005)19:2(119).

[136] P. Demian, "Information retrieval for the management of civil engineering design content," 2011.

[137] Y. Ye and G. Fischer, "Supporting reuse by delivering task-relevant and personalized information," in *Proceedings of the 24th International*

Conference on Software Engineering, in ICSE ’02. New York, NY, USA: Association for Computing Machinery, May 2002, pp. 513–523. doi: 10.1145/581339.581402.

[138] B. Larsen, A. Tombros, and S. Malik, “Is XML retrieval meaningful to users? searcher preferences for full documents vs. elements,” in *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, in SIGIR ’06. New York, NY, USA: Association for Computing Machinery, Aug. 2006, pp. 663–664. doi: 10.1145/1148170.1148306.

[139] N. Pharo, “The effect of granularity and order in XML element retrieval,” *Information Processing & Management*, vol. 44, no. 5, pp. 1732–1740, Sep. 2008, doi: 10.1016/j.ipm.2008.05.004.

[140] S. Betsi, M. Lalmas, A. Tombros, and T. Tsikrika, *User expectations from XML element retrieval*. 2006, p. 612. doi: 10.1145/1148170.1148280.

[141] P. Demian and R. Fruchter, “Methodology for Usability Evaluation of Corporate Memory Design Reuse Systems,” *Journal of Computing in Civil Engineering*, vol. 20, no. 6, pp. 377–389, Nov. 2006, doi: 10.1061/(ASCE)0887-3801(2006)20:6(377).

[142] H. R. Nemati, D. M. Steiger, L. S. Iyer, and R. T. Herschel, “Knowledge warehouse: an architectural integration of knowledge management, decision support, artificial intelligence and data warehousing,” *Decision Support Systems*, vol. 33, no. 2, pp. 143–161, Jun. 2002, doi: 10.1016/S0167-9236(01)00141-5.

[143] T.-C. Chou, P.-L. Chang, Y.-P. Cheng, and C.-T. Tsai, “A path model linking organizational knowledge attributes, information processing capabilities, and perceived usability,” *Information & Management*, vol. 44, no. 4, pp. 408–417, Jun. 2007, doi: 10.1016/j.im.2007.03.003.

[144] R. Matkar and A. Parab, “Ontology based expert systems – replication of human learning,” S. J. Pise, Ed., New Delhi: Springer India, 2011, pp. 43–47. doi: 10.1007/978-81-8489-989-4_7.

[145] BrainKart, “Rule Based Architecture of an Expert System,” *BrainKart*. https://www.brainkart.com/article/Rule-Based-Architecture-of-an-Expert-System_8931/ (accessed May 31, 2023).

[146] Muhanad Tahrir Younis, “Chapter Three: Rule-Based Expert Systems,” Mustansiriyah University. [Online]. Available: https://uomustansiriyah.edu.iq/media/lectures/6/6_2022_04_02!07_55_32_PM.pdf

[147] George Gesior, “AI Then and Now – Expert Systems,” 2018. https://www.linkedin.com/pulse/ai-now-expert-systems-george-gesior/ (accessed May 31, 2023).

[148] Pawara Siriwardhane, “An Introduction to Expert System Shells,” 2022. https://medium.com/nerd-for-tech/an-introduction-to-expert-system-shells-530043914ec0#86b9 (accessed May 31, 2023).

[149] B. V. Gompel, “Capturing Engineering Knowledge : Concepts of Automation in Bridge Design,” 2018. Accessed: May 31, 2023. [Online]. Available: https://www.semanticscholar.org/paper/Capturing-

Engineering-Knowledge-%3A-Concepts-of-in-Gompel/a7423490ff0a9e423214d992b7a35de58d8489f9

[150] M. Sandberg, "Towards a Knowledge-Based Engineering Methodology for Construction," Sep. 2015, pp. 1–8. doi: 10.1061/9780784479377.001.

[151] R. Hoog, R. Martil, B. Wielinga, C. Bright, W. Velde, and T. Ross, "The Common KADS model set," Feb. 1998.

[152] M. Stokes, "Managing Engineering Knowledge: MOKA-Methodology for Knowledge Based Engineering Applications," 2001. Accessed: May 31, 2023. [Online]. Available: https://www.semanticscholar.org/paper/Managing-Engineering-Knowledge%3A-MOKA-Methodology-Stokes/5b8d8d834c686d14c1057da70dd65f172c9f6163

[153] J. Berri, "Towards a framework for Collective Intelligence," *2010 Fifth International Conference on Digital Information Management (ICDIM)*, pp. 454–459, Jul. 2010, doi: 10.1109/ICDIM.2010.5662446.

[154] S. Austin, A. Baldwin, and A. Newton, "Manipulating the flow of design information to improve the programming of building design," *Construction Management and Economics*, vol. 12, no. 5, pp. 445–455, Sep. 1994, doi: 10.1080/01446199400000054.

[155] B. Hollins and G. Hollins, "Hurst (1993). Design Management Processes that can be used by Practitioners," in *Proceedings of 9th International Conference on Engineering Design, ICED*, 1993, pp. 656–663.

[156] S. AUSTIN, A. BALDWIN, and A. NEWTON, "A Data Flow Model to Plan and Manage the Building Design Process," *Journal of Engineering Design*, vol. 7, no. 1, pp. 3–25, Mar. 1996, doi: 10.1080/09544829608907924.

[157] Autodesk, "Anatomy of a Good Generative Design Process," *Generative Design Primer*, 2021. https://www.generativedesign.org/01-introduction/01-02_generative-design/01-02-05_anatomy-of-a-good-generative-design-process (accessed May 31, 2023).

[158] P. Slatter, "Building expert systems: cognitive emulation," Jan. 1987. Accessed: May 31, 2023. [Online]. Available: https://www.semanticscholar.org/paper/Building-expert-systems%3A-cognitive-emulation-Slatter/921294cdf63f476c98c31a07c5f90dc25e3c38eb

[159] L. Medsker, M. Tan, and E. Turban, "Knowledge acquisition from multiple experts: Problems and issues," *Expert Systems with Applications*, vol. 9, no. 1, pp. 35–40, Jan. 1995, doi: 10.1016/0957-4174(94)00046-X.

[160] A. D. Preece, "Towards a methodology for evaluating expert systems," *Expert Systems*, vol. 7, no. 4, pp. 215–223, Nov. 1990, doi: 10.1111/j.1468-0394.1990.tb00234.x.

[161] J. M. Garza, C. Ibbs, and E. East, "Knowledge Elicitation Techniques for Construction Scheduling," 1991. Accessed: May 31, 2023. [Online]. Available: https://www.semanticscholar.org/paper/Knowledge-Elicitation-Techniques-for-Construction-Garza-Ibbs/a190bc8624e057d2497fc5bf7cc47c9ef482f71b

[162] Z. Zhou, Y. M. Goh, and L. Shen, “Overview and Analysis of Ontology Studies Supporting Development of the Construction Industry,” *Journal of Computing in Civil Engineering*, vol. 30, no. 6, p. 04016026, Nov. 2016, doi: 10.1061/(ASCE)CP.1943-5487.0000594.

[163] N. El-Gohary, A. Asce, and T. El-Diraby, “Domain Ontology for Processes in Infrastructure and Construction,” *Journal of Construction Engineering and Management*, vol. 136, Jul. 2010, doi: 10.1061/(ASCE)CO.1943-7862.0000178.

[164] T. E. El-Diraby, “Domain Ontology for Construction Knowledge,” *Journal of Construction Engineering and Management*, vol. 139, no. 7, pp. 768–784, Jul. 2013, doi: 10.1061/(ASCE)CO.1943-7862.0000646.

[165] buildingSMART, “Industry Foundation Classes (IFC),” *buildingSMART International*. https://www.buildingsmart.org/standards/bsi-standards/industry-foundation-classes/ (accessed Jun. 02, 2023).

[166] C. Lima, T. El-Diraby, B. Fiés, A. Zarli, and E. Ferneley, *The E-Cognos Project: Current Status and Future Directions of an Ontology-Enabled IT Solution Infrastructure Supporting Knowledge Management in Construction*. 2003. doi: 10.1061/40671(2003)103.

[167] C. M. Eastman and G. V. Shirley, “Management of Design Information Flows,” in *Management of Design: Engineering and Management Perspectives*, S. Dasu and C. Eastman, Eds., Dordrecht: Springer Netherlands, 1994, pp. 255–277. doi: 10.1007/978-94-011-1390-8_14.

[168] H. Dong, F. K. Hussain, and E. Chang, “An ontology-based real-time project monitoring system in the cloud,” Aug. 2011, Accessed: Jun. 04, 2023. [Online]. Available: https://opus.lib.uts.edu.au/handle/10453/111314

[169] M. Niknam and S. Karshenas, “A shared ontology approach to semantic representation of BIM data,” *Automation in Construction*, vol. 80, pp. 22–36, Aug. 2017, doi: 10.1016/j.autcon.2017.03.013.

[170] S. Simoff and M. Maher, “Ontology-Based Multimedia Data Mining for Design Information Retrieval,” *Computing in Civil Engineering*, 1998, Accessed: Jun. 04, 2023. [Online]. Available: https://www.semanticscholar.org/paper/Ontology-Based-Multimedia-Data-Mining-for-Design-Simoff-Maher/80bcaa895e0becff33b6191f8149d906ddf9c716

[171] G. Ren, R. Ding, and H. Li, “Building an ontological knowledgebase for bridge maintenance,” *Advances in Engineering Software*, vol. 130, pp. 24–40, Apr. 2019, doi: 10.1016/j.advengsoft.2019.02.001.

[172] R. Li, T. Mo, J. Yang, S. Jiang, T. Li, and Y. Liu, “Ontologies-Based Domain Knowledge Modeling and Heterogeneous Sensor Data Integration for Bridge Health Monitoring Systems,” *IEEE Transactions on Industrial Informatics*, vol. 17, no. 1, pp. 321–332, Jan. 2021, doi: 10.1109/TII.2020.2967561.

[173] S. Zhang, F. Boukamp, and J. Teizer, “Ontology-based semantic modeling of construction safety knowledge: Towards automated safety planning for job hazard analysis (JHA),” *Automation in Construction*, vol. 52, pp. 29–41, Apr. 2015, doi: 10.1016/j.autcon.2015.02.005.

[174] J. Wang, “Total Constraint Management for Improving Construction Work Flow in Liquefied Natural Gas Industry,” Thesis, Curtin University, 2018. Accessed: Jun. 04, 2023. [Online]. Available: https://espace.curtin.edu.au/handle/20.500.11937/73516

[175] F. H. Abanda, J. H. M. Tah, and D. Duce, “PV-TONS: A photovoltaic technology ontology system for the design of PV-systems,” *Engineering Applications of Artificial Intelligence*, vol. 26, no. 4, pp. 1399–1412, Apr. 2013, doi: 10.1016/j.engappai.2012.10.010.

[176] X. Wang, T. N. Wong, and Z.-P. Fan, “Ontology-based supply chain decision support for steel manufacturers in China,” *Expert Systems with Applications*, vol. 40, no. 18, pp. 7519–7533, 2013, doi: 10.1016/j.eswa.2013.07.061.

[177] 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, doi: 10.1016/j.eswa.2012.02.002.

[178] C. Chapman, S. Preston, M. Pinfold, and G. Smith, “Utilising enterprise knowledge with knowledge-based engineering,” *IJCAT*, vol. 28, pp. 169–179, Jan. 2007, doi: 10.1504/IJCAT.2007.013354.

[179] T.-S. Liau and W.-C. Wang, “Representations of Building Design Process with Iterations,” presented at the 17th International Symposium on Automation and Robotics in Construction, Taipei, Taiwan, Sep. 2000. doi: 10.22260/ISARC2000/0176.

[180] S. Austin, A. Baldwin, B. Li, and P. Waskett, “Analytical design planning technique: a model of the detailed building design process,” *Design Studies*, vol. 20, no. 3, pp. 279–296, May 1999, doi: 10.1016/S0142-694X(98)00038-6.

[181] U.S. DEPARTMENT OF COMMERCE, *INTEGRATION DEFINITION FOR FUNCTION MODELING (IDEF0)*. FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION, 1993.

[182] F. J. O’Donnell and A. H. B. Duffy, “Modelling design development performance,” *International Journal of Operations & Production Management*, vol. 22, no. 11, pp. 1198–1221, Jan. 2002, doi: 10.1108/01443570210450301.

[183] L. Wang and F. Leite, “Process Knowledge Capture in BIM-Based Mechanical, Electrical, and Plumbing Design Coordination Meetings,” *Journal of Computing in Civil Engineering*, vol. 30, no. 2, p. 04015017, Mar. 2016, doi: 10.1061/(ASCE)CP.1943-5487.0000484.

[184] M. Shafiq, J. Matthews, and S. Lockley, “A study of BIM collaboration requirements and available features in existing model collaboration systems,” *Electronic Journal of Information Technology in Construction (ITcon)*, vol. 18, pp. 148–161, Aug. 2013.

[185] G. Lee, C. M. Eastman, and R. Sacks, “Eliciting information for product modeling using process modeling,” *Data & Knowledge Engineering*, vol. 62, no. 2, pp. 292–307, Aug. 2007, doi: 10.1016/j.datak.2006.08.005.

[186] V. Knotten, F. Svalestuen, G. K. Hansen, and O. Lædre, “Design Management in the Building Process - A Review of Current Literature,” *Procedia Economics and Finance*, vol. 21, pp. 120–127, Jan. 2015, doi: 10.1016/S2212-5671(15)00158-6.

[187] B. S. Bell and S. W. J. Kozlowski, “A Typology of Virtual Teams: Implications for Effective Leadership,” *Group & Organization Management*, vol. 27, no. 1, pp. 14–49, Mar. 2002, doi: 10.1177/1059601102027001003.

[188] V. Knotten, F. Svalestuen, S. Aslesen, and H. Dammerud, “Integrated methodology for design management–a research project to improve design management for the AEC industry in Norway,” presented at the Proceedings of the 22nd Annual Conference of the International Group for Lean Construction, 2014, pp. 1391–1400.

[189] buildingSMART, “IfcBridgePart (bSDD).” 2023. Accessed: Jun. 06, 2023. [IFC4.3]. Available: https://search.bsdd.buildingsmart.org/Classification/Index/71088

[190] Matti Hannus and Kari Pietilainen, “Implementation concerns of process modelling tools,” 1995, Accessed: Jun. 06, 2023. [Online]. Available: https://itc.scix.net/paper/w78-1995-449

[191] “SA help: Activities, Processes, Sub-Processes, and Tasks.” https://support.unicomsi.com/manuals/systemarchitect/11482/starthelp.html#page/Architecting_and_designing/BusinessProcessAnalysis.03.005.html (accessed Jun. 06, 2023).

[192] E. Ganesan, “Process Hierarchy and Granularity Definition in Enterprise Process Modeling.” Rochester, NY, Jan. 18, 2011. Accessed: Jun. 06, 2023. [Online]. Available: https://papers.ssrn.com/abstract=1929715

[193] “Definition - Activity vs. Task - Definitions in Process Mapping and Projects,” *Elsmar Cove Quality and Business Standards Discussions*, Dec. 21, 2010. https://elsmar.com/elsmarqualityforum/threads/activity-vs-task-definitions-in-process-mapping-and-projects.45345/ (accessed Jun. 07, 2023).

[194] Theiio, “What is a task, an activity, a process? - Theiio | Task, Activity, Process,” *Theiio*, Oct. 02, 2019. https://theiio.com/task-activity-process-what-are-they/ (accessed Jun. 07, 2023).

[195] Stefano Damiani, “IDEF0-Activity Modeling,” presented at the Metodi di Supporto alle Decisioni Manageriali, Università Roma Tre, 2018. Accessed: Jun. 09, 2023. [Online]. Available: http://damiani.inf.uniroma3.it/CORSI/MSDM/index_file/idef0-v2.pdf

[196] “Function model,” *Wikipedia*. Mar. 18, 2023. Accessed: Jun. 08, 2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Function_model&oldid=1145277454#Functional_decomposition

[197] C. T. Formoso, “A knowledge based framework for planning house building projects,” 1991. Accessed: Jun. 08, 2023. [Online]. Available: https://lume.ufrgs.br/handle/10183/18667

[198] “What goes into a Generative Design Process?,” *Generative Design Primer*. https://www.generativedesign.org/01-introduction/01-

02_generative-design/01-02-03_what-goes-into-a-generative-design-process (accessed Jun. 08, 2023).

[199] P. R. Waskett, "Management of the building design process with the Analytical Design Planning Technique," PhD Thesis, Loughborough University, 1999.

[200] B. J. Hicks, S. J. Culley, R. D. Allen, and G. Mullineux, "A framework for the requirements of capturing, storing and reusing information and knowledge in engineering design," *International Journal of Information Management*, vol. 22, no. 4, pp. 263–280, Aug. 2002, doi: 10.1016/S0268-4012(02)00012-9.

[201] A. Snyder, "Encapsulation and inheritance in object-oriented programming languages," *SIGPLAN Not.*, vol. 21, no. 11, pp. 38–45, Jun. 1986, doi: 10.1145/960112.28702.

[202] Graham, "Why inheritance never made any sense | Structure and Interpretation of Computer Programmers," *Structure and Interpretation of Computer Programmers*, Dec. 03, 2021. https://www.sicpers.info/2018/03/why-inheritance-never-made-any-sense/ (accessed Jun. 08, 2023).

[203] T. Browning, "Applying The Design Structure Matrix To System Decomposition And Integration Problems: A Review And New Directions," *Engineering Management, IEEE Transactions on*, vol. 48, pp. 292–306, Sep. 2001, doi: 10.1109/17.946528.

[204] N. Chiriac, K. Hölttä-Otto, D. Lysy, and E. Suk Suh, "Level of modularity and different levels of system granularity," 2011.

[205] A. Merakli, "Pure Functions: High Cohesion, Low Coupling," *Medium*, Oct. 30, 2022. https://ammarmerakli.medium.com/pure-functions-high-cohesion-low-coupling-174b0a47ef24 (accessed Jun. 17, 2023).

[206] C. Tsagkani and A. Tsalgatidou, "Process model abstraction for rapid comprehension of complex business processes," *Information Systems*, vol. 103, p. 101818, Jan. 2022, doi: 10.1016/j.is.2021.101818.

[207] Z. Li, Q. A. Rahman, R. Ferrari, and N. H. Madhavji, "Does Requirements Clustering Lead to Modular Design?," in *Requirements Engineering: Foundation for Software Quality*, M. Glinz and P. Heymans, Eds., in Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2009, pp. 233–239. doi: 10.1007/978-3-642-02050-6_20.

[208] L. J. Kaetzel and J. R. Clifton, "Expert/knowledge based systems for materials in the construction industry: State-of-the-art report," *Materials and Structures*, vol. 28, no. 3, pp. 160–174, Apr. 1995, doi: 10.1007/BF02473222.

[209] Javier Béjar Alonso, "Knowledge Representation," Computer Science Department, Facultat d'Informàtica de Barcelona - Universitat Politècnica de Catalunya, 2009. [Online]. Available: https://www.cs.upc.edu/~bejar/ia/transpas/teoria.mti/3-RC2-Representacion_Frames-eng.pdf

[210] P. Harmon, "Object-oriented AI: a commercial perspective," *Commun. ACM*, vol. 38, no. 11, pp. 80–86, Nov. 1995, doi: 10.1145/219717.219783.

[211] C. Rattanaprateep and S. Chittayasothorn, “A frame-based object-relational database expert system architecture and implementation,” in *Proceedings of the 5th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases*, in AIKED’06. Stevens Point, Wisconsin, USA: World Scientific and Engineering Academy and Society (WSEAS), Feb. 2006, pp. 327–332.

[212] “Frame (artificial intelligence),” *Wikipedia*. May 13, 2023. Accessed: Jun. 10, 2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Frame_(artificial_intelligence)&oldid=1154549514#Comparison_of_frames_and_objects

[213] Y. Zhang, J. Liu, and K. Hou, “Building a Knowledge Base of Bridge Maintenance Using Knowledge Graph,” *Advances in Civil Engineering*, vol. 2023, pp. 1–16, Apr. 2023, doi: 10.1155/2023/6047489.

[214] David Sorrentino, “Integrating Semantic Networks and object-oriented model to represent and manage context,” Reykjavik University, 2012.

[215] H. Shiva Kumar, C. S. Krishnamoorthy, and N. Rajagopalan, “A process model for knowledge-based concrete bridge design,” *Engineering Applications of Artificial Intelligence*, vol. 8, no. 4, pp. 435–447, Aug. 1995, doi: 10.1016/0952-1976(95)00019-W.

[216] Betty M. Miller, *Object-Oriented Expert Systems and Their Applications to Sedimentary Basin Analysis*. U.S. Geological Survey, Reston, VA, 1993.

[217] John A. Bullinaria, “IAI : Semantic Networks and Frames,” School of Computer Science, University of Birmingham, 2005. [Online]. Available: https://www.cs.bham.ac.uk/~jxb/IAI/w6.pdf

[218] H. Harmse, “The Correspondence between DLs/OWL and OO,” *Henriette’s Notes*, Sep. 09, 2017. https://henrietteharmse.com/2017/09/09/the-correspondence-between-dlsowl-and-oo/ (accessed Jun. 10, 2023).

[219] Tara O’Grady, “Chapter 2.2 | Semantic Networks and Frames.” CSC 416 - Artificial Intelligence Programming Languages, SUNY Oswego, 2016. [Online]. Available: https://www.cs.oswego.edu/~togrady/csc416/hw/416miningCh2.2.pdf

[220] H. Wang *et al.*, *Frames and OWL side by side*. 2006.

[221] N. Staff, “The Database Model Showdown: An RDBMS vs. Graph Comparison,” *Graph Database & Analytics*, Aug. 03, 2015. https://neo4j.com/blog/database-model-comparison/ (accessed Jun. 10, 2023).

[222] Agustin Martinez, “Graph Database – Folder IT,” 2022. https://folderit.net/graph-databases/ (accessed Jun. 10, 2023).

[223] Jesús Barrasa, “Building a semantic graph in Neo4j,” *Graph-backed thoughts*, Apr. 06, 2016. https://jbarrasa.com/2016/04/06/building-a-semantic-graph-in-neo4j/ (accessed Jun. 10, 2023).

[224] Michael DeBellis, “Semantic Web vs. Property Graphs,” *MichaelDeBellis*, May 30, 2022. https://www.michaeldebellis.com/post/owlvspropgraphs (accessed Jun. 10, 2023).

[225] R. Stevens, C. A. Goble, and S. Bechhofer, "Ontology-based knowledge representation for bioinformatics," *Brief Bioinform*, vol. 1, no. 4, pp. 398–414, Nov. 2000, doi: 10.1093/bib/1.4.398.

[226] W. Gielingh, *General AEC reference model (GARM) an aid for the integration of application specific product definition models*, vol. 126. 1988.

[227] N. Noy and D. Mcguinness, "Ontology Development 101: A Guide to Creating Your First Ontology," *Knowledge Systems Laboratory*, vol. 32, Jan. 2001.

[228] M. Debellis, *A Practical Guide to Building OWL Ontologies Using Protégé 5.5 and Plugins*. 2021.

[229] D. Allen, "Graph Data Modeling: Categorical Variables," *Neo4j Developer Blog*, Oct. 23, 2020. https://medium.com/neo4j/graph-data-modeling-categorical-variables-dd8a2845d5e0 (accessed Jun. 12, 2023).

[230] T. Fuchino, T. Takamura, and R. Batres, "Development of Engineering Ontology on the Basis of IDEF0 Activity Model," in *Knowledge-Based Intelligent Information and Engineering Systems*, R. Khosla, R. J. Howlett, and L. C. Jain, Eds., in Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2005, pp. 162–168. doi: 10.1007/11552413_24.

[231] C. Walton, *Agency and the Semantic Web*, 1st edition. Oxford: Oxford University Press, 2006.

[232] C. Golbreich, "Combining Rule and Ontology Reasoners for the Semantic Web," in *Rules and Rule Markup Languages for the Semantic Web*, G. Antoniou and H. Boley, Eds., in Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2004, pp. 6–22. doi: 10.1007/978-3-540-30504-0_2.

[233] M. O'Connor and A. Das, *SQWRL: a query language for OWL*. 2009.

[234] B. Curtis (Curt) (CIV), "[protege-user] SPARQL subquery in Protege SNAP-SPARQL tab," Mar. 03, 2020. Accessed: Jun. 14, 2023. [Online]. Available: https://mailman.stanford.edu/pipermail/protege-user/2020-March/011298.html

[235] C. J. Moore, J. C. Miles, and D. W. G. Rees, "Decision support for conceptual bridge design," *Artificial Intelligence in Engineering*, vol. 11, no. 3, pp. 259–272, Jul. 1997, doi: 10.1016/S0954-1810(96)00041-6.

[236] P. Schönfelder, T. Al-Wesabi, A. Bach, and M. König, *Information Extraction from Text Documents for the Semantic Enrichment of Building Information Models of Bridges*. 2022. doi: 10.22260/ISARC2022/0026.

[237] M. Häußler and A. Borrmann, "Knowledge-based engineering in the context of railway design by integrating BIM, BPMN, DMN and the methodology for knowledge-based engineering applications (MOKA)," *Journal of Information Technology in Construction (ITcon)*, vol. 26, no. 12, pp. 193–226, May 2021, doi: 10.36680/j.itcon.2021.012.

[238] M. Owoc, "Contextual Knowledge Granularity," Jan. 2011.

Appendix A: SPAQRL Query

```
#Get all blocks in the ontology, group by block names, and show all the processes of each block

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 proc: <urn:webprotege:ontology:25418ca8-24cf-4a1c-91f8-d78be63bbd6f#>



SELECT ?block (COUNT(?process) AS ?total) (GROUP_CONCAT(REPLACE(STR(?process), "urn:webprotege:ontology:25418ca8-24cf-4a1c-91f8-d78be63bbd6f#", ""); SEPARATOR=", ") AS ?names)

WHERE {

  ?process proc:hasICOM ?block.

}

GROUP BY ?block

ORDER BY ?block
```

```
#Get all processes, order by next activity position (by counting previous activities),  filter all the children activities (only take parents)

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 proc: <urn:webprotege:ontology:25418ca8-24cf-4a1c-91f8-d78be63bbd6f#>



SELECT ?process (COUNT(?Prprocess) AS ?co) (GROUP_CONCAT(REPLACE(STR(?Prprocess),
"urn:webprotege:ontology:25418ca8-24cf-4a1c-91f8-
d78be63bbd6f#", ""); SEPARATOR=", ") AS ?names)

WHERE {?process a proc:Process

Optional{      ?process proc:hasPreviousActivity ?Prprocess. }


Optional{    ?process proc:isChildActivityOf ?childActivity.}


FILTER (!bound(?childActivity))

}

GROUP BY ?process

ORDER BY ?co
```

#Get all processes, get variantactivity of chose activity that created the wanted part, filter the variants that's not the chosen

```
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 proc: <urn:webprotege:ontology:25418ca8-24cf-4a1c-
91f8-d78be63bbd6f#>


SELECT ?process ?c

WHERE {

  ?process rdf:type proc:Process

  optional {?select proc:createPart proc:Pile }

optional{?select proc:hasVariantActivity ?c}

MINUS {

    ?process proc:hasVariantActivity ?select .
  }

}

#Combine the last 2 queries

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 proc: <urn:webprotege:ontology:25418ca8-24cf-4a1c-
91f8-d78be63bbd6f#>


SELECT distinct ?process (COUNT(?Prprocess) AS ?co)

WHERE {

  ?process rdf:type proc:Process

Optional{      ?process proc:hasPreviousActivity ?Prprocess. }

Optional{    ?process proc:isChildActivityOf ?childActivity.}


FILTER (!bound(?childActivity))

  optional {?select proc:createPart proc:Pile }

optional{?select proc:hasVariantActivity ?c}


MINUS {

    ?process proc:hasVariantActivity ?select .

  }

  optional {?select1 proc:createPart proc:Bearing }

optional{?select1 proc:hasVariantActivity ?c1}
```

```
MINUS {

   ?process proc:hasVariantActivity ?select1 .

  }


}

GROUP BY ?process

ORDER BY ?co




#add block name

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 proc: <urn:webprotege:ontology:25418ca8-24cf-4a1c-91f8-d78be63bbd6f#>


SELECT distinct ?process (COUNT(?Prprocess) AS ?co) (GROUP_CONCAT(REPLACE(STR(?block), "urn:webprotege:ontology:25418ca8-24cf-4a1c-91f8-d78be63bbd6f#", ""); SEPARATOR=", ") AS ?names)

WHERE {

  ?process rdf:type proc:Process

Optional{      ?process proc:hasPreviousActivity ?Prprocess. }

Optional{    ?process proc:isChildActivityOf ?childActivity.}

Optional{     ?process proc:hasICOM ?block.}

FILTER (!bound(?childActivity))
```

```
  optional {?select proc:createPart proc:Pile }
optional{?select proc:hasVariantActivity ?c}

MINUS {
    ?process proc:hasVariantActivity ?select .
  }
  optional {?select1 proc:createPart proc:Bearing }
optional{?select1 proc:hasVariantActivity ?c1}

MINUS {
    ?process proc:hasVariantActivity ?select1 .
  }

}
GROUP BY ?process
ORDER BY ?co
```