LLM-augmented semantic web technologies for knowledge sharing and reuse in construction: A proof-of-concept framework

Zhen Zhang[1], Yang Zou[1,2], Johannes Dimyadi[3], and Brian H.W. Guo[4]

[1]Department of Civil and Environmental Engineering, University of Auckland, New Zealand
[2]School of Architecture, Building and Civil Engineering, Loughborough University, UK
[3]Codify Asset Solutions (CAS) Ltd, New Zealand
[4]Department of Civil and Environmental Engineering, University of Canterbury, New Zealand
zhang.zhen@auckland.ac.nz, y.zou2@lboro.ac.uk, jdimyadi@cas.net.nz, brian.guo@canterbury.ac.nz

Abstract

Effective knowledge sharing and reuse across construction projects are frequently impeded by various project conditions and user requirements. Semantic web technologies (SWTs) provide a structured method for managing reusable knowledge in such dynamic environments. Nevertheless, the broader adoption of SWTs has been constrained by the limited availability of skilled knowledge engineers and the complexity of knowledge model maintenance over time. Large language models (LLMs) offer a potential to significantly reduce the barriers of implementing SWTs, which is largely ignored in current research and practices. This paper introduces an SWT-LLM integration framework to facilitate dynamic knowledge sharing and reuse in the construction industry. This is a proof-of-concept study that demonstrates how LLMs, grounded in a reusable IFC-based ontology, can automate the generation of executable semantic web artifacts (SWRL, SPARQL, SHACL) from natural language requirements. A case study of modular product design (MPD) was conducted to evaluate the proposed framework, and the results indicate a high level of performance, with an average combined score of 0.87 (on a scale of 0 to 1) across key knowledge sharing and reuse tasks with GPT-4o and DeepSeek-V3.2.

Keywords –

Knowledge sharing and reuse; Large language model (LLM); Semantic web technologies (SWTs); Ontology; Modular product design (MPD)

1 Introduction

Knowledge sharing and reuse in construction are challenged by the fragmented organization and changing context among projects. To address this challenge, semantic web technologies (SWTs) have been adopted in managing construction domain knowledge. SWTs are a set of tools for capturing and reusing engineering knowledge constructed in an ontological model with concepts, relationships and properties and adapting them for specific project contexts and requirements [1]. However, the implementation of SWTs to facilitate knowledge sharing and reuse is hindered by the poor collaboration between domain experts and knowledge engineers to develop and maintain an effective ontology and database across projects.

Large language models (LLMs) have demonstrated strong capabilities in learning and disseminating knowledge. In the context of construction knowledge management, they hold significant potential to improve knowledge sharing and reuse [2]. However, their effectiveness is constrained by the highly dynamic and context-specific nature of construction projects. Fine-tuning LLMs for each unique scenario is not only time-intensive but also cost-prohibitive. To address this, a promising solution is to augment LLMs with external knowledge bases using SWTs [3], which can enable adaptive, context-aware responses without extensive retraining. These technologies allow for the preservation and evolution of project-specific knowledge within a shared ontological framework. Despite this potential, there is a notable research gap: the integration of SWTs with LLMs in the construction sector remains underexplored. Few studies have investigated how semantic technologies can be systematically combined with LLMs to support flexible, scalable knowledge management tailored to construction projects.

To advance SWTs using LLMs, this study proposed a SWT-LLM integration framework to automate key tasks of knowledge sharing and reuse in construction, such as customized rule development, project-specific knowledge adaptation, and constraint-based compliance checking. Ultimately, this proof-of-concept framework seeks to lower the barriers to implementing SWTs in the

construction industry. Firstly, Section 2 provides some necessary background about the SWTs and LLM in construction and how the research is designed. Then, Section 3 illustrates the integration of SWTs and LLMs. In Section 4, a case study about reusing MPD knowledge is conducted to evaluate this framework. Finally, findings are discussed in Section 5, and the paper is concluded in Section 6.

2 Background

2.1 SWTs for construction

In construction projects, knowledge is shared and reused based on specific project contexts and client requirements. It means that some knowledge cannot be directly applied to different projects without adaptation. For example, the functional requirements of a building influence the selection of spatial spans, which in turn determine the dimensions of beams and columns in space modules. These structural specifications subsequently affect the dimensions of required connections. Although this can be achieved by using some tools like Revit to establish their parametric relationships, the development of Revit families can be time-consuming, and their application has lower flexibility and scalability for integrated collaboration due to data exchange issues. By using SWTs, the construction knowledge and information can be organized in a semantically consistent ontology and adapted based on dynamic project conditions for effective sharing and reuse, such as the design of prefabricated façades [4] and timber houses [5].

The implementation of SWTs can be seen as a set of processes for the development and maintenance of an ontological knowledge model and its database. It is implemented using several interrelated SWTs, including: (1) Resource Description Framework (RDF) for data story; (2) Web Ontology Language (OWL) for representing complex and rich domain knowledge through formal semantics; (3) SPARQL for querying and manipulating RDF/OWL-based knowledge according to user-defined requirements; (4) Semantic Web Rule Language (SWRL) for defining custom inference rules that extend OWL axioms to derive new knowledge; and (5) Shapes Constraint Language (SHACL) for expressing and validating constraints that enforce compliance with user-specified requirements. Ontologies in this framework can be tailored either manually using SPARQL queries or automatically by SWRL rules. Their conformity to defined requirements can then be validated using SHACL constraints.

However, the widespread industrial adoption of SWTs still encounters difficulties [6]. The construction industry is highly decentralized and organized around individual projects, preventing the long-term accumulation and standardization of domain expertise. Additionally, construction suffers from a lack of collaboration between domain experts and skilled knowledge engineers, making it difficult to capture knowledge into digital systems.

2.2 Enhancing SWTs with LLMs

Integrating ontologies into LLMs has emerged as a valuable and effective approach for enhancing information retrieval and understanding in the construction domain [3]. Conversely, knowledge management processes are also expected to benefit from these ontology-augmented LLMs, as they can operate within the same enriched knowledge context.

Pan et al. [7] summarized several research directions of augmenting knowledge graph (KG) completion and construction using LLMs, such as entity discovery and relation extraction [8], but did not clarify the methods of knowledge updating. Allen et al. [2] proposed LLMs as components or tools enabling natural language in the practice of SWTs to assist knowledge engineers or knowledge users in their daily work. For example, users can generate and edit an ontology through conversation with an LLM-based chatbot [9]. In the construction industry, Durmus et al. [10] tried to develop building safety ontologies using LLMs, showing a potential of LLM-enhanced SWT implementation. Yoon et al. [11] utilized the ontology-enhanced LLMs to modify the building operation data in Turtle syntax. However, it should be recognized that directly generating ontology through LLMs is problematic because the data in Turtle format is hard to maintain when context is changing and affecting other interrelated instances [12]. In other studies [13–15], LLMs were used to transform natural language into SPARQL queries for RDF data manipulation but were not applied to the construction sector.

These strategies and methods can potentially reduce the implementation barriers of SWTs where professional knowledge engineers are highly limited. However, there is a lack of research on the systematic combination between SWTs and LLMs, which can particularly address the need for dynamic knowledge sharing and reuse within the construction industry. Specifically, the interactive mechanism among SWTs should first be systematically explored, including not only SPARQL query execution but also OWL-based reasoning and data validation. Then, the integration of LLM into these interactions can be developed and the optimisation of embedding LLMs can be measured in the case studies relating to construction knowledge sharing and reuse.

2.3 Research scope and design

This study employs a proof-of-concept methodology

to evaluate the feasibility of integrating SWTs and LLMs within a controlled yet representative environment. A case study is conducted on modular product design (MPD) in the context of off-site construction (OSC), a domain where MPD has gained traction as a strategic approach to balance product standardization with user-driven customization for effective manufacturing and assembly [16]. Despite its potential, MPD remains underdeveloped largely due to the complexities of knowledge reuse across diverse projects [17]. For instance, new project requirements must be embedded within an existing MPD knowledge model in a way that updates relevant design parameters while remaining compliant with predefined manufacturing and assembly rules. Furthermore, any adapted MPD outcomes must be validated against a set of specifications to ensure compliance and readiness for delivery.

It is important to emphasize that the primary objective of this paper is to assess the feasibility of the proposed framework for supporting knowledge sharing and reuse tasks, including customized rule development, project-specific knowledge adaptation, and constraint-based compliance checking. The focus is not on industrial implementation or deployment, but rather on validating the conceptual integration and methodological soundness of the proposed framework.

3 Proposed framework

In this proposed framework, the reasoning capacities of LLMs and SWTs are integrated to share and reuse knowledge with complex semantic structures in construction. Traditionally, project-specific requirements, such as contextual conditions, local regulations, and user preferences, are manually formed into SPARQL queries, SHACL constraints, and SWRL rules. These formal representations are then applied to a predefined ontology model comprising a terminology box (TBox), which defines domain concepts, relationships, and properties, and an assertion box (ABox), which contains project-specific data instances. To ensure these queries, constraints, and rules are correctly developed, close collaboration between domain experts and knowledge engineers is important. LLMs are integrated with SWTs to facilitate these traditional processes, bridging the gap of human-based collaboration.

This integration framework is designed to balance the rigid requirements of knowledge base operations with the inherently uncertain outputs of LLMs. A concept-focused retrieval strategy has been adopted to retrieve related sub-graph from the whole ontology. Relevant RDF triples are retrieved from both the TBox and ABox to provide a robust contextual foundation for LLM augmentation. These context-enhanced LLMs are then employed to generate SPARQL queries, SHACL constraints, and SWRL rules. Ultimately, the formalized outputs facilitate the updating and compliance checking of the existing ontological knowledge model and underlying database.

The coded SWT implementation processes are summarized in Table 1, and the corresponding graphical illustration is shown in Figure 1.

Table 1 Coded SWT implementation processes

Code	Description
(a)	Manually develop SPARQL queries based on project requirements
(b)	Manually develop SHACL constraints based on project requirements
(c)	Manually develop SWRL rules based on project requirements and axioms
(d)	Manually develop SHACL constraints referring to the TBox
(e)	Manually develop SWRL rules referring to the TBox
(f)	SHACL constraints are implemented through SPARQL queries at the back end
(g)	SPARQL queries directly apply to the ABox
(h)	ABox is the instantiation of TBox
(i)	SWRL rules directly apply to the ABox
(j)	Requirements embedding into LLMs
(k)	LLM-enhanced SPARQL queries generation
(m)	LLM-enhanced SHACL constraints generation
(n)	Global knowledge embedding
(p)	Case-specific knowledge embedding
(q)	SWRL rules generation

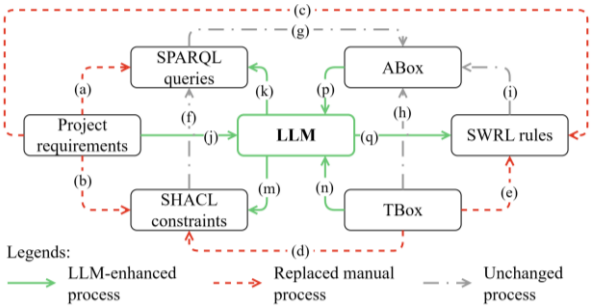

Figure 1. LLM-integrated process re-engineering for SWT implementation

According to the findings reported in [14] and our preliminary experiments [18], LLMs have been exhibiting an acceptable understanding of the basic syntax used in SWTs. However, it is important to acknowledge that the outputs generated by LLMs are typically unstructured, while automated SWT implementation demands outputs that are strictly structured and syntactically valid.

Due to concerns regarding cost and flexibility, this study does not explore fine-tuning or modifying the internal architecture of LLMs. Instead, it adopts few-shot prompting as a practical approach to guide LLMs in understanding SWT-specific tasks and adhering to the domain-specific syntactic constraints, which has demonstrated effectiveness in aligning LLM outputs with the structured generation requirements [19]. When dealing with ambiguous natural language from different project stakeholders, these inputs are firstly aligned with the classes, object properties and data properties predefined in the TBox to avoid LLM hallucinations.

4 Implementation and evaluation

To evaluate the proposed framework, a case study was conducted focusing on modular connection design in the context of off-site construction. In this case, LLMs were integrated to support the automated generation of SPARQL queries, SWRL rules, and SHACL constraints. These outputs were used to adapt an existing ontology model, enabling effective knowledge reuse across varying project conditions.

4.1 Case study overview

Three modular connection design options exhibiting superior performance in terms of manufacturability and ease of assembly were selected for evaluation [20]. Figure 2 illustrates the primary configurations of the selected designs, including the assembly of columns and beams using plug-in devices, plates, and bolt sets.

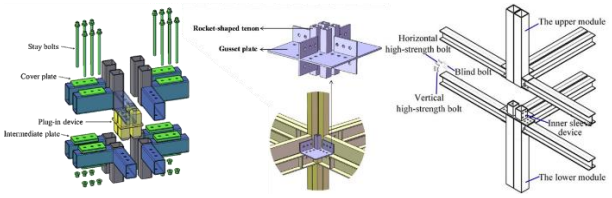

Figure 2. Selected connection design options

The experimental pipeline consists of four main steps: (1) developing a foundational ontology based on selected case studies, (2) formulating project requirement entries (PREs) in natural language, (3) generating ontology adaptation code using the enhanced LLMs, and (4) testing and evaluating the generated code within a developed prototype system. PREs were formulated using natural language based on the developed ontology and design scenarios. As an initial test, a set of exemplary PREs presented in Table 2 was used to adapt the existing ontology and RDF database to dynamic project-specific contexts. The main scenarios include: (1) developing globally reusable rules that can be applied for automatic case adaptation using SWRL (PRE1-3), (2) finding a reusable case and modifying it with project-specific facts using SPARQL (PRE4-6), and (3) specifying the information delivery requirements for real-time compliance checking using SHACL (PRE7-10).

Table 2 Exemplary PREs for test

No.	Project requirement entries (PREs)
1	If an instance of a plugin device is assembled with an instance of a column, the tenon width of the plugin device should be column width-2*column thickness-5mm.
2	If an instance of a plugin device is assembled with an instance of a column, the tenon height of the plugin device should be 1.5 times (column width) ^2
3	If an intermediate plate is bolted with some beams by some bolts, then the minimum width of the intermediate plate is the width of the beam + 3*diameter of the bolts.
4	Find the instances of option1 connecting 8 columns.
5	Add 2 instances of column type1 and 4 instances of column type2
6	Let the width of intermediate plate type1_1 be 200 mm, and let the width of intermediate plate type1_2 be 150 mm.
7	The tenon height of the plugin device must be larger than 200 mm.
8	The width of the intermediate plate must be larger than its minimum width.
9	Exactly one plugin device is required in each connection design option.

The high-level structure of the foundational ontology is depicted in Figure 3. High-level structure of the ontology. It is developed based on the latest Industry Foundation Classes (IFC) schema [21]. Each connection design is composed of multiple components, such as plates and bolts, which are assembled and connected to other structural elements, including columns and beams.

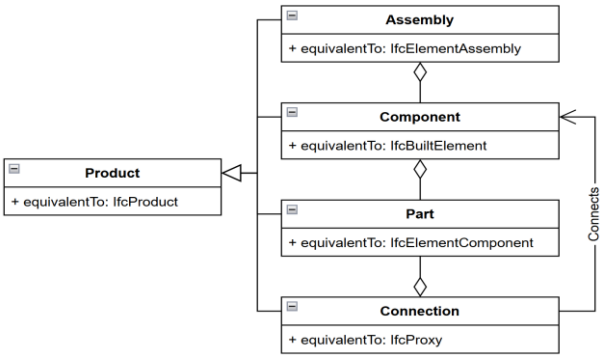

Figure 3. High-level structure of the ontology

4.2 Experiment illustration

By way of illustration, globally applicable rules can first be defined. For example, one of them can be: "*if an intermediate plate is bolted to one or more beams using a set of bolts, then the minimum width of the intermediate plate equals the beam width plus three times the bolt diameter*" (PRE3). Subsequently, a designer may specify the actual plate width according to project requirements; for example, "*the width of the intermediate plate type 1 is set to 200 mm and 150 mm*" (PRE6). Finally, the asserted design parameters are verified against the constraint that "*the actual width of the intermediate plate must be larger than or equal to its minimum required width*" (PRE8).

The implementation of this workflow can be augmented by LLMs, which translate natural-language specifications into corresponding executable code to enable automated knowledge adaptation driven by an ontology-based reasoning engine.

4.2.1 SWRL-based reasoning

Firstly, LLMs were utilized to generate SWRL rules that can be applied to all instances in the ABox. The example of PRE3 establishing the universally applicable relationship between assembled intermediate plates, beams and bolts is shown in Figure 4. Similarly, PRE1 and PRE2 were also transformed into SWRL rules using enhanced LLMs.

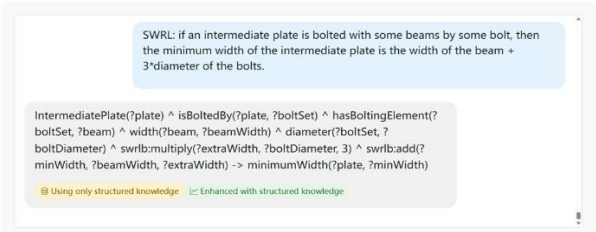

Figure 4. LLM-generated SWRL for PRE3

The outputs were then embedded in the OWL-based reasoning process to dynamically generate the inferred parameters. In Figure 5, for instance, the minimum width of an intermediate plate was inferred by the embedded SWRL rule based on the existing factual data of the associated beam and bolt set.

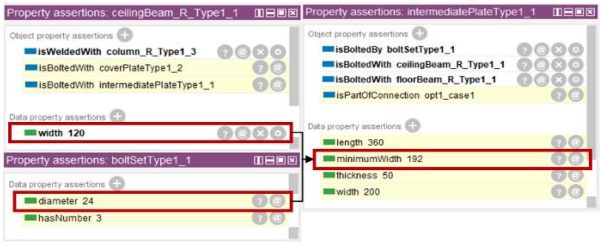

Figure 5. Property inference using SWRL

4.2.2 Fact assertion using SPARQL

In real-world construction projects, designers often assign specific property values to reflect varying contextual conditions and project-specific requirements, such as structural performance criteria, material preferences, or regulatory constraints. These property values must be accurately represented within the knowledge model to support effective decision-making and automation. Figure 6 shows an example of generating a SPARQL query of PRE6 using LLMs. Then, this LLM-generated SPARQL query was checked and processed successfully by the RDF service module shown in Figure 7. Factual property values, 200 mm and 150 mm, of width have been assigned to the instances of intermediate plate type 1. These inserted project facts will be checked by SHACL constraints defined by clients, project managers and regulatory documents such as standards and local policies.

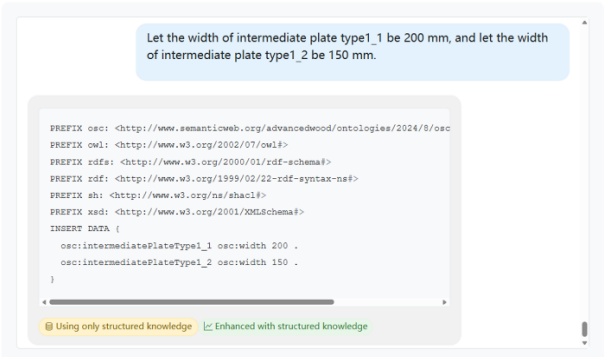

Figure 6. LLM-generated SPARQL for PRE6

Figure 7. Fact assertion using SPARQL

4.2.3 Compliance checking with SHACL

Structured SHACL constraints can also be generated by the augmented LLMs. Figure 8 presents the LLM-generated SHACL constraints used to verify the compliance of width for the intermediate plate assembled in a modular connection design (PRE8). Based on the inferred minimum width of the intermediate plate in Section 4.2.1 and two of their asserted factual parameters in Section 4.2.2, the prototype system can conduct a compliance checking process to validate the design

outcomes.

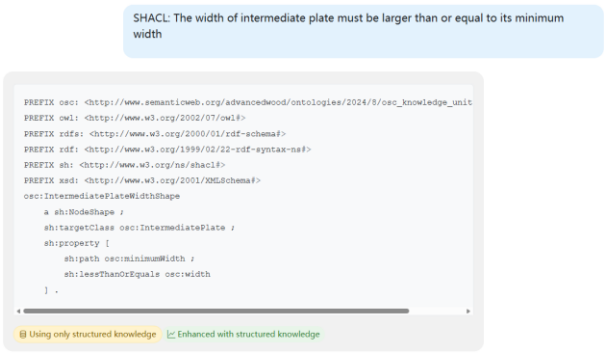

Figure 8. LLM-generated SHACL for PRE8

As shown in Figure 9, since the inferred minimum width of the intermediate plate is 192 mm, the validations for intermediate plate type 1_1 (with a width of 200 mm) and type 1_2 (with a width of 150 mm) passed and failed, respectively.

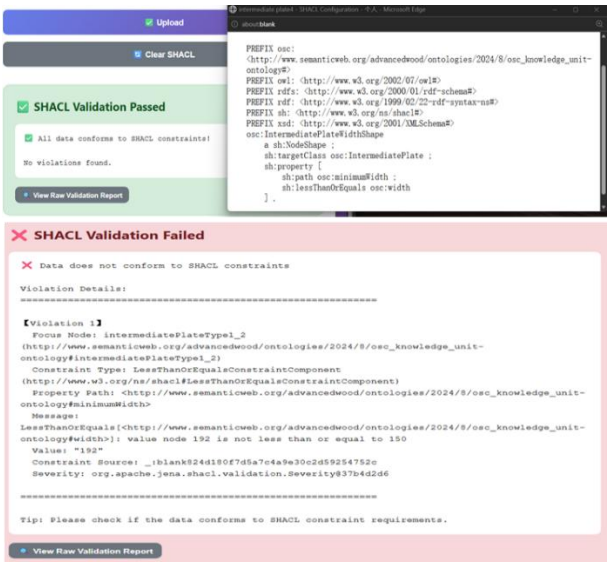

Figure 9. Compliance checking with SHACL

4.3 Evaluation and results

To initially evaluate the reliability and robustness of the proposed framework and system with SWTs and LLMs, the 9 exemplary PREs were extended to 30 PREs with additional four paraphrases for each of them, getting 150 entries. Then, each entry, for example, a paraphrase of PRE8: "*The intermediate plate width must exceed its minimum allowable value*", was tested by three stable and publicly available LLMs using standard RAG and our proposed framework, respectively. Finally, 900 rounds of tests were conducted using the developed prototype system. The dataset covers seven types of PREs to be evaluated for MPD knowledge sharing and reuse, including (1) individual assertion, (2) attribute and relation assertion, (3) attribute calculation, (4) logic inference of attributes and relations, (5) individual quantity assertion and regulation, (6) data value and range constraints, and (7) data value enumeration.

The three LLMs employed in this study include: GPT-4o, GPT-4o-mini, and DeepSeek-V3.2. GPT-4o represents one of the most advanced and stable commercially available LLMs, while GPT-4o-mini serves as its lightweight variant. Comparing these two models enables evaluation of whether the proposed framework requires the capabilities of a more powerful LLM. DeepSeek-V3.2, a state-of-the-art open-source model with performance comparable to GPT-4o, is included to assess the feasibility of local deployment, particularly in scenarios where data privacy is a concern.

In LLM-enhanced SWTs implementation, the core capability of LLMs is their ability to transform natural language text into executable code in SWRL, SPARQL, and SHACL syntaxes, based on a consistent semantic context. Accordingly, this study evaluates LLM performance across three tasks: (1) text-to-SWRL (T2SW), (2) text-to-SPARQL (T2SP), and (3) text-to-SHACL (T2SH).

The experimental results are derived from a combination of answer parse (AP) validation and the standard F1 score [14]. The AP metric assesses the syntactic correctness of the generated code, specifically, whether it can be successfully parsed and executed by the knowledge system. This is a binary measure, with a value of 1 indicating successful parsing and 0 indicating failure. While syntactic correctness is essential for execution, it does not ensure semantic validity, only semantically correct queries yield meaningful and relevant results.

Semantic accuracy is measured using the standard F1 score, which is calculated based on precision and recall by comparing the generated code to a ground truth reference. The ground truth consists of code that faithfully represents the intended meaning of the input and includes all required data elements. The formula for computing the F1 score is provided in Equation (1).

$$F1 = \frac{2 \times precision \times recall}{precision + recall} \quad (1)$$

To capture both syntactic and semantic dimensions of performance, a *combined score* is calculated as defined in Equation (2), assigning a weight of 20% to syntactic correctness (AP) and 80% to semantic accuracy (F1 score). This weighting scheme reflects the practical importance of generating code that is not only executable but also semantically faithful to the input [14].

$$Combined = 0.2AP + 0.8F1 \quad (2)$$

Table 3 presents the initial evaluation results of LLM-

enhanced SWT implementation tasks, assessing the performance of the selected LLMs across the three key transformation tasks. The average scores (Ave.) are also computed to provide a holistic measure of each model's effectiveness.

Table 3 Results of LLM-enhanced SWT implementation

LLM	T2SW	T2SP	T2SH	Ave.
GPT-4o	**0.96**	0.66	0.90	0.84
/Standard RAG	0.34	0.22	0.76	0.44
GPT-4o-mini	0.38	0.37	0.60	0.45
/Standard RAG	0.35	0.19	0.27	0.27
DeepSeek-V3.2	0.84	**0.96**	**0.93**	**0.91**
/Standard RAG	0.39	0.25	0.81	0.48

According to the results, the proposed approach demonstrated consistent high scores across the three SWT implementation tasks and improved performance compared with standard RAG. Specifically, GPT-4o achieved strong performance in T2SP (0.96) and T2SH (0.90), yielding an overall average of 0.84. This result reflects its superior ability to generate both syntactically correct and semantically accurate executable code. However, GPT-4o-mini, the lightweight variant, demonstrated moderate performance, with scores ranging from 0.38 (T2SW) to 0.60 (T2SH), and an overall average of 0.45. This indicates a noticeable trade-off in performance for efficiency and model size. DeepSeek-V3.2 lagged behind GPT-4o in T2SW (scoring 0.84) and exceeded GPT-4o in T2SH (0.93) and T2SP (0.96), leading to the highest average score of 0.91.

Additionally, each PRE was transformed into semantic web artifacts manually and with help of the LLM-enhanced knowledge engineering system, respectively. In average, it costs around 3 minutes to manually develop an executable semantic web artifact from scratch. In contrast, the average time consuming with the proposed system for 900 rounds of execution is 5.51 seconds, which significantly improves the time efficiency of ontology-based knowledge modelling for MPD knowledge sharing and reuse.

5 Discussion

The previous research used LLMs directly to generate ontological data [9,12]. This process, however, was treated largely as a black box, where the internal reasoning and logic of the model could not be easily traced or validated. The primary contribution of this work is a feasibility demonstration of an ontology-grounded LLM-SWT integration framework. This framework breaks the barriers of data interoperability of traditional BIM environments and rule customization flexibility of rule-based checking methods by managing all semantic data in one consistent and software neutral framework and establishing links among different data sources from multiple project stakeholders. We show, through a controlled case study, that LLMs can reliably generate semantically valid SWT code, thereby reducing the dependency on specialized knowledge engineers and providing a template for scalable, white-box knowledge automation in construction.

More specifically, this framework restricts LLM outputs to explicitly structured artifacts that conform to ontological definitions, enabling the outputs to be parsed, executed, and validated using standard SWTs. It ensures that LLMs act not as autonomous knowledge generators, but as ontology-aware code synthesizers. As a result, every step of the knowledge transformation pipeline, including rule-based reasoning, fact assertion, and compliance checking, can be reproduced, audited, and debugged. This level of transparency enhances the reliability and maintainability of the system, especially in dynamic project contexts where consistent adaptation and validation of knowledge is essential.

Some typical syntactical errors that always occur in LLM generation are addressed through few-shot prompting, minimizing semantic hallucinations in the outputs. For example, the system prompt enforces ontology-aligned variable naming and restricts predicate usage to those explicitly defined in the TBox. These safeguards significantly reduce ambiguity and prevent invalid or misleading rules from being generated. Consequently, the SWT–LLM integration framework bridges the gap between the flexibility of LLMs and the formal rigor of semantic web reasoning, enabling scalable, automated, and verifiable knowledge engineering for dynamic contexts in construction.

6 Conclusion and future work

This study presented and validated a proof-of-concept framework for integrating LLMs with SWTs to automate knowledge sharing and reuse tasks in construction. The results demonstrate the technical feasibility of using advanced LLMs (GPT-4o, DeepSeek-V3.2) as semantic code synthesisers within an ontology-driven environment.

The study was limited to a narrow domain and a small set of requirement patterns. Evaluation focused on code correctness rather than real-world engineering outcomes. The method also relies on well-structured ontologies and prompt templates, which may reduce flexibility in diverse settings.

Future research will expand validation across construction domains, support dynamic product configuration, and improve adaptability to changing project contexts. Efforts will also focus on automating rule creation, enhancing ontology maintenance, and incorporating validation safeguards for practical deployment.

References

[1] G.L. Rocca, Knowledge based engineering: Between AI and CAD. Review of a language based technology to support engineering design. *Advanced Engineering Informatics*, 26 (2):159–179, 2012.

[2] B.P. Allen, L. Stork, P. Groth, Knowledge Engineering using Large Language Models. *arXiv preprint*, arXiv:2310.00637 (2023).

[3] M. Li, Z. Wang, BuildingGPT: Query building semantic data using large language models and vector-graph retrieval-augmented generation. *Building and Environment*, 287:113855, 2025.

[4] J. Montali, M. Overend, P.M. Pelken, M. Sauchelli, Knowledge-Based Engineering in the design for manufacture of prefabricated façades: current gaps and future trends. Architectural Engineering and Design Management, 14 (1-2):78–94, 2018.

[5] M. Sandberg. Knowledge-based engineering in construction: The prefabricated timber housing case. *Journal of Information Technology in Construction*, 13:408–420, 2008.

[6] J. Pokojski, K. Szustakiewicz, Ł. Woźnicki, K. Oleksiński, and J. Pruszyński. Industrial application of knowledge-based engineering in commercial CAD/CAE systems. *Journal of Industrial Information Integration*, 25:100255, 2022.

[7] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, and X. Wu. Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering*, 36(7):3580–3599, 2024.

[8] L.-P. Meyer, C. Stadler, J. Frey, N. Radtke, K. Junghanns, R. Meissner, G. Dziwis, K. Bulert, and M. Martin. LLM-assisted knowledge graph engineering: Experiments with ChatGPT. In *Proceedings of the First Working Conference on Artificial Intelligence Development for a Resilient and Sustainable Tomorrow*, pages 103–115, Wiesbaden, Germany, 2024.

[9] B. Zhang, V. A. Carriero, K. Schreiberhuber, S. Tsaneva, L. S. González, J. Kim, and J. de Berardinis. OntoChat: A framework for conversational ontology engineering using language models. In *Proceedings of the ESWC 2024 Satellite Events*, pages 102–121, Cham, Switzerland, 2025. Springer Nature.

[10] D. Durmus. Potential of large language models for ontology development in construction domain. Master's thesis, Technical University of Munich, 2024.

[11] S. Yoon, J. Song, and J. Li. Ontology-enabled AI agent-driven intelligent digital twins for building operations and maintenance. *Journal of Building Engineering*, 108:110686, 2025.

[12] V. K. Kommineni, B. König-Ries, and S. Samuel. From human experts to machines: An LLM supported approach to ontology and knowledge graph construction. *arXiv preprint*, arXiv:2403.08345, 2024.

[13] M. Mountantonakis and Y. Tzitzikas. Generating SPARQL queries over CIDOC-CRM using a two-stage ontology path patterns method in LLM prompts. *Journal on Computing and Cultural Heritage*, 18(1), 2025.

[14] L.-P. Meyer, J. Frey, F. Brei, and N. Arndt. Assessing SPARQL capabilities of large language models. *arXiv preprint*, arXiv:2501.05925, 2025.

[15] M. Hornsteiner, M. Kreussel, C. Steindl, F. Ebner, P. Empl, and S. Schönig. Real-time text-to-Cypher query generation with large language models for graph databases. *Future Internet*, 16(2):53, 2024.

[16] I. Kim, J. Shin, S. H. Shah, and S. U. Rehman. Client-centered detached modular housing: Natural language processing-enabled design recommender system. *Journal of Computational Design and Engineering*, 11(1):137–157, 2024.

[17] J. Cao, D. F. Bucher, D. M. Hall, and J. Lessing. Cross-phase product configurator for modular buildings using kit-of-parts. *Automation in Construction*, 123:103437, 2021.

[18] Z. Zhang, Y. Zou, B. H. Guo, L. Jiang, J. Dimyadi, and R. Davies. Enhancing knowledge capture and reuse for offsite construction through a container-based knowledge approach. In *Proceedings of the Sixth International Conference on Civil and Building Engineering Informatics*, pages 530–542, 2025.

[19] I.-V. Hernandez-Camero, E. Garcia-Lopez, A. Garcia-Cabot, and S. Caro-Alvaro. Context-aware few-shot learning SPARQL query generation from natural language on an aviation knowledge graph. *Machine Learning and Knowledge Extraction*, 7(1), 2025.

[20] X. Liu, C. Hou, J. Peng, Y. Wang, and K. J. R. Rasmussen. Recent advancements of inter-module connections for modular buildings: An overview and multi-dimensional assessment. *Engineering Structures*, 335:120378, 2025.

[21] International Organization for Standardization. Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries—Part 1: Data schema (ISO 16739-1). ISO, Geneva, Switzerland, 2024.