2025 European Conference on Computing in Construction

CIB W78 Conference on IT in Construction

Porto, Portugal
July 14-17, 2025

TEXT2STRUCTURES: COMMUNICATIVE AGENTS FOR EARLY STAGE STRUCTURAL DESIGN

Edward de Groot[1], Ranjith K. Soman[1], and Daniel M. Hall[1]
[1]Delft University of Technology, Delft, Netherlands

Abstract

Communicative agents powered by Large Language Models have potential to transform structural engineering design. This study introduces Text2Structures, a multi-agent system for early conceptual design. Using Retrieval Augmented Generation, agents generate a set of structural design recommendations. The demonstration of a simple parking structure design verifies that agents can recommend materials, structural systems, and dimensions for floors and beams. Using Text2Structures, users can input brief requirements in natural language, integrate regulatory and prescriptive documents from local jurisdictions, and receive transparent and justified structural recommendations. Although further refinements are needed, this study lays a foundation for communicative agents in structural design.

Introduction

In structural engineering, advanced design and calculation software has long been used by engineers. As the recent development of artificial intelligence (AI) continues, the primary concern for structural engineers is the lack of clarity and transparency in AI's decision-making processes. Explainability in engineering is critical because it facilitates communication with project stakeholders and establishes accountability, mainly when errors occur. Traditional AI tools often fail in this regard, making it difficult to understand how specific design choices are made. In addition, engineers have limited control over AI system inputs, such as local building codes or relevant construction examples. Most software lacks the flexibility to prioritize specific documents, which can hinder outputs that are accurate for local jurisdictions and contexts. Finally, existing tools require specialized expertise to operate effectively, creating a steep learning curve and reducing usability for engineers without extensive technical knowledge. This disconnect between the complexity of AI software and the practical needs of engineers highlights a significant gap in current solutions.

A subset of AI is communicative agents, which are particularly interesting because they can communicate in natural language with humans. These agents operate under a coordination workflow that manages their inputs, outputs, and data flows, enabling them to work autonomously with tools like internet access and financial resources without requiring constant human supervision (Qian et al., 2024). Communicative agents are part of recent advancements in AI that have brought new possibilities, mainly through Large Language Models (LLMs) like ChatGPT, which have significantly intensified interest in AI applications (Maslej et al., 2023).

Communicative agents are designed to interact with each other using natural language. The resulting multi-agent system resembles a collaborative team with individual agents making unique contributions, further enhancing their ability to solve complex problems (Qian et al., 2023). Communicative agents can simulate human behavior and can write, review, and execute code. Using customized agent workflows, it is possible to develop frameworks for the automatic execution of programming tasks (Hong et al., 2023; Chen et al., 2023; Zhou et al., 2023; Qian et al., 2023; Wu et al., 2023; Li et al., 2023).

However, there are significant limitations in the application of communicative agents in structural engineering. The field demands a comprehensive approach to managing complex and sometimes conflicting structural design requirements, including ensuring structural integrity, safety, cost-effectiveness, aesthetic appeal, and sustainability. One key consideration of structural engineering is the authority of the building code and other design guidance tools. Therefore, there is an opportunity to use an approach such as Retrieval-Augmented Generation (RAG), which can allow agents to refer to authoritative knowledge bases outside of their LLM training data.

Recent promising studies have been conducted on the application of communicative agents in the built environment. For example, I-design (Çelen et al., 2024) applies communicative agents for the process of interior design. Text2BIM (Du et al., 2024) looks at communicative agents for BIM design; however, the system does not explain why certain design choices are made and cannot be used for structural decisions. In general, there has been little exploration that considers the specific needs of structural engineering workflows and best practices. Therefore, this paper addresses the following question:

How can a communicative multi-agent system be designed and implemented for the purposes of structural

engineering?

To address this question, the research focuses on two key aspects. First, the paper examines how individual agents can be developed to perform specialized tasks in structural engineering. Second, it explores the necessary steps for integration of LLM and RAG into a complete multi-agent system; in other words, the framework needed to "bring the team together".

This paper is structured as follows. Section 2 provides background information on multi-agent systems, communicative agents, their existing applications, and the structural design process. Section 3 details the design and development of the multi-agent workflow, Text2Structures, in the context of structural engineering. Section 4 demonstrates the multi-agent workflow by executing the initial stages of the conceptual design process for a multi-level parking structure. Section 5 discusses the results, addresses research limitations, and identifies potential directions for future studies. Finally, Section 6 concludes the paper by summarizing key contributions and highlighting the importance for future advancements in the field.

Background

Multi-agent Systems

A Multi-agent system (MAS) can decompose a large-scale problem by several agents cooperating to reach a particular objective. Such teams of agents are useful when problems are too significant for a single agent to solve (Khodabandelu and Park, 2021).A heterogeneous team of agents can achieve better results than a single or homogeneous team (Khodabandelu and Park, 2021; Marcolino et al., 2013). This is because diverse agents are more effective than uniform copies of the most powerful agents. At the same time, system designers of MAS must understand the often complex interdependence between members of a heterogeneous team. (Hsu et al., 2016)

The design of MAS using communicative agents will need to consider the following characteristics of MAS:

- *Environment:* Agents operate within a specific environment. In a physical environment, agents perform tangible tasks, such as sweeping the floor. A sandbox environment is a virtual simulated world where agents can experiment with actions. In other cases, no environment is set, and the focus shifts to the communication between agents (Guo et al., 2024).

- *Hierarchy and Decision Making:* Typically, some form of hierarchical structure is present for decision making, meaning that agents are organized into levels with some agents having authority over others. For example, a leader agent can guides or plan tasks while follower agents then execute instructions based on the leader's guidance (Han et al., 2024). Agents can be directed only to communicate in their layer or the layer above, or centralized with only communication to the leader. Such structures are common in scenarios requiring coordinated efforts directed by a central authority (Guo et al., 2024). In equi-level structures, agents operate at the same hierarchical level and collaborate without centralized leadership; it can, therefore, also be seen as a decentralized structure. These agents can have the same, neutral, or opposing objectives and negotiate or collaborate to achieve their goals. This structure emphasizes collective decision-making and shared responsibilities among agents (Guo et al., 2024; Han et al., 2024). Nested structures combine elements of both hierarchical and equi-level systems, allowing for sub-structures within the central system. These can handle complex tasks by breaking them down into smaller tasks managed by sub-groups of agents (Han et al., 2024)

- *Memory:* Consideration of memory management in MAS is needed to maintain contextual coherence and learning from past interactions. Agents use memory to store and retrieve information about their environment, past actions, and interactions with other agents. This information helps agents adapt strategies, improve decision-making, and enhance system performance. Memory can be managed through in-context learning, where agents retain short-term information relevant to the current task, or through external databases, where long-term information is stored for future reference (Guo et al., 2024; Han et al., 2024).

- *Planning:* Planning in MAS involves the development of strategies and action sequences that the agents will follow to achieve their goals. Planning can be either global, which concerns the overall tasks, or local, which is about a sub-task (Han et al., 2024).

Communicative Agents

Communicative agents are a subset of MAS. One key differentiator is that communicative agents have the ability to use natural language in order to exchange information for coordinating actions, making decisions, and solving problems. Thus, most recent studies on communicative agents use LLMs (Maslej et al., 2023).

The most common application of communicative agents is in the field of software development. Researchers use a sandbox environment and assign different software development roles to agents. In this way, agents can optimize the design process (Qian et al., 2023). A different domain is embodied agents, where LLM-based MAS is used for communication issues involving multiple robots (Chen et al., 2023). Given the advanced communication capabilities, LLM-MAS is also used for world simulation, societal simulation, gaming, and the economy. The agents role play by having diverse roles and viewpoints. In a societal simulation, agents demonstrate human-like behavior and interaction (Park et al., 2023).

In built environment research, the *I-design* system is based upon communicative agents (Çelen et al., 2024). The approach is for interior design that allows users to generate and visualize their design using natural language. In the framework, I-design uses five distinct agents with predefined roles for generating 3D indoor scenes from user input: interior designers, interior architect, engineering agent, layout corrector, and layout refiner. The first agent is the interior designer, who proposes a selection of objects based on the user input. The interior architect uses the objects selected by the designer to determine a spatial arrangement in the room. After this step, the engineering agent creates a valid graph, ensuring structural validity. The layout corrector corrects invalid connections in the scene graph. The last agent is the layout refiner, who refines the relationship between objects, enhancing the accuracy of the layout. I-design operates with a "Divide & Conquer" organizational structure to manage complexity and improve performance. Each agent focuses on a specific sub-task, using the outputs of the previous agents. This method includes a feedback loop to eliminate unwanted behaviors and address hallucination issues. The agents use the retrieval of objects to develop a design. Communication is through JSON objects facilitating a structured data exchange. This ensures that each agent's contribution builds on the previous ones, maintaining a coherent and integrated development. I-design uses a framework named AutoGen(Microsoft AutoGen Team, 2023) for the agent communication, with each agent using the GPT-4. I-design is a successful demonstration of communicative agents that can be applied to design tasks within the built environment.

Structural engineering design process

Before automation, it is essential to understand the nature of work for engineering professionals. During preliminary design, structural engineers focus on the overall structural system, not individual components or elements. The goal is to develop a global structural design that meets the project brief (IstructE, 2022). A common practice in this phase is to design multiple variants, of which one is to be further elaborated (Crielaard and Terwel, 2020).

The early stage structural design process has been described in a preliminary design flow chart created by Crielaard and Terwel (2020) This flowchart identifies ten steps to perform in structural design, of which the first four are relevant to the preliminary design: 1) problem analysis, 2) schematic of the load-bearing system, 3) material choice, and 4) estimation of profile dimensions.

There are two key points where a decision has to be made: the selection of the load bearing system and the choice of material. Both are dependent on the problem analysis, but also on each other. The remaining steps are action points, referring to calculations, starting with dimensioning the profile dimensions and ending with a unity check, indicating whether the structural design is OK. In this flowchart, an iteration is possible when the structural engineer notices an insufficient unity check. For example, a different profile dimension can be chosen. In addition to the pure structural aspects, relevant evaluation criteria such as cost and environmental effects can also be considered. In this way, alternatives can be compared to make an informed decision (Crielaard and Terwel, 2020).

Text2Structures

The multi-agent workflow for Text2Structures consists of four steps:

1. Selecting a programming framework
2. Developing individual agents
3. Integrating a LLM and Retrieval-Augmented Generation RAG
4. Designing the multi-agent workflow

The technical framework (Figure 1) provides an overview, including user-agent, agent-agent, and agent-external documentation interactions.

Programming framework: Developing a multi-agent system requires a suitable programming framework that supports defining agent roles, which aligns well with the structured steps of conceptual design. Several possible programming frameworks were inspected, including Chatdev and Meta-GPT. All reviewed frameworks maintained conversation records, and most offered documentation.One key criteria was the ability to support human interaction. From these criteria, the programming framework AutoGen was selected because it was the most accessible (requiring no account) and offered the most comprehensive documentation for building tailored prototypes.
Agents: Agents are assigned specialized tasks to maintain clarity and control. The workflow consists of eight agents: The Lead Engineer oversees structural agents and assigns tasks, executing predefined functions, while the Assistant Agent initiates workflows but does not use an LLM. The Planner Agent communicates with the user to determine boundary conditions, and the Client (User Proxy) directs interactions, requiring human input mode set to ALWAYS. The Material Agent selects construction materials based on a function call utilizing RAG, while the Load-Bearing Agent determines the structural layout based on the Material Agent's decisions. The Dimensioning Agent specifies spans and depth ratios for floors and beams, and the Writer Agent summarizes conversations between structural agents. A sequence of nested chats is employed, inspired by AutoGen's example Microsoft AutoGen Team (2024). The Lead Engineer handles inner monologues with structural agents, while the Assistant Agent initiates and manages workflow scalability.
LLM and RAG : A locally run server was chosen for ease of setup and function calling capabilities, utilizing LiteLLM and Ollama for LLM inference Ollama Team (2024). The model used is Llama3.1 (8B parameters,

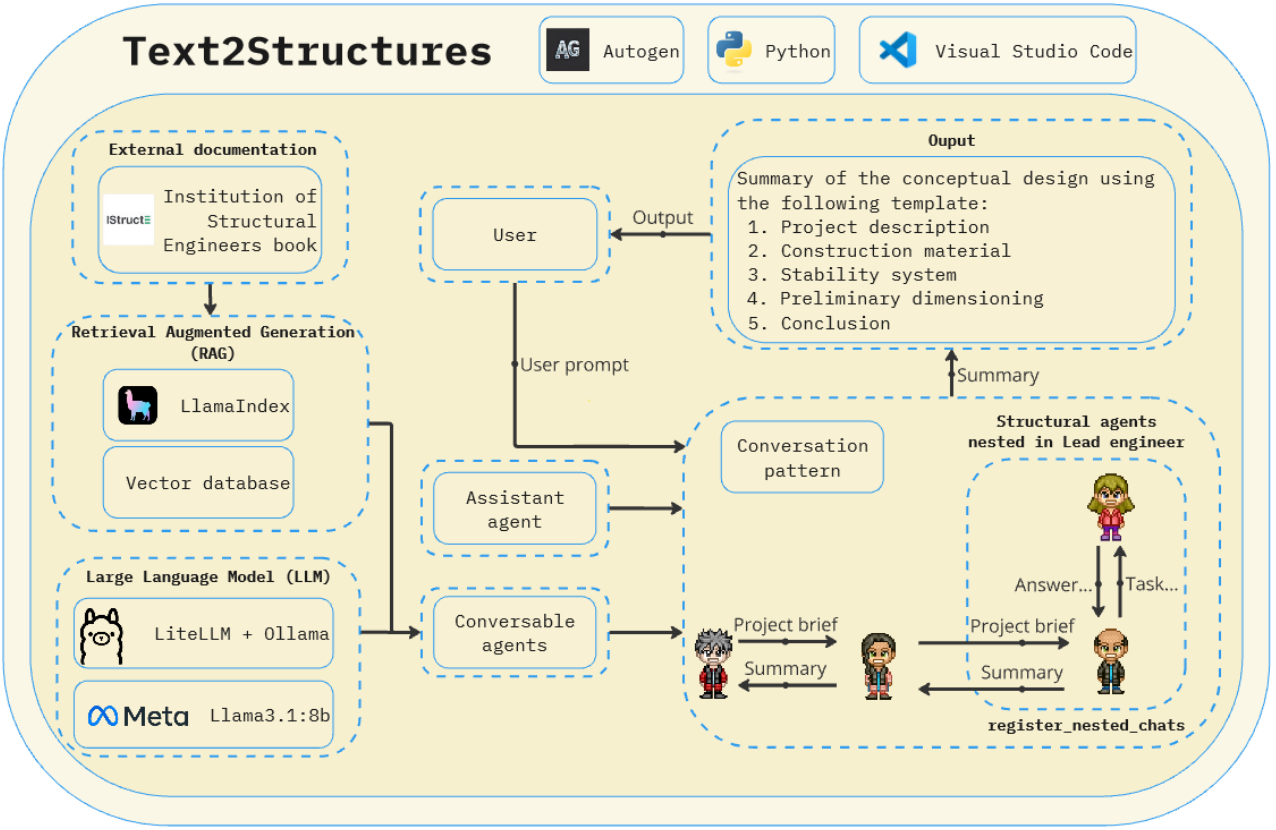

Figure 1: Text2Structures Technical framework

Table 1: Overview of Multi-Agent Frameworks

Framework	Role Definition	Human Interaction	Conversation Record	GitHub Repository
Chatdev	Predefined	+	+	+
AutoGen	Predefined	+	+	+
Meta-GPT	Predefined	+	+	+
Generative Agents	Predefined	-	+	-
Multi-Agent Collaboration	Predefined	-	+	-
Auto Agents	Dynamic	-	+	+
Camel	Predefined	-	+	+
Agents	Predefined	+	+	+

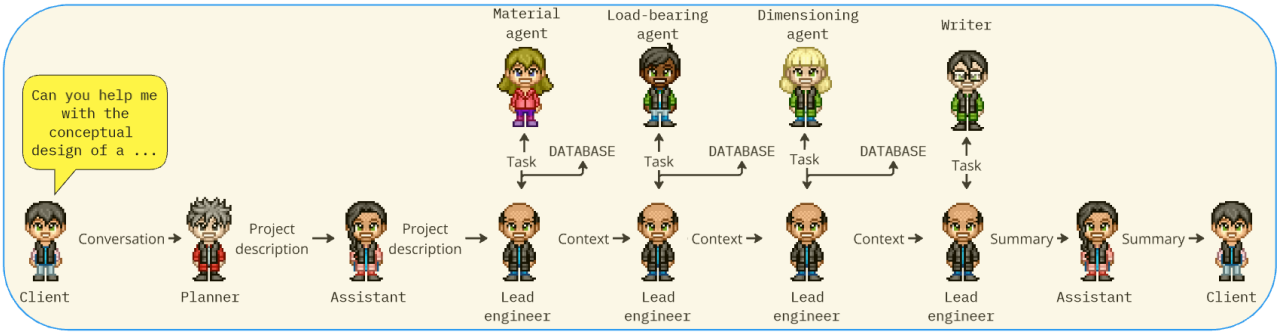

Figure 2: Multi Agent Workflow

4.7GB) Ollama Team (2024), Gemma, and Mistral-Nemo running locally with a temperature setting of 1 to balance creativity and predictability. RAG was implemented to enhance agent responses by providing access to external documents via LlamaIndex and Llamaparse for structured/unstructured data processing. In AutoGen RAG Integration, predefined functions were registered with agents Microsoft AutoGen Team (2023), allowing the Lead Engineer to execute function calls for the Material Agent. Structural agents share a common database, except for the Load-Bearing Agent, which has additional access to stability-related chapters.

Multiagent workflow: Text2Structures employs a customized chat pattern, utilizing a sequence of nested chats inspired by AutoGen. This approach enables the lead engineer to engage in an inner monologue with structural agents, leveraging other agents for problem-solving. Implemented through the 'register nested chats' function, the chat sequence is initiated by the assistant agent, while the lead engineer manages the 'chat queue', directing conversations and settings. The inner monologue, limited to two turns, ensures efficient execution and response generation, with the final summary being the last message rather than a reflection to prevent information loss. A structured process guides agent interactions: the client initiates a prompt, and the planner agent refines project boundaries before handing the brief to the assistant agent, who forwards it to the lead engineer. The lead engineer then sequentially assigns tasks to material, load-bearing, and dimensioning agents, each building upon prior responses . These agents follow a predefined workflow based on IstructE guidelines and database references. The writer agent compiles insights into a structured summary, which is relayed back through the lead engineer, assistant agent, and finally to the user. This structured approach ensures scalability, with the assistant agent orchestrating workflows and enabling additional nested interactions as needed (see Figure 2).

Demonstration

Text2Structures was used to complete the conceptual design of a parking garage. Agents were tasked to assist a structural engineer by proposing a design option that includes the primary construction material, load-bearing system, and preliminary dimensions. The purpose is to give the engineer a starting point, which can then be further refined and calculated by the professional.

The brief was to design a two-story parking garage (20–40 meters) with a 50-year lifespan. The foundation design was excluded, and there is an assumption that the site has no impact from adjacent buildings. The design prompt was phrased in three different ways for different runs (see Table 2), to note differences in the outcomes.

The base user prompt was slightly adjusted and given to the agents for three different runs, as seen in Table 2. In addition, three different LLMs are used to execute the multiagent workflow: Llama 3.1, Gemma, and Mistral-Nemo. These models were selected primarily because they are available through Ollama, which was a requirement for running the system locally on a standard laptop. Among the models hosted on Ollama, these three were among the most frequently pulled, suggesting strong community preference and reliable performance. Additionally, they support function calling and are available in sizes that can run on a basic laptop. The results show how each of the three LLMs performed, including the key choices in the conceptual design made by the agents, which can be seen in Table 3.

The results and process were then evaluated to assess the agents' structural decision-making, focusing on realism and relevance.The results show differences in how each model responds to the prompts. Surprisingly, the Gemma 2 model behaved differently than expected. While Gemma 2 provides reasoning, it noted that it could not determine the availability and cost of materials, recommending that an expert make the final decision. Additionally, it pointed out that building services would impact decisions, which, although true, was intentionally excluded from the scope of this research. The Llama and Mistral-Nemo models performed similarly, though Llama preferred hybrid structures when working with steel. Overall, all the models had difficulty suggesting appropriate grid dimensions for the specified prompt. When answering, they referred to an example in the literature. However, this was not directly suitable for the given situation. The slight variation in the prompt did not indicate such different responses.

In two test runs, the material agent proposed concrete once and steel once. Both could work, given the broad project brief. However, the agent's reasoning sometimes misstated facts. For example, it claimed concrete inherently resists harsh environmental exposure without clarifying design details. The load-bearing agent suggested two-way spanning floors in one run but forgot about load transfer. In the other run, it considered stability systems more carefully, resulting in a hybrid steel concept.

The dimensioning agent offered rules of thumb for grid layouts or standard beam sizes, but rarely followed through with basic calculations. It often repeated guidelines without applying them to the stated geometry. This can still be helpful as a starting point, but it is not a full engineering design.

Discussion

Overall, some runs demonstrated agents could successfully provide high-level structural concepts. One observation is that while human engineers typically sketch early ideas and iterate quickly (a mode of working that differs from a rigid step-by-step sequence), these agents worked in a sequential pattern. Agents seldom revisited or refined earlier choices.

From a technical standpoint, we succeeded in using multiple agents, enabling them to read documentation via RAG tools, and running everything locally on a mid-range laptop. Some tasks—like memorizing prior decisions—remained only partly fulfilled. Agents often contradicted

Table 2: Prompts and runs

Prompt	
1	Develop a structural conceptual design of a simple car park. Consider the following: The building height should be suitable for two stories. The building is 20 by 40 meters. There are no adjacent buildings. Design life will be 50 years. Do not consider any other factors.
2	Create a conceptual structural design for a car park with two stories. The building's footprint will be 30 by 45 meters, and there are no adjacent buildings. The design should have a lifespan of 50 years, without taking into account site-specific conditions.
3	Develop a basic conceptual design for a 2-story car park. Consider the following: The building dimensions are 15 by 40 meters. No neighboring buildings are present. The design life will be 50 years. Focus on structural design only, without other constraints.

Table 3: Agent key choices

Model	Run	Material	Structural system	Grid dimensions	Floor type	Beam type
Llama3.1:8B	1	Steel	Hybrid system	-	Concrete slab	Steel beam
	2	Concrete	Loadbearing	Column spacing 7.5m in one direction and 16m in the other	Slab on ribbed deck	RC beams
	3	Steel	Hybrid system	-	One-way spanning RC slab	Steel beam
Gemma2:9B	1	-	-	Column spacing 7.5m in one direction and 16m in the other	Hollow core RC slab	Secondary steel beam
	2	-	-	Multiples of module sizes	One-way spanning RC slab	-
	3	Steel	Steel columns and beams	Multiples of modules sizes	One way RC slab	-
Mistral-nemo:12B	1	Precast concrete	Transfer structure	-	Flat RC slab	RC beam
	2	Steel	Steel frame	3 x 2.5m and 2 x 5 m + 6m	-	Steel beams
	3	Steel	Steel frame	Regular grid dimensions	deep composite metal deck slab	Steel beams

[-] means to be determined, or an agent refers to a specific table when reasoning

previous materials or dimensions. Structurally, they did select core elements, propose systems, and reference dimensioning basics. However, they performed minimal arithmetic and rarely fine-tuned designs. Despite these shortcomings, the approach was successful in speeding up conceptual work by generating well-reasoned ideas grounded in textual references.

Limitations and Future Research

This research used only four of the six structural steps outlined by IStructE. Foundations and off-site versus on-site planning were outside the prototype's scope. Another limitation is platform dependence. Some local LLM solutions, like those in LiteLLM and Ollama, run best on macOS. Windows compatibility is reportedly in development, but not yet mature at the time of writing.

An intriguing area for future work is training LLMs with real design reports. Feeding them boundary conditions, design logic, and final outcomes could improve their engineering sense. Such training must prevent the agents from copying entire designs, but rather guide them to reason about genuine constraints and code requirements. Fine-tuning temperature settings might encourage creativity while ensuring compliance with structural norms.

The Text2Structures system could also be further extended to the remaining six steps outlined by Crielaard and Terwel (2020), namely: 5) determine the loads, 6) determine the load per element, 7) determine load combinations and reaction forces, 8) check ultimate limit state (strength and stability), 9) check service limit state (stiffness), and 10) overall final check.

We also see potential in pairing "reasoning" agents with "coding" agents that generate quick BIM models, as shown in other text-to-BIM studies (Du et al., 2024). These coding agents could automatically translate an agent's conceptual decisions into 3D elements. A crucial step would be building a robust feedback loop where a "reviewer" agent checks each design proposal against local standards or code-based heuristics. If something fails, it could prompt a targeted revision rather than wait until all tasks are done.

Lastly, implementing a user interface would broaden adoption beyond specialized developers. A future version of AutoGen Studio might let engineers run a structured sequence of agent dialogs behind a simple chatbot-like screen. This would make the system more approachable, allowing engineers to leverage AI reasoning while retaining oversight of each step.

Conclusion

This study explored the role of communicative agents in structural engineering, demonstrating their potential in automating early design tasks such as material selection, load-bearing system identification, and preliminary dimensioning. While the agents generated structured design concepts and integrated textual references effectively, limitations in reasoning consistency and iterative refinement highlight the need for further improvements. Despite these challenges, the approach accelerates conceptual design and provides a structured framework for AI-assisted engineering.

By leveraging multi-agent systems with RAG, this research provides insights into AI-driven workflows for structural engineering. The ability to translate textual descriptions into structured design suggestions enhances transparency and usability. The findings also highlight the importance of balancing automation with human oversight to ensure engineering accuracy.

Future work should focus on improving agents' reasoning with real design reports, integrating coding agents for BIM model generation, and developing reviewer agents for validation against engineering standards. Additionally, creating an intuitive user interface, such as an AutoGen Studio integration, could make these systems more accessible to practicing engineers. Advancing these aspects will strengthen AI's role in structural design and improve its practical adoption.

References

Çelen, A., Han, G., Schindler, K., Van Gool, L., Armeni, I., Obukhov, A., and Wang, X. (2024). I-design: Personalized llm interior designer. arXiv preprint arXiv:2404.02838.

Chen, G., Dong, S., Shu, Y., Zhang, G., Sesay, J., Karlsson, B. F., Fu, J., and Shi, Y. (2023). Autoagents: A framework for automatic agent generation. arXiv preprint arXiv:2309.17288.

Crielaard, R. and Terwel, K. C. (2020). Ontwerpen van Constructies en Funderingen 2. TU Delft.

Du, C., Esser, S., Nousias, S., and Borrmann, A. (2024). Text2bim: Generating building models using a large language model-based multi-agent framework. arXiv preprint arXiv:2408.08054.

Guo, T., Chen, X., Wang, Y., Chang, R., Pei, S., Chawla, N. V., Wiest, O., and Zhang, X. (2024). Large language model based multi-agents: A survey of progress and challenges. arXiv preprint arXiv:2402.01680.

Han, S., Zhang, Q., Yao, Y., Jin, W., Xu, Z., and He, C. (2024). Llm multi-agent systems: Challenges and open problems. arXiv preprint arXiv:2402.03578.

Hong, S., Zheng, X., Chen, J., Cheng, Y., Wang, J., Zhang, C., Wang, Z., Yau, S. K. S., Lin, Z., Zhou, L., et al. (2023). Metagpt: Meta programming for multi-agent collaborative framework. arXiv preprint arXiv:2308.00352.

Hsu, S.-C., Weng, K.-W., Cui, Q., and Rand, W. (2016). Understanding the complexity of project team member selection through agent-based modeling. International Journal of Project Management, 34(1):82–93.

IstructE (2022). The Structural Plan of Work 2020 - The Institution of Structural Engineers.

Khodabandelu, A. and Park, J. (2021). Agent-based modeling and simulation in construction. Automation in Construction, 131:103882.

Li, G., Hammoud, H. A. A. K., Itani, H., Khizbullin, D., and Ghanem, B. (2023). Camel: Communicative agents for" mind" exploration of large scale language model society. arXiv preprint arXiv:2303.17760.

Marcolino, L. S., Jiang, A. X., and Tambe, M. (2013). Multi-agent team formation: Diversity beats strength? In IJCAI, volume 13.

Maslej, N., Fattorini, L., Brynjolfsson, E., Etchemendy, J., Ligett, K., Lyons, T., Manyika, J., Ngo, H., Niebles, J. C., Parli, V., Shoham, Y., Wald, R., Clark, J., and Perrault, R. (2023). Artificial intelligence index report 2023.

Microsoft AutoGen Team (2023). Tool Use | AutoGen. `https://microsoft.github.io/autogen/docs/tutorial/tool-use`. Accessed: February 2025.

Microsoft AutoGen Team (2024). Solving Complex Tasks with a Sequence of Nested Chats | AutoGen. `https://microsoft.github.io/autogen/docs/notebooks/agentchat_nested_sequential_chats`. Accessed: February 2025.

Ollama Team (2024). Llama3.1. `https://ollama.com/library/llama3.1`. Accessed: February 2025.

Park, J. S., O'Brien, J., Cai, C. J., Morris, M. R., Liang, P., and Bernstein, M. S. (2023). Generative agents: Interactive simulacra of human behavior. In Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology, pages 1–22.

Qian, C., Cong, X., Yang, C., Chen, W., Su, Y., Xu, J., Liu, Z., and Sun, M. (2023). Communicative agents for software development. arXiv preprint arXiv:2307.07924.

Qian, C., Liu, W., Liu, H., Chen, N., Dang, Y., Li, J., and Sun, M. (2024). ChatDev: Communicative Agents for Software Development. In Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 15174–15186.

Wu, Q., Bansal, G., Zhang, J., Wu, Y., Zhang, S., Zhu, E., Li, B., Jiang, L., Zhang, X., and Wang, C. (2023). Autogen: Enabling next-gen llm applications via multi-agent conversation framework. arXiv preprint arXiv:2308.08155.

Zhou, W., Jiang, Y. E., Li, L., Wu, J., Wang, T., Qiu, S., Zhang, J., Chen, J., Wu, R., Wang, S., et al. (2023). Agents: An open-source framework for autonomous language agents. arXiv preprint arXiv:2309.07870.