Proceedings of the IASS 2024 Symposium
Redefining the Art of Structural Design
August 26-30, 2024, Zurich Switzerland
Philippe Block, Giulia Boller, Catherine DeWolf,
Jacqueline Pauli, Walter Kaufmann (eds.)

Applications of an LLM to Scale and Automate Computational Workflows for Civil Structural Design

Nicholas WILLIAMS*, Sujal Kodamadanchirayil SURESH, Luke HUGHES, Brian KILEEN, Theodore GALANOS.

*Aurecon Group
Lv 8, 850 Collins St, Melbourne 3008
AUSTRALIA
nicholas.williams@aurecongroup.com

Abstract

This paper outlines practice-based research into the integration of Generative AI within engineering workflows. We focus on tools built on a Large Language Model (LLM), demonstrating an ability to streamline data processing from unstructured language into structured data for inputs to design and analysis software for bridge components. Through this simple but powerful step, we highlight the improvements to robustness and reusability of these custom tools over conventional tools used within computational and digital design practices. We explore the opportunities for more extensive automation within these workflows, in turn enhancing efficiency and creating potential for optimization within the Architecture, Engineering and Construction (AEC) practices. This includes opportunities for LLMs to enable orchestration across design and delivery processes.

Keywords: GenerativeAI in Civil Design, Generative Design, Structural Design, Automated design; Digital design workflows.

1. Introduction

Artificial Intelligence is not a new technology but the accessibility and broad applicability of the recent form of Generative AI, through publicly available tools like ChatGPT, has captured broad attention. Broad predictions productivity improvements enabled by this technology are being made across industries [1], including for engineering and design services [2].

This paper addresses a tool built on an LLM that is integrated within an engineering workflow to interpret between an engineer and design and analysis software. In this role, the LLM transforms unstructured natural language into structured data for analysis and use within parametric schemas. This is a critical function in day-to-day engineering design practice and can complement other software tools to help augment a human designer.

2. Hypothesis – That LLMs can enhance engineering workflows by simplifying human-machine interfaces

LLMs are general purpose tools, trained on general datasets and applied to a range of tasks which are based on natural language, such as the writing of emails, creating reports and synthesizing content from documents. This focus on natural language can lead to assumptions that engineering tasks, focused on mathematical analysis and drawings, are not well suited to use of GenAI technologies.

There is significant research through both commercial software businesses and academia into the integration of LLMs within their commercial products. Examples in the AEC sector include Autodesk

who are addressing conceptual design activities [3]. Within academia, research includes the proposal of novel design methods for generative design [4]. Each of these cases targets design decision-making.

We undertook this research within a commercial engineering and design business, to test and practical constraints in how LLMs could be brought into core day-to-day engineering design tasks. We propose that one practical application centres on the simplification of human-machine interfaces in the design workflow. We further propose that for there are opportunities to make use of orchestration [5] to enhance users when working through complex engineering workflows.

2.1 Testing generic workflows across sample projects

In an effort to align progress during the experiment the following high-level process was drafted and is captured in Figure 1 below.

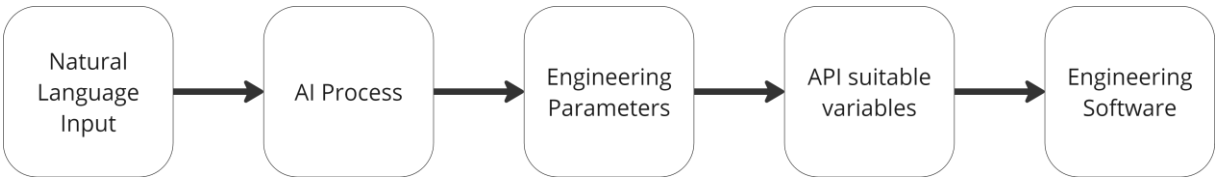


Figure 1: High level process workflow diagram

The intent of the experiment was to allow a user to enter natural language inputs into a tool and have this semantically interpreted by AI to then enable specific engineering parameters to be extracted from the text. In turn these engineering parameters are converted into variables suitable for entry into engineering software.

3. Methodology

3.1. Background tools and Assumptions

A linear reinforced concrete section design was selected for this experiment, an example that is typical to practice, with minimal opportunity for outcomes to be misunderstood, or inadequate engineering parameters or outputs to be produced. Autodesk Structural Bridge Design 2024 (ASBD) was the preferred software for this experiment with the availability of simple parameter naming conventions which are already pre-determined and documented by the software package itself. A JSON file (text-based data input) is the file format used by ASBD, a file which can be programmatically created, through processing of the natural language inputs. Several sample exports of this JSON input (Figure 2) and output files were examined. ASBD allowed input of a JSON file via the command line.

```
"process" : {
"bridgeName" : "My bridge",
"commands" : {
"commandArray" : [
{
"commandType" : "SECTION_ANA_BENDING",
"commandID" : "DS_MP4",
"wantReport" : true,
"analysisDetails" : {
"sectionRef" : "SS1",
"loadcase": "SL1",
"resistance": "MY",
"analysisType": "STRESS_STRAIN",
"neutralAxisType": "FREE",
"ignoreCompressionReinforcement": false,
"calculationType": "BENDING",
"includeUnits": true
```

Figure 2: Example extract from template JSON file from ASBD

3.2. Leveraging bespoke applications built on LLMs

The research builds on the development of a software tool named "Bamboo". Based on the ChatGPT LLM, this is a tool for extracting structured outputs from unstructured text or natural language inputs. This tool addresses key issues encountered within the broad application of digital workflows within design. Generative design models require the use of data (parameters, variables, fields) that are in the exact structure and format that the tool expects. Further to this, designers face issues including the use of highly specialized analysis software which limits cross-disciplinary collaboration [ref], and the use of parametric tools in design and the interface to, requirements for literacy in specific computer interfaces such as visual programming languages [ref].

This often means that although generative models can get very close to returning data in the structure that is expected, or predicted to be in the form that is expected, they are quite unfit for automation. Bamboo provides a bridge between a user that can describe what they want and a tool or application.

Bamboo in its current form provides access to pre-built solutions which allow:

- **Entity search**: Perform automated search operations on a document, or other text-based input for an entity which match that defined via our schema objects. This also allows semantic based comparison to harness the contextual understanding possible within LLM based tools.
- **Validation:** Allows for automatic validation of the returned entities against prepopulated criteria, which can include graphical confirmation of status. This process can also provide semantically accurate feedback to the user on errors or omissions within the input data.
- **Data security:** Ability to operate within OpenAI instances that maintain data security.

3.2.1 Natural language processing techniques

Bamboo turns unstructured text data into structured data using use of natural language processing techniques. The parameters for the entity schema are created within the Bamboo instance and is configured with a simple configuration file using an Azure deployment of OpenAI models (i.e. ChatGTP>=3.5). Details of entity schema used for this experiment are shown within Figure 3 below. A Bamboo Entity comprises of a name, a description, and properties. The Entity name and description are strings that contextually guide the LLM to identify the relevant data the user is trying to extract and the Entity properties describe the shape of the data Bamboo will return. Bamboo validates the returned data so that the output data is guaranteed to have the exact structure specified by these properties, allowing downstream data-driven processes to be executed on top of unstructured data.

```
# Bamboo entity schema
[[entity]]
    name = "Rectangular concrete section"
    description = "A specification of a rectangular concrete section section (possibly
with reinforcement)"
    unique = true

    [[entity.property]]
        name = "Concrete Grade (MPa)"
        description = "The strength of the concrete in MPa"
        type = "number"

    [[entity.property]]
        name = "Reo Grade (MPa)"
        description = "The strength of the reinforcement in MPa (default is 500 Mpa)"
        type = "number"

    [[entity.property]]
        name = "width"
        description = "The width of the section in mm"
        type = "number"
        required = true

    [[entity.property]]
        name = "depth"
        description = "The depth of the section in mm"
        type = "number"
        required = true

    [[entity.property]]
        name = "Bottom Reo"
        description = "The type of reinforcement in the bottom in format: N{diameter}-
{spacing}, e.g. N24's at 150cc bottom -> N24-150"
        type = "string"
```

Figure 3: Example of Entity schema file contents

3.3. Refined Workflow Diagram

As the experiment progressed it became clear that the original high level process workflow diagram outlined earlier in Figure 1 within this paper required some refinement and clarification. This evolved into the following process diagram which more accurately captures inputs, processes, connections between operation and outputs.

Focusing on the development of process diagram serves acted as an effective sanity check on the progress to date and allowed all parties to the experiment to confirm that they do fully understand the scope, limitations, and extents of the development.

While process diagrams such as this are commonplace in the field of software development they are not as prevalent in the AEC industry generally. They can however rapidly help clarify the scope and purpose of a 'tool' and as such should be considered as a critical part of future experiments and developments such as this.

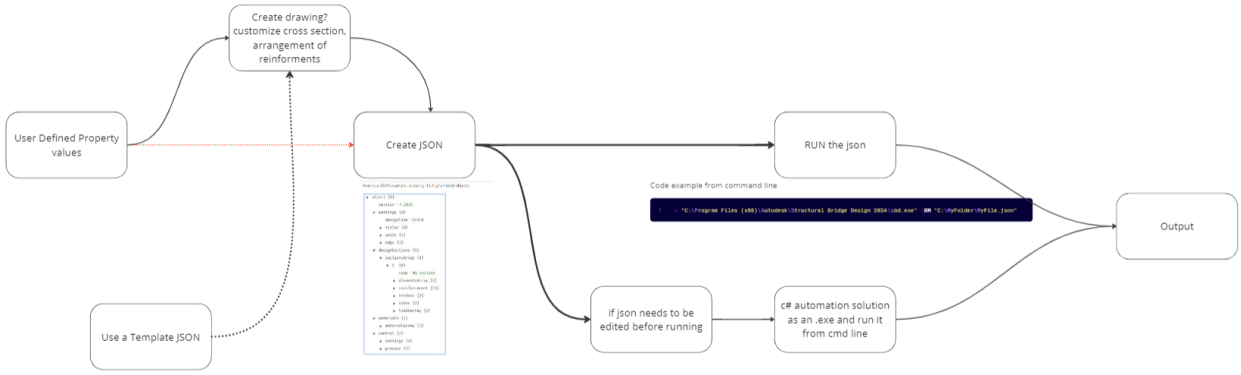


Figure 4: Example extract from template JSON file from ASBD

3.4. Interfaces

Bamboo served as the foundation of the tool which was developed and provided the user with a clear and clean UI while facilitating the programmatic interface between the engineering software package and the LLM processing itself. Following a review of the API available for the chosen engineering software, it was identified that the most reliable approach was to work backwards from the parameters defined within the engineering software and use these as the basis for our development works.

The documentation also provided specific examples of how the engineering analysis processes could be run without the need for the user to open the software to run the analysis themselves. This allowed our development to focus on only one aspect of UI (Bamboo) and carry out the running of the tasks purely as background operations not visible to the end user.

3.4.1. Front End & LLM Processing

The front end was created using an open-source app framework called Streamlit. For simplicity the UI only needed to comprise of an input box to allow the user to submit their text input.

An example of an input that could be written is provided below:

Please design a rectangular concrete section on a meter width basis. The element shall be 325mm deep and use 40MPa concrete strength. Reinforcement is to be N24's at 150mm bottom, N16's at the top 150mm. 50mm cover to reinforcement all round. Reo grade to be 500

The parameters gathered from this natural language input is used to plot a diagram of the bridge section. A user can validate this with the image that is generated. If further fine tuning is required the user is able to edit these parameters in the sidebar or the prompt and resubmit.

3.4.2. Connection to Engineering software

To interface with the engineering software, a JSON file format was the preferred way forward. To have a basis from which to work, it was possible to manually create a design section within the user interface of ASBD and then export this as a template JSON file.

The JSON structure consists of five objects. The relevant objects for our purpose are settings, design sections, materials and control. Sub-objects are found below each and are shown for information within Figure 4 below. With a well understood JSON structure, various examples were analysed and separate templates created. This allowed the experiment to cater for and choose between a design for a rectangular concrete and circular concrete cross sections from within the natural language inputs themselves. The JSON structure and content expected by the software was then constructed from the parameters parsed from the prompt entered by the user in the Bamboo interface.

Figure 5 : Standard JSON structure from ASBD

3.4.2. Outputs returned to the user

Through the API it is possible to return images, diagrams and outputs back to user in a format which is consistent with the UI based version of ASBD. Examples of these are contained in Figure 6 & Figure 7.

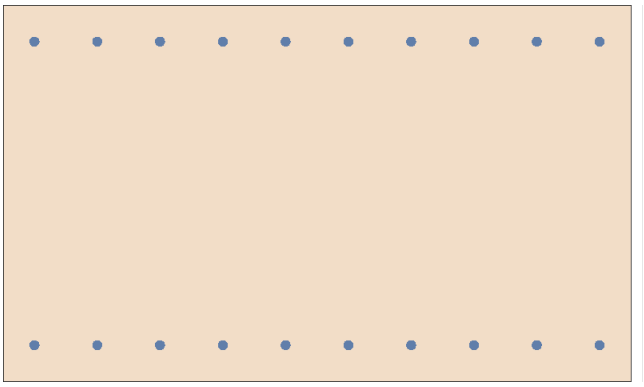


Figure 6 : ASBD Cross section reinforcement layout output

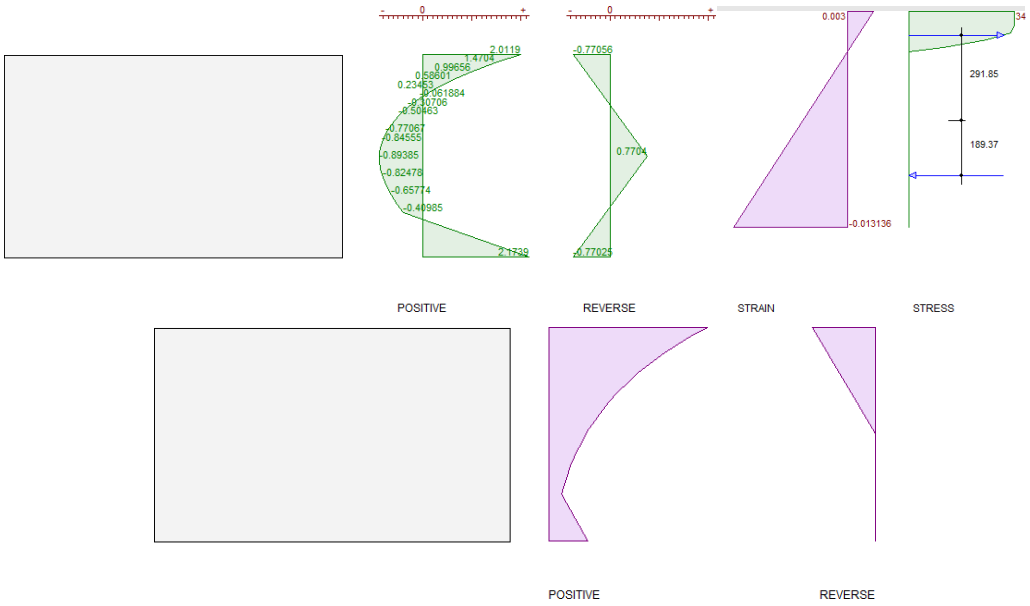
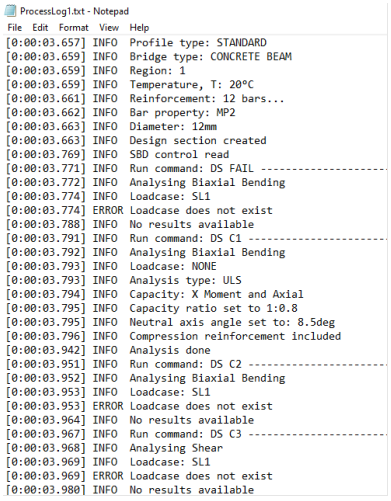


Figure 7 : Example ASBD cross-section analysis result outputs

It is also possible to return analysis logs and other debugging outputs to assist the user in the event of an unexpected output or finding. For example, log outputs define analysis results in a text format.



ProcessLog1.txt - Notepad

File Edit Format View Help

```
[0:00:03.657] INFO  Profile type: STANDARD
[0:00:03.659] INFO  Bridge type: CONCRETE BEAM
[0:00:03.659] INFO  Region: 1
[0:00:03.659] INFO  Temperature, T: 20°C
[0:00:03.661] INFO  Reinforcement: 12 bars...
[0:00:03.662] INFO  Bar property: MP2
[0:00:03.663] INFO  Diameter: 12mm
[0:00:03.663] INFO  Design section created
[0:00:03.769] INFO  SBD control read
[0:00:03.771] INFO  Run command: DS FAIL ----------
[0:00:03.772] INFO  Analysing Biaxial Bending
[0:00:03.774] INFO  Loadcase: SL1
[0:00:03.774] ERROR Loadcase does not exist
[0:00:03.788] INFO  No results available
[0:00:03.791] INFO  Run command: DS C1 ----------
[0:00:03.792] INFO  Analysing Biaxial Bending
[0:00:03.793] INFO  Loadcase: NONE
[0:00:03.793] INFO  Analysis type: ULS
[0:00:03.794] INFO  Capacity: X Moment and Axial
[0:00:03.795] INFO  Capacity ratio set to 1:0.8
[0:00:03.795] INFO  Neutral axis angle set to: 8.5deg
[0:00:03.796] INFO  Compression reinforcement included
[0:00:03.942] INFO  Analysis done
[0:00:03.951] INFO  Run command: DS C2 ----------
[0:00:03.952] INFO  Analysing Biaxial Bending
[0:00:03.953] INFO  Loadcase: SL1
[0:00:03.953] ERROR Loadcase does not exist
[0:00:03.964] INFO  No results available
[0:00:03.967] INFO  Run command: DS C3 ----------
[0:00:03.968] INFO  Analysing Shear
[0:00:03.969] INFO  Loadcase: SL1
[0:00:03.969] ERROR Loadcase does not exist
[0:00:03.980] INFO  No results available
```

Figure 8 : Example ASBD analysis log outputs

The most valuable aspect of the API access was the ability to export a fully developed ASBD file. This enables an engineer to use natural language to automate the production of a full analysis file and can then open and edit this file as if it were created manually using the ASBD software user interface itself. Examples of an output file which has been reopened within ASBD and which is fully editable is shown within Figure 9 below.

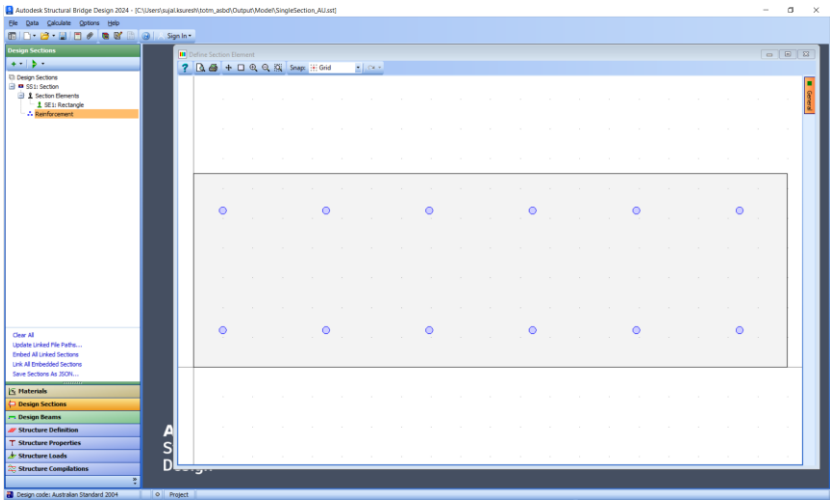


Figure 9 : Example ASBD Analysis log outputs

5. Discussion

5.1 Robust and reusable tools for future workflows

The experiment has demonstrated a viable and practical method for connecting natural language input with commonly available engineering analysis and design software using LLM tools as the intermediary. We present the user with a chat-bot style interface into which natural language inputs can be written and then parse engineering parameters from this written content. Once parameters are extracted, it is then possible to programmatically pass these via API to the Autodesk Structural Bridge Design software, run an analysis and extract calculations reports and outputs, and provide these directly back to the user.

This textbox interface presents a very low barrier to users, irrespective of technical background or discipline. This contrasts with specialised engineering design and analysis software, which currently has limitations in cross-disciplinary design workflows - structural design software is generally targeted at users who are themselves structural designers and engineers. It also contrasts with computational design tools which, while more readily used across disciplines, typically require specialised teaching and skills to be used effectively and accurately [7].

During development of the application, it was also possible to embed visual feedback to the user on the completeness and accuracy of the input they had entered. This was achieved via a checkmark or cross symbol appear beside each parameter within the user interface. Where a variable was able to be extracted, a checkmark was rendered and where the variable could not be determined a cross was rendered. While this was originally implemented as user guidance only, it became clear during testing that this visual feedback also has a basic ability to teach to user how to improve the input that they were entering into the tool. As such there is the potential to develop tools that can serve a specific engineering function while in parallel acting as learning and development aids for less experienced engineers.

Beyond this, the ability to reuse and repurpose custom tools is demonstrated to an extremely high level within the experiment. For example, to use the same parameters to conduct an alternative type of analysis it simply requires the retraining of the model (using few-shot prompting) for the analysis entity only. All other parameters and model training used within the tool could remain unchanged in that instance. This implies that if there is a need to expand the scope or coverage of the tool to carry out an additional function then this can be achieved with a minimal amount of adjustment to the development of the tool.

Expanding the tool to achieve this functionality opens the opportunity to carry out more complex analysis tasks on components such as the variety of standard precast prestressed bridge beams, or more intricate structural component interfaces. This would allow for the tool to carry out multiple simultaneous analysis tasks and provide a comparison set of results back to the user. This could in turn allow for the process of value engineering and optimisation to occur right from the outset of design and

in turn allow for a reduction in the physical materials required for a given set of design parameters. It is also worth noting that this added functionality and application capability can be achieved without any noticeable additional complexity for the end user. The processing, reasoning and selection of more complex analysis tasks can all be embedded within the Generative AI aspects of the application.

5.2 Opportunities for further research

Following from this experiment are numerous areas for further exploration. An immediate set of steps that are being considered to extend this experiment include:

- Integration of voice or speech recognition to the chat bot interface to allow the user to speak commands to the tool as if it were a member of their team to which a task was delegated. This could allow for spot checks and other high level 'on the fly' analysis tasks to be captured and recorded and actioned with outputs exactly when they are identified, rather than being added to a list of follow up actions to be carried out by a designer at a later stage.
- Expansion of the tool into other analysis types within the ASBD cross section design module which would allow the tool to be used for a wider variety of design tasks by the user. For example, a circular pile or column section is often designed using an Interaction Curve analysis method rather than plain bending analysis method. This could then be selected by the user or prompted by the interface from the content entered by the user for these types of elements.
- Expansion of the tool to allow it to process and generate all iterations of reinforcement layout or structural depth within a set range of values. i.e. Run a series of analysis processes to generate all outputs from a prompt such as "Report on the capacity of a 200 thick 40MPa for all bar diameter configurations between 12mm and 32mm. Consider the spacing of reinforcement to be 150c/c for all cases." This scenario would be well suited to situations where there are multiple similar elements on a project which should all be optimized individually and independently of one another. In this situation the user may simply select and report on the relevant result for each element from the group of results generated.
- Development of a process to allow iterative design optimization based on the results of analysis runs i.e. Run a series of analysis processes to find the 'best' design outcome from a prompt such as "Carry out a design for a 200 thick 40MPa slab to carry a maximum ULS bending moment of 112kNm/m and a ULS shear force of 65kN/m. This is well suited to the optimization of an individual structural component, where the external loading to be designed for is known but the reinforcement design has not yet been completed.
- Experimentation with evolutionary style 'scoring' of outputs to allow potential for semi-autonomous design optimization. By preloading of criteria and associated weighting within the development process it would be possible to direct design outputs in a particular direction to suit variable client or project needs i.e. limit the quantity of reinforcement used, keep maximum concrete volume to a minimum, limit overall embodied carbon within the final design etc.

Beyond these simple extensions, there is opportunity to consider the implementation of 'agents' each of which would be loaded with the entity descriptions required for the distinct type of analysis or function they are tasked with running. These agents would enable a more readily modular kit of parts and present opportunities for 'orchestration' within design workflows. While orchestration is practiced within multiple industry segments [5], it has only limited application within the AEC industry [6].

Furthermore, we can identify opportunities for a digital agent to drive orchestration itself, trained on how to select the appropriate agent based on the content and semantic meaning of the input from the user. In the context of this experiment the agents could be developed to handle tasks across multiple iterations and permutations. As examples, these could include:

- different physical cross sections to be analysed
- different clients and their associated technical standards or requirements
- different regions or counties with their associated standards and requirements

- designs with outputs formatted for a different purpose i.e. concept, tender or detailed design

There is also the potential for the orchestration agents to themselves be developed with different overall goals or objectives such as:

- refine and present a design that uses the least overall concrete in the section
- refine and present a design that uses the least overall reinforcement in the section
- refine and present a design with the least embodied carbon across all components

The use of an entity schema within the development of the agent and overarching orchestration agent selection allows for a very high level of modularity. Beyond the precise requirements of many digital and computational interfaces, these present increased opportunities for functions to be added without the need to amend the functionality of any other agents which are already developed and functional.

6. Conclusion

This paper demonstrates the application of an LLM based tool within engineering design workflows. The role of the LLM centers on interpretation of user inputs from natural language to design software, taking unstructured language into a structured data. This is a simple but powerful step, minimizing common issues that arise through specialized and parametric software. It further builds a level of robustness and reusability into the tools being developed.

This initial demonstration sets a basis for broader testing of LLMs within engineering design workflows. Projects demand the integration of multiple disciplines and software packages to produce varied outputs. The ability for LLMs interpret data and reliably structure it as parameters required for another tool, suggests that these tools could streamline interfaces at multiple places within a workflow. This would augment a designer by reducing data transformation tasks. Through this, future research will look at how a design might further 'orchestrate' design workflows.

Acknowledgements

This research has been funded through internal investment within the Aurecon Group.

References

[1] Chui, M., Hazan, E., Roberts, R. et. al.(2023), "The Economic Potential of Generative AI, the Next Productivity Frontier", https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/the-economic-potential-of-generative-ai-the-next-productivity-frontier#/

[2] M. Sandburg, R. Gerth, W. Lu, G. Jansson, J . Mukkavaara and T. Olofsson, (2016) "Design automation in construction – an overview" Conference: Proceedings of the 33rd CIB W78 Conference, Brisbane, Australia.

[3] MA, K.Grandi, D.et al. (2023), "Conceptual Design using Large Language Models", ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, https://www.research.autodesk.com/app/uploads/2023/08/Conceptual-Design-Generation-Using-Large-Language-Models.pdf

[4] Balmer, V. et.al (2024), "Design Space Exploration and Explanation via Conditional Variational Autoencoders in Meta-Model-Based Conceptual Design of Pedestrian Bridges", Automation in Construction, Vol 163, https://doi.org/10.1016/j.autcon.2024.105411

[5] S. Rasal and E. J. Hauer, (2024) "Navigating Complexity: Orchestrated Problem Solving with Multi-Agent LLMs" arXiv:2402.16713v1 , https://arxiv.org/abs/2402.16713

[6] Peeters, R. and Bizer, C. (2024), "Entity Matching using Large Language Models" arXiv:2310.11244v2, https://arxiv.org/abs/2310.11244

[7] Belesky, P. (2018) "The Green Grasshopper: Approaches to Teaching Computation Design Methods in Landscape Architecture", Journal of Digital Landscape Architecture, 3-2018, pp. 406-41