

A CASE STUDY OF CONTEXT OPERATIONS WITH LARGE LANGUAGE MODELS FOR THE SYSTEMS ARCHITECTURE DESIGN OF A HEAVY SAR UAV

Vinicius Kaster Marini[1] & Petter Krus[2]

[1]Department of Mechanical Engineering, Centre of Technology,
Federal University of Santa Maria, Santa Maria/RS, Brazil
[2]Fluid Power and Mechatronic Systems, Department of Management and Engineering,
Linköping University, Linköping, Sweden

Abstract

The complexity in developing modern aircraft systems challenges design engineers on the synthesis and development of system architecture models that are supportive to the integration task. This contribution introduces a framework of formal operations for assembling context in aircraft systems engineering suported by large language models (LLMs). To demonstrate the framework in operation, a case study is carried out with the task of modelling the component architecture design for a heavy search-and-rescue UAV. The context workload is assembled in four treatments with requesting at five separate LLMs to generate a system architecture model of the UAV. The responses provide examples of possible SAR UAV architecture configurations, and enable the validation of model codeblocks on codeblock length and element counts. The case study approach brings up the essential role of exemplars as convergence element to balance the diverging effect of preceding context, and the role of modelling validation methodologies to improve the outcomes from LLMs to speeding up the modelling tasks in aircraft architecture design.

Keywords: Systems architecture, Aircraft systems, Large language models, Context operations

1. Introduction

The implementation of artificial intelligence (AI) to augment systems design and modelling for novel aircraft influences to ways forward in developing the aircraft design process. This contribution aims to demonstrate the use artificial intelligence (GenAI) capabilities at aircraft system design tasks. These tools have attracted the attention of system and design engineers ([1, 2]) for the potential support to early design and system modelling, a context where other design automation techniques fell short. However, their use in engineering and design is subject to challenges regarding their probabilistic approach to content ([3]). While synthesis of assurance arguments makes a potential use case for LLMs, their use in engineering and design modelling tasks requires careful review of the LLM outcomes towards design work products ([4]).

Then, *how to improve the accuracy of GenAI to design intent towards system architecture modelling*?

This contribution presents the theme of LLMs in aircraft systems design within the use case of system architecture modelling from awareness to the state-of-the-art ([5]) towards the systematization of context input.

1.1 Related work

Then, related work involves the following use cases: assistance to systems modelling; assistance in system tools; example&rule assistance; complementary knowledge; assisted safety-driven methods; assisted reliability&safety methods; exploration of dependencies; and HMI for design assistance.

Assistance to systems modelling: Explorations on using LLMs with systems models demonstrate the early use of developing language models along system modelling frameworks. [6] experiment with prompting at LLMs with focus on a single system modelling task, supported by model templates intended to provide exemplars. The integration of modelling frameworks through exemplars helps extract useful information for model-building.

Assistance in system tools: LLMs with chatbox tools can work within model-based systems engineering (MBSE) environments such as reported by [7] and [2], where the use of LLMs benefits from capturing the modelling framework in the MBSE environment to generate of system models from concept to architecture. This is also the focus in [8], who work with prompting at the interface between the LLM tool and the modelling environment with adding a design element ontology to support the synthesis of design specifications.

Example&rule assistance: [1] explores the use of LLMs within aircraft concept design with support of structured templates and domain-specific rules to generate system configurations onto prompting the LLM to compose and generate system models with considering these inputs. This approach evolves from early contributions by using a preliminary domain question refined to a systematic prompt with topic structure, aiming to convey design intent and composition rules on objects and their mutual relations in the intended model.

HMI for design assistance: Counter to the perception that the designer is to be automated out of the process, [5] used concept maps to be parsed/splitted to enable sequenced prompt chains on mission design information for aircraft design. [9] explores ways of working with LLMs in different modes of operation: firstly, the direct generation of models from prompting; then, aircraft architecture models generated through on-the-fly generated application code that instantiates LLM-generated modules.

These use case propositions demonstrate the diversity of situations where LLM-supported GenAI tools could help systems design. At the same time, this contribution aims to further develop knowledge and practice on their use by providing an a formal structured approach to LLM operation, and an example on how it works.

2. LLM function

The context operations are hereby defined through analysing the workings of a chatbox application ([10]) towards assembling the LLM call. Here, we use equation clauses to present the functionality and parameters of LLMs, and denote all references to them as LLM_X. The following characteristics apply:

2.1 Choice of LLM function

To understand the context intake functionality of large language models (LLMs), we formalize its workings as a function class $'LLM'_X$, where each specific model plays the role of a function; $'X'$ may be replaced by any particular LLM at the discretion of the user. Equation 1 approaches LLMs in representing the ability of the professional user to choose one particular model among several available:

$$\begin{aligned} LLM_X = \vee\{&M_{GPT_m}, M_{Claude_m}, M_{Gemini_m}, \dots, \\ &M_{Deepseek_m}, M_{Qwen_m}, M_{Kimi_m}, \dots, \\ &M_{Mistral_m}, N_{Command_m}, \dots\} \end{aligned} \tag{1}$$

Here, M refers to each being a model, and m refers to one of several versions in a specific model lineup. One can then understand M as an instance of LLM_X which can perform advanced language processing operations, including design and engineering tasks, where LLM_X involves the processing of any chosen M at answering to a request by the user.

2.2 Context operation to LLM function

The context workload components for $C_{Q(i)}$ shall follow certain order in order to maximise the compliance by the LLM_X function to the intent of the query. The context workload $C_{Q(i)}$ to the LLM_X function results from assembling these context modules onto a message to the model. The model will process the workload $C_{Q(i)}$ onto an answer $A_{M(i)}$ from its internals.
Then, Equation 2 displays the formulation of each single call to LLM_X.

$$\underbrace{R_{P(i)}}_{policy} || \underbrace{P_{P(i)}}_{reference} || \underbrace{Q_{P(i)}}_{question} = C_{Q(i)} \tag{2}$$

$$A_{M(i)} = LLM_X\{C_{Q(i)}\}$$

Here, $R_{P(i)}$ is placed first for the policy component to set rules over the whole of the call. The $P_{P(i)}$ component will convey imported reference units to provide complementary guidelines towards the answer. Then, the core question unit $Q_{P(i)}$ includes the query statement as expression of intent and can include specifics such as μ-Templates ([11]) and other exemplars besides the actual question.

2.3 Context assembly units

The context operations will be hereby defined regarding an assembly viewpoint, considering a modular context defined from the concatenation of composed parts. Our intent is to demonstrate the context-building components towards the enhanced chatbox. Before proceeding with the example, the following context operations are introduced:

2.3.1 Rule and policy operations

The user can compose/import the global prompt G_{sp} in the form of equation 3 to request the LLM to follow a set of rules, without adding a boundary prompt. Here, the call to LLM_X takes place by the (I) message line, and G_{sp} will be forwarded at all times with no complement by the B_{sp} component.

$$R_{P(i)} = \underbrace{G_{sp}}_{global} \tag{3}$$

The user can also compose/import the boundary prompt B_{sp} as complementary guidance to LLM_X alongside G_{sp} to the $R_{P(i)}$ policy component, in the form of equation 4. Here, the call to LLM_X takes place by the (II) message line, and B_{sp} context component can be forwarded along the G_{sp} global prompt.

$$R_{P(i)} = \underbrace{G_{sp}}_{global} || \underbrace{B_{sp(i)}}_{boundary} \tag{4}$$

Here, $R_{P(i)}$ can carry the single global prompt, or can carry both global and boundary prompts. These complement each other regarding the order in which they are assembled: G_{sp} conveys role, background, attitude and style directives for LLM_X, whereas $B_{sp(i)}$ forwards guidelines to how LLM_X shall assemble the answer such as work process, topic structure, stylesheet and generic codeblock format as applicable.

2.3.2 Imported references

These context units can be individually composed/imported by the user at anytime before the call. Here, an imported context unit $P_{r(i)}$ serves as reference towards the call $'i'$. to LLM_X. It becomes available as context for calling LLM_X, according to the equation 5 below.

$$P_{P(i)} = \underbrace{P_{r(i)}}_{single} \tag{5}$$

At the same time, the reference component can include several $'j'$ context units towards the call $'i'$, from the first to the final $'p'_{th}$ reference, each making a context unit $P_{p(j,i)}$ to LLM_X. These $'j'$ units become all available as context for calling LLM_X, according to the equation 6 below.

$$P_{P(i)} = ||_{j=1}^{p} \underbrace{P_{r(j,i)}}_{ref.unit} \quad \Big| \; \forall \; j \in \mathbb{N} \tag{6}$$

Here, $P_{P(i)}$ can carry a single reference imported through $P_{r(i)}$, or several references imported through $P_{r(j,i)}$ - which means there are $'j'$ reference units assembled onto the context of the $'i'_{th}$ call to LLM_X. This formulation does only regard the assembly of reference units towards the reference module $P_{P(i)}$ in the context workload $C_{Q(i)}$; the user makes all considerations of purpose and ordering about importing the reference units, one or multiple at a time.

2.3.3 Core question operations

The context assembly to LLM_X calls involves the crafting of a question for each request. The user needs to compose the core question $Q_{P(i)}$ towards calling LLM_X, as in display with Equation **??**. Here, the user can compose a single query statement $Q_{c(i)}$ as in display by Equation 7. For that purpose, the query statement will contain guidewords such as *'what'*, *'where'*, *'how'*, or elaborated requests such as with including *'please explain'* or *'I need to know'* clauses.

$$Q_{P(i)} = \underbrace{Q_{c(i)}}_{query} \tag{7}$$

At the same time, the user may figure the query statement can be enhanced by *prompt vectors*. Here, the question $Q_{P(i)}$ as shown by equation 8 aggregates the clauses for $Q_{c(i)}$, and then the $O_{v(i)}$, repeated times to each $Q_{v(m,i)}$ elements, and then $Q_{P(i)}$ will involve the whole aggregated question with all vectors asking the LLM_X function to provide an answer.

$$Q_{P(i)} = ||_{m=1}^{p} \quad \underbrace{Q_{c(i)}}_{query} \quad || \quad \{ \; \underbrace{O_{v(i)}}_{operator} \; || \; \underbrace{Q_{v(m,i)}}_{vectors_{col}} \; \} \quad \Big| \quad m,i \in \mathbb{N} \tag{8}$$

The prompt vectoring formulation in Equation 8 applies to the assembly of a single request to LLM_X function, and enables comprehensive responses upon the capability of individual LLM_X functions. The individual answer provided by LLM_X will address all single specifics as defined in prompt vectors, because the query statement $Q_{c(i)}$ and the operator $O_{v(i)}$ are replicated at all times along each prompt vector.

3. Case study

This section demonstrates the outputs of system configuration requests enabled by the context operations formally defined in section 2.2. To demonstrate how context operations work, this paper involves a modelling case regarding the component architecture of a hybrid SAR UAV, which is hereby denoted as *Suthern-Cross* as in Table 1.

This includes a depiction of an as-current Search-and-Rescue operation over flooded area with an helicopter - the SAR Drone is expected to have similar capability of recovering victims. The minimum modelling scope involves the following aircraft subsystems: energy and propulsion; hoist, winch and rescue bed; flight control systems; navigation and communications components. These are set into a dictionary to validate the models.

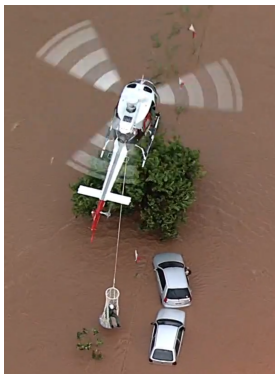
3.1 Context workload workflow

The context workload is assembled in four treatments with requesting at five separate LLMs to generate a system architecture model of the UAV. The treatments are designed to be cumulative, where each treatment adds more context units to the workload for LLM_X to process.

Considering this workflow, there are different context operation possibilities upon user choice over a string of calls to LLM_X, as Figure 1 displays the system policy composition operations.

Table 1 – Context operations demonstration assignments regarding systems characteristics

	Heavy SAR UAV - Suthern-Cross
Application	Lifting and transport of people and equipment.
Assignment	Operation: Search-and-rescue operations to recover victims from flash flood areas to medical care.
Key constraints	Agressive weather, station-keeping on hover, energy budget, low noise footprint, regulatory compliance.
LLM Assignment	Help functional decomposition and architecture design under professional supervision towards defining system/concept architecture.



Picture source: `https://youtu.be/QVyptRhdDgA?si=ygtpMefwQXCHfO54&t=84`, accessed on 2026-06-05.

The user will first compose/import the components he expects to forward to LLM_x; here, he can add G_{sp} alone, or can add G_{sp} with B_{sp} towards setting R_{sp}. Then, the application composes the rulework context R_{sp} from the rulework components available, and sends the rulework to LLM_X at the request of the user.

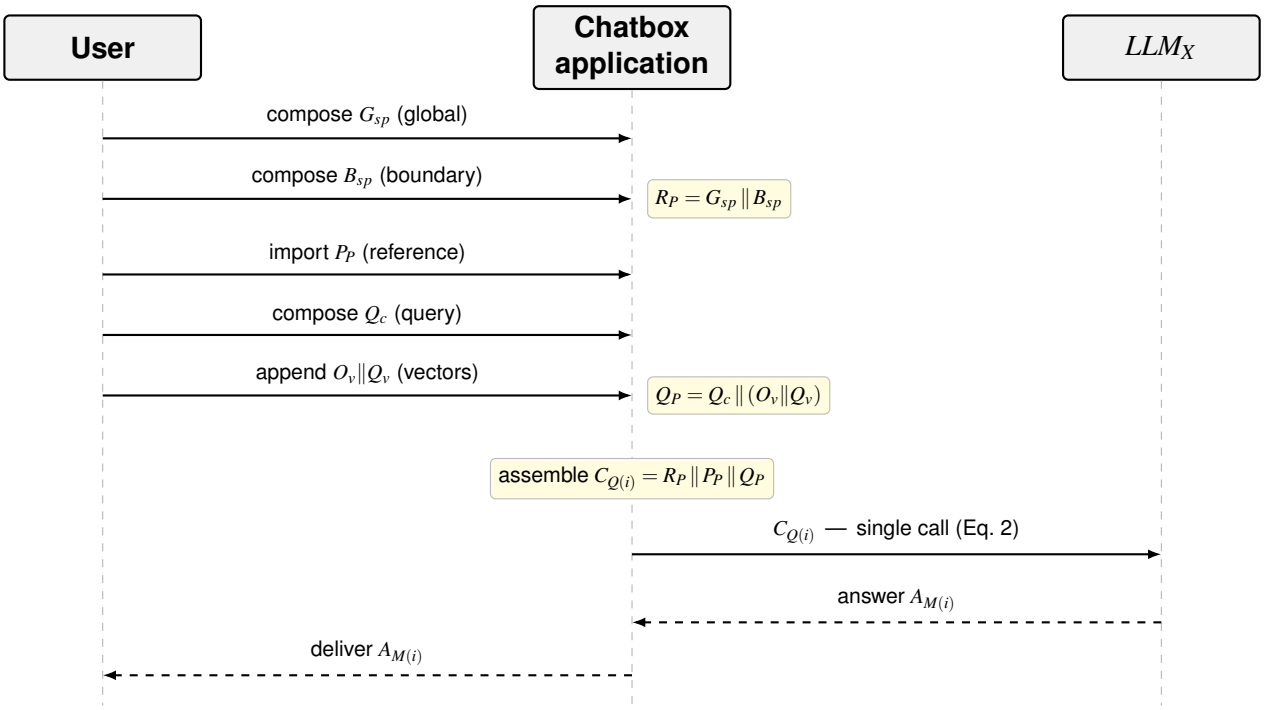


Figure 1 – Sequence diagram of the single LLM_X call. The user composes or imports the five context units (the columns of Table 2); the chatbox application assembles them into the workload $C_{Q(i)} = R_P \| P_P \| Q_P$ (Eq. 2), issues a single call to LLM_X, and returns the answer $A_{M(i)}$.

The responses provide examples of possible SAR UAV architecture configurations, and enable the validation of model codeblocks on codeblock length and element counts. The case study approach brings up the essential role of exemplars as convergence element to balance the diverging effect of preceding context, and the role of modelling validation methodologies to improve the outcomes from LLMs to speeding up the modelling tasks in aircraft architecture design.

3.2 Context workload treatments

The context treatments involve selective use of context units onto requesting LLM_X for the generation of the intended work product. The treatments are designed to be cumulative, where each treatment adds more context units to the workload[1] for LLM_X to process. The Table 2 displays the treatments below:

Table 2 – Context workload treatments.

	G_{sp} **Global**	B_{sp} **Boundary**	P_P **Reference**	Q_c **Query**	$O_v \| \| Q_v$ **Vectors**
Content	Role directives, mission scenario, Validation	Modelling scope, principles, rules, requirements	Generated benchmarking analysis.	Work product request, and requirements.	μ-Template subsystem exemplars.
T1				•••> 127 Tk	
T2	•++>	•++>		•••> 2485 Tk	
T3	•++>	•++>		•••>	•++> 7645 Tk
T4	•++>	•++>	•++>	•••>	•++> 20555 Tk
Input	Typed/pasted prose & topics, saveable.	Typed/pasted prose & topics, saveable.	Pasted source text import or DOCX, PDF	Typed/pasted or imported prose & topics	Paste into or import

This experiment uses `plantUML` as modelling language considering its use with prior examples of LLM-aided modelling experiments, such as by [6] and [1].

3.3 Model use treatments

The LLM assortment was defined with mind to considering processing resource: one model running on local GPU[2], two models from the cloud[3] that can be run on high-end local desktop GPUs to 48Gb RAM, and two frontier models to be run from the cloud. Table 3 displays the metrics, their descriptions and their units.

Table 3 – Model structure metrics - characterization of models.

	M1	M2	M3	M4	M5
	Qwen3 8b	**ChatGPT OSS-20b**	**Nemotron3 super-120b-a12b**	**Kimi K2.5**	**Claude Sonnet4.6**
	[12]	[13]	[14]	[15]	[16]
Resource	Local/GPU	Cloud/Nvidia	Cloud/Nvidia	Cloud/Ollama	Cloud/Anthropic
Architecture	Distilled	MoE	MoE	MoE	Undisclosed
[Param.]	8×10^9	2×10^{10}	$1,2 \times 10^{11}$	1×10^{12}	
Context [Tk]	$3,27 \times 10^4$	$1,28 \times 10^5$	$1,00 \times 10^6$	$2,56 \times 10^5$	$1,00 \times 10^6$
Answer [Tk]	$4,09 \times 10^3$	$3,27 \times 10^4$	$6,40 \times 10^4$	$6,40 \times 10^4$	$6,40 \times 10^4$

The API interface forwards the assembled context workloads to the selected LLM_X models in Table 3. The procedure generates 20 workloads from context and model treatments, where each resulting answer is exported for rendering with `plantUML`, followed by evaluation of syntax and modelling.

[1]The context intake length in tokens [Tk] is the workload that utilizes context length capacity in each of the models called. Table 3 displays model characteristics regarding context length and answer length capacities for reference.

[2]The local model runs with a 4Gb Nvidia Quadro T1000 GPU and a 2.7-4.0 GHz 12-core Intel Core-i7 CPU with 16Gb memory.

[3]Cloud bandwidth intake takes place at 50 Mbps and answer yield takes place at 500 Mbps.

3.4 Model answer verification

The modelling response can provide a work product the interpreter is able to render, yield a system model that represents the intended system architecture with sufficient detail, and produce a configuration that complies with required functionality. Here, `plantUML` syntax errors prevent the interpreter from rendering the intended model. At the same time, LLM limitations can produce a representation with insufficient detail and lacking compliance to system requirements.
The system architecture models of the Heavy SAR UAV have been verified under the criteria in Table 4 below:

Table 4 – Validation criteria for system architecture models.

Syntax misses	`id` used in relationship, not declared with `as`.	Same *as part* used multiple times in code.	Malformed arrow, unicode in `id`, stray characters.	Unbalanced *{}*, unclosed package, miss *start/end*.	Malformed soft/ hard-*port* name, or signature.
Element misses	even-Propellers (motor+prop); single power source +distr.+battery	Structural elements like frame, arms, pylons, thermal, landing gear.	Ancillary systems such as parachute, hoist+winch+bed, comms, datalink.	Flightctrl+autopilot IMU, INS, GNSS, airspeed, radar-alt, and odometry/wind.	Sensors such as EO/IR + thermal +multispectral and LIDAR.
Port/flow misses	source out-type $\neq$ sink in-type (e.g., *EL* $\rightarrow$ *MR* without converter).	output flow feeding output port, or sink flow emitting upstream.	Declared port with no flow, or flow into undeclared port.	Component signature conflicts with actual flow input.	Same component pair connected multiple times or A$\rightarrow$A.
Compliance errors	Elements astray with no connections in the model.	Unintended duplicates such as two *Ctrl_Flight* as one triplex.	No terminal output in model, such as (propeller$\rightarrow$*AERO* or hoist$\rightarrow$payload).	Source never reaches bus/fusion, or absent/broken feedback loop.	Fuel-source even-props or redundancy violations.

These come from the differences between raw models from LLM_X output and corrected models to syntax, along dictionaries and semantic rules that were made operational as a Python script. The question/answer statistics for each model response to workload were generated by a token count monitor within the chatbox application.

4. Results

The generative modelling approach supported by modular context assembly has been experimented on 20 parallel model runs along the workload treatments from section 3.2 and the models called in section 3.3. To demonstrate the results from the case study, this section first presents selected examples of model responses by LLMs, and then presents a comparative analysis of model length and element counts.

4.1 Modelling examples

Figure 2 displays the model produced by (M5) Claude Sonnet 4.6 for the first treatment (T1) with no ancillaries to the question. This is a class model with system packages. This model has 38 high-level components with specifications in class properties (+) and 46 flows mostly expressing overall interdependence between elements. While possibly displaying a design specification of a heavy a SAR UAV, it deviates from the intent of a component architecture.
To support that purpose, treatments (T3) and (T4) involve the use of μ-Templates [1] as exemplars embedded in prompt vectors, with examples in figure 3a, 3b and 3c. These are aggregated to the query statement as prompt vectors following equation 8 to steer the processing of the LLM.
Figure 4 displays the raw model produced by ChatGPT-OSS-20b for the third treatment, which involved the policy and prompt vector ancillaries (T3). This model has 28 individual elements and 32 flows; power transmission and distribution elements appear with two sets of soft ports – combining lower level components.
Then, Figure 5 displays a corrected model with basis on the one above. This corrective treatment involves layout modifications to the powertrain components, whereas the hoist system, electronics and communication systems remained the same, including the port-combination components.

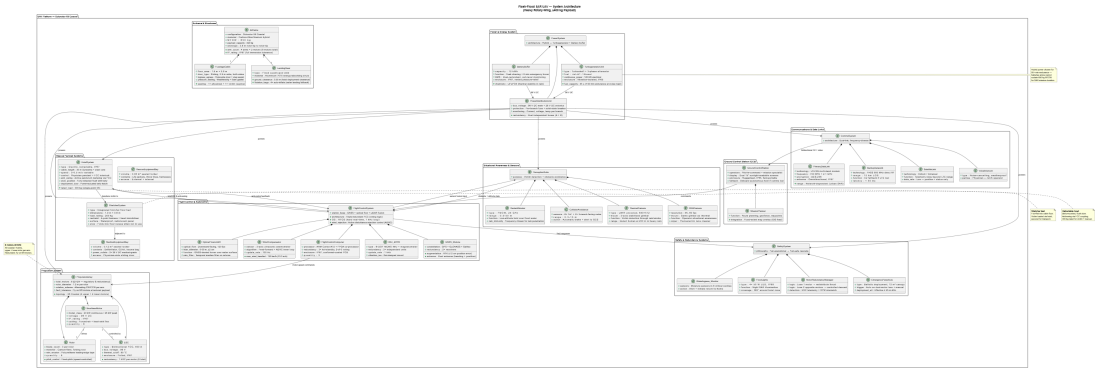
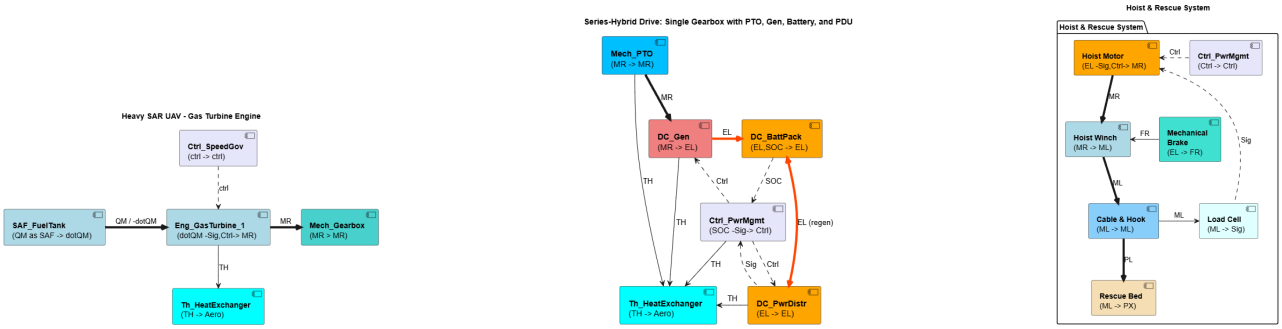


Figure 2 – Model requested from Claude Sonnet 4.6 for T1 context.



(a) SAF Gas Engine.

(b) PTO and distribution.

(c) Hoist system.

Figure 3 – System model errors from LLM_X answer products.

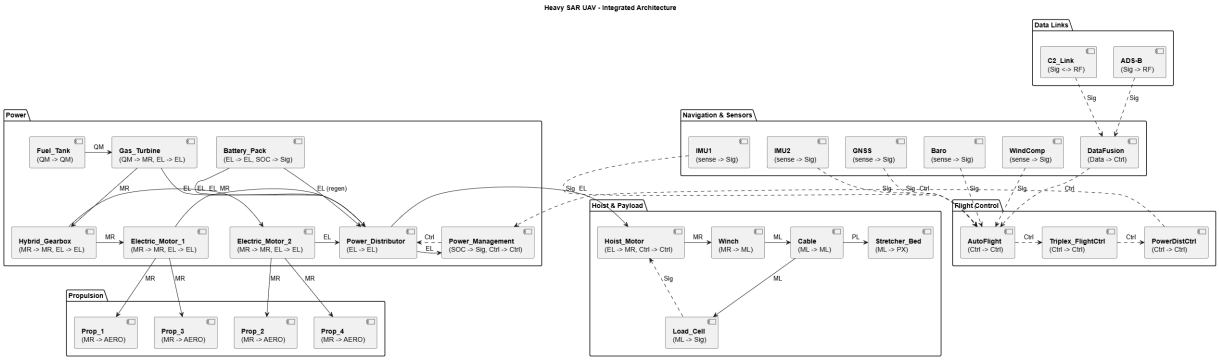


Figure 4 – Model requested from GPT-OSS-20b for T3 context.

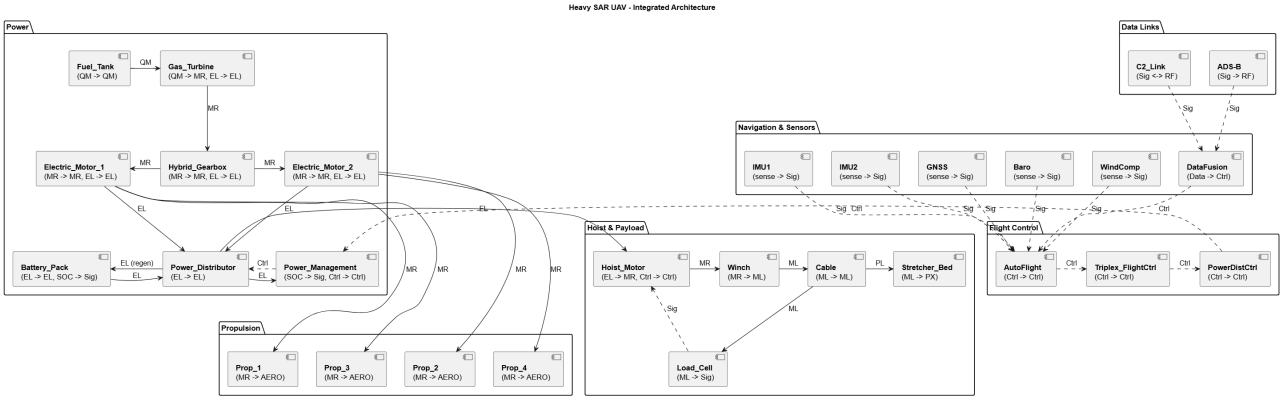


Figure 5 – Model corrected from that of GPT-OSS-20b for T3 context.

4.2 Modelling analysis

This section is intended to present the results of the experiment and discuss its outcome. Figure 6 displays the result of model length from processing context workload. Here, policy (T2) has mostly increased the modelling output from LLM_X with exception of the top frontier model (M5). The introduction of μ-Templates as prompt vectors into the question (T3) reduced the model length; the addition of reference (T4) did not have so significant effect.

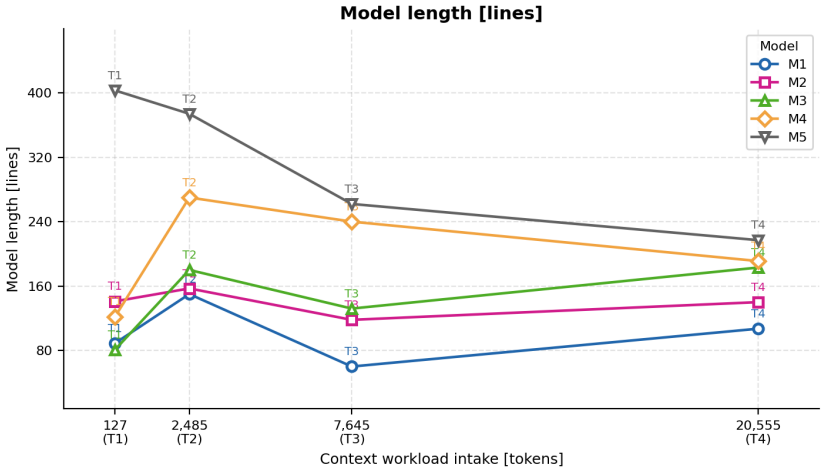
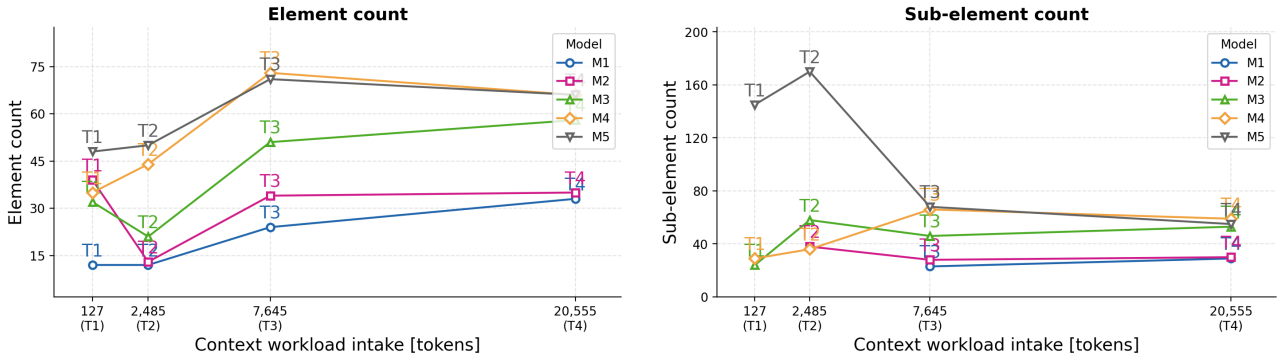


Figure 6 – Model length as result of processing context workloads.

Then, an increase in element count from figure 7a with the introduction of policy (T2) and prompt vectors (T3) corresponds to a decrease in sub-element count from figure 7b in these treatments. The counts did not change significantly from treatment (T3) to treatment (T4) showing the effect of μ-Templates in steering the representation of system architecture.



(a) Element count by workload treatment.

(b) Subelement by workload treatment.

Figure 7 – Element and subelement counts by workload treatment.

There are significant effects from treatments regarding element misses and port/flow mismatches from LLM_X yields. Element misses were consistently high on smaller LLMs, and policy and guidelines along the question (T2) made component misses to increase. Then, μ-Templates as prompt vectors practically annulled component misses in (T3) and (T4) treatments.

4.3 Discussion of results

This paper presented modelling samples towards architecting a heavy SAR UAV system, and results from a quantitative analysis over the generated models. This case study experiment involved our running 20 queries spanning context workloads and selected LLMs that were requested to generate system architecture `plantUML` codeblock models. The analysis part involved setting up modelling metrics to verify the output of the LLMs; here, we focus element count and syntax requirements for the verification.



Figure 8 – AI-generated Heavy SAR UAV rendering (source: Gemini).

The performance of frontier models such as Kimi K2.5 (M4) – rendering of its (T3) treatment shown in figure 9 – and Claude Sonnet 4.6 (M5) in modelling architectures with support of μ-Templates – and that of similar-scale models such as Deepseek and ChatGPT-5.x from previous experience of ours in [10] – bears no remarks upon their higher scale and longer context capacities.

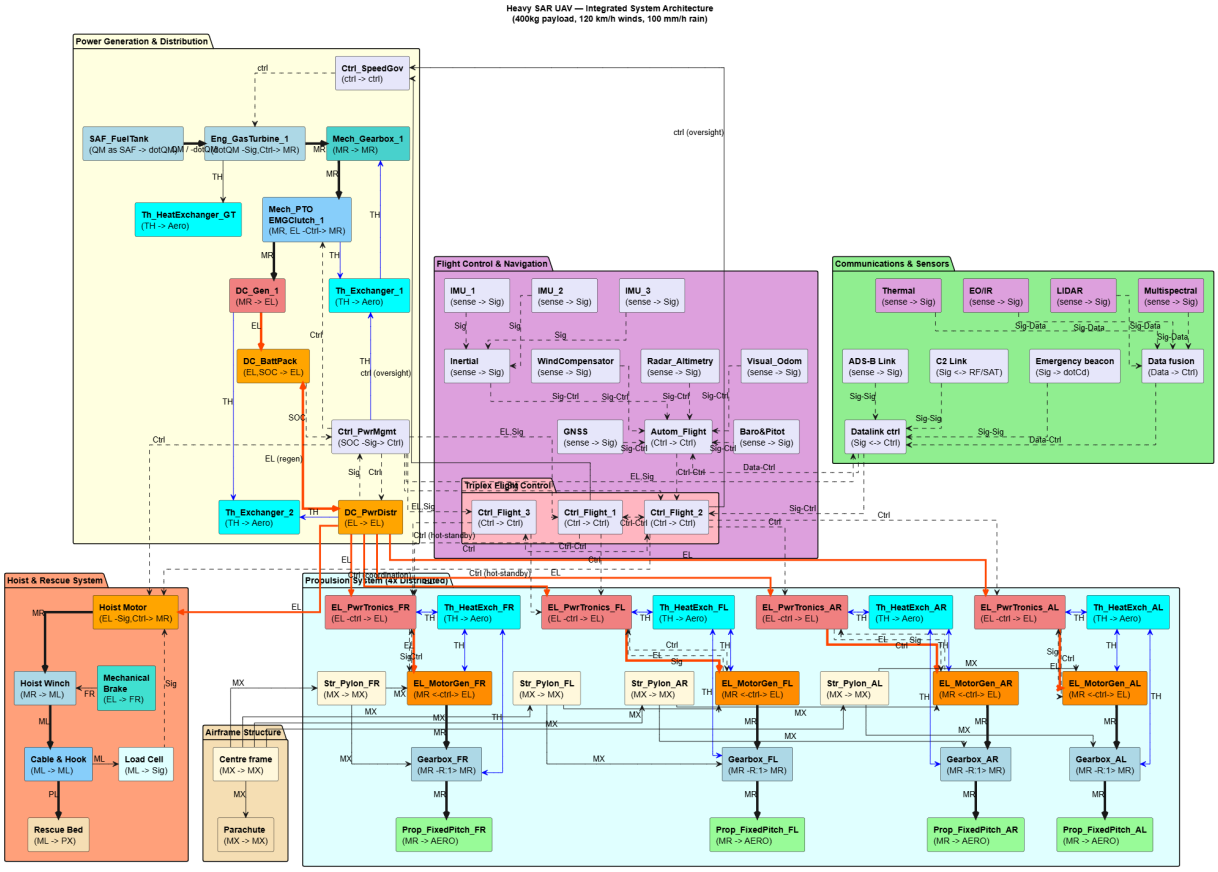


Figure 9 – Model requested from Kimi K2.5 for T3 context.

This demonstrates μ-templates have a convergence effect to a desired model setup regarding its format, structure and compliance. However, in models such as Qwen3, GPT-OSS and Nemotron, the effects of context operations deserve attention. Here, Qwen3-8b (M1) and GPT-OSS-20b (M2), both with maximum 32767 Tk of context capacity, were significantly utilized – especially in treatments T3 and T4 with including μ-Template exemplars.

Their performance comes not without their responses coming briefly, either with subcomponents declared within component blocks – which deviate from the purpose of architecting – or with omitting components entirely yet complying with code syntax and modelling conventions. Here, one expects a topology of components, effect flows and ports that supports component-level specification and designing with mind to later integration tasks. At the same time, such responses can be fixed to completeness, accuracy and compliance with a few minutes of juggling model code.
The whole dataset reveals the effect of the primacy and recency, characteristics of information units within a context workload to an LLM that were revealed by [17] upon finding out on the limitations of LLMs under high utilization of their context length capacity. These effects have something to do with context units that *precede* the question having a divergence effect, and context units that are within the question, which actually *succeed* the query statement – that carries the actual request and thereby the modelling intent – and make both convergence and enforcement effects to expected patterns in both coding and modelling.
Understanding the capabilities of LLMs of a scale spectrum from *giga* to *tera*-scale enables proceeding to investigate and take advantage of novel context handling techniques to improve the capability of artificial intelligence onto systems modelling to design intent.

5. Conclusions

This contribution has presented a framework of context operations on which to prompt at large language models. Being implemented in a chatbox application as that shown by [10], this framework was successfully operated towards a case study of aircraft design, here being a heavy SAR UAV. This case study contributes with evolving from modelling examples towards the experimentation with modelling requests, and the verification of their modelling results through measuring their architectural properties.
The evaluation of modelling quality by large language models from the synthesis of modular context workloads enabled us to discuss on the modelling capability of LLMs under a single query. The modelling results presented from the experiment are on par with current understanding about the workings of LLMs to system design. At the same time, these modelling results enable the development of a learning database on how to support systems modelling with generative artificial intelligence.

6. Contact Author Email Address

vinicius.marini@ufsm.br

7. Copyright Statement

The authors confirm that they, and/or their company or organization, hold copyright on all of the original material included in this paper. The authors also confirm that they have obtained permission, from the copyright holder of any third party material included in this paper, to publish it as part of their paper. The authors confirm that they give permission, or have obtained permission from the copyright holder of this paper, for the publication and distribution of this paper as part of the ICAS proceedings or as individual off-prints from the proceedings.

References

[1] Petter Krus. Large language model in aircraft system design. In *34th Congress of the International Council of the Aeronautical Sciences, ICAS 2024*, 2024.

[2] Brian Johns, Kristina Carroll, Casey Medina, et al. AI systems modeling enhancer (AI-SME): Initial investigations into a chatgpt-enabled MBSE modeling assistant. *INCOSE International Symposium*, 34(1):1149–1168, 2024.

[3] Timm Teubner, Christoph M Flath, Christof Weinhardt, Wil Van Der Aalst, and Oliver Hinz. Welcome to the era of chatgpt et al. *Business & information systems engineering*, 65(2):95–101, 2023.

[4] Alejandro Pradas-Gomez, Petter Krus, Massimo Panarotto, and Ola Isaksson. Large language models in complex system design. In *DESIGN 2024 International design conference*, volume 4, pages pp. 2197–2206. Cavtat, Croatia: Design Society, 2024.

[5] Vinicius K. Marini, Jens Alfredson, and Petter Krus. Context of collaborative human-machine systems architecture design for enhanced functionality awareness and balanced command and control authority.

In *Proceedings of the 12th Swedish Aersopace Technology Congress - FT2025*. Stockholm, Sweden: Swedish Society for Aeronautics and Astronautics (FTF), 2025.

[6] Javier Cámara, Javier Troya, Lola Burgueño, and Antonio Vallecillo. On the assessment of generative AI in modeling tasks: an experience report with ChatGPT and UML. *Software and Systems Modeling*, 22(3):781–793, June 2023.

[7] John K. DeHart. Leveraging large language models for direct interaction with SysML v2. *INCOSE International Symposium*, 34(1):2168–2185, 2024.

[8] Louis Richard Timperley, Lucy Berthoud, Chris Snider, and Theo Tryfonas. Assessment of large language models for use in generative design of model based spacecraft system architectures. *Journal of Engineering Design*, 36(4):550–570, 2025.

[9] Petter Krus. Augmenting aerospace system design using large language models. In *Proceedings of the 12th Swedish Aersopace Technology Congress - FT2025*. Stockholm, Sweden: Swedish Society for Aeronautics and Astronautics (FTF), 2025.

[10] Vinicius Kaster Marini and Petter Krus. Synthesizing aircraft system specifications with document-managed Large Language Model outputs from one-shot system inquiry prompt chain. In *10th CEAS Aerospace Europe Conference, 28th AIDAA International Congress*, 2025.

[11] Petter Krus. Using large language models for fluid power system design. *JFPS International Journal of Fluid Power System*, 19(2):74–79, 2026.

[12] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[13] OpenAI. gpt-oss-120b & gpt-oss-20b model card, 2025.

[14] NVIDIA. Nemotron 3 super: Open, efficient mixture-of-experts hybrid mamba-transformer model for agentic reasoning, 2026.

[15] Ollama. Kimi k2.5 model card. Ollama model card, 2026. Accessed: 2026-06-04.

[16] Anthropic. Introducing Claude Sonnet 4.6. Anthropic Product Announcement, February 2026.

[17] Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the association for computational linguistics*, 12:157–173, 2024.