FT2025
SWEDISH AEROSPACE AT THE CROSSROADS
The 12th Swedish Aerospace Technology Congress
FT2025 in Stockholm
October 14-15, 2025

Context of collaborative human-machine systems architecture design for enhanced functionality awareness and balanced command and control authority

Vinicius Kaster Marini[1], **Jens Alfredsson**[2], **and Petter Krus**[3]

[1]*Department of Mechanical Engineering, Federal University of Santa Maria, Santa Maria/RS, Brazil*
E-mail:, vinicius.marini@ufsm.br

[2]*Aeronautics, Saab AB, Linköping, Sweden*
E-mail:, Jens.alfredson@saabgroup.com

[3]*Fluid Power and Mechatronic Systems, Linköping University, Linköping, Sweden*
E-mail:, petter.krus@liu.se

Abstract

Aircraft control systems and their interfaces have been subjects of significant research effort, yet there are gaps in human-machine collaboration related to functionality requirement awareness in the development process, and to command-and-control authority of the system being designed. While their command, control and integration characteristics allow the expectation of near-future autonomous mobility systems, the path in this direction has been difficult because of increased exposure to hazards deriving from emerging system functionality and integration issues, and reduced situation awareness of system operators. This contribution establishes the context of a systems architecture framework to reduce uncertainty about whether intended functionality will match the operating context and its circumstances, and to support developing context-matching systems architecture with balanced command-and-control in human-machine collaboration. The context hereby established involves human-machine collaboration between acting engineer and large language models as supportive agents, proceeding through the generation of system specifications, and then enabling human verification and authentication of design output considering the purpose requirements for system functionality. The contribution provides a pilot example declaring design intent through prompt questions. This is processed into design synthesis directives for a large language model, resulting in systems specification outputs from context towards requirements. The example helps assessing the reasons why human-machine collaboration in the design process and system design can support functional and situational awareness, information efficiency and operating effectiveness.

Keywords: 1, design process 2, large language models 3, human-machine collaboration

1 Introduction

The growing reliance on intelligent automation makes human-machine collaboration a critical goal to succeeding with systems deployment, by asserting complementary roles of people and automated systems upon executing a task [1]. In this context, the following issues deserve attention: firstly, that designed automation authority may be unsafe in time- and range-critical operation circumstances; and, secondly, that automation-enabled efficiencies could be too hard to reach by pilots and crew during operation [2].

When their implications reach back to the design process, design engineers find themselves accountable for solving the problem, while the complexity and intricacy of multi-domain integration across engineeering computing tools give rise to uncertainty on whether designed automation will behave as intended [3]. Then, the competences and architecture of next-generation automation systems require innovative engineering design resources that take advantage of novel technology [4] to enhance the design of aircraft automation systems onto fostering flight safety and improving operational efficiency, through effective collaboration with personnel.

There is a growing interest on how *large language models* (LLMs) could be useful and effective within systems engineering and design. Their *transformer*[1] working principle [5], with basis on very large-scale *neural networks*, can process natural language input and generate answers with person-like competence [6]. However, the nature of their processing activity and the quality of their output give rise to critique on their use in general [7], as well as to significant questions about whether they are fit to safety-critical engineering tasks within aerospace mobility [8].

Notwithstanding these issues, the potential capabilities of LLMs draw our interest on using them to navigate across automated aircraft systems design. Here, the departure point is the hypothesis that human-machine collaboration applies to both the design and the operation of aircraft systems. We set forward under the refining consideration that information technology and automation are significantly involved, and bear influence on the outcome of both design and operation activities related to modern aircraft systems.

Hence, this paper presents the concept of operations for a systems design framework that processes design intent declarations in a structured way and parses them to prompting a *large language model* (LLM) for early-phase system design tasks. This framework leverages *prompt engineering*[2] [9] onto the generation of structured system specifications. The staged several-shot prompt pattern allows instructing the LLM tool to enable the traceability and the connection of metrics that influence aircraft automation effectiveness to succeed at an air mobility task.

This approach uses an example case on the system design of aircraft intended for search-and-rescue missions, with regard to the specific case of flood disasters over riverside regions that result from 100-year rainfall [10]. Personal awareness of the author on this particular case - living in the region through the time of occurrence - and references to the mission category support the construction of a visual model, which works as a reference to structuring systematic and traceable chain-of-thought prompts. This approach is seen to reduce output uncertainty, which enables proceeding towards support to the synthesis of balanced human-machine collaboration.

The content is then organized as follows: this introduction is succeeded by context and related work. The methodology section presents the research approach with the architecture framework and supporting processes/tools, and is succeeded by the pilot example that illustrates the application of the framework. The results section discusses and evaluates the outcome from using the framework, succeeded by a broader discussion about implications and limitations of the approach. Lastly, the conclusion completes this paper with a summary and directions for future research.

[1] *Transformer* is a neural network architecture that focuses attention rather than recurrence and makes the principle that enables LLMs to interpret natural language content and generate conversation-like output.

[2] *Prompt engineering* is an emerging field of inquiry within information systems research that promotes a systematic approach to study and develop *natural language* interactions and instructions to elicit responses from LLMs.

2 Related work

The draw of LLMs to engineering communities reflects an evolved capability to solve engineering tasks from *natural language processing* (NLP) algorithms. While these could not yield problem-solving abilities until recently, language models evolved in attention capability through using the *transformer* architecture [5] in models such as *bidirectional encoder representations from transformers* (BERT) and the series of *generative pretrained transformer* (GPT) models. The exponential leaps in scale by the evolution of these language models enabled new content generation from their solving complex tasks - the so-called *emergent ability*. Yet the main draw in their functionality comes as developing LLMs such as GPT-3 could provide answers in conversation form after training [11].

The use of LLMs in aircraft systems design has potential to answer questions dearly held by industry in complexity-tackling and time-to-market of design processes [12], which other design automation approaches could not answer so far because many system design tasks are ill-suited for design optimization or KBE techniques alone. Because of concerns manifested in the governance and use of *artificial intelligence* (AI) in complex systems [13], as well as the use of LLMs in engineering activities such as assurance cases [8], the need for a closer look into human-machine collaboration [14] applies both to operating machines and processes to their design and development.

This section will look at the following viewpoints: the application of LLMs to generate system models and specifications; the application of LLMs to generate design task information such as requirements and assurance cases; and the state of the art in human-machine collaboration in design and operation environments.

2.1 Use cases of LLMs for model synthesis

This exploration viewpoint reflects the engagement of trials with LLMs in generating system specification models with significant completeness and maturity. Examples in this viewpoint look to use LLMs in interoperation with systems modelling frameworks to generate specifications, directly or through intermediate language parsing. Four of the examples worked within the aerospace domain, whereas other examples worked other complex trades such as autonomous vehicles and healthcare operations, with different prompting patterns and verification methods as shown in Table 1.

Tikayat Ray and colleagues use aircraft design requirements from certification standards, which they parse with LLMs and *boilerplate* text structures to requirement statements [15]. Ofsa and Topcu experiment with a structured few-shot prompt is associated to reference mission information to generate a list of mission stakeholders containing the description of their roles [16]. Then, Balu and colleagues approach the assembly of a *resource-augmented generation* (RAG)-framework including a publicly-available dataset for an automated driving perception system alongside instructions adopted from technical standards within the domain of vehicle automation [17].

Table 1: Comparative overview of LLM use cases for model synthesis.

Reference	Prompt pattern & LLM type	Auxiliary resources	Answer results	Validation & correction
Tikayat Ray et al. [15]	BERT over source data	Reference requirements in documentation	Processed requirements on *boilerplates*	Requirement grammar structure inspection with demonstration
Ofsa & Topcu [16]	Single few-shot to GPT-3.5	Reference mission statement with prompt	List of mission stakeholders with descriptions	Comparison of results across several prompts under same prompt-to-answer treatment
Balu et al. [17]	Two few-shot to GPT-3.5	Reference requirements from RAG database	Processed system design requirements	Comparison of results across some different prompt-to-answer treatments
Cámara et al. [18]	Single one-shot to GPT-3*	UML-model references from research team	Processed UML models specifications	Comparison of prompt trials and review and correction of resulting UML-models
García Alarcia et al. [19]	Single several-shot to GPT-4	Mission statements & OPM template from research team	Processed OPM mission specification	Co-sine similarity verification between resulting mission models and reference model
Crabb & Jones [20]	Single one-shot to GPT-3.5	Scenario briefing promp reference from research team	XMI requirement and system models to MBSE tool	Expert scoring review across participant cohorts and applicable corrections
DeHart [21]	Four one-shot to GPT-4	SysML and Python code models from researcher	Simulation of model coupled to system requirements	Demonstration of effort and complexity against usual practice
Johns et al. [22]	Three several-shot to GPT-4T	JSON-model references from research team	SysML list of requirements and block (bdd, ibd) models	Expert qualitative review of resulting man-made and generated system models
Von Heissen et al. [23]	Two several-shot to n/d[3]	JSON-model references from research team	SysML-based requirement, function, logical specifications	Expert scoring review of resulting system models without correction
Makatura et al. [24]	Two or more several-shot to GPT-4	FeatureScript and OpenJSCAD models from research team	Generated three-dimensional body geometries	Demonstration of effort and complexity against usual practice
Pradas-Gomez et al. [25]	Two or more several-shot to GPT-4	UML model and Python-KBE templates	Generated UML-architecture and 3-D geometries	Expert qualitative review of resulting UML and KBE about model capability

Cámara and colleagues experiment with unstructured prompting to find out LLM capabilities, and proceeds to systematic prompting with support of reference *unified modelling language* (UML) models to generate new ones [18]. García Alarcia and colleagues prompt the LLM to read mission statement inputs along a generic *object-process methodology* (OPM) model [26] template, and generate specific technical requirements for a similar space mission [19].

Crabb and Jones experiment with the synthesis of a requirements list along a use case diagram for a health care system to generate *XML metadata interchange* (XMI) models [20]. DeHart explores with processing a specific problem for use in a MBSE environment to generate python language code conveying a system component model, by using the structural analysis of a beam as example [21]. Johns and colleagues work with few-shot prompts to GPT-4 Turbo along with a set of *systems modelling language* (SysML) model templates for a rocket-propelled aircraft: a system hierarchy, a requirements list, and an internal block diagram [22]. Von Heissen and colleagues make an application plugin that structures a single multi-shot prompt with a few-shot user request field and in-application model templates available for each system specification - requirement, function and logical [23].

Makatura and colleagues explore different use cases for LLMs related to designing part geometries and production processes. This involves the testing and verification of prompt patterns and the demonstration of design possibilities in making geometry, configuring design parameters, design-for-manufacturing, along with structural analysis [24]. Pradas-Gomez and colleagues explore use cases for LLMs in the design of complex systems, namely the generation of UML system architecture models and the synthesis of *knowledge-based engineering* (KBE)-based part geometry models. Their approach uses systematic prompting technique adapted to each case, for the prompts specify the expected elements of the subsystem architecture and the part geometry models. The coordination between professional engineers and LLM agents requires attention [25].

2.2 Use cases of LLMs for design activities

This exploration viewpoint reflects the engagement of trials with LLMs performing design activities such as requirements engineering, safety assurance analyses, and geometry development, looking to explore and test their suitability. This viewpoint shows examples shown in Table 2, which look to investigate the use LLMs on their potential to fulfilling a

Table 2: Comparative overview of LLM use cases for design activities.

Reference	Prompt pattern & LLM type	Target task & design domain	Auxiliary resources	Answer results & validation
Norheim et al. [27]	No prompting found in work	Requirements engineering in aerospace	Available research datasets on requirements	Framing and benchmarking of requirement studies for LLM use
Nouri et al. [28, 29]	Multiple several-shot to custom GPT-4	Hazard analyses and synthesis of safety requirements	Document specifications and design figures from RAG database	Expert review indicates LLM as good for preliminary use, outputs need review
Geissler et al. [30]	Multiple several-shot to GPT-3.5T	Failure scenario estimation in safety engineering	Reference models, information and code from RAG database	Comparison across different prompt-to-answer treatments deemed satisfactory by team
Odu et al. [31]	Multiple one-shot to GPT-4o	Safety assurance argument synthesis in safety engineering	Processed GSN patterns and software engineering guides	Similarity of prompt outputs indicates difficulties with mixed-cardinality relationships
Sivakumar et al. [32]	Single several-shot to GPT-4	Mission statements & OPM template from research team	GSN syntax and semantic instructions from engineering guides	Co-sine similarity verification indicates semantic understanding with assumptions and rationale errors
Qi & et al. [33]	Multiple several-shot to GPT-4	Safety engineering of automotive and energy automated systems	Previous STPA studies of automotive and energy systems	Verification of added information indicates system knowledge and prompting as key aspects
Charalampidou et al. [34]	Several one-shot to GPT-4	Safety engineering of automated unmanned aerial vehicle	References about the system architecture of the vehicle	Verification of generated scenarios upon several-shot prompting indicates tendency to exhaustion

design task, or about how LLMs could be set deliver work products for engineering tasks to an expected degree of work quality. Here, our focus is set on early design tasks with focus on requirements and safety engineering, both aspects of significant complexity that draw significant resources from aerospace engineering teams.

Norheim and colleagues look to assess the use of LLMs on requirements engineering tasks. Their approach frames the use of language models on a requirements engineering process regarding progression from informal statements to actual system components [27]. Nouri and colleagues experiment the use of LLMs to carry out a *hazard analysis and risk assessment* (HARA) task to fulfil a functional safety requirement. The approach involves a first exploration phase with testing prompt statements, and a second phase involved the setup of a prompt pipeline for the LLM routine and generate the work product on an automated automotive functionality feature [28].

Geissler and colleagues experiment with processing a primary prompt onto multiple several-shot prompts eliciting information from a RAG database including the intended system model, safety engineering documentation, and analytical function codes for critical path, fault propagation, single-point failure finding, to process an answer [30]. Odu and colleagues approach the use of LLMs work form the extraction of safety case patterns from literature regarding the cases of airborne collision avoidance, and other submarine, medicine and software systems to generate safety cases [31]. Sivakumar and colleagues explore the possibilities with safety cases by starting from a prompt pipeline to GPT-4 with rule-setting questions followed by safety-case generation requests to elicit the safety case output [32].

Qi and colleagues approach an automated emergency brake system [35] as basis for using LLMs with *systems-theoretic process analysis* (STPA) [36] on different query approaches regarding the communication direction and the prompt recurrence, first to model the control structure and then to identify the unsafe control actions. The prompt experiment was performed in one-way single-shot, one-way multi-shot, and two-way multi-shot including stepwise review [33]. Charalampidou and colleagues consider the STPA method on a fully automated *unmanned aerial vehicle* (UAV) with a phased query approach prompting LLMs along general guidelines of the method, from defining the system to obtaining safety specifications against unsafe control actions thereby identified. The method involves placing reference documents about the system specification, and text-based explanations about the system and its actions [34]. To tackle the cognitive effort, Geissler and colleagues [30] suggest delegating questions to an LLM agent from a single prompt, while Qi and colleagues [33] find a staged prompting strategy yields better results.

2.3 Holistic human-machine collaboration

The arrival of *artificial intelligence* (AI) to mainstream technical systems and especially to complex systems such as in aerospace calls the attention to the capabilities of this technology in support to automation, to the extent that it can enable autonomy in some applications [37]. When AI emerges as a major influence to the development of systems that can operate in an increasingly independent manner, Human-AI teaming addresses the need to ensure coordination and collaboration between people and machines [38]. When used well, AI components have potential to provide enhanced cognition (more information bandwidth) and complex intelligence to people and teams (better reasoning and decision support).

Nevertheless, AI-based systems have shown behaviour that calls for increased oversight and the need to enable the design of collaborative human-machine systems. The systems approach is evolving from a 'control loop' to a 'collaborative loop' where personnel and agents can work together in *joint cognitive systems* [39]. This also involves the need for a more dialogic and flexible approach to systems supporting human-machine collaboration, in systems design as well as in the designed system that will operate [40]. This collaboration is needs planning, structuring and oversight to preserve human authority over critical decisions, while leveraging computational capabilities of efficiency and scalability.

Systems design has seen the implementation of AI techniques well before operations, by the means of design automation techniques, which can help the design process by handling complexity, exploring alternatives, and maintaining consistency across specifications [41, 42]. However, while design automation research succeeds in its application to market, there is difficulty to replicate more complex and complicated propositions with accumulated knowledge, especially in technological systems with sociotechnical components [43] where mathematical and logical methods see limitations on their suitability to operation and system design tasks.

While the implementation of *neural networks* (NNs) can provide better adaptability, optimization-focused design automation methods still remain restricted to specific, niche design problems [44]. Other use of NNs regard the focus on geometry design optimization through reinforcement learning, where a reward function guides the algorithm to optimize specific geometric aspects [45]. However, cross-domain design modelling and problem-solving support (actor > activity > requirement > function > component) has been eluding researchers and engineers looking to use such models.

The draw of LLMs to engineers stems from their improved capability over other NLP algorithms to solve complex tasks: whereas the BERT had 330M parameters, GPT-2 model had 1.5B parameters, and the next-generation GPT-3 was developed to 175B parameters. These factors enabled evolving LLM capabilities towards generating new content upon the solution of complex tasks - the so-called *emergent ability*. Yet the main draw in their functionality comes as developing LLMs such as GPT-3 could provide answers in conversation form after training [11]. The effective use of LLMs requires coordination between professional engineers and computing agents for their successful [25].

Here, *Human-autonomy teaming* (HAT) involves the sharing of activities among personnel and autonomous agents in interdependent work, and comprises key research areas such as the definition of agent characteristics, the human-machine team composition, and means and protocols of communication [46]. Effective HAT requires means of maintaining *situation awareness* (SA) for personnel and the AI cobots that share the task, encompassing task assignment, actor and agent roles, and teamwork SA [38]. A cognitive perspective on Human-Autonomy command emphasizes the need for flexibility and versatility about executing the intended activities, such as when managing various assignments at hand [47].

3 Methodology

This section outlines the development approach employed for the proposed systems architecture framework and provides a contextual view on the how the framework is developing and on the selection of an example case to demonstrate it in first hand [10]. This framework is ongoing development through beginning with a manual approach and proceeding to automate the generative process to the use of *large language models* (LLMs). To frame the demonstration approach in this paper, this section firstly presents the framework elements with support of *visual tools*, model-parsing applications and *large language models* (LLMs); then, it describes the chain-of-thought prompting approach around a modular strategy and the policy definition that enables it; by last, the text describes how prompting is structured across phases and system aspect blocks.

3.1 Research development method

The research approach is an exploratory study about the capability of LLM models to enable human-machine collaboration by the design process in working towards better human-machine collaboration in operation. The stepped approach started with a literature review to help form the picture about the state of development of LLMs and their use in engineering and design. Then, it evolved to the second step with unstructured prompting with LLM models such as GPT-4, GPT-4o, and GPT-4.1 about generic system development cases in a narrow system scope.

The third step of the approach is the structured prompting about the generic search-and-rescue mission through manual typing over the chatbox where a partial scope of the design assignment is tried with testing the intended conversation functionalities. a fourth step involved the development of the CXL-Markdown prompt parser, for enabling the intended visual approach; then, the last step is the use of the full prompting functionality to the LLM tools on the intended mission case, which then takes the form of a case study.

This paper is going to focus the display of the results from this last step, while the perceptions from the previous steps may be useful for discussing the general use of the LLMs.

3.2 Resulting framework composition

The framework has three major components: the first one is the use of a visual approach to define the scope of inquiry and the prompting pattern. This visual approach can be implemented through an in-house application or through an external model provider. In its current form, the visual approach works through the use of CmapTools [48], whose user interface shown in Figure 1 provides and environment for the construction of knowledge models in so-called concept maps. For this demonstration, CmapTools enables defining a visual diagram reflecting a prompt intent, and its export onto a CXL-format derived from *extensible markup language* (XML). This file carries the information regarding the concept nodes, the linkphrases and the connections between them, which will be read onto LLM prompts.

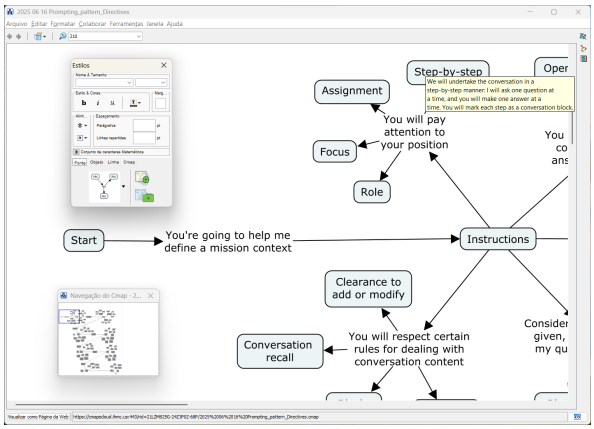

Figure 1: The CmapTools interface.

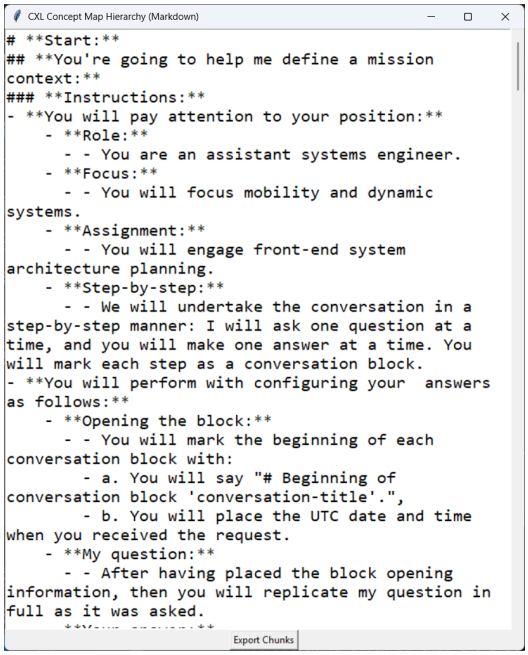

Figure 2: The CXL-Markdown prompt sequence result

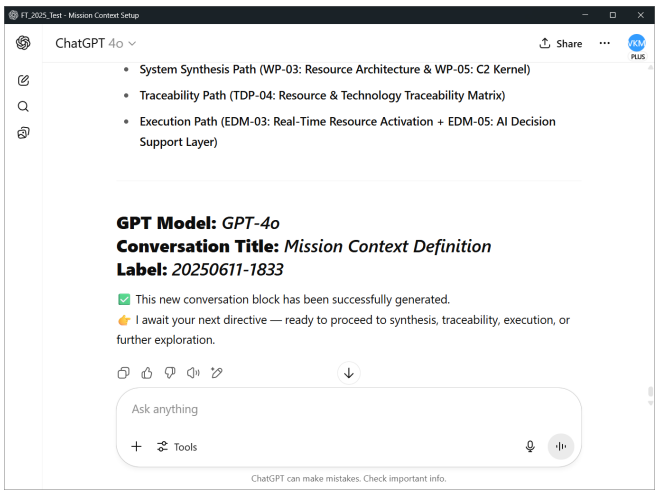

Figure 3: The ChatGPT chatbox

The second major component is a model-parsing Python routine that opens the CXL-file and reads the XML-code out of it, so that to convert the information in concept blocks, link-phrases and their comments into a topic sequence and then to the *Markdown* language for the purpose of making the prompt chain. This application reads the entire CXL-model through with following its nodes and links, and parses it onto a multiple several-shot prompt chain. Being directly derived from the visual model, the resulting prompt chain shown in Figure 2 will inherently carry the system design formulation phases as sets of characteristics; these characteristics are divided into few several-shot prompt chunks that make specific aspects of the intended system by the press of a button.

At this development level, the third component is the chatbox application for the LLM, hereby shown in Figure 3. Here, three ChatGPT models shall be used for comparison: *ChatGPT-4o* [49], *ChatGPT-o4-mini* [50] and *ChatGPT-4.1*. These models are selected for their specific characteristics: *GPT-4o* is the general-purpose model intended for most tasks, *GPT-o4-mini* is a reasoning-optimized model with higher throughput, which is useful for technical problem-solving, and *GPT-4.1* is the most capable of them, so far. Preliminary tests before these runs have shown that different LLM-models, can yield different answer patterns with different tones and emphases depending on the selected model.

The LLM-algorithm is operated through the authors' own ChatGPT user account, whose access enables reaching the GPT-chatbox where prompt patterns will be set. While some context information can permeate between conversations, specific instruction is given so that the system 'forgets' other conversations. By receiving the prompt chunks set onto the chatbox, the algorithm makes a conversation record; the conversation is continued with a sequence of prompt chunks whose answers will be accumulated through the conversation. As the GPT-chatbox enables scrolling about and through the conversation, the GPT-chatbox also works as answer review interface for the ongoing system development conversation.

3.3 Planning the conversation

This part is derived from the following research steps: firstly an unstructured prompt phase with asking questions about generic examples, where a chain-of-thought pattern is used with several prompt requests; then a structured prompt phase with manual input into the chatbox, and lastly the visual prompt planning approach. The conversation is firstly planned in regard to setting a conversation policy with defining the role of the LLM and the rules of engagement for the conversation, where each prompt will be always additive.

The conversation policy section has two concepts around which all rules of engagement are defined: first comes the 'Instructions' concept with a visual shown in Figure 4 and then the 'About the conversations' concept. The instructions part defines the role and the assignment of the LLM agent, including answer composition, answer referencing to enable traceability and update policy. The second part sets the conversation up in terms of direct prompts and derived prompts, their sequence and their relationships to the intended inquiry.

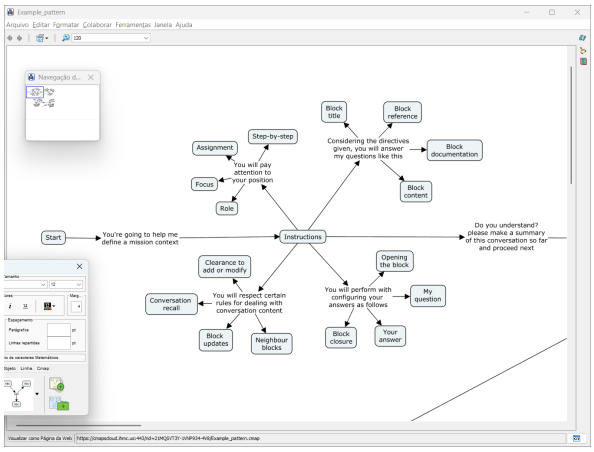

Figure 4: The first part of the policy, in visual form.

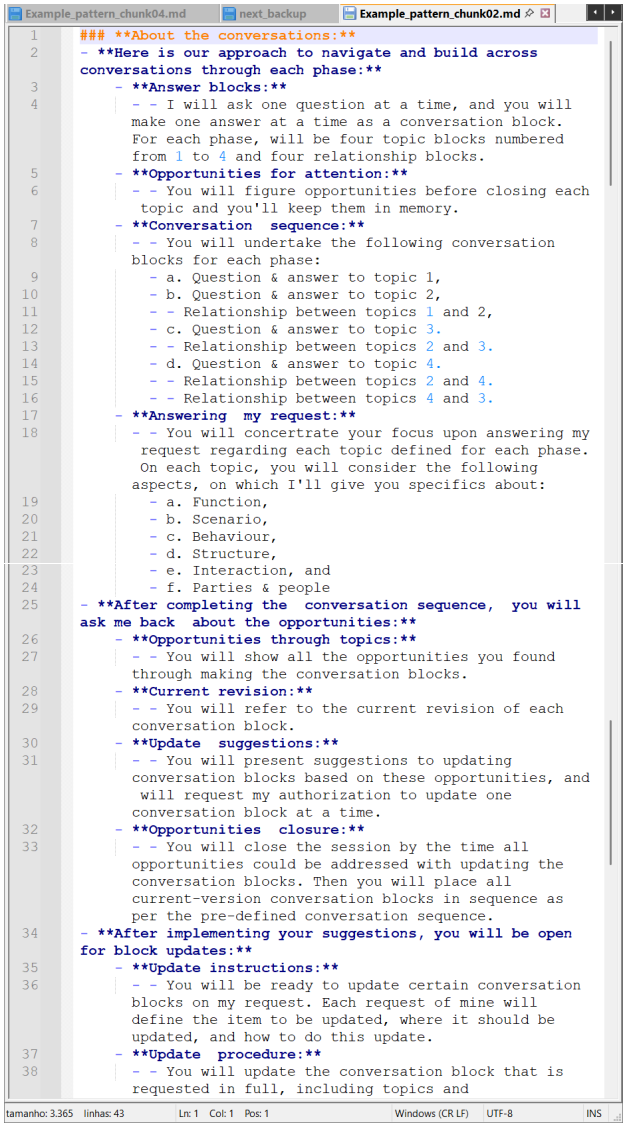

Figure 5: Excerpt of the policy in markdown.

Here, direct prompts work specific aspects at certain stage of system development, and derived prompts define specific relationships between these aspects. It also defines the opening for opportunities to generate system design content, and two sessions after completing the intended system aspects: an ask-me-back session for augmentation possibilities and an update session for the correction of inaccuracies. Each following section has two sets of questions: the first on the aspects for direct prompting, and the second on generating the relationship between the directly-prompted ones.

As from the 'About the conversations' concept in the previous section, the algorithm receives the prompts as defined in Figure 5 on which it shall perform an ask-me back session and set a waiting state for review and update requests. This process will repeat for all other sections until completing the scope of system specification inquiry in this demonstration.

The ask-me-back and the update opening reflect the steps planned for human-machine collaboration. While the algorithm is set to give suggestions about opportunities it figures about, the last step is the human-requested update. The conversation policy definition is necessary to make sure the LLM will do the marking and chunking of conversation blocks and will distinguish between different conversation sections, and will also take note of revision number for each conversation block that is defined.

4 Pilot example

As mentioned in the introduction, the approach shall engage the development of a system architecture for the search-and-rescue mission type, with considering the specific case of victim recovery from flooding by riverside communities after a 100-year rainfall [10]. In reality, this episode was characterized by 420 mm of rain between the 29th of April and the 5th of May, 2024, in the centre region of the southern state of Rio Grande do Sul in Brazil, and recurred from rainfall intensity only witnessed back in 1941. The consequences of the rainfall are in display by Figure 6, by the city of Cruzeiro do Sul at a distance of 140 km from Porto Alegre that is the state capital. The flood made 57 fatalities by the first five days, and a total 187 fatalities in total [52].

The prompt chain evolution will be presented in regard to its context and requirement definitions, as the conversation policy is described in 3.3. Each section is generated after a specific prompt of its own, which is chunked off from the whole chain with support of the CXL-to-markdown model parser. With the prompts chunked, they are copied and parsed onto the GPT chatbox window.

4.1 Context and requirement definitions

With Figure 7 showing the full visual model prior to prompt conversion, it displays the conversation policy by the elements in the uppr half, whereas the elements corresponding to the general context definition and the requirement definition are situated by the bottom of the figure. The context definition is the front-end for prompting about the *search-and-rescue* (SAR) mission, and has its aspects requested in general form.

Figure 6: Flooding in Cruzeiro do Sul, RS after the 100-year rainfall [51].

As the exercise involves no RAG to getting external data, the case selection involved reasonable information from the public domain.

While the first pipeline of context definition works for 'priming' the LLM to communicate about general aspects of search-and-rescue missions, the requirement definition that comes in second works differently. Because the prompts were chunked by the means of the algorithm, it was possible to set a preamble prompt to the algorithm in regard to the requirement definition stage, requesting information about the tragedy. The requirement definition prompt chain is started with a preamble prompt shown in Figure 8 requesting the LLM to be aware of the flood case and consider it for the questions ahead regarding requirement definition.

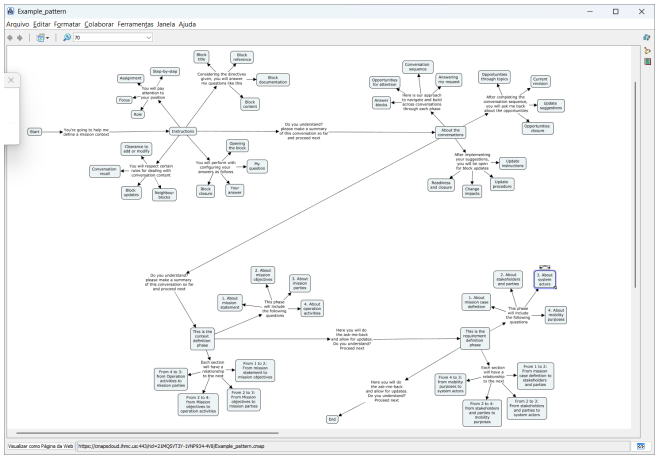

Figure 7: The full visual route before conversion.

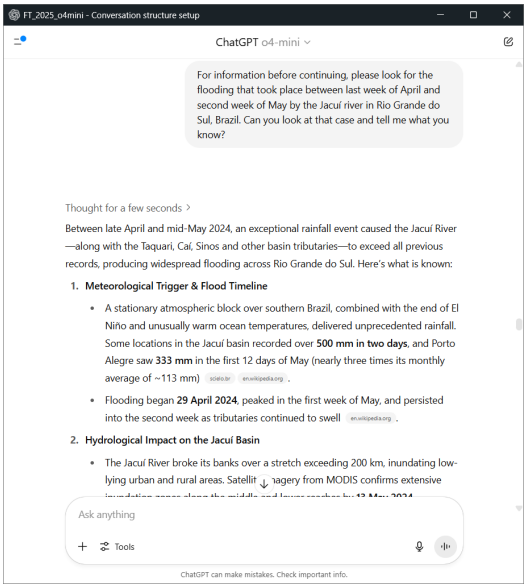

Figure 8: Incorporation of reference into conversation.

After the preamble prompt, a manual request is set so the model confirms it understood the data and includes the data on-record as a new conversation block as shown in Figure 9. Then the prompts are followed on as usual, with the same information, and allowing the LLM to use the data as its context of its reasoning and inquiry. The inquiry involves the identification of mission aspects that need to be taken into account in search-and-rescue effort aiming at the recovery of victims in riverside areas, considering flooding rivers with high kinetic energy.

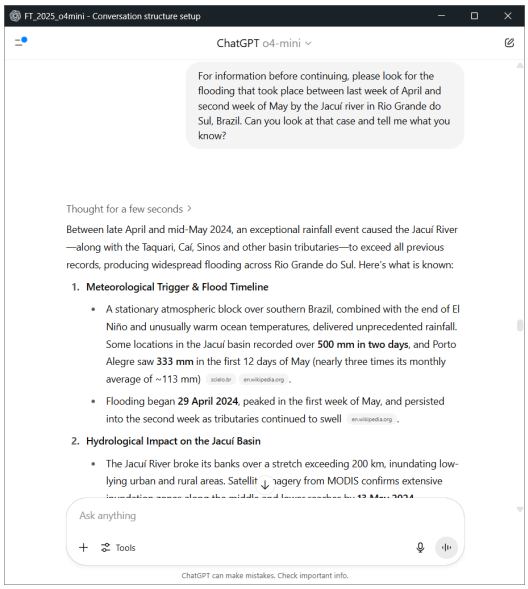

Figure 9: Incorporation of reference into conversation.

4.2 Air mobility and aircraft considerations

While the prompt chain considers the mission design in first place, here the model makes references to expectations from air mobility in such a case. By the end, a manual request is set about references to 'air mobility' and 'aircraft', and the algorithm returns a summary answer that contains the references it made to the assumptions, requirements and means of implementation it mentioned throughout the conversation. In each of the three versions tested - GPT-4o, GPT-04mini and GPT-4.1 - the script recalled the references it made to air mobility and aircraft throughout all conversation blocks.

Each of the scripts has a particular way of reporting. GPT-4.1 as shown in Figure 10 focuses the verbal description of use cases, payloads and typical operation patterns, aircraft crew and personnel - it pointed out at cross-training between UAVs and helicopters, then the capabilities that stand out and the expectations of stakeholders, observations about human-machine teaming and their monitoring needs by last.

GPT-4o follows a similar pattern to that from GPT-4.1, with focus on general aspects, key parties and aircraft roles; then it proceeds to the references by the requirement definition phase, and defines general roles and capabilities of aircraft with identifying the air force as main stakeholder, and by last it identifies a mobility cluster - possibly the local airbase at Santa Maria - along matching UAVs to use cases of their advantage.

Then, GPT-o4mini mentions categories of use cases, supportive technologies, and roles and responsibilities by parties; it proceeds to identifying aircraft and devices for air domain activities; it lists parties' expectations and then useable types of aircraft. The o4mini model has also identified references to ai mobility and aircraft in relationships, where early-defined capabilities appear in significant number. At the end, it identifies aircraft types and dscribes the most promising roles for their use.

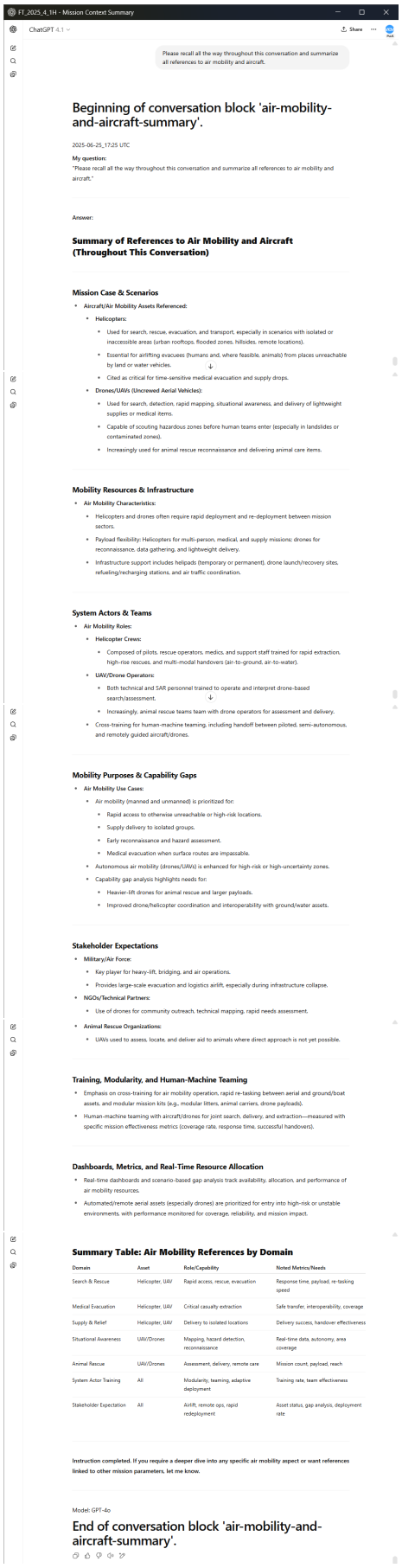

Figure 10: Aircraft reference summary by GPT-4.1.

5 Discussion and future work

This paper has approached the context of collaborative human-machine systems architecture design with a visual approach for planning prompts to *large language models* (LLM) to engage in systems engineering until systems architecture. This paper has reviewed the uses of LLMs in the design process so far, and then presented the fundamental components to the framework: a visual modelling application, a model parsing application and a prompt throughput device, here being, respectively, the CmapTools applicaiton, a XML-Markdown parser application and the GPT-chatbox from an accessible user account.

While some other works in literature also consider the parsing of models, this approach is about making a visual path to prompting for the purpose of buildling foundational conversations, that served in this example to collect relevant references about use cases and intended capabilities of aircraft and can serve to generate more elaborate system models as with the time investment. Such a prompt route enables a visual approach to prompting because of reading a visual graph onto a markdown text that can be used for prompting. Using three parallel conversations with three different models brings some insight about how to develop the use of LLMs as design assistants.

The first aspect of human-machine collaboration is the establishment of a conversation policy. This conversation is additive, and does not involve erasing prior references. Nevertheless, models with shorter context windows may suffer if the conversation proceeds at once. Therefore, splitting prompts and distributing the generative language processing onto shorter conversations is the way to develop, which enlightens our methodology choices in proceeding with development. This paper has been done with awareness about the possible failure modes by LLM models, and it is for that reason that the conversation policy looks to establish traceable and revisable conversation blocks.

The path to ensuring acceptable reliability of LLMs as design assistants has two ways: one, by LLM developers, who can expand benchmarks such as context window and maximum token size to enable a higher throughput capacity. Both GPT-4o and GPT-o4mini choked in few moments of their conversations due to the quantity of text they were accessing within the conversation. This choking revealed itself by GPT-o4mini ignoring a part of a prompt requested at it and then accepting a repeated prompt request with no issue- yet processing it in a separate process window rather than in the mein conversation interface.

The other way is about the distribution of prompting workload. When the prompts may get too complex, the use of language processing framework techniques such as *retrieval-augmented generation* (RAG), and prompt distribution to agents relieves the context window of LLM processors and improves the rational use of tokens, meaning that a more advanced way of processing information becomes accessible. At the same time, less computer resources and less energy could be spent if reducing the strain on LLMs.

Much of the prompting accuracy comes from... prompting, which means prompt engineering is another area to look at closely. The quest for trustworthy and reliable automation begins at our fingers, both in the design process context as well as in the operation context. There is now a foundation from which depat and to pursue these improvements to build upon. While GPT-models can trace and make individual conversation models, it shall be an external application to include an undersigning module or a user-bound SHA-code to authenticate the professional partner of the LLM model, meaning that there is someone who reviewed the LLM model and is ready to stand by it.

6 Conclusion

This paper has presented the demonstration of a human-machine collaboration for system architecting, looking to enhance functional awareness by engineers, and to find improvements in balanced control authority. The establishment of information about its mission assignment with basis on information about a real case brings use cases, capacity ranges, specific capabilities and parties expectations to form early design requirements about the design of one or more novel aircraft towards the mission case thereby identified.

The balance of command and control and the increased awareness to functionality come together with developing the prompt pattern, both in terms of distribution and allocation of processing resources and with focus on the relevant information to aircraft innovation processes where design engineers and operator representatives may face together.

7 Acknowledgments

This work undergoes through sponsorship by CISB/CNPq aerospace development program under the grant number 200944/2024-0, in execution with Linköping University in Sweden and by qualification leave granted by UFSM in Brazil.

References

[1] Patrick Millot and Guy A. Boy. Human-machine cooperation: a solution for life-critical systems? *Work*, 41(S1):4552–4559 pp., March 2012.

[2] Mary L. Cummings. Automation bias in intelligent time critical decision support systems. In Don Harris, editor, *Decision making in aviation*, pages 289–294. London: Routledge, 2017.

[3] Martin Törngren and Paul T. Grogan. How to deal with the complexity of future cyber-physical systems? *Designs*, 2(4):40 pp., 2018.

[4] Hualong Chen, Yuanqiao Wen, Man Zhu, Yamin Huang, Changshi Xiao, Wei Tao, and Axel Hahn. From automation system to autonomous system: An architecture perspective. *Journal of Marine Science and Engineering*, 9(6):645, 24 pp., 2021.

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[6] Timm Teubner, Christoph M Flath, Christof Weinhardt, Wil Van Der Aalst, and Oliver Hinz. Welcome to the era of chatgpt et al. the prospects of large language models. *Business & Information Systems Engineering*, 65(2):95–101 pp., 2023.

[7] Michael T. Hicks, James Humphries, and Joe Slater. Chatgpt is bullshit. *Ethics and Information Technology*, 26(2):1–10 pp., 2024.

[8] Mallory S. Graydon and Sarah M. Lehman. Examining proposed uses of LLMs to produce or assess assurance arguments. Technical memorandum, NASA/TM-2025-0001849, National Aeronautics and Space Administration, March 2025.

[9] Douglas C. Schmidt, Jesse Spencer-Smith, Quchen Fu, and Jules White. Towards a catalog of prompt patterns to enhance the discipline of prompt engineering. *ACM SIGAda Ada Letters*, 43(2):43–51 pp., 2024.

[10] Iury T. Simoes-Sousa, Carolina M. L. Camargo, Juliana Távora, Agata Piffer-Braga, J. Thomas Farrar, and Tamlin M. Pavelsky. The May 2024 flood disaster in southern Brazil: Causes, impacts, and swot-based volume estimation. *Geophysical Research Letters*, 52(4):e2024GL112442,, 2025.

[11] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, et al. A survey of large language models. *ArXiv*, abs/2303.18223, 2023.

[12] A. L. Smith and Nicholas S. Bardell. A driving need for design automation within aerospace engineering. In *11th Australian International Aerospace Congress*. Melbourne, Australia: Royal Aeronautical Society, Australian division, 2005.

[13] Mary L. Cummings. Identifying ai hazards and responsibility gaps. *IEEE Access*, 13:54338–54349, 2025.

[14] Mica R. Endsley. From here to autonomy: Lessons learned from human–automation research. *Human Factors*, 59(1):pp. 5–27, 2017.

[15] Archana Tikayat Ray, Bjorn F. Cole, Olivia J. Pinon Fischer, Anirudh P. Bhat, Ryan T. White, and Dimitri N. Mavris. Agile methodology for the standardization of engineering requirements using large language models. *Systems*, 11(7), 2023.

[16] Max Ofsa and Taylan G. Topcu. An empirical exploration of chatgpt's ability to support problem formulation tasks for mission engineering and a documentation of its performance variability. In *22nd Annual Conference on Systems Engineering Research*. Long beach, CA: Stevens Institute of Technology and University of South California, 2025.

[17] Balahari Vignesh Balu, Florian Geissler, Francesco Carella, Joao-Vitor Zacchi, Josef Jiru, Nuria Mata, and Reinhard Stolle. Towards automated safety requirements derivation using agent-based rag. *Proceedings of the AAAI Symposium Series*, 5(1):299–307, May 2025.

[18] 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.

[19] Ramón M. García-Alarcia, Pietro Russo, Alfredo Renga, and Alessandro Golkar. Bringing systems engineering models to large language models: An integration of opm with an llm for design assistants. In *Proceedings of the 12th International Conference on Model-Based Software and Systems Engineering - MBSE-AI Integration*, pages 334–345. INSTICC, SciTePress, 2024.

[20] Erin . Crabb and Matthew T. Jones. Accelerating model-based systems engineering by harnessing generative ai. In *2024 19th Annual System of Systems Engineering Conference (SoSE)*, pages 110–115, 2024.

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

[22] Brian Johns, Kristina Carroll, Casey Medina, Rae Lewark, and James Walliser. Ai systems modeling enhancer (ai-sme): Initial investigations into a chatgpt-enabled mbse modeling assistant. *INCOSE International Symposium*, 34(1):1149–1168, 2024.

[23] Oliver Von Heissen, Fabian Hanke, Isaac Mpidi Bita, Aschot Hovemann, Roman Dumitrescu, et al. Toward intelligent generation of system architectures. *Proceedings of NordDesign 2024*, pages 504–513, 2024.

[24] Liane Makatura, Michael Foshey, Bohan Wang, Felix Hahnlein, Pingchuan Ma, Bolei Deng, et al. How can large language models help humans in design and manufacturing? *ArXiv*, abs/2307.14377, 2023.

[25] 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.

[26] Dov Dori. *Object-Process Methodology: A Holistic Systems Paradigm; with CD-ROM*. Springer Science & Business Media, 2002.

[27] Johannes J. Norheim, Eric Rebentisch, Dekai Xiao, Lorenz Draeger, Alain Kerbrat, and Olivier L. de Weck. Challenges in applying large language models to requirements engineering tasks. *Design Science*, 10:e16, 2024.

[28] Ali Nouri, Beatriz Cabrero-Daniel, Fredrik Torner, Hakan Sivencrona, and Christian Berger. Welcome your new ai teammate: On safety analysis by leashing large language models. In *Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering - Software Engineering for AI*, CAIN '24, page 172–177, New York, NY, USA, 2024. Association for Computing Machinery.

[29] Ali Nouri, Beatriz Cabrero-Daniel, Fredrik Törner, Håkan Sivencrona, and Christian Berger. Engineering safety requirements for autonomous driving with large language models. In *2024 IEEE 32nd International Requirements Engineering Conference (RE)*, pages 218–228. IEEE, 2024.

[30] Florian Geissler, Karsten Roscher, and Mario Trapp. Concept-guided llm agents for human-ai safety codesign. *Proceedings of the AAAI Symposium Series*, 3(1):100–104, May 2024.

[31] Oluwafemi Odu, Alvine B. Belle, Song Wang, Segla Kpodjedo, Timothy C. Lethbridge, and Hadi Hemmati. Automatic instantiation of assurance cases from patterns using large language models. *Journal of Systems and Software*, 222:no.112353, 2025.

[32] Mithila Sivakumar, Alvine Boaye Belle, Jinjun Shan, and Kimya Khakzad Shahandashti. Exploring the capabilities of large language models for the generation of safety cases: the case of gpt-4. In *2024 IEEE 32nd International Requirements Engineering Conference Workshops (REW)*, pages 35–45, 2024.

[33] Yi Qi, Xingyu Zhao, Siddartha Khastgir, and Xiaowei Huang. Safety analysis in the era of large language models: A case study of stpa using chatgpt. *Machine Learning with Applications*, 19:no. 100622, 2025.

[34] Stavroula Charalampidou, Apostolos Zeleskidis, and Ioannis M. Dokas. Hazard analysis in the era of ai: Assessing the usefulness of chatgpt4 in stpa hazard analysis. *Safety Science*, 178:no. 106608, 2024.

[35] Liangliang Sun, Yan-Fu Li, and Enrico Zio. Comparison of the hazop, fmea, fram, and stpa methods for the hazard analysis of automatic emergency brake systems. *ASCE-ASME J Risk and Uncert in Engrg Sys Part B Mech Engrg*, 8(3):no. 031104, 10 2021.

[36] Nancy G. Leveson. Safety analysis in early concept development and requirements generation. *INCOSE International Symposium*, 28(1):441–455, 2018.

[37] Carlos C. Insaurralde. Intelligent autonomy for aerospace engineering systems. In *2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC)*, pages pp. 1–10, 2018.

[38] Mica R. Endsley. Supporting human-ai teams:transparency, explainability, and situation awareness. *Computers in Human Behavior*, 140:no. 107574, 2023.

[39] Erik Hollnagel and David D. Woods. *Joint cognitive systems: Foundations of cognitive systems engineering*. CRC press, 2005.

[40] Guy A. Boy. An epistemological approach to human systems integration. *Technology in Society*, 74:102298, 2023.

[41] Panos Y. Papalambros. The optimization paradigm in engineering design: promises and challenges. *Computer-Aided Design*, 34(12):939–951, 2002.

[42] Eugen Rigger, Kristina Shea, and Tino Stanković. Method for identification and integration of design automation tasks in industrial contexts. *Advanced Engineering Informatics*, 52:101558, 2022.

[43] New perspectives on Design Automation: Celebrating the 40th anniversary of the ASME Design Automation Conference. *Journal of Mechanical Design*, 137(5):050301, 05 2015.

[44] Niccolo' Batini, Niccolo Becattini, and Gaetano Cascini. Turbomachinery design: checking artificial neural networks suitability for design automation. In *ICED 23 The 24th International Conference on Engineering Design*, volume 3, page pp. 3651–3660. Design Society: Bordeaux, France, 2023.

[45] Fabian Dworschak, Sebastian Dietze, Maximilian Wittmann, Benjamin Schleich, and Sandro Wartzack. Reinforcement learning for engineering design automation. *Advanced Engineering Informatics*, 52:101612, 2022.

[46] Thomas O'Neill, Nathan McNeese, Amy Barron, and Beau Schelble. Human–autonomy teaming: A review and analysis of the empirical literature. *Human Factors*, 64(5):904–938, 2022.

[47] Arne Norlander. Human-autonomy command (hac): Directing and coordinating human-machine systems as cognitive capabilities. In *2024 IEEE International Conference on Recent Advances in Systems Science and Engineering (RASSE)*, pages 1–9, 2024.

[48] Alberto J Cañas and Joseph D Novak. Concept mapping using cmaptools to enhance meaningful learning. In *Knowledge cartography: Software tools and mapping techniques*, pages 23–45. Springer, 2014.

[49] OpenAI. Gpt-4o system card, 2024. Accessed: 2025-06-23.

[50] OpenAI. Introducing openai o3 and o4-mini, 2025. Accessed: 2025-06-23.

[51] Marcio Steiner. Flooding in cruzeiro do sul 2024, 2024.

[52] Wikipedia contributors. 2024 rio grande do sul floods, 2025. Accessed: 2025-06-25.