Ontology-Driven LLM Assistance for Task-Oriented Systems Engineering

Jean-Marie Gauthier, Eric Jenn and Ramon Conejo
IRT Saint Exupéry, 3 Rue Tarfaya, 31400 Toulouse, France
{*firstname.lastname*}*@irt-saintexupery.com*

Keywords: Systems Engineering, Large Language Model, Agent, Systems Modelling, Modelling Assistant, Ontology.

Abstract: This paper presents an LLM-based assistant integrated within an experimental modelling platform to support Systems Engineering tasks. Leveraging an ontology-driven approach, the assistant guides engineers through Systems Engineering tasks using an iterative prompting technique that builds task-specific context from prior steps. Our approach combines prompt engineering, few-shot learning, Chain of Thought reasoning, and Retrieval-Augmented Generation to generate accurate and relevant outputs without fine-tuning. A dual-chatbot system aids in task completion. The evaluation of the assistant's effectiveness in the development of a robotic system demonstrates its potential to enhance Systems Engineering process efficiency and support decision-making.

1 INTRODUCTION

1.1 Context and Motivations

Model-Based Systems Engineering (MBSE), as a multidisciplinary approach, offers the promise to structure engineering data with the support of modelling. It aims at facilitating activities encompassing system requirements, design, analysis, as well as verification and validation, during the conceptual design phase throughout the subsequent development stages and life cycle phases (INCOSE, 2007).

At the same time, one of the most notable advancements in the artificial intelligence domain is the emergence of Large Language Models (LLMs). Powered by deep learning techniques, those models have demonstrated exceptional capabilities in generating human-like text, images from natural language input. These models, exemplified by OpenAI's Generative Pre-trained Transformer (GPT) series (Brown et al., 2020), have garnered widespread attention for their ability to generate natural language text or images at an unprecedented scale (Ramesh et al., 2021).

Despite the increasing adoption of MBSE approaches in the industry, its full deployment remains limited due to various challenges, ranging from technological hurdles to organizational barriers (Chami and Bruel, 2018). These challenges often stem from the complexity of integrating MBSE into existing workflows and the need for specialized knowledge and training. To overcome these obstacles, one of the main leads is the integration of Artificial Intelligence (AI) into SE and MBSE processes (Chami et al., 2022). The intersection of MBSE and AI presents an opportunity to improve the way systems engineering tasks such as requirements writing, models authoring, refinement, and analysis are performed.

Specifically, the integration of Large Language Models (LLMs) with Systems Engineering (SE) and Model-Based Systems Engineering (MBSE) offers the potential to automate and enhance key SE and MBSE tasks (Alarcia et al., 2024). A significant number of applications of AI to MBSE use cases have been identified in the literature (Schräder et al., 2022), and within the Systems Engineering communities (SELive, 2023). Moreover, industrial companies have initiated exploration and experimentation to understand the possibilities and constraints associated with implementing AI, and especially GPT-like models, in the systems engineering domain. In particular, LLMs were tested as an assistant from requirements authoring (Andrew, 2023) (Tikayat Ray et al., 2023) and more complex Requirements Engineering tasks (Arora et al., 2024), to MBSE assistance, such as SysML V2 model edition (Fabien, 2023) and PlantUML model generation from prompts (Cámara et al., 2023). These examples show that LLMs can streamline and enhance SE and MBSE processes by enabling automated generation and refinement of models from natural language inputs.

Gauthier, J.-M., Jenn, E. and Conejo, R.
Ontology-Driven LLM Assistance for Task-Oriented Systems Engineering.
DOI: 10.5220/0013441100003896

In *Proceedings of the 13th International Conference on Model-Based Software and Systems Engineering (MODELSWARD 2025)*, pages 383-394
ISBN: 978-989-758-729-0; ISSN: 2184-4348

However, while LLMs have the potential to reduce manual effort and increase the accuracy of SE and MBSE, practical challenges remain. Indeed, the need for reliable traceability, precision, and consistency in SE and MBSE tasks emphasizes the importance of further research on how LLMs can effectively support these objectives without extensive fine-tuning. In addition, managing large datasets, handling context limits, and achieving seamless integration within SE-specific processes, are challenges that need to be addressed.

1.2 Research Questions and Contributions

To address the limits identified in the previous section, we propose to answer to the following main research questions. The first research question is to identify contexts, system engineering phases, and activities in which LLMs can be effectively applied, and add value. Specifically, the study seeks to identify the stages and contexts in which LLMs can be effectively applied. It addresses questions regarding their ability to enhance task completion, improve efficiency, reduce human error, and support the consistency and completeness of SE data.

RQ1: Which Systems Engineering activities can benefit the most from the integration of LLMs?

RQ2: What implementation strategies can be employed to effectively integrate LLMs into these SE activities?

RQ3: What tangible benefits do LLMs offer for enhancing the identified Systems Engineering activities?

To address these questions, the paper introduces an experimental platform designed to evaluate the integration of LLMs into SE tasks, particularly in the context of requirements engineering. While the focus remains on textual artifacts, such as needs and requirements, the study also explores the potential for scalability and adaptability to other SE domains.

Our contribution presents results obtained in the AIDEAS and EasyMOD projects carried out at IRT Saint Exupery. AIDEAS aims to evaluate the benefits of LLM-based conversational agents to realize selected engineering tasks in the field of aerospace and space systems including SE tasks, technological surveys, etc. Besides providing technical solutions to implement those agents using LLMs, a significant part of the study focuses on understanding how tasks distribute between the human and nonhuman agents and how this collaboration can be evaluated and, eventually, improved. The contributions of our study are as follows:

1. **Processes and Tasks Identification:** We use LLM to identify the relevant ISO15288 technical processes (iso, 2015) necessary to meet an engineering objective expressed by a systems engineer. The LLM further decomposes each process into specific tasks, creating a task hierarchy. This sets the foundation for thorough task completion and traceability throughout the project.
2. **Ontology-Driven Task Assistant:** Using a Systems Engineering ontology, the LLM identifies the required inputs—such as specific data, stakeholder requirements, or prior analysis results—that are essential to initiate and perform the task effectively. Additionally, the LLM determines the outputs that each task should produce, such as technical specifications, refined requirements, or system component definitions. By defining tasks with these input and output expectations, the ontology-driven approach enhances the LLM's ability to deliver precise, relevant guidance and ensures that each task contributes systematically to the overall engineering objectives.
3. **Completeness Checking and Task Validation:** We implemented a completeness checker to ensure each task meets SE criteria and is thoroughly documented, leveraging LLM capabilities for final validation.
4. **Dual-Chatbot Assistance:** Two assistant chatbots were developed. A guided questioning assistant helps engineers refine requirements, clarify specifications, and streamline data management, while a task-specific assistant provides open-ended support, expanding the LLM's capacity for structured and unstructured queries.

1.3 Paper Overview

The remainder of this paper is organized as follows. Section 2 provides background on Large Language Models (LLMs), including their foundational principles and relevant advancements. Section 3 outlines our methodology and presents the approach for implementing the ontology-driven LLM solution. Section 4 showcases insights from a robotic system case study, and discusses the results. Section 5 reviews related work on the integration of LLMs for Model-Based Systems Engineering (MBSE). Finally, Section 6 concludes the paper and outlines future work.

2 BACKGROUND ON LANGUAGE MODELS

In this section, we present the foundational knowledge and context required to comprehend the following content. First, foundations on LLMs are presented. Then, a focus is made on Augmented Language Models (ALMs) that attempt to address the limitations of LLMs.

2.1 Large Language Models

Large Language Models (LLMs) are a class of AI models that predict the next character of a sequence, or "piece of word" based on the input text. They work by leveraging advanced deep learning techniques, specifically transformer architectures, to process and generate human-like text. The following paragraphs explain the main concepts related to LLMs.

2.1.1 Transformer Architecture

LLMs are based on the Transformer architecture, which uses self-attention mechanisms and feed-forward neural networks to process input efficiently and capture long-range dependencies (Vaswani et al., 2017). The self-attention mechanism allows the model to weigh the importance of different parts of a sequence, enabling a nuanced understanding of context and relationships between tokens. Input text is decomposed into smaller units called "tokens", like words or sub-words, and represented as high-dimensional vectors, allowing the model to predict the next token in a sequence based on preceding context.

2.1.2 Pre-Training and Fine-Tuning

LLMs are pre-trained on vast text datasets, learning to predict the next token in a sequence, which helps them grasp syntax, semantics, and contextual relationships. This large-scale training enables them to generate coherent, contextually relevant text. They can then be fine-tuned on smaller, task-specific datasets to adapt their capabilities for specialized applications.

2.1.3 Prompting, Zero-Shot, and Few-Shot Settings

The effective use of LLMs is based on prompt engineering, where the input prompts guide the model's text generation. Creating optimal prompts requires an understanding of the model's capabilities and limitations, which is still a research problem in its own. Researchers experiment with prompt length, formats, and contextual cues to improve the quality and relevance of the output. Well-engineered prompts are particularly important in specialized fields, such as Systems Engineering, where accurate and contextually appropriate responses are essential.

Prompting techniques are typically divided into zero-shot and few-shot methods. In a zero-shot setting (Kojima et al., 2022), the model generalizes to unseen tasks without specific training examples, relying solely on prior knowledge. In contrast, few-shot learning (Brown et al., 2020) involves providing a small number of task examples to help the model adapt to and perform specific tasks more effectively.

2.2 Augmented Language Models

LLMs excel in capturing complex linguistic patterns, understanding context, and generating human-like text across various tasks. However, they face challenges, including the generation of non-factual yet plausible responses (hallucinations) and the need for large-scale storage to handle task-related knowledge beyond their limited context. These issues stem from the LLMs' reliance on a single parametric model and constrained context size, which impacts their accuracy and practical use.

To address these limitations, Augmented Language Models (ALMs) have emerged. ALMs, as defined by (Lecun et al., 2023), integrate reasoning and external tools to enhance the model's capabilities. Reasoning strategies in ALMs decompose complex tasks into manageable subtasks, often leveraging step-by-step processes (Qiao et al., 2022). Techniques like Chain of Thought (CoT) prompting have been shown to significantly improve reasoning, especially for multi-step tasks, by guiding the model through a structured approach that enhances its performance over traditional prompting methods (Wei et al., 2022).

2.3 Assessment Regarding Our Contributions

In this study, we leverage Large Language Models (LLMs) in their base form, without fine-tuning, due to the substantial computational resources and specialized Systems Engineering (SE) datasets required for fine-tuning, which are not available. To overcome this limitation, our approach integrates several techniques to enhance the LLM's ability to perform SE tasks effectively:

- **Prompt Engineering and Few-Shot Setting:** We use prompt engineering to tailor the LLM's behavior to Systems Engineering tasks, guiding it to

generate SE-specific responses. Few-shot learning is employed to enhance the model's generalization across various SE tasks by providing a few examples, enabling the LLM to instantiate data for new, unseen tasks efficiently.

- **Chain of Thought (CoT) Reasoning:** We improve the model's performance by applying CoT reasoning to break complex tasks down into smaller steps.
- **Retrieval-Augmented Generation (RAG):** We use RAG to enable the LLM to access external, task-specific knowledge on demand. By combining generative capabilities with information retrieval, the RAG ensures more accurate, contextually relevant outputs without the need for time-consuming fine-tuning.

3 LLM-BASED ASSISTANT FOR SYSTEMS ENGINEERING

This paper presents an approach that leverages LLMs to assist in Systems Engineering processes. Our approach combines LLMs with an SE ontology, generating task-specific guidance aligned with a user-defined SE objectives. The LLM decomposes each SE process into tasks, identifies required inputs and outputs, sets completion criteria to ensure alignment with SE standards, and provides guidance to achieve the tasks.

The SE assistant is integrated in a systems engineering platform called EasyMOD. This platform enables engineers to create system architectures, automatically extract views for interactive review document creation, and map system analysis results onto system architectures. In this paper, we focus on the assistant component of EasyMOD. An overview of our systems engineering assistance approach is presented, particularly emphasizing the iterative prompting approach that builds the context for specific systems engineering tasks. By iterative prompting, we mean a step-by-step approach where each task is guided by context constructed from the outputs of previous steps, ensuring continuity and informed progression.

3.1 Processes and Tasks Identification

The first step of our approach involves capturing the systems engineering objective defined by the engineer and proposing an appropriate process to achieve it. Figure 1 outlines the workflow using ISO 15288 technical processes as a baseline. It begins with two types of prompts: the **context prompt** (also known as *system prompt*), which specifies the Systems Engineering role that the AI should assume while generating responses. The **constraints prompt** (also known as *user prompt*) provides specific instructions or queries that the LLM must address within the context established by the **context prompt**. It further captures the engineer's SE Objective, as the starting prompt, the tasks to be performed by the LLM, and the expected JSON format as answer.

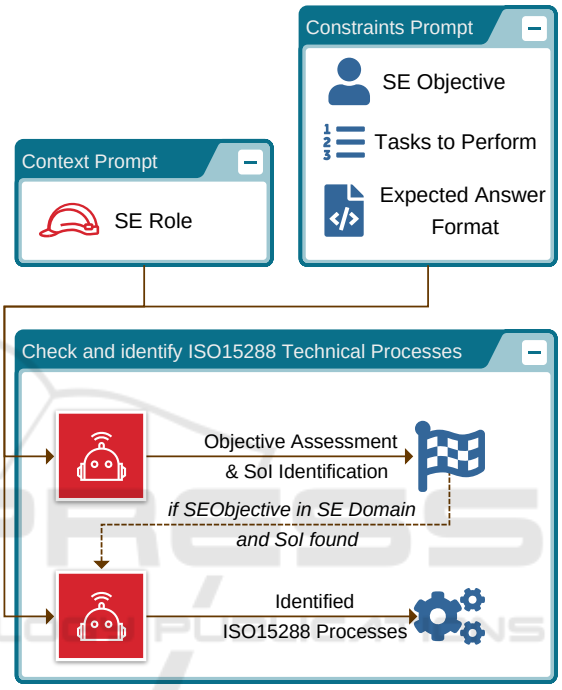

Figure 1: Prompts for ISO 15288 processes identification.

Using these inputs, the assistant validates whether the SE objective aligns with the Systems Engineering domain and identifies the corresponding System of Interest (SoI). Upon successful assessment, the assistant identifies and outputs the relevant ISO 15288 technical processes, which form the foundation for subsequent task decomposition and validation.

Then the Systems Engineering processes are broken down into tasks. The SE role, objective, and identified list of processes are used to establish the context for the assistant. The user selects one of the processes from this list, which, with the tasks to perform and the expected answer format, guides the LLM to generate a task breakdown.

3.2 Ontology-Driven Task Assistant

Central to our approach, an SE ontology called Systems Engineering Information Model (SEIM), which the assistant uses to extract and define the essential

concepts for each Systems Engineering task. By integrating this ontology, the assistant has access to a structured description of the relationships between the key concepts in SE. An excerpt of this ontology is shown in Figure 2. The concepts in this example include Needs, Stakeholder Requirements, Functions, System Requirements, and Abstract Constituents, all of which are linked with associations to establish traceability.

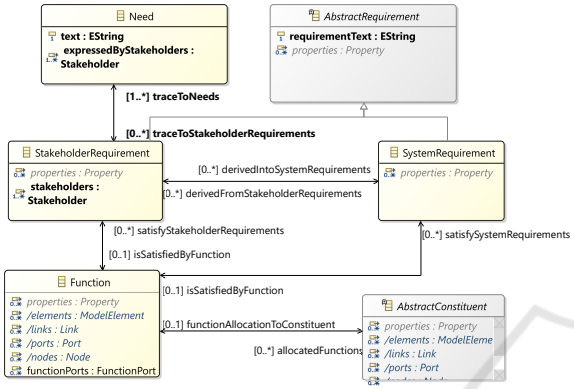

Figure 2: Ontology Excerpt as a Metamodel.

Additionally, certain concepts within the ontology are supplemented with specific documentation tailored for the LLM. For example, stakeholder requirements and system requirements are enriched with best practices and standardized patterns to ensure the writing of high-quality requirements.

The Figure 3 illustrates how the ontology is used by the assistant to identify the engineering items that must be consumed and produced by each task, ensuring alignment with the overall system objectives. The **context prompt** includes details such as the SE Role, SE Objective, a list of available processes and tasks, and the user-selected task for focused assistance. The **constraints prompt** includes the SEIM ontology, the task the assistant shall follow, and the required LLM's output format. The assistant uses these inputs to identify the necessary SEIM Input/Output (I/O) concepts related to the selected task.

For example, the task of translating stakeholder needs into stakeholder requirements requires the input concepts of *Stakeholders* and *Needs* and produces *Stakeholder Requirements* as the output.

3.3 Task Completeness Checking

After tasks are defined, ensuring their completeness and correctness is crucial. This is achieved by the completeness checker, which evaluates whether all required task outputs have been produced based on predefined criteria. The checker leverages LLMs to identify missing elements, and provide feedback for iterative refinement of the SE task.

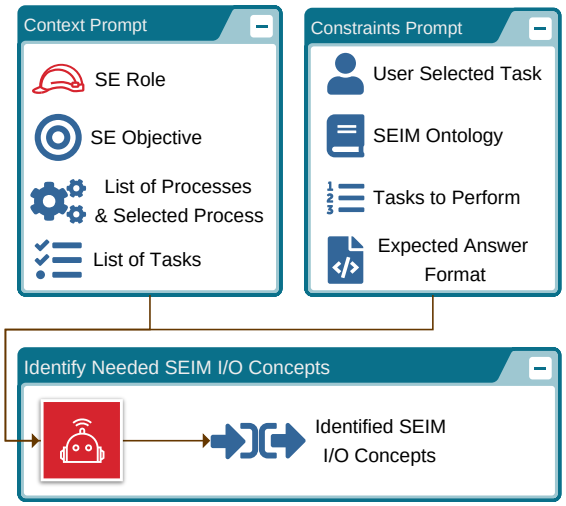

Figure 3: Prompts for task's I/O identification.

To extract the completeness criteria, we use a chain of thought (COT). The first phase involves defining the SE task's completeness criteria. We propose to use the association links between the task's output concepts and its input concepts, as illustrated in Figure 4. For instance, if the task aims at producing stakeholder requirements from needs, then a completeness criterion is to ensure that each need is traced to stakeholder requirements. Once these traceability links are established, the assistant is further tasked with defining completeness criteria grounded in these links, while also incorporating additional criteria related to the data generated by the task.

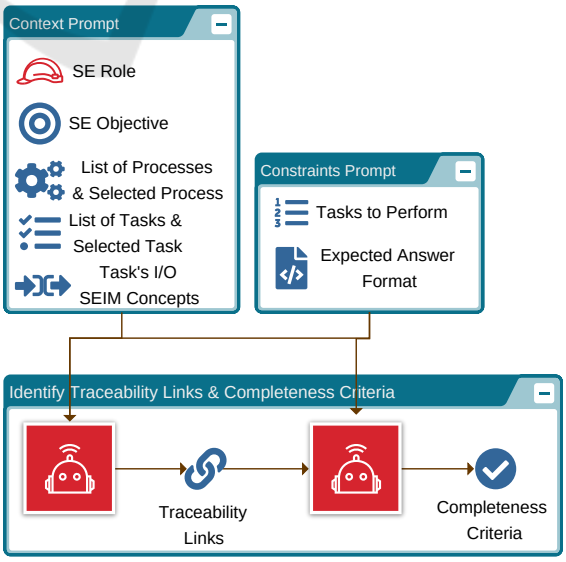

Figure 4: Prompts and COT for completeness criteria identification.

The second phase illustrated by Figure 5 involves conducting a difference analysis between the successive versions of the data. This analysis serves to identify and categorize changes made during the task, including elements that have been added, removed, or updated. By systematically comparing the two versions, this process provides insights into the evolution of the task's outputs, highlighting modifications and enabling engineers to track progress, ensure consistency, and address any unintended alterations.

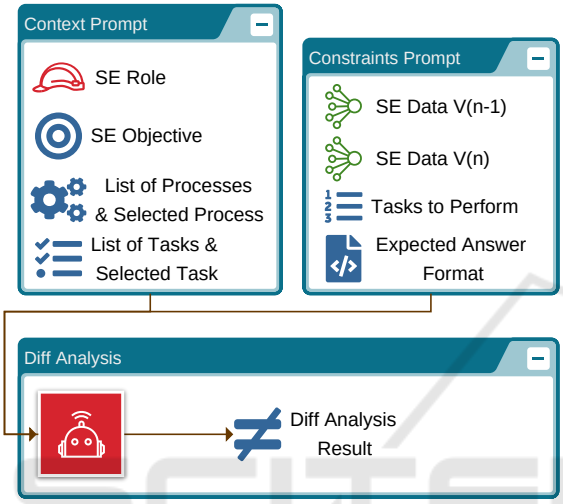

Figure 5: Prompts for diff analysis.

Finally, the completeness criteria and the results of the difference analysis are provided as inputs to the assistant to conduct a task completeness assessment. Figure 6 illustrates this process, highlighting the integration of completeness criteria, difference analysis results, and the most recent completion assessment. Leveraging these inputs, the assistant evaluates the degree to which the task meets the predefined criteria, adjusting the completion percentage as necessary. Depending on the assessment, the completion percentage is calculated, reflecting the current state of the task's outputs relative to the defined goals.

3.4 Dual-Chatbot Assistance

To further enhance user support, we propose a dual-chatbot system, comprising a guided questioning assistant and an open task-specific assistant. The guided Question/Answer (Q/A) chatbot assists in task completeness by asking focused questions aimed at satisfying the predefined completeness criteria.

The Q/A chatbot, as illustrated in Figure 7, assists systems engineers in completing specific tasks by asking context-aware, targeted questions. Its functionality is driven by a detailed context prompt that aggregates critical information, such as the assistant's role, the global engineer's SE objective, and the hierarchical relationship between tasks and processes, ensuring tailored guidance aligned with the engineer's responsibilities. The prompt further integrates domain-specific knowledge through SEIM ontology concepts, references existing Systems Engineering data for consistency, and outlines task completion criteria to ensure the chatbot's questioning focuses on achieving successful task outcomes.

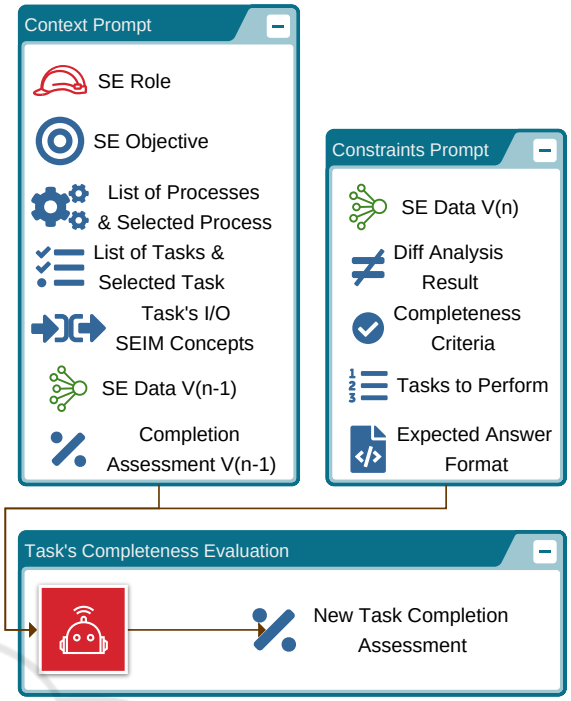

Figure 6: Prompts for task's completeness evaluation.

In addition, the open-ended chatbot provides broader support for addressing complex or unstructured tasks, offering flexibility and adaptability.

4 EXPERIMENTS, RESULTS, AND DISCUSSION

This section presents the experimental evaluation of our approach. We detail the experimental platform developed to test the proposed methodology, followed by its application to a robotic system use case. We also discuss the challenges and lessons learned during the experiment, including insights on task completeness, and consistency. Through these experiments, we aim to assess the practical value of LLM integration in real-world SE applications.

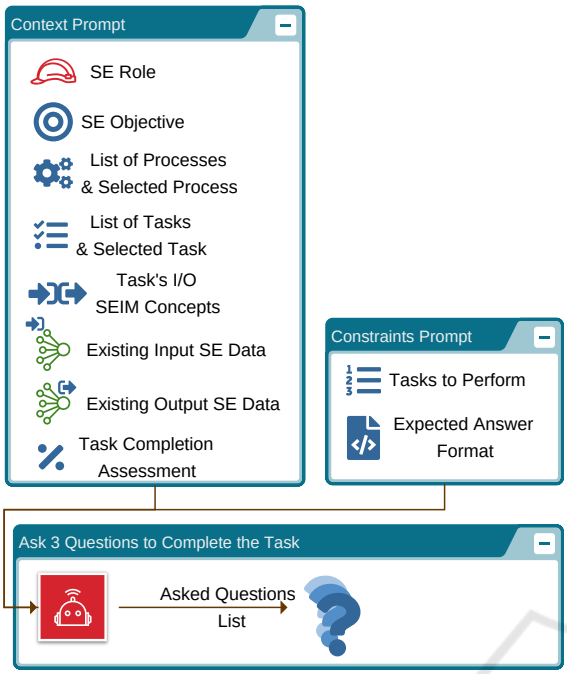

Figure 7: Prompts to ask a list of questions.

4.1 Experimental Platform

The main interface of the experimental platform is shown in Figure 8. The interface is divided into several key elements. On the top-left side of the screen, is the Process Overview panel, which highlights the current SE process and task being undertaken, such as translating stakeholder requirements into system requirements. Below this panel, the Ontology Manager enables users to interact with predefined SE concepts, ensuring alignment with the knowledge base.

The central section of the interface features an engineering data table, which lists the required and produced SE data, such as traceability links to original needs, and derived system requirements. Above the table is the task completeness interface. This section allows users to track progress and identify missing elements.

To the right, the Dual-Chatbot System is displayed in a panel. The guided questioning chatbot helps users clarify inputs, outputs, and requirements for task completion by asking focused questions, ensuring traceability and alignment with task objectives. For instance, it prompts the user to define system functions necessary to satisfy derived requirements. Additionally, the open-ended chatbot assistant offers broader support for unstructured or complex tasks by providing suggestions, explanations, or additional context where needed.

4.2 Application on a Robotic Use Case

To validate our approach, we applied it to a robotic system under development at IRT. The robot serves as a demonstration platform integrating advanced technologies developed by the IRT and its partners. It is designed to autonomously guide visitors within the IRT premises while showcasing integrated technological solutions. Its development emphasizes the reuse of existing hardware and software components, with minimal new development, prioritizing the integration of technologies to highlight their value.

The robot's mission includes several key operational capacities:

- Responding to verbal commands from visitors, such as adjusting speed, stopping, or resuming movement.
- Dynamically adjusting its displacement speed to match the visitor's pace.
- Monitoring the visitor's proximity, requesting them to return if they move too far, and notifying the supervision operator and host if the visitor remains out of range for an extended period.
- Sending an automated email to the host upon mission completion and returning to a designated parking area.

We started the experiment with the following engineering objective:

> *We are a team of engineers in charge of the specification, the design and the implementation of a robot (named BIRD) which mission is to welcome visitors and to bring them to the right place. The robot will be deployed in a research institute named IRT (Institut de Recherche Technologique) at Toulouse in a building named B612. The robot will serve also to demonstrate our know-hows and will embed technological bricks we are developing at the IRT.*

We present here the results obtained using the tool. For interested readers, the complete set of data is publicly available in our public repository[1].

- **System Mission Identification:** First, we identified several system's missions listed on Table 1. The missions SM-001, SM-002, and SM-003 are relevant and well-formed, effectively capturing the robot's primary objective. However, SM-004, which focuses on accessibility features, appears more aligned with a specific system feature rather

[1] https://sahara.irt-saintexupery.com/jean-marie.gauthier/Data_LLM/src/branch/master/BIRD_LLM.pdf

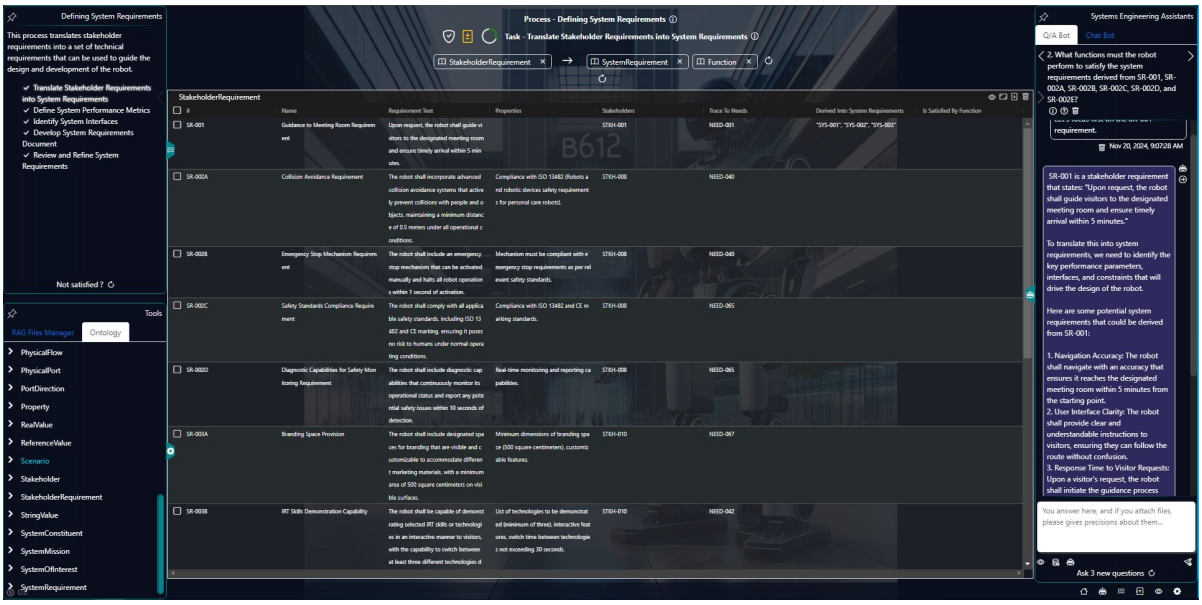

Figure 8: Experimental platform main interface.

than a high-level mission. For the rest of the experiment we focused on SM-001 only.

- **Stakeholder Identification:** Starting from an initial set of 7 stakeholders provided by the systems engineer, the tool allowed use to identify 11 additional stakeholders. All stakeholders were relevant. An excerpt of the resulting stakeholder is shown on Table 2.
- **Needs Capture:** Starting from an initial set of 31 needs provided by the systems engineer, the tool allowed use to identify 43 additional needs, and to link all them to the corresponding stakeholder. The Q/A chatbot assisted us in capturing all stakeholder needs by prompting questions about stakeholders who had not yet expressed any needs. All the obtained needs were relevant. An excerpt of the resulting needs is shown on Table 3.
- **Needs Analysis:** We used the open chatbot to analyze the set of 74 identified needs, specifically focusing on detecting conflicts. Our assistant successfully identified 11 potential conflicts. A detailed record of the needs analysis discussion is provided in the complete data document.
- **Stakeholder Requirements Definition:** The tool assisted in drafting 39 stakeholder requirements, ensuring complete traceability to the original needs expressed by stakeholders. All requirements are measurable and include verification and validation tasks within their documentation. While significant progress has been made, this task remains ongoing as further refinement and validation are still needed. An excerpt of the stakeholder requirements is presented in Table 4.
- **Task Completeness and Consistency Checks:** By leveraging, the ontology-driven and LLM-assisted completeness checker, and the Q/A chatbot, the tools assisted in ensuring that all needs and requirements were logically consistent, aligned with the system's mission, and free from hallucinations so far.

This experiment highlights the efficiency of our approach in supporting SE tasks, such as requirement elicitation, and traceability management. The robot use case also underscores the scalability of our methodology in handling diverse and interconnected SE tasks while maintaining consistency across data sets and ensuring system definition completeness.

4.3 Discussion

The experiments conducted focused primarily on textual artifacts—needs and requirements—where LLM integration was shown to be highly effective. By identifying 18 stakeholders, extracting 74 needs, detecting 11 conflicts, and drafting 29 stakeholder requirements, the tool demonstrated its capacity to enhance task execution, efficiency, and completeness, addressing **RQ1**. Specifically, the Q/A chatbot's guided questions ensured gaps were filled, while the ontology-driven framework structured unstructured data into a coherent and traceable form. This success builds confidence for extending the tool to architecture-related activities involving system models, reinforcing the LLM's potential to automate SE tasks and aug-

Table 1: Identified robot's missions.

ID	Name	Description
SM-001	Visitor Guidance	To assist visitors by providing directions and escorting them to specific locations within the B612 building.
SM-002	Demonstration of Technology	To showcase the technological advancements and capabilities developed at IRT.
SM-003	Operational Support	To assist IRT staff and reception staff in their daily operations by performing tasks such as message delivery and fetching items.
SM-004	Accessibility Features	To provide special accessibility features for visitors with disabilities.

Table 2: Extract of resulting stakeholders.

ID	Name	Documentation
STKH-001	Visitors	Individuals visiting the IRT within the B612 building, interacting with the robot for guidance to specific locations.
STKH-002	Supervisor	Responsible for monitoring and controlling the operation of one or more robots.
STKH-003	Maintenance Operator	In charge of the system's maintenance, particularly addressing potential breakdowns.
...	...	...
STKH-018	Cleaning Staff	Responsible for maintaining cleanliness and order within the operational environment of the robot, ensuring that the robot's pathways are free of obstacles and debris.

ment MBSE practices. However, the experiments did not yet explore scenarios where engineering artifacts from previous or similar systems exist. Addressing **RQ1** further, such reuse could optimize task completion by leveraging historical data for faster and more accurate generation of requirements or models. Future work will also focus on validating results through comparison with human-generated benchmarks to assess the tool's accuracy and reliability.

The study highlights that even without fine-tuning, Large Language Models (LLMs) can adequately perform many general Systems Engineering (SE) tasks, providing an efficient and cost-effective solution. This addresses **RQ2**, as it demonstrates how LLMs can be practically implemented using prompt engineering and ontology-driven frameworks to assist in specific SE activities without the need for expensive fine-tuning. Importantly, our approach remains highly generic and adaptable to different SE processes, as even the tasks themselves are dynamically generated by the LLM. This flexibility answers part of **RQ2**, as it highlights the tool's process-agnostic nature. The same framework can be seamlessly applied to standards like ISO 15288 or ARP4754A without requiring infrastructure modifications. Such adaptability ensures the tool's broader applicability across diverse SE contexts, which is a key advantage over traditional SE tools.

However, the results also raise an important consideration for **RQ3**: whether the performance improvement achieved through fine-tuning justifies the additional investment in terms of data and computational resources. This remains an open avenue for future work. In addition, a key concern tied to **RQ3** is the quality of the outputs generated by LLMs. While the assistant provided relevant and useful support during the experiments, rigorous verification protocols remain to be defined. For safety- or cost-critical systems, where small errors in assumptions or data can have severe consequences, a dedicated validation framework is essential. Human-generated benchmarks will also play a critical role in systematically evaluating and comparing the assistant's outputs. Hence, as of now, LLMs should serve as a support tool rather than a replacement for human judgment, particularly in high-stakes domains. This balance ensures the risk of undetected errors is mitigated while still reaping the benefits of automation.

This study underscores that LLMs are particularly suited to automating "tedious" SE tasks, which are repetitive, time-consuming, and prone to human error—such as structuring data, ensuring traceability, and detecting inconsistencies. By addressing these tasks, the assistant allows systems engineers to focus on higher-value activities like strategic decision-making and system innovation. This directly an-

Table 3: Extract of resulting needs.

ID	Name	Need Text	Stakeholders
NEED-001	Guidance to Meeting Room	As a Visitor, I need to be guided to the meeting room / office in order to reach the meeting room at the appropriate time.	STKH-001
NEED-002	Guidance to Rest Room	As a Visitor, I need to be guided to the nearest rest room upon request.	STKH-001
...	...	...	...
NEED-073	Emergency Stop and Manual Override	As Cleaning Staff, I need to have the ability to stop the robot in case it obstructs the cleaning process or in emergencies.	STKH-018
NEED-074	Network Connectivity	As IT Support Team, I need the robot to have reliable network connectivity for remote monitoring and updates.	STKH-009

Table 4: Extract of resulting stakeholder requirements.

ID	Name	Text	Trace to Needs
SR-001	Guidance to Meeting Room Requirement	Upon request, the robot shall guide visitors to the designated meeting room and ensure timely arrival within 5 minutes.	NEED-001
SR-004	Guidance to Rest Room Requirement	Upon request, the robot shall guide visitors to the nearest rest room and ensure timely arrival within 3 minutes.	NEED-002
...	...	...	...
SR-027	Emergency Stop Capability Requirement	The robot shall allow the Supervisor to initiate an emergency stop, halting all operations immediately within 1 second of the command being issued.	NEED-014

swers **RQ3** by showcasing tangible benefits in efficiency and task completion while maintaining alignment with SE standards and objectives.

Finally, the findings demonstrate the relevance of LLMs for SE tasks, their practical implementation in a process-agnostic platform, and their ability to enhance task execution, consistency, and traceability. However, challenges remain in ensuring validation protocols, including comparisons with human-generated benchmarks, and exploring the reuse of existing SE data, which will be crucial for broader adoption and future advancements.

5 RELATED WORK

The increasing complexity of cyber-physical systems necessitates model-based systems engineering (MBSE), but its broader adaptation requires specialized training. User-centric systems engineering and AI can make this process more accessible. The research in (Bader et al., 2024) demonstrates that a GPT model, trained on UML component diagrams, can effectively understand and generate complex UML relationships, though challenges like extensive XMI data and context limitations remain.

The paper (Fuchs et al., 2024) explores how large language models (LLMs) can automate model creation within a new declarative framework. Users can interact with the system using natural language, representing a significant shift in model development. The paper provides examples demonstrating how LLMs can improve the speed and cost of tasks like object creation, content review, and artifact generation by 10 to 15 times.

An intelligent workflow engine that integrates an AI chatbot to enhance traditional workflow management systems in presented in (Reitenbach et al., 2024). By using Natural Language Processing and Large Language Models, the chatbot serves as a user-friendly interface, simplifying workflow creation and decision-making.

The study in (du Plooy and Oosthuizen, 2023) explores how GPT-4 can enhance the development of system dynamics simulations within systems engineering. Findings reveal that while GPT-4 significantly improves simulation construction, error reduc-

tion, and learning speed, it also struggles with identifying errors and generating high-quality expansion ideas. Technical challenges include AI's limited ability to handle specialized domains and the potential for errors due to data quality.

In (Timperley et al., 2024), researchers evaluated the use of TEXT-DAVINCI-003 LLM for generating spacecraft system architectures for Earth observation missions, achieving at least two-thirds traceability of design elements back to requirements across three cases. Consistent outputs suggested potential for diverse missions, though function generation challenges emphasized the need for expert oversight to refine outputs. Higher-token-count prompts improved traceability and alignment but increased computational costs. Lower classifier confidence for modes indicated textual variation impacts, while high precision over recall reflected training data limitations. Manual assessments validated architecture quality, with future work aimed at automated evaluation and ontology refinements to better map LLM output to MBSE models. Data security concerns also suggested the benefit of self-hosted, fine-tuned LLMs.

Existing research demonstrates the potential of LLMs to assist in systems engineering by automating specific tasks, such as UML generation (Bader et al., 2024), declarative model creation (Fuchs et al., 2024), and intelligent workflow simplification (Reitenbach et al., 2024). However, these approaches often address isolated tasks without establishing a robust foundational context for the overall systems engineering process.

Our work complements these efforts by focusing on building a comprehensive and traceable context for SE tasks through ontology-driven task guidance, iterative prompting, and completeness checking. This foundational approach ensures the alignment of tasks with SE frameworks like ISO 15288, enabling more reliable outputs before diving into detailed atomic tasks. By integrating traceability and error-recovery mechanisms, our methodology provides the structured groundwork necessary to support and enhance future detailed SE workflows, bridging the gap between high-level guidance and fine-grained task execution.

6 CONCLUSION AND PERSPECTIVES

This paper presents an ontology-driven, LLM-based assistant integrated into the EasyMOD platform to support task-oriented Systems Engineering (SE) processes. By combining context-aware and iterative prompting with ontology-based task structuring, the assistant ensures alignment with SE objectives, continuity across tasks, and the consistent generation of high-quality data. The experimental platform and dual-chatbot system provide structured guidance and open-ended support, enhancing task completeness, efficiency, and user engagement.

The study demonstrates that LLMs are particularly effective in automating tedious and repetitive SE tasks, such as stakeholder identification, needs capture, requirements definition, and conflict detection. The results, validated on a robotic system case study, highlight notable benefits in improving task efficiency, ensuring traceability, and reducing human error. Additionally, the tool proves capable of transforming unstructured textual data into structured outputs aligned with a formal SE ontology. However, challenges remain, particularly in handling safety- and cost-critical systems where rigorous verification protocols are required. Furthermore, the current approach has yet to explore scenarios where pre-existing engineering artifacts from similar systems can be reused to accelerate task execution.

From a methodological perspective, the proposed solution is highly generic and adaptable, capable of dynamically generating tasks without relying on predefined workflows. This flexibility enables the tool to seamlessly integrate with various SE standards, enhancing its applicability across different industrial domains. Nonetheless, the dependency on predefined ontologies and the absence of user feedback mechanisms for iterative refinement are improvement areas.

Future work will broaden the scope of SE tasks addressed by the assistant, extending beyond requirements elicitation and quality assessment to include architecture definition, model-based engineering, and other SE lifecycle phases. This expansion will assess the assistant's applicability across diverse workflows and improve its utility for complex, interconnected tasks. To support this, we plan to integrate graph-based Retrieval-Augmented Generation (RAG) to enhance task-specific knowledge retrieval and strengthen interactions with external tools and EasyMOD modules, such as model editing and review processes.

Future validation efforts will focus on developing human-generated benchmarks to rigorously evaluate the assistant's performance across larger, more diverse case studies. This will include analyses of task completeness, consistency, and alignment with SE objectives. Additionally, human-machine interaction studies and ergonomic evaluations will assess the tool's practical benefits and refine its capabilities. By addressing these areas, future work aims to solidify

the assistant's role as a robust support system for Systems Engineering practitioners.

ACKNOWLEDGEMENTS

The authors thanks the Fondation de Recherche pour l'Aeronautique et l'Espace (FRAE) for contributing to the funding of the work presented in this paper. We are also grateful, to Airbus Commercial Aircraft, Airbus Defense and Space, and the French ANR for contributing to the funding of the EasyMOD project.

REFERENCES

(2015). ISO/IEC/IEEE International Standard - Systems and Software Engineering – System Life Cycle Processes.

Alarcia, R. M. G., Russo, P., Renga, A., and Golkar, A. (2024). 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.

Andrew, G. (2023). Future consideration of generative artificial intelligence (ai) for systems engineering design. *26th National Defense Industrial Association Systems Engineering Conference*.

Arora, C., Grundy, J., and Abdelrazek, M. (2024). Advancing requirements engineering through generative ai: Assessing the role of llms. In *Generative AI for Effective Software Development*, pages 129–148. Springer.

Bader, E., Vereno, D., and Neureiter, C. (2024). Facilitating user-centric model-based systems engineering using generative ai. In *MODELSWARD*.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Cámara, J., Troya, J., Burgueño, L., and Vallecillo, A. (2023). On the assessment of generative ai in modeling tasks: an experience report with chatgpt and uml. *Software and Systems Modeling*, 22(3):781–793.

Chami, M., Abdoun, N., and Bruel, J.-M. (2022). Artificial intelligence capabilities for effective model-based systems engineering: A vision paper. *INCOSE International Symposium*, 32(1):1160–1174.

Chami, M. and Bruel, J.-M. (2018). A survey on mbse adoption challenges. In *INCOSE EMEA Sector Systems Engineering Conference (INCOSE EMEASEC 2018)*. Wiley Interscience Publications.

du Plooy, C. and Oosthuizen, R. (2023). Ai usefulness in systems modelling and simulation: gpt-4 application. *South African Journal of Industrial Engineering*, 34(3):286–303.

Fabien, B. (2023). Exploration AI and MBSE: Use Cases in Aircraft Design. *INCOSE Next AI Explorer*.

Fuchs, J., Helmerich, C., and Holland, S. (2024). Transforming system modeling with declarative methods and generative ai. In *AIAA SCITECH 2024 Forum*, page 1054.

INCOSE (2007). *Systems Engineering Vision 2020*. International Council on Systems Engineering (INCOSE), 2nd edition.

Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., and Iwasawa, Y. (2022). Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Lecun, Y., Dessì, R., Lomeli, M., Nalmpantis, C., Pasunuru, R., Raileanu, R., Rozière, B., Schick, T., Dwivedi-Yu, J., Celikyilmaz, A., et al. (2023). Augmented language models: a survey. *arXiv preprint arXiv:2302.07842*.

Qiao, S., Ou, Y., Zhang, N., Chen, X., Yao, Y., Deng, S., Tan, C., Huang, F., and Chen, H. (2022). Reasoning with language model prompting: A survey. *arXiv preprint arXiv:2212.09597*.

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., and Sutskever, I. (2021). Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR.

Reitenbach, S., Siggel, M., and Bolemant, M. (2024). Enhanced workflow management using an artificial intelligence chatbot. In *AIAA SCITECH 2024 Forum*, page 0917.

Schräder, E., Bernijazov, R., Foullois, M., Hillebrand, M., Kaiser, L., and Dumitrescu, R. (2022). Examples of ai-based assistance systems in context of model-based systems engineering. In *2022 IEEE International Symposium on Systems Engineering (ISSE)*, pages 1–8.

SELive (2023). Artificial intelligence (ai) in model-based systems engineering. *https://www.selive.de/ai-in-mbse/ [last visited:2023-11-27]*.

Tikayat Ray, A., Cole, B. F., Pinon Fischer, O. J., Bhat, A. P., White, R. T., and Mavris, D. N. (2023). Agile methodology for the standardization of engineering requirements using large language models. *Systems*, 11(7).

Timperley, L., Berthoud, L., Snider, C., and Tryfonas, T. (2024). Assessment of large language models for use in generative design of model based spacecraft system architectures. *Available at SSRN 4823264*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. (2022). Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.