# From Documents to Models: Adopting MBSE Practices in Complex Embedded Systems and Leveraging Large Language Models

University of Turku
Department of Computing
Master of Science (Tech) Thesis
Software Engineering
December 2024
Samu Saloranta

UNIVERSITY OF TURKU
Department of Computing

Samu Saloranta: From Documents to Models: Adopting MBSE Practices in Complex Embedded Systems and Leveraging Large Language Models

Master of Science (Tech) Thesis, 96 p.
Software Engineering
December 2024

---

This thesis investigates the transition from Document-Based Systems Engineering to Model-Based Systems Engineering while using Base Transceiver Station development at Nokia as a case study. Additionally, the thesis explores the potential of Large Language Models to facilitate this transition. The research includes a theoretical analysis through a literature review on MBSE and a case study of initial MBSE adoption efforts at Nokia. A proof-of-concept plugin for the MagicDraw modeling tool that integrates OpenAI's GPT-4o to assist in systems engineering tasks is developed and evaluated.

Multiple methodologies are employed in the thesis including literature review, case study analysis, proof-of-concept implementation, and experimental evaluation. The developed plugin performed well in the evaluation, achieving high accuracy in model understanding and model modification tasks, while minimizing token usage of the model representation. The results suggest that LLMs have potential in assisting in systems engineering tasks in modeling tools. Furthermore, MBSE adoption can benefit BTS development through improved system understanding and traceability, although organizations face challenges including high upfront investment and resistance to change.

Identifying theoretical insights into MBSE adoption benefits and challenges as well as practical solutions through AI integration are the main contributions of this thesis, while also identifying some areas for future research in real-world applications of LLM assistance in systems engineering.

Keywords: MBSE, SysML, System Engineering, MagicDraw, AI, LLM, Generative AI, Embedded Systems

# Contents

# 1 Introduction

## 1.1 Background and Motivation

Currently, many different teams at Nokia are working on initial efforts to transition from document-based system engineering (DBSE) to model-based system engineering (MBSE). This is a massive change in how the systems would be managed and developed. It will also almost certainly have large implications for how many people will have to change their way of working in the future. Some initial work has already started on developing the model of the existing and complex Base Transceiver Station (BTS). However, the modeling efforts have not been without difficulties, which is unsurprising as the system is a BTS. This large-scale embedded system has been developed using a traditional Document-Based Systems Engineering approach for a long time. This is a big motivation for this thesis, to analyze the benefits and challenges associated with transitioning from document-based system engineering to model-based system engineering and to come up with potential strategies to solve and lessen these challenges. This thesis will also show an innovative solution for the problem of the difficult learning curve of MBSE, which is to use a plugin powered by an LLM (Large-Language Model) in a modeling tool to make learning modeling easier within the tool.

## 1.2 Research Question and Objectives

This thesis aims to address the following research questions:

**RQ1:** How can a transition from Document-Based Systems Engineering to Model-Based Systems Engineering benefit the development of a long-established, complex embedded system, such as a Base Transceiver Station (BTS), and what challenges and key considerations should be taken into account during this transition?

**RQ2:** What approach can be used to create an efficient textual representation of a SysML model inside a modeling tool (MagicDraw) that is interpretable by a Large Language Model (LLM) like OpenAI's GPT-4o?

**RQ3:** Can an LLM integrated within a modeling tool (MagicDraw) assist in model understanding, query answering, and modification?

**RQ4:** How can the integration of LLMs with MBSE tools impact the adoption and productivity of MBSE practices within organizations?

To address these research questions, the objectives of this thesis are:

- Conduct a literature review to identify benefits, best practices, and challenges in MBSE adoption
- Analyze the challenges faced in the initial MBSE modeling efforts for the BTS system at Nokia
- Identify key considerations and potential strategies for transitioning from document-based practices to MBSE in the context of complex embedded systems like a Base Transceiver Station (BTS)
- Design and implement a MagicDraw plugin that enables effective communication between the SysML model and an LLM.

- Evaluate the LLM's capability to understand, query, and modify SysML models within MagicDraw through the plugin
- Assess the potential impact of the LLM-powered plugin on MBSE adoption and productivity at Nokia, considering benefits and limitations

## 1.3 Research Methods

This section will present the research methods that will be used in this thesis to perform the research.

### 1.3.1 Literature Review

A literature review will be conducted to gather information about MBSE. Literature will be reviewed from various academic databases such as IEEE Xplore, ResearchGate, and Google Scholar. Materials from other sources for example technical reports and conference proceedings will also be used in the thesis. Many different keywords will be used to find literature that is relevant to this thesis research, including "MBSE", "SysML", "Model-Based Systems Engineering", "MBSE AI", "SysML AI", SysML LLM", and "MBSE LLM". The purpose of this research is to look for challenges, benefits, and potential solutions for adopting MBSE. The literature review will also include current research on integrating LLMs with MBSE.

### 1.3.2 Case Study

An ongoing effort at Nokia will be examined, in which teams are starting to transition from using a DBSE approach to an MBSE approach in the development of Base Transceiver Stations (BTS). The Base Transceiver Station (BTS) is a complex embedded system that has been developed over several years by many teams. This presents a real-world context for investigating the challenges and potential solutions

for MBSE adoption. The current effort into building the model of the BTS will be analyzed and potential strategies will be given to combat the challenges that have been faced.

### 1.3.3 Proof-of-Concept (PoC) Plugin Implementation

To study the potential of integrating an AI with a modeling tool, a Proof-of-Concept plugin will be made for the modeling tool MagicDraw which will be integrated with GPT-4o. The plugin will communicate between the SysML model inside MagicDraw and an LLM (OpenAI's GPT-4o) API. It will also build a textual representation of the state of the model and send it to GPT-4o. This is required to give GPT-4o an understanding of the model state so it can respond to the user's queries and requests. The plugin will also allow the LLM to make changes to the model by adding, removing, or modifying the model's elements and diagrams based on user instructions. Implementing this plugin for MagicDraw is necessary to answer the research questions as there does not currently exist a plugin or functionality inside MagicDraw to use an AI assistant for modeling activities.

### 1.3.4 Experiments and Analysis

Testing and evaluation will be done to test the effectiveness of the plugin in model understanding and modification tasks. The testing will include simple tests that have a definite, correct answer and explorative tests which are more subjective and closer to real-world use case scenarios. Additionally, the performance and token usage of the plugin will be evaluated by measuring response time and token usage with models of different sizes. These different kinds of testing will give a balanced evaluation of the plugin's capabilities to work alongside human users as an assistant for modeling-related activities.

## 1.4 Thesis Structure

The thesis is made up of six chapters including this introduction chapter. The other chapters address specific aspects of the research to build a comprehensive understanding of transitioning from DBSE to MBSE and the integration of LLMs into MBSE tools.

**Chapter 1: Introduction** has provided background information and the main motivation for the research. The chapter also outlined the research questions and objectives, as well as described the research methods used in the study.

**Chapter 2: Literature Review and Background Information** includes a review of the existing literature on systems engineering and MBSE. The chapter starts by going through the literature on traditional Document-Based Systems Engineering (DBSE), discussing its principles, practices, and challenges. After that, the chapter explores MBSE and its core principles, and the SysML modeling language as the most widely-used modeling language for MBSE. Literature is also reviewed and examined to find reported benefits and challenges of adopting MBSE. Chapter 2 additionally gives background information on the modeling tools used in this thesis, MagicDraw, and Teamwork Cloud. This chapter gives valuable background information for addressing RQ1.

**Chapter 3: Transitioning to MBSE for BTS Development: A Case Study Analysis** presents an analysis of the initial MBSE modeling efforts for the BTS at Nokia. Next, the benefits of transitioning to MBSE for the development of a BTS are identified and discussed. The chapter also examines key considerations that should be made for a successful transition to MBSE for BTS development. The chapter continues to discuss the challenges that are expected during this MBSE adoption process. This chapter directly addresses RQ1 by analyzing the initial MBSE modeling efforts for the BTS at Nokia.

**Chapter 4: LLMs in MBSE and Plugin Implementation** explores the potential of LLMs in improving the productivity and learning curve for MBSE modeling tools like MagicDraw. First, an introduction is given to LLMs, their recent developments, and their capabilities. Next, the literature is presented and previous research on integrating AI with modeling tools is explored. This is then followed up by describing the design and implementation of the LLM-powered MagicDraw plugin as well as the approach used in creating the textual model representation that is sent to the LLM. This chapter's focus is to address RQ2 and RQ3 by detailing the approach used to create a textual representation of SysML models that is interpretable by LLMs (addressing RQ2) and by explaining how the plugin integrates with MagicDraw to support model understanding query answering, and modification (addressing RQ3).

**Chapter 5: Evaluation of LLM Integration in MagicDraw and Its Impact** evaluates the capabilities of the LLM-powered plugin for MagicDraw. Chapter 5 presents the testing methodology that was used and the results of the experiments that were conducted. The plugin's abilities in model understanding, query answering, and model modification tasks are measured to gauge its usefulness in improving both the productivity and learning curve of using MagicDraw. The chapter also discusses the potential benefits of LLM integration for MBSE processes as well as the potential impact on the MBSE learning curve and productivity. Limitations of the plugin and the testing for it are given as well as areas for future development. Also, the broader implications of integrating LLMs with MBSE tools for MBSE adoption are considered at the end of the chapter. This chapter primarily addresses RQ3 and RQ4 by evaluating model understanding, query answering, and model modification capabilities (addressing RQ3) and by discussing the potential benefits of LLM integration for MBSE processes (addressing RQ4). RQ2 is additionally ad-

dressed by evaluating the efficiency of the approach used in the plugin to create a textual representation of a SysML model.

**Chapter 6: Conclusions and Recommendations** is the last chapter of the thesis and it provides a summary of the key findings of the research. Limitations of the study are also discussed and possible future research directions are proposed. This chapter addresses all of the research questions by summarizing the findings of the thesis.

# 2 Literature Review and Background Information

## 2.1 Traditional, Document-Based Systems Engineering (DBSE)

As this thesis will discuss Model-Based Systems Engineering the concepts of a system and systems engineering must be understood first. A system can be defined as a technology that is an integrated set of elements, subsystems, or assemblies that accomplish a defined objective [1]. Systems engineering can be defined as a "perspective, a process and a profession ... focused on defining customer needs and required functionality early in the development cycle, documenting requirements, and then proceeding with design synthesis and system validation while considering the complete problem: operations, cost, and schedule, performance, training and support, test, manufacturing, and disposal." [1] DBSE refers to a systems engineering approach that is based on documents and it can be used interchangeably with the term systems engineering. This is because traditionally systems engineering has relied heavily on document-based approaches to capture and communicate system information. In this thesis, DBSE, traditional systems engineering, and systems engineering are used interchangeably and refer to the same concept. [2]

The systems engineering process follows what is known as the system life cycle

(SLC), which is the combined and sequential phases to engineer, produce, operate, and retire a system [2]. Document-Based Systems Engineering (DBSE) is a traditional approach in systems engineering in which the main artifacts that are produced are documents. These documents are created and maintained to describe parts of the system like its requirements, design, verification, validation, and other aspects. [3] The documents can be specifications, interface control documents, system description documents, trade studies, analysis reports, or verification plans for example, which can be generated and maintained throughout the entire system lifecycle [2]. The documents created in DBSE are often standalone and they might have limited integration between other documents. This can lead to challenges in maintaining traceability, consistency, and an overall understanding of the system the documentation is describing. [4]

DBSE can be used in engineering small and large systems, but there are more challenges associated with utilizing DBSE in large and complex systems. As the system grows more complex, the amount of documents that have to be created and maintained increases. With the increasing amount of documents, it becomes more and more difficult to make sure that all parts of the systems are aligned and up-to-date [3]. This problem might be caused by the fact that the information about the system is often so scattered across many documents in such a way that it is very challenging to acquire a good understanding of the whole system. Not having a holistic view of the entire system can cause problems and increase the risk of errors and omissions.

Traditional DBSE approaches can also suffer from problems related to the documentation workflows often being so linear and sequential. The documents are static by their very nature, which can make it difficult to accommodate changes quickly. For example, when a requirement changes, it might require updates to be made in many different documents, which would require a lot of additional manual effort to

make sure that all the documents are kept consistent.

The main challenges of DBSE include:

- **Traceability Issues:** It can be very hard to make sure that every document is always up-to-date and reflective of the state of the system. This can lead to errors, omissions, and inconsistencies between the various documents and the system. [3]

- **Lack of Integration:** The system documents are often created independently, which might cause a lack of integration between the different parts of the system. This can result in a very fragmented view or understanding of the system as a whole. [4]

- **Maintenance Overhead:** As the system evolves, keeping all documents up-to-date requires significant effort, leading to increased maintenance overhead. It takes a lot of effort to constantly apply the process of writing, reviewing, updating, and publishing the documents. [3]

- **Limited Design Semantics:** Information about the system in the documents is captured with natural language and illustrations. However, they might not be completely suitable for describing complex design concepts and they could potentially lead to misinterpretation or over-specification. [3]

- **Communication Challenges:** Information spread across multiple documents can make it difficult to understand the system. Having an incorrect or incomplete understanding of the system can make it difficult to communicate with others about the system effectively. It can take a lot of additional effort to correct these misunderstandings, which hampers effective communication about the system. [2]

- **Measuring Change Impact:** Assessing the impact of changes to the system

can be very complex and time-consuming, especially in situations when there are changes present in multiple documents. [5]

- **Measuring Integrity:** Metrics of the system description like integrity, completeness, quality, and accuracy are mostly gathered by manual inspection. [5]

In complex systems development, these challenges can become quite apparent, as in these cases multiple teams and a lot of stakeholders can be involved. These challenges can also become very apparent in situations where complying with industry standards is crucial, like in the case of the automotive industry with ISO26262 and Automotive SPICE. [3]

The effect of all of these challenges added up together can become very visible. With DBSE there has been a need for improved traceability, better management of the system's complexity, better communication between stakeholders, and a more efficient way of handling changes in the system. These factors have motivated people to search for different approaches, which has led to the development of MBSE.

## 2.2 Model-Based Systems Engineering (MBSE): Principles and Practices

Model-Based Systems Engineering (MBSE) represents a big departure in systems engineering from document-centric to model-centric approaches. This section will present some of the key principles and practices of MBSE. MBSE can be defined as the formalized application of modeling principles, methods, languages, and tools to the whole lifecycle of large, complex interdisciplinary, sociotechnical systems [6]. A simplified definition provided by Mellor et al. describes MBSE as "simply the notion that we can construct a model of a system that we can transform into the real thing" [6].

The International Council on Systems Engineering (INCOSE) has been emphasizing the benefits of MBSE for a while now. INCOSE envisions a transition from traditional document-based systems engineering to a model-based approach where models become the main artifacts of the systems engineering process. There are expectations that this shift will improve the ability to capture, analyze, share, and manage system information more effectively [7].

### 2.2.1 Core Principles of MBSE

Here are some key principles and practices of MBSE that guide its application:

1. **Centralization of System Models**: A big emphasis in MBSE is put on creating and utilizing comprehensive models of a system as a single source of truth. [8]

2. **Comprehensive System Representation**: In MBSE there is more of a focus on the interactions and relationships between different system elements rather than individual components. This approach of focusing on the whole system rather than single components makes it easier to identify potential issues earlier and makes sure that all elements work together well. [8]

3. **Layered Approach**: Going through layers of increasing detail, making sure that all domains (requirements, behavior, architecture, and verification) are addressed at each level before moving to the next. [9]

4. **Purpose-Driven Views**: In the model, every view should have its own defined purpose and scope that would preferably be specific to the needs of the stakeholder. [10]

5. **Lifecycle Continuity**: MBSE can, and is meant to be utilized across the whole system life cycle from the start till to the end. The fact that MBSE

can be used throughout the whole life cycle of a system enables connections between all phases of development, which improves traceability at every level. [8]

6. **Enhanced Communication and Collaboration**: In MBSE models themselves work as a common language for all project participants. This can help all of the stakeholders from different disciplines to get involved in the design process, improving communication and collaboration. [8]

7. **Reusability and Scalability**: Reusability and scalability are promoted in MBSE thanks to model components and patterns being reusable in other places of the model or even in different models. This makes the development of future systems more efficient and makes managing complexity in very large projects more manageable.

8. **Continuous Verification and Validation**: The MBSE approach helps enable continuous verification and validation at each layer of the model. For example, requirements can be traced to model elements and tested, allowing for early detection of errors and problems. [9]

MBSE is a different way of performing systems engineering than traditional document-based systems engineering. In MBSE, a model of the system is built to describe the system in great detail. The model itself can be used to support analysis, specification, design, verification, and validation of the system. In MBSE, models are used to replace the system engineering work that has been done traditionally with a document-based approach. The main output, or the artifact that is gotten from MBSE is a coherent model of the system. [11]

In MBSE, the model of the given system can be seen as the sole source of truth for the system. This is contrary to the way the truth about the system is handled in document-based system engineering, where the truth about the system is spread

across numerous documents of different types. This fact about document-based system engineering can cause concerns about the level of traceability we have of the system; it can be difficult to trace how different parts of the system interact with other parts of the system when those details are described in completely different documents. This is especially true for complex systems that might have hundreds if not thousands of large documents detailing different aspects of the system. [3]

One of the main objectives and the main motivations for adopting MBSE have to do with the model of the system being a great source for facilitating understanding across different stakeholder groups. [11]

## 2.3 SysML: The Language of MBSE

OMG Systems Modeling Language, or as it is better known, SysML, is a general-purpose language meant for graphical modeling. The language supports the analysis, specification, design, verification, and validation of complex systems. It was developed as a subset of Unified Modeling Language (UML) as well as an extension of it and its focus is to address the specific needs of systems engineering instead of software development [11]. According to a study by Ma et al. SysML is also the most widely used modeling language for Model-Based Systems Engineering, which might be due to it being specifically designed to have great coverage of system aspects making it ideal for systems engineering [12].

### 2.3.1 Brief History and Development

SysML started being developed in 2001 when the International Council on Systems Engineering (INCOSE) realized a need for a revision of UML that is designed with systems engineering applications in mind. After that in 2003, the Object Management Group (OMG) joined together with INCOSE to issue a request for a proposal

for a UML profile for systems engineering. Because of this, the SysML Partners consortium was formed and they developed the first SysML specification [13].

SysML (v1.0) was first released in 2007 and several new versions of SysML have since been released. This thesis will be using the SysML 1.6 specification, whose specification was published in December 2019. This is not the newest version of SysML however, as the formal SysML version 1.7 specification was published in June 2024 and SysML version 2.0 specification is in beta [14].

### 2.3.2 Key Features and Diagrams

Diagrams are a key part of SysML, as they provide the engineers with a way to visualize and model various aspects of a system. They can be used to show graphical insights about the system to the various stakeholders. Different diagram types can be used to show off different aspects of the system that is being modeled. There are a couple of different ways to categorize these different SysML diagram types, but I will be organizing them into the following four main pillars [11]:

1. Structure
2. Behavior
3. Requirements
4. Parametrics

Nine different diagram types are included in the SysML diagram taxonomy:

1. **Structure Diagrams**:

   **Block Definition Diagram (BDD):** Shows system hierarchy and classifications.

**Internal Block Diagram (IBD):** Depicts the internal structure within a block and its connections.

**Package Diagram:** Organizes model elements into packages.

2. **Behavior Diagrams**:

**Activity Diagram:** Represents the flow of data and control between activities.

**Sequence Diagram:** Shows the interaction between parts that are working together in a system.

**State Machine Diagram:** Describes state transitions and actions in response to events.

**Use Case Diagram:** Provides a high-level description of system functionality.

3. **Requirements Diagram**: Captures requirements hierarchies and their relationships to other model elements.

4. **Parametric Diagram**: Represents constraints on system parameter values to support engineering analysis.

### 2.3.3 SysML Extensions to UML

As SysML is designed for system engineering needs, it extends UML by introducing new modeling elements and making changes to existing ones. Here are some of the most notable extensions:

**Blocks**: This is the most fundamental unit used to depict structure in SysML. It is an extension of UML's structured classes to represent any level of the system hierarchy [13].

**Ports**: Flow ports are introduced to SysML to provide the user the ability to model the flow of matter, energy, or data between different blocks, in addition to UML's standard ports [11].

**Requirements Modeling**: As requirements are essential for systems engineering, SysML adds its own requirements diagram as well as related constructs to capture and trace requirements through the whole system model [15].

**Parametrics**: To allow the user to model engineering analyses and performance constraints SysML introduces parametric diagrams as well as constraint blocks [11].

**Allocations**: SysML provides allocations relationships so that mappings can be made between different model elements, like functions to components or logical to physical architectures [13].

**Value Types and Units**: SysML introduces value types that include both units and dimensions, enhancing the ability to model quantitative properties. This helps make sure that the number values inside the model are interpreted correctly and also allows for automated unit consistency checks, which reduces the probability of errors due to mismatches in units.

### 2.3.4 Usage of SysML in Practice

SysML is very widely used in different industries like aerospace, automotive, defense, healthcare, and telecommunications. Engineers can use SysML in many different activities:

**Capture and Manage Requirements**: Engineers can model requirements inside SysML, which is how they can ensure full coverage and traceability throughout the whole design process.

**Design System Architecture**: Both logical and physical architectures can be defined with the help of SysML and used to model system hierarchies, interfaces, and interactions.

**Perform Trade Studies**: Parametric models can be used by engineers to evaluate different design alternatives against some predefined performance criteria.

**Support Verification and Validation**: Requirements can be linked to test cases and simulations inside the model and when they are linked, SysML facilitates systematic verification and validation.

**Enable Communication**: SysML diagrams are a visual way of being able to represent the system through various diagram types. The SysML diagrams can help communicate complex system designs to stakeholders who might not be familiar with technical details.

**Integrate with Other Tools**: A model-based development environment can be supported with SysML by integrating the SysML models with simulation tools, code generators, and other engineering software.

### 2.3.5 Benefits of SysML in MBSE

SysML is used extensively in MBSE for many good reasons. Here are some of the advantages of using SysML in MBSE:

**Standardization**: SysML provides a standard language for systems engineering, which improves communication among both stakeholders and tools [16].

**Comprehensive System Representation**: SysML allows users to model various aspects of the system, including structure, behavior, requirements, and constraints, all within a single integrated model [11].

**Traceability**: The allocation and requirement relationships in SysML provide traceability between different model elements and across the system life cycle [13].

**Tool Support**: SysML is widely supported by most MBSE modeling tools [5].

**Integration with Other Disciplines**: SysML is rooted in UML, which allows integration with software engineering practices and tools [11].

### 2.3.6 Challenges and Limitations

Along with its benefits, there are some challenges with SysML:

**SysML-C1: Learning Curve**: As SysML is quite comprehensive it can be quite difficult for new users to learn and use it effectively. However, it might be easier to learn for people who are already familiar with UML, as SysML is based on UML [5].

**SysML-C2: Tool Dependence**: How effective SysML modeling is depends mostly on the capabilities of the MBSE modeling tool that is chosen [16].

**SysML-C3: Model Complexity**: Large and complex SysML models can become more difficult to manage and navigate as they continue to grow [13].

**SysML-C4: Customization Needs**: There might be a need for some organizations to customize SysML to fit their specific domains or processes, which can lead to interoperability challenges [11].

If SysML is the language being used for modeling in MBSE, having a good understanding of SysML and the things it is capable of is very important for properly implementing MBSE practices. This is why minimizing these challenges is important for the success of the efforts of transitioning to MBSE. Chapters 4 and 5 will present

one potential solution to minimizing mainly the challenges SysML-C1, and SysML-C2 by leveraging the utility of LLMs in a modeling program.

## 2.4 MBSE Adoption: Benefits and Challenges Reported in Literature

In the literature, there are reports of several potential benefits of adopting Model-Based Systems Engineering, as well as challenges that organizations face in this process. This section will summarize the key findings from existing studies on MBSE adoption.

### 2.4.1 Benefits of MBSE in Literature

Various studies have identified purported benefits that organizations have realized or expect to realize from MBSE adoption:

**B1: Improved Communication and Shared Understanding:** MBSE models provide a common language and representation that improves communication among stakeholders and ensures that everyone has a shared understanding of the system [17], [18].

**B2: Enhanced Traceability and Consistency:** MBSE facilitates maintaining clear traceability between requirements, design elements, and verification activities, which can reduce the number of errors and omissions [9].

**B3: Improved Verification and Validation Capability**: MBSE was found to have a high positive perception for verification and validation capabilities [19].

**B4: Reduced Time and Cost**: MBSE can reduce the development time and costs, especially in later stages of projects after the model or models themselves have been developed [2], [19].

**B5: Better Decision Making**: Approaches that were using MBSE were reported to help aid in decision-making and design reasoning [19].

However, it is important to acknowledge that the empirical evidence supporting the benefits of MBSE can vary quite a bit. A systematic review by Campo et al. found that 47% of claims about MBSE benefits were completely based on author opinions instead of empirical evidence. Also, 14% of the claims provided at least some metrics to support their statements about MBSE. [19]

In another systematic literature review, Henderson and Salado examined the literature on MBSE and its benefits from 1998 to 2019. The keywords were limited to "Model-Based Systems Engineering" "Model-Based Systems Engineering" OR "MBSE" and the only inclusion criteria was mentioning MBSE benefits. They analyzed 360 papers, each of which cited benefits of MBSE for an aggregated count of 1233 individually claimed benefits [17].

Out of all the papers that had claimed benefits of MBSE, 240 papers (66.7%) were identified citing benefits to MBSE without supporting evidence of any kind (claims had not been formally measured nor observed as part of an actual implementation of MBSE). 109 papers (30.3%) were identified including referenced benefits of MBSE from another source, which means that the claim is based on a reference source instead of the author's opinions. 36 papers (10.0%) included observed benefits of MBSE. These kinds of informally observed benefits of MBSE can be useful or valuable, however, to prove these observations to be true and not anecdotal they need to be formally measured with scientific scrutiny [17].

Only 2 papers (0.6%) were identified having measured the benefits of MBSE with both of them having some issues with their methodology. The first one used qualitative scoring by industry experts to assess the benefits, but details were not given on the elicitation process. Also, the benefit/cost ratio was based on estimations instead of actual project data, and the comparison between MBSE and document-

based approaches lacked clarity on team characteristics and potential biases [20]. The second paper predefined challenges that MBSE was expected to address, which has the potential to bias the team's focus. Additionally, the study assumed MBSE was the only factor in achieving success without considering other variables that could have influenced the outcomes. [17], [21]

The literature review indicated that there was a lack of empirical evidence present in the publicly available literature between the years 1998 and 2019 that supports the hypothesis that MBSE is beneficial for the development of engineered systems. Additionally, the most frequently cited benefit, better communication/ information sharing, makes up a little over 10% of the total benefits cited, which suggests that there is not an agreement in the community on what the main benefits of MBSE are according to the literature review. MBSE possessing a lack of empirical evidence does not however mean that it is disadvantageous or that it does not have benefits, it means that we do not know if is beneficial or not. To provide empirical evidence of MBSE's benefits, the benefits would have to be measured formally with scientific scrutiny. [17]

There are however some other studies that have provided concrete data demonstrating the advantages of MBSE in complex systems development.

Rogers and Mitchell [22] conducted quite a comprehensive case study on the application of MBSE in the Submarine Warfare Federated Tactical Systems (SWFTS) program, which involves an evolutionary development of a complex system-of-systems (SoS). They compared the traditional systems engineering approach with an MBSE approach and observed improvements:

**B6: Reduction in Systems Engineering Effort:** Compared to the traditional systems engineering approach the MBSE approach led to an 18% decrease in the systems engineering effort required per requirement change. This gain in efficiency was attributed to the improved ability to manage and process

interface requirements changes using various model-based methods [22].

**B7: Early Defect Detection:** Changing the approach to MBSE allowed for earlier detection of defects during the integration phase rather than during platform testing. A 37% reduction in defects was discovered during platform testing, which indicates that issues were identified and resolved earlier in the development life cycle. It is better to detect defects earlier rather than later as defects found later cost more to fix and resolve [22].

**B8: Overall Quality Improvement:** Using MBSE the total amount of interface defects per requirement was reduced by 9%. This gives more evidence that MBSE can also improve the quality of the systems engineering artifacts on top of enhancing efficiency, which can lead to a more reliable system overall [22].

This case study by Rogers and Mitchell also provided an analysis of the economics of MBSE and demonstrated a positive Return on Investment (ROI) from adopting MBSE. There were cost savings found from the MBSE approach that were attributed to the reduced labor hours and earlier defect detection and resolution. These savings were able to make up for the initial investment in MBSE tools and training [22].

### 2.4.2 MBSE Adoption Challenges in Literature

According to the literature, numerous purported benefits set MBSE apart from DBSE, but many challenges have been faced in attempting to adopt it. A survey was conducted by Chami and Bruel to identify and analyze the most common, main challenges in MBSE adoption [23]. The following are the key challenges they identified:

**C1: Awareness and Change Resistance:** The human factor plays an important role in adopting MBSE, especially when the key players have different levels of

MBSE knowledge and not enough time is given for training. This can increase the resistance to change as existing approaches can feel safer.

**C2: Purpose and Scope Definition:** Deciding and defining a clear purpose and scope for MBSE adoption is important but difficult in real-world applications. It is not easy to describe exactly why and what should be modeled before starting to deploy.

**C3: Upfront Investment:** Adopting MBSE is not free, as it requires a substantial upfront investment, especially if it has not been considered before. This will include such things as determining an effective investment strategy, accurate cost estimation, and quantifying its estimated return on investment.

**C4: Executive Level Sponsorship:** The increased popularity of MBSE has increased the readiness for executive-level support for MBSE initiatives, but there are still conflicting goals in MBSE adoption. Some employees mostly care about low-cost adoption, while others are more focused on maximizing adoption quality and more interested in long-term solutions.

**C5: Adoption Strategy:** There are two main approaches for adopting MBSE, which are off-cycle adoption and on-cycle adoption. Choosing the wrong strategy can hurt the benefits of MBSE. Off-cycle adoption refers to adopting MBSE in a sandbox environment, which is considered ideal as it requires less budget and time. On-cycle adoption refers to adopting MBSE directly on productive projects, which is a lot more challenging and introduces new costs for running projects.

**C6: Method Definition and Extension** One important factor to consider when adopting MBSE is the method that will be used. It can often be necessary to customize an appropriate method according to the needs. It is challenging

to set up the required method, document it, and facilitate it with modeling rules, guidelines, and customizations to tools and training materials. Also, new method extensions might be needed in the future which can bring further challenges.

**C7: Modularity and Reusability:** There is a problem in many organizations in which an isolated reuse approach is used, where a set of data is copied and pasted from one context to the next. The unfortunate fact is that this can still happen even in MBSE and its system models, which ultimately results in losing the "source of truth" as soon as the copied source or pasted target is changed.

**C8: Complexity Management:** The systems grow more and more complex as the number of components, functions, and interactions grows. The increasing number of model elements and the dependencies between the elements and the model(s) can take the existing methods and tools to their limits.

**C9: Tool Dependency and Integration:** To adopt MBSE the companies need to pick a set of tools and training material for those tools so that everyone is working on the model(s) with the same set of tools. This is not an easy thing to decide on as there exists no tool that does everything perfectly. Additionally, the integration between systems modeling tools and other tools like requirement tools is still solved with specific solutions.

**C10: Different Methodologies**: Models created by different teams may be very hard to integrate into other models or to interpret by others if the different teams are using different methodologies.

**C11: Large Models Visualization:** As a system model becomes larger and grows in complexity, it starts to become harder to navigate, understand, and

present in a clear way. In a growing model, it can become increasingly challenging for team members to have a good understanding of the whole model. Also, finding specific information or understanding relationships between different parts can become more difficult.

**C12: Integration with Legacy Systems**: Most organizations very likely have existing systems and documentation that would need to be included in the MBSE environment in a way that makes sense. Migrating this old legacy data can take a lot of extra time and may even require complex custom solutions

## 2.5 MagicDraw and Teamwork Cloud: The MBSE Tooling Environment

MagicDraw is a well-known modeling tool that has been developed by No Magic, who are now part of Dassault Systemes. The tool can support different modeling languages like UML and most notably for us, SysML. It is the tool that we are utilizing for this research as it is widely used in the industry for MBSE applications and it is the tool that has been used in the preliminary efforts to model the BTS. This section will give a short introduction to the MagicDraw tool as well as its collaboration platform Teamwork Cloud and their relevance to this study.

### 2.5.1 Overview of MagicDraw

MagicDraw is a modeling platform that supports SysML modeling. MagicDraw provides its users with a user-friendly interface, and the ability to create diagrams and features that allow for modeling collaboratively. Some of the key features of MagicDraw are:

- Full support for UML and SysML

- Model validation and consistency checking and maintaining
- Customizable reporting and documentation generation
- Version control and team collaboration features
- Extensibility through plugins and APIs
- Source code reverse engineering
- Traceability between models with project usages
- Model analysis facilities

MagicDraw is quite flexible within the Software Development Life Cycle, giving the users the ability to adapt their Software Development Life Cycle to specific business requirements. Another factor that gives MagicDraw more flexibility is that it is not dependent on any given methodology. This allows companies to create their approach or adopt another existing methodology depending on their company-specific requirements [24].

### 2.5.2 SysML Plugin

Since SysML is being used as the modeling language of choice in the modeling of the BTS, we need our modeling tool, MagicDraw, to support the SysML standard. SysML is supported in MagicDraw through an official SysML plugin, which provides this support to MagicDraw. This plugin has been used in the efforts so far to model the BTS. The SysML plugin supports the specification, analysis, design, verification, and validation of complex systems [25].

This SysML plugin provides support for the SysML standard and these features improve the MagicDraw's capabilities for systems engineering, all of which are essential for the modeling efforts on the BTS [25].

### 2.5.3 Teamwork Cloud for Collaborative Modeling

Teamwork Cloud is designed to work well with MagicDraw while also adding collaborative functionality to work on models with others simultaneously. Teamwork Cloud has been used in the efforts so far to model the BTS. The following are the main features of Teamwork Cloud, that are the most relevant to this thesis:

- Centralized model repository for shared access
- Version control and branching capabilities
- Concurrent multi-user editing of models
- Access control and user management
- Scalability for handling large models

All of these features help manage the complex, large-scale, and collaborative nature of the BTS modeling project [26].

### 2.5.4 Relevance to This Research

MagicDraw, its SysML plugin, and Teamwork Cloud are relevant to this research for many reasons. The following are the main reasons:

**Industry Standard:** They are used widely in the industry, which ensures that this research applies to real-world situations.

**Extensibility:** MagicDraw provides the capability to extend its functionality with its plugin architecture, which allows anyone to build their plugin for MagicDraw. This is crucial for the LLM integration proof-of-concept, which will be presented in Chapter 4 and evaluated in Chapter 5. This also helps minimize the challenge associated with SysML of tool dependence, SysML-C2, as the limits of the modeling tool can be extended.

**Model Representation:** MagicDraw's internal model representation and its API make it possible to extract the information about the model in a format that can be sent to and processed by LLMs.

**Scalability:** MagicDraw's and Teamwork Cloud's capability to handle large, and complex models is very important for modeling systems like the BTS. Also, project usage is another feature that helps scale up the project by maintaining traceability between multiple different models of the system.

**Collaboration:** Teamwork Cloud and its features can facilitate multiple teams with multiple people modeling the BTS collaboratively at the same time.

**System Engineering Support:** The SysML plugin provides MagicDraw support to the SysML standard, which facilitates systems engineering practices that are important for working on complex systems like BTS.

### 2.5.5 Potential Challenges in Using MagicDraw and Teamwork Cloud for Large-Scale Systems

These tools are powerful for modeling and performing MBSE activities, but there still exist challenges with modeling large-scale systems like a BTS. Here are some potential challenges:

**Performance:** There can be problems with large models leading to slower performance when there are too many elements, relationships, and diagrams in the model. Issues can also arise when too many people are working on the same model at the same time in Teamwork Cloud.

**Collaborative Complexity:** Teamwork Cloud's models are meant to be shared and worked on collaboratively, but sometimes coordinating work on a shared model among various teams can prove to be challenging.

**Learning Curve:** There is quite a steep learning curve for new users to learn the syntax of SysML, features of MagicDraw and Teamwork Cloud, and modeling the system. This challenge is one of the motivations for our research into integrating an LLM with a modeling tool, MagicDraw.

# 3 Transitioning to MBSE for BTS Development: A Case Study Analysis

## 3.1 Introduction to the BTS Case Study

### 3.1.1 The BTS System to be Modeled

When it comes to cellular networks, a Base Transceiver Station (BTS) is a vital piece of equipment that enables wireless communication between user equipment like mobile devices and the broader network. The BTS is responsible for crucial tasks like facilitating wireless communication through the transmission and reception of radio signals, handling complex signal processing, and managing different aspects of cellular connectivity.

BTS architecture can be very complex and involve many different domains. The BTS includes various systems, units, and components like an antenna system, a radio frequency unit, a baseband unit, a power supply, a cooling system, a back-haul connection, and an Operations and Maintenance (O&M) interface. An antenna system transmits and receives radio signals to and from mobile devices and it can include many antennas for different frequencies and technologies like 2G, 3G, 4G, and 5G. A radio frequency unit processes the radio signals, including transmitters, receivers, and power amplifiers, and it also converts between analog radio frequency signals and digital baseband signals. A baseband unit handles digital signal process-

ing, manages radio resources and protocols, and interfaces with the core network. A power supply provides power to all BTS components and the cooling system manages temperature to ensure continuous good performance of equipment. A backhaul connection links the BTS to the core network and can be a fiber optic, microwave, or some other high-capacity connection. An Operations and Maintenance interface allows remote monitoring and management of the BTS.

### 3.1.2 Challenges in the Initial Modeling Efforts for BTS

One of the biggest challenges is the upfront time investment that is required to learn a whole new way of working. Adopting MBSE practices means that everyone working on the system will be required to have a sufficiently good understanding of SysML and its diagrams to perform the modeling work properly. The people who will be working on the model itself and making diagrams for it will additionally need to be good at using the modeling tool and will have to understand and follow the MBSE methodology chosen. Most people working on the models of the BTS were not familiar with MBSE or SysML previously, which has caused difficulties in understanding how the system should be modeled according to the SysML standard. This also exemplifies the steep learning curve challenge of SysML (SysML-C1) and the challenge of awareness and change resistance (C1) identified in Chapter 2. This means that a lot of upfront investment has to be spent on educating employees on MBSE, SysML, and modeling with MagicDraw. This upfront investment can for example be in the form of formal training by a third party that is paid for the employees by the employers or in the form of dedicated work time spent on learning.

There have also been different opinions on how the system should be modeled. This challenge reflects both the method definition challenge (C6) as well as the challenge of different methodologies (C10) seen in MBSE adoption literature. Not having a common understanding and agreement on the way that the system should

be modeled can cause problems, because if the people working on the models of the system have different approaches to modeling the system, then the various parts modeled by different people might end up not being compatible with each other. This can create problems with the models of the system, which requires additional time in fixing and consolidating the different parts of the models.

Another challenge is that the BTS is an already existing, large, and complex system that has been developed for a long time using Document-Based Systems Engineering approaches instead of MBSE. This means that the system can not be modeled from the ground up as it is being developed, which would make it easier to ensure that the system corresponds perfectly with the system that is being developed at the same time.

### 3.1.3 Current Processes in the Systems Engineering of the BTS

When it comes to the systems engineering of the BTS, specification plays a key role. One of the most important processes that is being used in the specification is CFAM (Common Feature Analysis Module), which describes the common process for feature specification. The process is used as a standardized approach for specifying new features across different product lines. It is used to help analyze the impact of a new feature on existing systems and it defines requirements for implementation. CFAM is typically started after the decision to proceed with developing the feature.

CFAM process uses four checkpoints to track progress, CP1: Feature scope and initial user scenarios, CP2: Detailed system-level requirements, CP3: Entity-level requirements and analysis, and CP4: Final comprehensive specification.

A CFAM document is produced in the process to serve as a comprehensive specification for the feature. Specifically, it captures the analysis, impacts, and requirements for implementing the feature across different levels of the system architecture.

A major reason and motivation for starting to adopt MBSE and model the BTS was to increase the efficiency of the specification process. This necessarily means, that the models of the BTS should be integrated into the current processes for the specification of the system or that the processes should be changed so that they use models of the system as the main sources of information by design.

## 3.2 Potential Benefits of Transitioning to MBSE for BTS Development

### 3.2.1 Improved System Understanding and Communication

In the development of a complex system like a BTS, which involves multiple domains, a major challenge for the stakeholders can be maintaining a complete understanding using a document-based systems engineering approach. In our literature review performed in Chapter 2, improved communication and shared understanding (B1) were identified as some of the most frequently cited benefits of MBSE. This benefit can be seen in many different ways in the context of developing a BTS.

Having a central model that can serve as a single source of truth for the BTS, including its subsystems and the interactions between them can be beneficial in addressing the challenge of information about the system being too scattered across many documents. This aligns with the improved communication and shared understanding benefit (B1) that has been reported widely in MBSE literature.

The model of the BTS can also provide multiple different types of views that are meant for the needs of different stakeholders, which can include views of a higher abstraction level meant for system architects as well as very detailed, technical views meant for teams responsible for the implementation.

Communication across different domains working on the BTS can become more

effective as all its subsystems are represented in a single, standardized modeling language.

SysML diagrams are a great visual way to be able to communicate complex relationships between the many components of the BTS, both logical and physical, improving the ability of the teams working on different subsystems to understand their interfaces and dependencies.

In a document-based approach, the new members may have more difficulty learning about the system from many disjointed documents, some of which they might miss. This is why a model of the BTS could also potentially be helpful in terms of helping new members understand the system structure more quickly, addressing the challenge of knowledge transfer.

### 3.2.2 Enhanced Traceability and Consistency

Maintaining traceability between requirements, design components, and implementation is crucial, especially for complex systems like BTS. The benefit of enhanced traceability and consistency (B2) was identified in our literature review and is very relevant for BTS development.

Changes that happen in one part of the system can be traced to their impacts on the other parts of the system, which exemplifies the enhanced traceability and consistency benefits (B2). This is very useful for managing the dependencies between BTS subsystems.

In MBSE the traceability between different levels of the system is made more explicit and clear, higher level system requirements can be traced down to individual specifications of components for example.

Traceability between the different aspects of the BTS can be ensured with a comprehensive model of the system, which can help reduce the risk of issues between different BTS subsystems.

### 3.2.3 Potential for Improved Validation and Verification

As the operations of BTS are safety-critical by their nature, the improved verification and validation benefit (B3) of MBSE is clearly relevant for BTS development.

The model of the BTS can be used to verify system behavior early in the development process, similar to the early defect detection benefit (B7) that was observed in the SWFTS program [22]. As mentioned in Chapter 2, a more systematic process for verification helped realize the overall quality improvement benefit (B8) in the SWFTS program.

By performing a model analysis some properties of the system can be verified automatically, reducing the need for manual verification and thus reducing the risk for human error.

Another potential benefit is that integration testing, in which different parts of the system are tested together, can be better planned by understanding the specifications of the interfaces from the model.

As mentioned in Chapter 2, a more systematic verification process helped realize the overall quality improvement benefit (B8) in the SWFTS program. [22]

### 3.2.4 Long-term Cost and Time Savings

Adopting MBSE does require a substantial amount of initial investment, however, many factors can contribute to savings in the long-term, as identified in our literature review with the reduced time and cost benefit (B4) and the reduction in systems engineering effort benefit (B6).

There can be a potential to reduce efforts in maintaining system documentation as the model becomes the main source of information about the BTS and any documents will be generated from the model.

Adopting MBSE has shown a potential for earlier detection of design issues through model analysis as identified in the SWFTS program contributing to a 37%

reduction in defects during platform testing.

Various model components can easily be reused across different BTS variants and generations, resulting in time savings.

Using MBSE has also been shown to improve efficiency in systems engineering activities, as observed in the reduction in systems engineering effort benefit (B6) demonstrated by the 18% decrease in effort per requirement change in the SWFTS program.

### 3.2.5 Facilitated Change Management and System Evolution

The BTS is not an unchanging, static system, instead, it is continuously evolving with new features and technologies. This is why managing change and system evolution is crucial for BTS development, aligning well with the improved decision-making benefit (B5) that was identified in our literature review.

The model facilitates systematic analysis of the changes that have been made across the whole system before implementation. To help with the version control of the system design, model versioning can be used to make handling different versions of the system design more manageable. With the help of models, different design alternatives can be evaluated more systematically.

A model of the system can be used to help in planning the integration of new features into the BTS by understanding all the interfaces that are affected by the new feature.

## 3.3 Key Considerations for MBSE Transition in BTS Development

Although there are many purported benefits for transitioning to an MBSE approach, there also exist multiple challenges as identified in our literature review that should

be considered during the transition in BTS development. This section will analyze how the identified challenges are relevant to BTS development and propose strategies to address them.

### 3.3.1 Awareness and Change Management

The development practices for BTS development have been established for a long time and BTS development involves a large number of stakeholders, which makes the challenge of awareness and change resistance (C1) very relevant. As previously mentioned in Section 3.1.2, most team members were not previously familiar with the concepts of MBSE or SysML, which has led to difficulties in adoption.

Additionally, the challenges of method definition (C6) and complexity management (C8) are very important to address due to the multi-domain and complex architecture of the BTS. Multiple levels of training could be useful in controlling the amount of training each stakeholder group needs.

To address this challenge of awareness and change resistance in the BTS context there could be multiple helpful strategies. To help motivate different stakeholder groups, different benefits of an MBSE transition could be emphasized. Enhanced motivation could help alleviate the resistance to change, making the transition process smoother. For example, an emphasis could be put on improved system overview and traceability for system architects, while an emphasis could be put on cleaner interface specifications and requirements for developers.

Another important factor to consider is that the value proposition of the transition should be clearly communicated. This can help justify the efforts being made in the transition to help offset the initial investment spent on it and decrease the resistance to change.

It is also important for the many key stakeholders responsible for various subsystems of the BTS to get involved early in the important decisions involving the

transition, including the methodology used in modeling the system. This would help ensure buy-in from the various stakeholders and would help with considering the many domain-specific needs of the various stakeholders.

### 3.3.2 Methodology and Training Strategy

Due to the multi-domain and complex architecture of the BTS, the challenges of method definition (C6) and complexity management (C8) are very important to address. Having multiple levels of training could be useful in managing the amount of training each stakeholder group requires. For example, system architects would likely require more comprehensive SysML and methodology training, while other domain engineers might only require training related to the specific modeling work that they are responsible for.

In order to ensure consistent modeling approaches used to model the BTS across different teams, common and clear modeling guidelines should be established. Additionally, to manage the scale of the models of the bTS, model decomposition strategies must be implemented to ensure that the sizes of the models of the BTS are kept reasonable.

### 3.3.3 Tool Integration and Legacy System Handling

The BTS has extensive documentation, which makes the challenges of tool dependency (C9) and integration with legacy systems (C12) very relevant for BTS development. To address these challenges, there are many factors to consider. MBSE tools and utilizing the models of the BTS should be integrated into the existing CFAM process to maintain continuity in the specification process.

Also, a strategy should be developed for how legacy documentation will be handled with the transition to MBSE. It is necessary to prioritize which existing documentation should be transformed into models. The strategy should consider how to

maintain traceability between legacy documents and new models, as well as how documentation will be generated from the models to support the existing stakeholders who rely on traditional documentation.

### 3.3.4 Model Organization and Scalability

As previously discussed, BTS is large and complex, which makes the challenges of modularity and reusability (C7) and large models visualization (C11) critical to address. To address these challenges, the BTS should be decomposed into models according to the chosen methodology. From a high-level view, each major subsystem should have its model, with clean interfaces between the decomposed models, and reusable component libraries for common elements.

MagicDraw provides guidelines for model decomposition. According to them, a model should be decomposed when the team size exceeds 20 concurrent users, the model complexity impacts performance, or when different teams need to work independently. [27]

To maintain traceability between the various models of the BTS, MagicDraw's project usage feature can be used. It is able to maintain relationships between the different decomposed models, allows the models to cross-reference each other, and supports performing an impact analysis across the entire system through all the models.

## 3.4 Conclusion

This chapter has addressed RQ1 by examining the potential benefits, challenges, and considerations that should be addressed when transitioning from DBSE to MBSE in the context of BTS development. The complexity and size of the BTS system make it a great candidate for MBSE adoption.

The analysis of the potential benefits shows that MBSE could specifically address current challenges in BTS development:

- A complete model representation would improve the management of complex dependencies between the different BTS subsystems more effectively compared to the current document-based approach.
- Similarly, a model-based approach could increase traceability, thus improving the management of changes across different BTS variants and generations.
- The CFAM process could be streamlined using the model-based approach by providing a single source of truth for feature specifications.
- Verification could be performed early through model analysis, which could help detect integration issues between different parts of the BTS before implementation.

There are also significant challenges that need to be addressed:

- Careful consideration is required during the transition due to the extensive amount of legacy documentation and established processes around BTS development.
- BTS development involves multiple teams from multiple domains, which necessitates well-planned training and a carefully considered methodology strategy.
- Due to the size and complexity of the BTS system a careful model decomposition and organization strategy is required.

The main considerations that were identified for a successful MBSE transition in BTS development emphasize the need for the following:

- Using a phased approach that gradually integrates with the existing processes.

- Designing clear guidelines for modeling that address domain-specific needs.
- Creating a proper model decomposition strategy that utilizes MagicDraw's project usage feature.
- Including comprehensive training material or teaching tailored to different stakeholder groups.

# 4 LLMs in MBSE and Plugin Implementation

## 4.1 Introduction to Large Language Models (LLMs)

This section will give a brief introduction to Large Language Models (LLMs) and their capabilities.

### 4.1.1 Brief Overview of LLMs

Large Language Models are a subcategory of AI (Artificial Intelligence) and they are a cornerstone of the field of natural language processing (NLP). These models are based on deep learning architectures with many different types of layers and they have been shown to demonstrate their notable capabilities in both understanding and being able to generate human-like text. This kind of behavior has been for a long time thought to have been outside the purview of machines but has recently become possible thanks to the advancements in LLMs. [28]

Tokenization is extensively used in LLMs as a pre-processing step as the text written in natural language needs to be made readable and understood by machines. Tokens can be words, subwords, characters, or symbols depending on the process used in the tokenization. The tokenization process determines how the text is broken down into tokens, which is important for LLMs as it is the deciding factor on how

the model processes and understands language [28]. In LLMs, a very important concept to understand is the context window, which is the amount of tokens that the model can process at once. The size of the window determines the amount of context that the model can have at any moment when generating or analyzing text. With a larger context window, more text can be analyzed or generated by the model. The context window size can vary quite a bit, a few years ago it may have been about 8000 and now for example OpenAI's most recent models have a context window size of 128 thousand [29] and context window sizes can go up to Google Gemini 1.5 Pro's 2-million-token context window [30].

LLMs themselves are neural networks that have been trained on extremely large amounts of textual data that have been converted into tokens, which is why they can often include billions of parameters. The way LLMs work is that they are always designed to predict the next token in a sequence, which allows them to generate text that is coherent and relevant in the context [28]. This use of tokens and predicting the next token has proven to be very flexible in its use cases as LLMs can be used to perform a very wide range of tasks related to language without necessarily needing to explicitly train the model on that specific task [31].

The transformer model is typically used in the architecture of LLMs. They use self-attention mechanisms to process input sequences, which enables the model to capture long-range dependencies within text. This is a very important ability, especially for understanding complex language structures [28]. LLMs that use the transformer architecture and have a very large scale can capture more intricate patterns and relationships inside language than smaller models [32].

### 4.1.2 Recent Developments and Capabilities

In recent years Large Language Models have seen massive advancements in terms of their capabilities. This has greatly increased the amount of potential use cases in

all kinds of different industries and domains, which really shows how versatile LLMs can be.

One of the largest and most significant developments in LLMs is the inception of what researchers call "emergent abilities". These abilities are defined as capabilities that are not present in smaller-scale models but appear as the models are scaled up in size and complexity. Larger models have been shown to have improved performance in tasks like arithmetic reasoning, commonsense reasoning, and even forms of logical deduction for example. Also, these abilities seem to manifest suddenly once the scale and complexity of a model reach beyond a certain point instead of improving incrementally step-by-step. [31]

There have also been large improvements in the models' few-shot and zero-shot learning capabilities that have been led by the scaling up of the LLMs. This results in the models being able to perform tasks with either very few examples in the case of few-shot or even with no specific examples in the case of zero-shot of the task in question. This has also increased the amount of potential applications of LLM, as they require less, or no fine-tuning to adapt to new tasks. [31]

The size of the LLMs' context windows has also been increasing. This can be very useful for many different use cases as a larger context window means the ability to feed more information to the model. A lot of use cases would be impossible without a sufficiently large context window and as the size of LLMs' context windows are ever-increasing more use cases for LLMs will surely become more plausible.

There are a couple of factors that have enabled the rapid advancement of LLMs in the past years. One of them is the diversity and the amount of data used when training the current LLMs. The shift into using text from all kinds of domains has improved the models' ability to generalize and be able to do an extremely wide variety of tasks. The continuous advancement in computation has also been crucial since the models are getting larger and more complex which is increasing their

computational demand. LLMs are trained with GPUs (Graphics Processing Units), TPUs (Tensor Processing Units), and NPUs (Neural Processing Units) as they are able to perform the types of calculations that are performed during the training of an LLM which is a lot of matrix calculations in parallel. Additionally, algorithmic innovations have been essential in propelling LLMs. The transformer architecture is a prime example of these algorithmic innovations as all the major LLMs seem to be currently based on it. There are also different kinds of transformers, which target specific aspects for optimization like enhancing the attention mechanism, refining pre-training objectives, and cutting down on parameter counts. [33]

Thanks to these factors among others LLMs have found many applications in various domains. LLMs can understand natural language, generate human-like text for various purposes, generate code, answer questions, and give suggestions for example. These capabilities make LLMs quite a powerful tool in many different industries like finance, healthcare, education, and software development. In the next section, we will be looking into a fairly novel idea of using an LLM to help in MBSE and modeling a system.

## 4.2 Potential of AI in MBSE

In this section, we will be specifically looking at how AI can be helpful in MBSE.

### 4.2.1 Current Research on Applications of AI in MBSE

There is some very recent research that has explored different applications of Artificial Intelligence (AI) in Model-Based Systems Engineering (MBSE).

In 2019, Chami et al. [34] proposed a framework for generating SysML model elements from semi-structured text using natural language processing and machine learning called SysDICE. They utilized named entity recognition (NER) in their

approach to extract SysML entities like actors, use cases, and blocks from text. Some of the key aspects included a data labeling tool for domain experts to annotate training data, an NLP (Natural Language Processing) framework to train and apply the NER model, and integration with SysML modeling tools to import the extracted entities. They showcased the approach on various use case descriptions from the rail sector, in which it managed to automatically extract relevant SysML elements.

Bader et al. [35] explored using a GPT model that was fine-tuned to generate UML component diagrams from natural language descriptions. Their approach included creating a dataset of labeled UML examples, fine-tuning GPT version 3.5 turbo on the UML dataset, and generating XMI code for new UML diagrams from text prompts. In the results of their research, they found that the model could accurately generate basic UML elements and relationships. A massive disadvantage that they found with this approach was however that XMI is extremely verbose which can easily become a problem with more model elements because of the LLMs' context window limitations, and they were also faced with issues with handling unique identifiers.

A pilot study was conducted by Crabb et al. [36] in which the goal was to assess the impact of integrating LLMs into the MBSE workflow. They had multiple groups of systems engineers create SysML models with varying levels of assistance from LLMs. There was a group with no LLM access which acted as the control group, a group with access to draft materials generated by an LLM, and a group with open access to query an LLM while performing modeling. They found that models created with open LLM access scored the highest on metrics like correctness and simplicity, experienced modelers with LLM access produced significantly more model artifacts in less time, and that LLM-generated draft models contained about 39% of the required content but it only took about 5% of the time compared to the others.

A study conducted by Timperley et al. [37] assessed the use of LLMs for generating the design of spacecraft system architectures. They developed a tool with Python to integrate OpenAI's GPT model with the Capella MBSE tool, enabling the generation of system functions, modes, and components from requirements. They held experiments on multiple spacecraft designs which showed that the generation of system modes and components achieved high traceability and alignment with the existing designs. They also showed that function generation was more challenging, with worse traceability to input requirements, and that the generated architectures were typically more detailed than their existing counterparts. The authors talked about the possibility that their method could be used as an "AI assistant" for spacecraft system engineers, with human input still remaining crucial in the process.

A proposal was given by Patet et al. [38] that integrating NLP-based tools with existing MBSE software could be used to ease MBSE adoption. They presented a framework for extracting requirements from unstructured documents with NLP and generating SysML model entities by training an NLP model on labeled data. The authors argued that these kinds of ML applications could be integrated with MBSE tools to greatly reduce the effort required by systems engineers when it comes to documenting requirements and generating SysML models.

These studies have shown some potential for AI to speed up various MBSE tasks like creating and working on the initial model, extracting requirements, and retrieving information. The common consensus among these studies seems to be that there are still limitations in handling complex relationships and that there is still going to be a continued need for human expertise in these tasks. AI assistants will most likely become more and more relevant and integrated into MBSE tools and workflows, which means that it will be the responsibility of the systems engineers to help guide the AI in this process to ensure good results.

### 4.2.2 Opportunities for AI in MBSE Processes

A vision paper was written by Mohammad Chami et al. [39] in which capabilities of AI for effective Model-Based Systems Engineering are explored. The paper identifies many capabilities that AI has that could benefit MBSE processes. Here are some of the most relevant capabilities for this thesis:

**Data Querying Techniques**: AI could be useful in supporting productive semantic querying techniques to answer common systems engineers' questions about the content of the model.

**Graphical Visualization**: AI could be used to improve the visualization of models graphically, especially for large and complex models, which can become quite difficult to comprehend.

**Knowledge Acquisition**: AI could be used to elicit domain experts' knowledge, analyze and put this knowledge formally into the model or models of the system, and validate this knowledge.

The International Council on Systems Engineering (INCOSE) has also talked about the capabilities of AI in MBSE in their Systems Engineering Vision 2035 [40]. They identified multiple opportunities for AI in MBSE processes:

**Enhanced Modeling and Simulation**: Major changes are being expected in systems engineering methods and tools by AI. This includes improved modeling, simulation, and visualization environments that give the systems engineers the ability to test and assess a multitude more alternative designs quicker and in a more thorough way.

**Intelligent Assistance**: The Systems Engineering Vision 2035 expects systems engineering tools to be augmented with AI by 2035, which would help to shift a lot of the routine tasks from the engineer to the computer. The benefit of

this would be that the systems engineers would have to spend less time on mindless work and they could instead spend more time on creative tasks.

**Better Decision Making**: Tools integrated with AI are predicted to help drive design activities in collaboration with systems engineering. This could potentially help avoid bad design choices and support the design intent.

**Natural Language Processing**: LLMs using Natural Language Processing (NLP) are expected to be used in helping systems engineers write improved specifications, remove ambiguity, identify requirements that are incompatible, and assess the impact of requirements on the final design.

## 4.3 Design and Implementation of LLM-Powered MagicDraw Plugin

This section will detail the design and implementation of a MagicDraw plugin integrated with an LLM.

The MagicDraw plugin integrated with a Large Language Model (LLM) is developed for MagicDraw version 2021x Refresh2. The LLM being used in the plugin has been developed by OpenAI and it belongs to OpenAI's GPT family of generative pre-trained transformer models. It chosen because it is widely used and is purported to perform well in a variety of tasks. The plugin is designed to work with SysML elements and diagrams, which is why the SysML plugin for MagicDraw is required to be installed in MagicDraw to use the plugin. The plugin is specifically designed to work with SysML because it is the modeling language used by the teams working on the BTS system modeling efforts.

### 4.3.1 Objectives and Requirements

The main objective of the plugin is to have GPT-4o help and support the plugin's users with activities related to the SysML models in MagicDraw. The plugin is supposed to help the user understand the model, give suggestions for improvements on the model, answer questions that the user has about the model, and to be able to make modifications to the opened model according to the user's requests.

There are many requirements for this plugin to meet its objectives. Here are the requirements for this plugin:

**R1: User Interface:** The plugin is required to have a user interface (UI) so that the user can send messages to OpenAI's API. The user should also be able to see the response from OpenAI's API so that they can receive helpful information about the model that the AI is supposed to produce. This means that the plugin needs a terminal in which the user can input text to send to OpenAI's API and the terminal should also have an output text field, which shows the messages that the user has sent and the messages that the plugin has received from OpenAI's API. The user should be able to open and close the plugin

**R2: Integration with OpenAI's API:** The plugin needs to be able to send messages to OpenAI's API and receive messages from it. This requires an API key to send and receive messages from OpenAI's API. The plugin needs to have a way to let the user enter their API key.

**R3: Model State Information:** Without giving the GPT-4o LLM any information about the model that the user is working on, it cannot possibly know anything about the model. This is why the plugin needs to be able to gather the currently opened model's elements, relationships, and diagrams and build a representation of a model state that it can send to OpenAI's API to provide

context on the model. This model state should be in text format and well understood by the GPT-4o model. This model state should aim to minimize token usage however, as LLMs have limited context windows and these context windows could be exceeded if the model is large in size and each element is represented in the model state representation with too many tokens.

**R4: Instructions Prompt:** The GPT-4o needs an instructions prompt that details exactly how it should act and reply to messages. This should include all the possibilities it has to make modifications to the model so that GPT-4o can format its modification instructions or commands correctly.

**R5: Model Modifier Commands:** Since the GPT-4o model is supposed to be able to make modifications to the model it should be sending these modifications instructions back to the plugin. This means we need to parse these modification instructions or commands in the plugin and implement ways to make the changes according to the modification commands that the GPT-4o sent the plugin.

**R6: User Experience:** The plugin should be easy to use and fast to learn, otherwise it might cause problems with productivity instead of improving it. It also should be responsive, the responses from the LLM should not take too long to get to the user.

### 4.3.2 Architecture and Key Components

MagicDraw enables anyone to extend the functionality of the program with plugins. There are many different types of plugins, but anyone can also make their plugin with the desired functionality. MagicDraw is written in Java and its plugins are also in Java, meaning that the java will also be used in this plugin that integrates GPT-4o. All the plugins the user wants to run must be placed in the plugins directory

inside the MagicDraw installation directory. The plugins are loaded into MagicDraw when the user starts up MagicDraw. Every plugin must contain the following resources: a directory, compiled java files that are packaged into a jar file, a plugin descriptor file, and it can also optionally include other files used by the plugin. As MagicDraw plugins are scanned during startup, MagicDraw's plugin manager searches the subdirectories inside the plugins directory. If a subdirectory contains the plugin descriptor file, the plugin manager reads the descriptor file. If the requirements specified in the descriptor file are fulfilled, then the plugin manager loads the class specified by the descriptor file and calls the init() method of the loaded class. The loaded class must be extended from the com.nomagic.magicdraw.plugins.Plugin class, which is the base abstract class for any MagicDraw plugin. [41]

Maven is used in the development of the plugin as the build automation tool to build and manage the project. It handles the dependencies that are required for the plugin. Using a build tool can speed up the development of the plugin, as it includes multiple .java files and many dependencies. The dependencies are listed in a pom.xml file, which Maven uses to load the necessary dependencies.

The plugin is divided into multiple parts, each of which with its responsibilities:

**Main Class:** This is the entry point for the plugin, as it extends com.nomagic.magicdraw.plugins.Plugin. It implements the init() method to initialize the plugin, set up menu items, and manage the TerminalPanel. This component partly addresses requirements R1 and R6 by setting up the menu elements to easily start up the plugin's terminal, which the user can use to communicate with the LLM.

**TerminalPanel:** This is the part that is responsible for the user interface which includes a terminal. The terminal has a text field for input, in which the user can enter messages. The terminal also has a text field for output, which prints out the user's messages as well as the response that the LLM sends to

the plugin. The terminal also has a button for sending the message, a popup menu to select which diagrams are sent with the message, and a toggle button to turn "command mode" on or off. When the command mode is on, the LLM will send commands to make modifications to the models by adding, removing, or changing elements. Java Swing components are used to implement the user interface. BlockingQueue is used for asynchronous message processing and SwingWorker is used for background tasks to maintain responsiveness in the UI. Asynchronous elements like BlockingQueue and SwingWorker are used in the design to address potential performance issues when dealing with large models or slow network responses. This component directly addresses requirements R1 and R6 by providing a simple, responsive, and intuitive user interface for the user to communicate with the LLM.

**ModelInformationGatherer:** This part is responsible for gathering a representation of the current model state. This means that ModelInformationGatherer needs to collect the necessary information on all the elements of the model that the user has currently opened in MagicDraw. ModelInformationGatherer traverses the MagicDraw model hierarchy using a depth-first search starting at the root element of the model, which is the model itself. Using a depth-first search approach ensures that the model whole model is covered comprehensively. All the elements are processed, turned into a textual representation, and stored in a set to avoid redundancy. GSON is used to serialize the elements into JSON. This component directly addresses requirement R3 by building a comprehensive yet still efficient representation of the model state.

**ModelModifier:** This part enables the LLM to create new elements, delete elements, and update elements. The ModelModifier is the part of the plugin that handles converting the messages from the LLM to commands. Each command that is parsed calls a method according to the content of the command mes-

sage sent by the LLM. The commands will include parameters, which will be passed to the methods called. These parameters can set the value of certain properties of the element in question. MagicDraw's SessionManager is used to enable the user to undo or redo the actions. This component addresses requirement R5 fully by implementing the capability for the plugin to modify the model based on the parsed instructions sent by the LLM.

**APIClient:** This part is responsible for the communication between the plugin and OpenAI's API. It uses OkHttp to ensure that the communication is asynchronous, which prevents UI blocking during API calls and allows the user to keep interacting with the plugin's UI. This component is responsible for addressing requirement R2 by handling all communication between the plugin and OpenAI's API.

### 4.3.3 Integration with MagicDraw's OpenAPI and OpenAI's API

For the plugin to work, it needs to be integrated with MagicDraw and GPT-4o API. The plugin's integration with MagicDraw leverages MagicDraw's OpenAPI, which provides comprehensive access to classes and interfaces that enable the plugin to interact with the modeling environment. To enable the plugin to access MagicDraw's OpenAPI, the plugin is required to include dependencies from the OpenAPI.

The plugin uses OpenAPI to get access to MagicDraw's element factories and model hierarchy classes to navigate, query, and modify the model. This is used in ModelInformationGatherer to build the model state representation and in ModelModifier to allow for operations like creating new elements, updating properties, and establishing relationships between elements. OpenAPI is also used in the plugin to access diagram information including the presentation elements within the dia-

grams known as symbols. This is essential to enable the GPT-4o model to perform analysis on diagrams and suggest modifications. To include the UI from the plugin into MagicDraw, the TerminalPanel part creates the terminal panel JPanel with Java Swing components and the terminal panel JPanel is added to MagicDraw's MainFrame from com.nomagic.magicdraw.ui.MainFrame.

The plugin is integrated with OpenAI's Assistants API, which allows the user to build AI assistants within applications. Another endpoint that OpenAI also has is the Chat Completions API, but Assistants API was chosen because it is better suited for more complex use cases like this plugin. Assistants API uses persistent threads to facilitate the conversation session between an Assistant and a user and the messages in Threads are automatically truncated to fit the content into the model's context window. The Assistants API also enables the Assistants to use tools that Chat Completions API does not have access to like code_interpreter and file_search, which makes it a more extendable option for the plugin. The LLM model that is used for the Assistant is GPT-4o-2024-08-06 as it is the most recent GPT model OpenAI has released and performs better than previous versions in a wide variety of different tasks. [42] [43] [44]

The integration with the Assistants API is implemented in the plugin inside the APIClient part. It handles all communication with OpenAI's API, which includes creating and managing conversation threads, sending messages, and retrieving responses. HTTP requests are made to the OpenAI API using the OkHttp library for efficient and reliable communication. To be able to communicate with the Assistant API, an API key is required as well as the Assistant ID. These are not stored in the code, instead they can be inputted by the user in the plugin's UI. These values are then encrypted using AES encryption and stored in a file the plugin uses to acquire the API key and the Assistant ID during run-time by decrypting them. This maintains security in the plugin by not exposing the API key and the Assistant ID in

plain text.

An essential component needed for the plugin to work is to give the GPT-4o model guidance on how it should respond to the user's messages as per R4. This guidance can be given through the Assistant's system instructions prompt. This prompt should define the role, responsibilities, and capabilities of the AI assistant within the context of MagicDraw and SysML modeling. This system instructions prompt will always remain in the LLM's context window so that it will always know the instructions given. The prompt includes the following:

**Role Definition:** The prompt establishes the model as a helpful assistant specifically designed for MagicDraw SysML modeling.

**Model State Understanding:** Instructions are given on how to interpret and utilize the model state information that is sent through the user's message.

**Command Mode:** Clear instructions are given on when and how to use the Commands Mode for modifying the model.

**Available Commands:** A list of all the commands the AI can use to interact with the MagicDraw model is provided, including actions such as creating new elements, diagrams, and relationships.

**Guidelines and Responsibilities:** Specific guidelines are given on how to use commands responsibly, maintain accuracy in paths and names, and adhere to SysML-specific terminology and concepts.

A paper by Chang et al. on utilizing the Socratic method in the prompting of an LLM [45] was used as a basis for the design of the system instructions prompt. The paper presents an approach to using the Socratic method in developing prompt templates that can improve the effectiveness of large language models. Various dialogue and reasoning techniques were identified in the paper which were shown to

be effective in multiple examples. Techniques from the paper that were used in the design of the system instructions prompt of our use case are Definition, Organization (Dialectic), Guided Assistance (Maieutics), Error Prevention (Elenchus), Adaptability (Generalization), and Focus on User Intent:

**Definition:** Definition is used in the beginning to specify terms and roles to ensure mutual understanding. In the system prompt, the role is explicitly defined: "You are a helpful assistant for MagicDraw SysML modeling". Also, terms like Model State, Command Mode, and Relationship Types that are used throughout the prompt are defined. The user's intent and primary goals are also outlined to set clear expectations.

**Organization (Dialectic):** The prompt is structured logically with headings for each section to make the information easy to navigate and understand.

**Guided Assistance (Maieutics):** Detailed and specific instructions and examples are given for each different type of command the plugin can process and modify the model with.

**Error Prevention (Elenchus):** Additional guidelines are given in the prompt to double-check paths, names, and command syntax to help avoid inconsistencies and mistakes.

**Adaptability (Generalization):** In the prompt, commands are presented with placeholders to be adaptable to various situations, and optional parameters helping generalize the command patterns to other tasks.

**Focus on User Intent:** The prompt emphasizes the importance of understanding as well as addressing the user's primary goals and intentions throughout the interaction.

This system instructions prompt directly addresses requirement R4 by giving clear instructions on how the LLM should reply to the messages sent by the plugin.

### 4.3.4 Plugin's Workflow

This section will outline the whole workflow of the plugin, from initialization to the end of a typical interaction cycle.

1. **Plugin Initialization:** MagicDraw loads the plugin and the plugin sets up its configuration and UI components.

2. **UI activation:** User opens up the AI terminal from MagicDraw's menu, which happens by adding the TerminalPanel to MagicDraw's main frame.

3. **User Input:** User types a message and potentially selects relevant diagrams to send to the LLM. Then the user sends the message.

4. **Model State Capture:** The plugin gathers the current state of the model and if diagrams were selected by the user then they will be processed and converted into images.

5. **Message Processing:** A single message is made by combining the user input and the model state, this message is then queued for processing.

6. **API Communication:** The plugin sends the message to OpenAI's Assistants API, after which the API processes the message and returns a response from the LLM model.

7. **Response Handling:** An API response is received and processed by the plugin and if the message by the user was sent with Command Mode on, the commands are extracted from the response.

8. **Model Modification (if applicable):** Commands are parsed by the plugin and each of the commands calls a specific method to modify the MagicDraw model.

9. **UI Updated:** The response is displayed in the TerminalPanel for the user to see. Now the UI is ready for the next interaction.

10. **Continuous Operation:** Steps 3-9 repeat for each user interaction.

11. **Cleanup:** The workflow is over when the user decides to close the AI Terminal or exits MagicDraw.

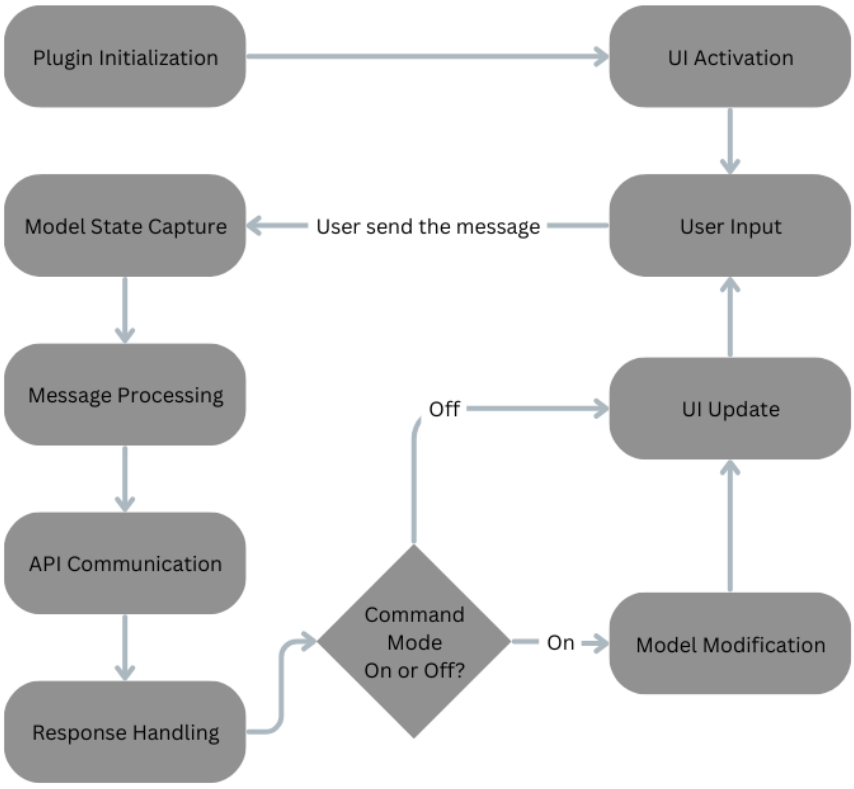


Figure 4.1: The plugin's workflow

## 4.4 Approach to Creating LLM-Interpretable SysML Model Representations

As mentioned earlier, it is imperative for the plugin to effectively help the user in modeling that the AI assistant has a good understanding of the model that the user is working on. This must include all the elements, diagrams, and relationships that are present in the model so that the AI assistant can give informed advice and make informed actions on the models that include all of these present.

### 4.4.1 Considerations for Building a Representation of a Model State

Some quite important factors need to be considered when trying to decide how the representation of a model state should be built.

One of these important considerations is the amount of data that a model state representation uses. This is important because the amount of data that a model state representation uses means that when the model state is sent to the AI assistant, the number of tokens that are consumed is large. This problem of the large amount of data that is required to represent the model was seen in the study by Bader et al. [35] in which the model was represented using the XMI (XML Metadata Interchange) standard which can describe UML and SysML models. The major problem with using this XML code however to build the representation of the model is that it is an extensive language and even simple information can cause a lot of textual overhead. This problem would become even greater with larger and more complex models, which could easily mean that the context window size of the LLM is not large enough to contain the whole representation of the model. I would also hypothesize, that overly verbose and extensive representation of the model's elements would lead to a poorer understanding of the model by the LLM, as more

of the information in the form of tokens is not informative or useful when it comes to understanding the elements in the model.

Another consideration that is directly linked to the size of the model representation is cost. The cost of using commercial LLMs or multimodal AI models can be calculated by knowing the number of tokens that are sent to the AI model and generated by the model as well as the price for tokens of both of these. For the plugin to be effective and worth using, it needs to bring more value than how much it costs to operate. This is why minimizing the size of the model representation is important to minimize the cost of operation. However, the model representation should not be minimized so much that the amount of value that using the plugin brings starts lowering faster than the model representation size is shrinking. A challenge with this calculation unfortunately is though, that quantifying the exact amount of value that using the plugin brings can be difficult. Also, the model representation should remain good enough that the plugin does not have a bad or incorrect understanding of the model which could cause the AI model to give erroneous advice to the user or make nonsensical modifications to the model.

The study by Bader et al. [35] found one more challenge, which is how IDs are handled. IDs are unique identifiers that each XMI element maintains, consisting of 37 characters each. The writers of the study were concerned with the LLM not effectively verifying the uniqueness of each ID. Also since the IDs are composed of random strings of characters, they do not follow natural language patterns. This can be a problem as LLMs typically use subword tokenization algorithms, which are optimized for natural language processing. Additionally, every character in the ID is likely to be tokenized separately or into small groups This could potentially result in high token consumption where each ID could use up to 37 tokens, which is very inefficient in terms of the model's context window utilization.

Also, the fact that IDs consist of 37 random characters means that each ID is

likely tokenized into a lot of tokens, up to 37 if no combinations of the neighboring random characters are tokenized together. Text is tokenized into words or sections of words that appear often in written language, so the IDs consisting of random characters make it unlikely that these IDs are tokenized in a compact way that does not consume a lot of tokens.

### 4.4.2 Methods for Extracting Relevant Information from SysML Models

MagicDraw has the functionality available to easily extract the XMI code of the model that the user is working on, which would be one way that the model state representation could be gathered for the plugin. But, for all the reasons listed in the previous subsection, an XMI representation of the model is not used in the plugin. This means that a novel method for gathering and building the representation of the model state needs to be designed and developed. The plugin will use MagicDraw's OpenAPI to achieve this.

The OpenAPI that MagicDraw offers its users has countless functionalities that can be used when developing plugins for MagicDraw. These functionalities include abilities to add or modify UI elements, add new actions, query the model elements of the model, and modify the elements of the model. When gathering the elements from the model, it must be ensured that all the relevant elements are gathered. This is why the gathering process will start at the root element of the model, which is the model itself. From the root model element, the other elements can be gathered by getting all the elements that the root model element owns. Elements in MagicDraw have countless different properties, one of the most important properties being "owned elements" which can be used to get all the elements that the element owns. In the plugin, all the elements of the model are gathered by recursively getting all the owned elements for each element using a depth-first approach starting from the

model's root element. This ensures that all the elements in the model are gathered.

Another factor that needs to be considered is the handling of default elements that are included in every SysML project. Since this plugin is specifically designed to work with SysML models in MagicDraw, we must consider how SysML models are handled in MagicDraw. SysML models in MagicDraw are created with the help of a SysML plugin, which includes a very large amount of elements in the model to make it conform to the SysML specification. These elements are crucial for the model to work like a SysML model, having all the SysML diagrams and elements, but sending all of this information of these elements that enable the SysML model would consume a lot of tokens and would not give any useful information to the AI model about the model that the user is working on. This is why these default elements that the SysML plugin inserts into all SysML models are excluded from being gathered and are not added to the model representation.

### 4.4.3 Techniques for Formatting Model Data for LLM Input

The model state needs to be formatted in such a way that the LLM can get a good understanding of the model so that it can give advice and modify the model well.

The LLM should have a good understanding of the hierarchy of the model and its elements. The location of an element in the model can be presented as a new property that I have made called "path", which shows all the elements that are owners of the element recursively. The "path" property is presented by showing all the parents from the parent of all the parents to the element itself using / as a delimiter: model/parent's_parent/parent/example_element. This helps give the LLM an understanding of the hierarchy of the elements in the model.

The representation of the model state will be sent as text as a part of the user's message to the LLM. The message to the LLM will clarify which part of the message includes the information about the model state and which part of the message the

user wrote. One important aspect to consider is that the text should be formatted so that it is clear to the LLM where one element ends and another begins. Otherwise, this can cause misunderstandings where the LLM might combine multiple elements to describe a single element. This is why every element will be formatted using JSONL (JSON Lines), where every line contains a single JSON (JavaScript Object Notation) object. JSON is extremely widely used, so the LLM should be able to parse the information in JSON well, as its training data most likely will have included a lot of JSON. In the plugin, each element will also be formatted as a single JSON object. The extracted properties of each element are placed into their own JSON object as key-value pairs, where the key is the name of the property and the value is the value of the property. For example. The key of the "name" property is "name" and the value will be the name of the element in question, and the key of the property "path" will be "path" and the value will be the path of the element like "model/usecases/usecase1". The elements formatted like JSON objects use "," as the delimiter, and the properties inside these elements also use "," as the delimiter, similarly to JSON. This format can be human-readable and can be used to give an understanding of the model, although this would be a very arduous way for humans to gain an understanding of the model. Thankfully, LLMs do not get tired like humans and therefore this format should hopefully be interpreted well by the LLM.

Elements in MagicDraw have a large amount of various properties. When creating a new element of any type, most likely almost all of these properties will not have a value assigned or will have the value as "False" in the case of boolean properties. These properties with no value assigned will very likely provide no value or very little value in terms of model understanding to the LLM. Additionally, providing these properties for every single element will use up a lot more tokens than is most likely necessary to give the model a good understanding of the model. For these

reasons, no properties with empty values will be sent as well as boolean properties with the value "False". This can be done because the LLM can be informed that if a boolean property has not been sent in the element's JSON object, the property's value is "False".

As previously discussed, there was a challenge that was considered with the handling of IDs for each element in the model. This is why the element's ID will not be sent as a part of the message to the LLM. The LLM should be able to identify the individual elements of the model with the element's name, path, type et cetera. in the format that the plugin uses, although this should be verified. The plugin's ability to identify elements with properties other than their IDs will be verified as a part of the evaluations done in the next chapter, Chapter 5.

### 4.4.4 Managing the Model State

The LLM should always be kept up-to-date with the state of the model so that it can answer the user's queries with up-to-date knowledge or make modifications to the model that make sense with the current version of the model. The most simple way to keep the LLM updated about the current model state of the model would be to send the whole, current model state representation to the LLM in every message the user sends. This would consume a lot of tokens for each message, however, increasing cost drastically. This is why the plugin uses an approach to keep track of the model state that considers the changes made to the model.

The full, initial model state is only sent to the LLM as a part of the first message that the user sends. This gives the LLM a full understanding of the model at the beginning of the conversation with the user. In the following messages that the user sends only the changes are tracked and added to the message sent to the LLM. If there are no changes then a message indicating that no changes have happened is included in the message sent to the LLM. If there are changes, then the elements

that have been added, changed, or removed will be included in the changes that are included in the message. The changes are made by keeping a copy of the previous version of the model state and comparing it to the current model state when the user starts sending a new message to the LLM. The individual elements in the model state are identified using their "element ID" property, which makes it possible to compare the specific elements from the previous model state version to the current version. The elements of the model being represented as their own JSON object makes it easy to implement the change tracking algorithm. If an element with the same element ID is not the same, then it has been changed. If the previous model state version includes an element ID that does not exist in the current version then the element has been removed. If the current model state version has an element ID that does not exist in the previous version then a new element has been added.

### 4.4.5 Handling of Different SysML Diagram Types and Elements

There are a lot of different types of diagrams and elements in SysML. The plugin needs to be able to represent each of them in a way that makes sense for the specific type of element or diagram. Implementing a method for each of the different types of elements would take a lot of time however, which is why the current version of the plugin only has implementations for methods that process only specific elements, and all the rest of the element types are processed with a generic method. This type of hybrid approach ensures that the most important element types are handled in a specific way while still making sure that all element types are handled at least in some way.

The generic method only processes elements that do not have their specific processing method and it includes the non-empty properties of the element that have been chosen as being most crucial for model understanding. For this Proof-of-

concept version of the plugin, the generic element processing method currently includes the following properties as long as they include a value: "name", "path", "incoming", "outgoing", "member", "memberEnd", "endType", "ownedEnd", "opposite", "appliedStereotype", "source", "target", "type", "stereotype", "multiplicity", and "ownedBehavior". "name" and "path" were chosen as they are fundamental properties for identifying and locating elements inside the model hierarchy. "incoming" and "outgoing" represent relationships and dependencies between elements, which are important for understanding how different parts of the model interact. "member" and "memberEnd" are used to represent associations and other relationship types. "endType" and "ownedEnd" help define the relationships between elements. "opposite" is used for bidirectional relationships, where the properties "source" and "target" are not used. "appliedStereotype" and "stereotype" are important to inform about the stereotypes present in the element, which allow for extending or customizing model elements. "source" and "target" are essential in directed relationships like dependencies and flows, as they tell the direction of the relationship. "type" is extremely important for understanding the nature of the element in question, helping in distinguishing between different types of elements. "multiplicity" defines the number of relationships that is associated with the element. "ownedBehavior" is a property that gives information about the associated behaviors of the element.

The elements in each diagram are included in the JSON object for the specific diagram in a condensed format. If all the elements present in diagrams were included in the JSON object for the diagrams, then that would waste a lot of tokens and would make each diagram's JSON object very complex and long. This is why only a condensed version of each element in a diagram is included to clarify which elements are in the diagram by including all the necessary properties to identify their full element JSON object that is separate from the diagram JSON object. The multiple

elements that are in the diagram are placed into "arrays" in the textual representation, which is formatted with square brackets "[]" and are delimited with "," like in JSON. Different types of diagrams are given different properties which can take an array of elements as a value. These properties can be for example "elements" and "relationships".

## 4.5 Conclusion

This chapter has described the design and the implementation of the LLM-powered MagicDraw plugin. The implementation includes solutions for efficient model state capture, token usage optimization, and integration with MagicDraw's OpenAPI and OpenAI's Assistants API. The next chapter will evaluate the capabilities of the plugin and the LLM used in the plugin, GPT-4o, to see how well the implementation of the plugin addresses RQ2 and RQ3. Similarly, RQ4 will be addressed by analyzing the evaluation results to assess the potential impact of LLM integration on MBSE adoption and productivity.

# 5 Evaluation of LLM Integration in MBSE and Its Impact

This chapter will evaluate the capabilities of the LLM-powered plugin for MagicDraw described in Chapter 4. This evaluation is conducted to address research questions RQ3 and RQ4, as well as verify aspects of RQ2 that the chosen approach for representing the model state is both efficient and interpretable by OpenAI's GPT-4o.

## 5.1 Expectations for the LLM-Powered Plugin

Before beginning the evaluation of the plugin, it should be made clear what capabilities are expected of the plugin. The expectations for the plugin are derived from its main purpose, which is to serve as an intelligent assistant for users in systems engineering activities. These expectations reflect the core capabilities that make such an assistant valuable in modeling. The plugin is expected to understand the model representation, give intelligent suggestions to the user, make intelligent modifications to the model, and be efficient in terms of time and token usage. This is why the expectations of the plugin have been divided into four categories: "Model Understanding and Query Response", "Intelligent Suggestion Generation", "Intelligent Model Modification", and "Performance and Token Usage". The main objectives of the plugin are to increase the productivity of the users' modeling efforts and to min-

imize the challenge associated with SysML of its steep learning curve, SysML-C1. For the plugin to fulfill these objectives, it should meet the following expectations. The expectations of the plugin are enumerated with the symbols from E1 to E6.

### 5.1.1 Model Understanding

**E1:** The LLM should be able to accurately interpret the SysML model structure, including elements, relationships, and hierarchies.

The plugin can not be an effective assistant if it is unable to understand the model that it is working on. This understanding is also crucial for all other capabilities.

These expectations align with RQ3 in terms of assessing the LLM's capability to understand and query SysML models inside MagicDraw. They also address RQ2 by validating whether the approach that has been chosen to build a textual representation of a SysML model is interpretable by an LLM.

### 5.1.2 Intelligent Suggestion Generation

The following are expected from the plugin and the LLM:

**E2:** The plugin should provide suggestions for systems engineering activities such as requirements analysis, functional analysis, interface analysis, and system decomposition.

These capabilities are expected, as an intelligent assistant should not just passively respond to queries but also be able to contribute to improving the model. By offering suggestions and identifying potential issues, the plugin can help maintain the quality and consistency of the model, which can be a challenge in complex systems.

This kind of functionality is also important to consider for evaluating how the integration of LLMs with MBSE tools can impact the adoption and productivity of MBSE practices, aligning with RQ4.

### 5.1.3 Intelligent Model Modification

The following model modification capabilities are expected:

**E3:** Ability to correctly perform operations like creating, updating, and deleting model elements based on the user's request.

**E4:** Ability to correctly handle commands that are complex and involve many steps and elements based on the user's request.

For the plugin to be truly able to assist in modeling activities, the plugin should be able to make changes to the model according to the engineer's directions. This capability could have a very positive effect in speeding up the modeling process, especially for tasks that are routine or repetitive.

These expectations address a part of RQ3 on the potential of integrating an LLM with an MBSE tool to assist in model modification.

### 5.1.4 Performance and Token Usage

The following are expected of the plugin when it comes to its performance and token usage:

**E5:** Response times should be reasonable for queries and model modifications.

**E6:** The token usage should remain reasonable, at least lower than if XMI was used for model representation.

For the plugin to be feasible and practical in real-world use, it needs to operate efficiently. Slow response times can make using the plugin less efficient and excessively high token usage can make using the plugin more costly, and more difficult with large models whose representation may be too large for the context window of the LLM. These expectations are important for addressing the practical aspects of RQ4. Additionally, RQ2 is addressed by assessing how efficient the model representation is.

## 5.2 Testing Methodology

The testing methodology is designed to test if the plugin meets the expectations outlined above and how effectively it is able to assist engineers in their modeling activities. In order to evaluate if the plugin meets these expectations, many testing approaches are combined to thoroughly assess the plugin's capabilities and performance in different scenarios.

1. **Simple tests:** The tests in this category will have specific inputs and only a single correct output that is expected from the plugin. These tests are designed to verify specific functionalities of the plugin and they should be repeatable and easy to validate.

2. **Explorative tests:** These tests are less simple and do not only have a specific correct output that is expected. Explorative tests are meant to be closer to a real-world usage scenario of the plugin and more representative of an actual use case of the plugin.

3. **Performance and token usage tests:** These tests will measure the response times and token usage for models of various sizes.

There are many benefits to using both simple and explorative testing to evaluate

the plugin's capabilities. Simple tests ensure that the plugin meets basic requirements well and explorative tests assess the plugin's performance in scenarios closer to real-world cases. The combination of both test types should give a more holistic view of the plugin's capabilities.

## 5.3 Execution of the Tests

This section will present how some tests were executed and their results. The tests are designed to evaluate how well the plugin fulfills its expectations.

### 5.3.1 Model Understanding and Query Response Tests

#### Simple Test: Looking Up Elements

The test will assess how well the plugin fulfills E1 by testing its interpretation of elements and their hierarchies. A model with different types of elements is prepared for this test. The user asks the LLM using the plugin the following question for ten different elements: "What is the path and type of element [specific element name]?"

| Correct Answer | Response |
|---|---|
| Model/element1 - Package | Model/element1 - Package |
| Model/element2 - Block | Model/element2 - Block |
| Model/element3 - Block | Model/element3 - Block |
| Model/element1/element4 - Package | Model/element1/element4 - Package |
| Model/element5 - UseCase | Model/element5 - UseCase |
| Model/element1/element4 /element6 - Block | Model/element1/element4/element6 - Block |
| Model/element1/element7 - Block | Model/element1/element7 - Block |
| Model/element1/element8 - Requirement | Model/element1/element8 - Requirement |
| Model/element1/element4/element9 - Activity | Model/element1/element4/element9 - Activity |
| Model/element10 - Constraint | Model/element10 - Constraint |

Table 5.1: The correct answers and the answers that the LLM gave are listed. The LLM was able to correctly identify 10/10 of the paths and types of the elements. Additionally, blocks element6 and element7 had the applied stereotype of interface block and constraint block respectively, both of which the LLM pointed out.

The LLM answered 100% of the questions correctly, indicating that it is able to

look up elements well and able to understand the type of the elements.

### Simple Test: Relationship Query

The test will assess how well the plugin fulfills E1 by testing its interpretation of the relationships. A model with elements and different types of relationships between them is prepared for this test. The user asks the LLM using the plugin the following question for ten different relationships: "What is the relationship between [element name A] and [element name B] and what is the direction of the relationship from the perspective of the first element?"

| Correct Answer | Response |
|---|---|
| Association - Outgoing | Association - Outgoing |
| Composition - Incoming | Composition - Incoming |
| Composition - Outgoing | Composition - Outgoing |
| Association - Incoming | Association - Outgoing |
| Usage - Outgoing | Usage - Outgoing |
| Generalization - Incoming | Generalization - Incoming |
| Information Flow - Incoming | Information Flow - Incoming |
| Abstraction(Satisfy) - Outgoing | Abstraction(Satisfy) - Outgoing |
| Abstraction(DeriveReqt) - Outgoing | Abstraction(DeriveReqt) - Outgoing |
| Abstraction(Trace) - Incoming | Abstraction(Trace) - Incoming |

Table 5.2: The correct answers and the answers that the LLM gave are listed. The LLM was able to correctly identify the type of each relationship (10/10) and correctly tell the relationship direction for all relationships except one (9/10). The LLM failed to correctly get the direction of a directed association.

The LLM correctly identified 100% of the types of relationships and 90% of the directions of the relationships. This shows that the LLM can understand the relationships of models quite well with the representation, although it did not correctly identify the direction of the directed association.

### 5.3.2 Intelligent Suggestion Generation

**Explorative Test: Intelligent Suggestion Generation**

This comprehensive test will evaluate how well the plugin fulfills the expectation E2 by using a medical device monitoring system as an example. A medical device monitoring system was chosen as the scenario for this test for a few reasons: it is a system where safety is critical, and it is fairly complex so it can be used to test various aspects of systems engineering for example.

For the sake of this test, a simplified and incomplete model of a medical device monitoring system was created to test the plugin's ability to suggest improvements and identify gaps in the model. Here is the structure of the model that this test uses:

- Package: "Patient_Monitoring_System"
  - Block: "Alert_System"
  - Block: "Display_Unit"
  - Block: "Network_Interface"
  - Block: "Power_Supply"
  - Block: "Sensor_Module"
  - Block: "Storage_Unit"
- Requirement: "SYS_REQ_1" with the requirement "The system shall monitor patient vital signs"
  - Requirement: "REQ_1" with the requirement "Display patient data"
  - Requirement: "REQ_2" with the requirement "Store patient records"
  - Requirement: "REQ_3" with the requirement "Alert on abnormal values"

- SysML Activity Diagram: "Main_Flow"
  - Activity: "Collect_Vital_Signs"
  - Activity: "Show_Data"

The user will ask multiple questions about the different aspects of the model. Due to the subjective and open-ended nature of the questions being asked, there are no specific correct answers. This is why such an evaluation criteria is used to give a score for each response to get a more objective measure of the quality of the answers. This assessment will still be subjective, as the author of this thesis evaluates the score for each response. This approach should still be useful in determining the approximate quality of the responses. To evaluate how good each answer is, the following evaluation criteria were created:

A score from 0 to 5 is used to evaluate each response:

**Score 5 (Excellent)**:

The response shows a complete understanding of the model state. It gives specific, actionable suggestions that can be directly implemented in SysML. The suggestions are technically correct and follow systems engineering principles. The response is comprehensive but still focused on the question asked.

**Score 4 (Very Good)**:

The response shows a very good understanding of the model state. The suggestions are mostly actionable with minimal interpretation needed. The suggestions are technically sound with slight imperfections. The response addresses the question well but may have missed minor aspects.

**Score 3 (Good)**:

The response shows a good understanding of the model state. The suggestions are useful but need some interpretation before implementation. The suggestions are technically valid but may have some gaps. The response addresses the main aspects of the question.

**Score 2 (Acceptable)**:

The response shows a sufficient understanding of the model state. The suggestions require a lot of interpretation but are still somewhat useful. The response contains some technical problems but the main advice is valid. The response partly addresses the question.

**Score 1 (Poor)**:

The response shows a very limited understanding of the model state. The suggestions are vague or difficult to implement. The response contains technical errors but parts of it might be salvageable. The response misses the most important aspects of the question.

**Score 0 (Completely Unacceptable)**:

The response shows no understanding of the model state or provides harmful suggestions. The suggestions cannot be implemented at all or they would damage the model. The response contains major technical errors or is completely unusable. The response fails to address the question or provides incorrect guidance.

**The Testing Results:** The user asks a total of 12 questions using the plugin from four different aspects of systems engineering: requirement analysis, system structure, behavioral analysis, and integration analysis. The questions are enumerated from Q1 to Q12 and the answers from A1 to A12.

**Requirements Analysis Questions:**

Q1: Please analyze the current requirements structure for this patient monitoring system. What issues do you see from a systems engineering perspective?

A1 Score: 4

Q2: Based on the analysis you made, what specific requirements should be added, and what relationships should be added between requirements?

A2 Score: 4

Q3: What requirements would be needed specifically for ensuring patient safety

and system reliability?

A3 Score: 5

**System Structure Questions:**

Q4: Please analyze the current block structure of the system. What issues do you see with the current decomposition?

A4 Score: 4

Q5: What logical groupings would you suggest for these components and what additional blocks might be needed?

A5 Score: 4

Q6: What interfaces should exist between these components? Please suggest the necessary ports and connections.

A6 Score: 5

Q7: What properties should each block have to properly specify its characteristics and requirements?

A7 Score: 4

**Behavioral Analysis Questions:**

Q8: Please analyze the current activity diagram. What essential functions are missing from this vital signs monitoring flow?

A8 Score: 4

Q9: What error handling and safety-critical functions should be added to this flow?

A9 Score: 5

Q10: How should data validation and system status monitoring be incorporated into this functional flow?

A10 Score: 4

**Integration Analysis Questions:**

Q11: Now that we have analyzed the requirements, structure, and behavior,

please analyze the overall consistency of the suggestions you have made. Are there any gaps or inconsistencies between these different aspects?

A11 Score: 3

Q12: What suggestions do you have for improving traceability between requirements, structural elements, and functional behaviors in the model?

A12 Score: 4

The responses received an average score of approximately 4.17 (83.3% of the maximum score), indicating very good performance in the responses. The result also indicates that the plugin fulfills expectation E2 well, providing useful suggestions for systems engineering. The biggest strengths of the responses were in the technical specifications and safety concerns while some of the weaknesses seemed to have been in integration analysis, where additional context may have been needed, and in the lack of specific suggestions.

### 5.3.3 Intelligent Model Modification

#### Simple Test: Element Creation

This test partly assesses how well the plugin fulfills E3 by testing its abilities to create model elements. An empty model is used as a starting point for this test and the Command Mode is turned on in the plugin's UI. The user will be asked to create ten elements of different types with various names and locations. Here are the commands the user sent:

- Command 1: "Make a package named package1 in the model"
- Command 2: "Make a block named block1 inside the package1 package"
- Command 3: "Make a requirement named requirement1 inside the model with the root model element as the parent"

- Command 4: "Make a package named package1.1 in package1"
- Command 5: "Make an activity element named activity1 inside package1.1"
- Command 6: "Make an actor element named actor1 inside package1"
- Command 7: "Make a use case element named usecase1 in package1.1"
- Command 8: "Make a property element named property1 inside block1"
- Command 9: "Make a constraint element named constraint1 inside the model with the root model element as the parent"
- Command 10: "Make a state machine element named statemachine1 in package1"

The plugin was able to successfully add each of these elements (10/10) with the correct names in the correct locations. This shows that the plugin and the LLM it uses can create simple elements in the model accurately and according to the instructions of the user.

**Simple Test: Element Deletion**

This test partly assesses how well the plugin fulfills E3 by testing its ability to delete model elements. The model from the last test involving element creation is used in this test. The user asks the LLM using the plugin to remove elements by telling to remove an element with a given name. The LLM will have to find the path of the element with the given name to create a valid element deletion command. The user requested to delete each of the elements from the last test. The user used the following format to request the deletion of an element: "Delete [elementName]", where elementName is the name of the given element.

The plugin was able to correctly remove every element (10/10). This shows that the plugin can remove the correct elements of various types that the user tells it to remove.

**Simple Test: Relationship Creation**

A model with various elements is prepared for this test. The user will be asked to create specific types of relationships between specific elements in the model. The following commands were used to determine whether the plugin can create the corresponding relationships between the correct elements correctly:

- Command 1: "Create an association relationship between block1 and block2"
- Command 2: "Create a composition relationship from block1 to block2"
- Command 3: "Create an aggregation relationship from block1 to block2"
- Command 4: "Create a realization relationship from block2 to interfaceBlock1"
- Command 5: "Create a generalization relationship from block2 to block1"
- Command 6: "Create a dependency relationship from block2 to block1"
- Command 7: "Create an include relationship from useCase1 to useCase2"
- Command 8: "Create an extend relationship from useCase3 to useCase1"
- Command 9: "Create a usage relationship from block2 to port1"
- Command 10: "Create an association between actor1 and useCase1"

The LLM successfully sent the correct responses to create the corresponding relationships and the plugin was successfully able to create each of the relationships in the model and in the correct diagram for each of the commands (10/10). This

shows that the plugin works as intended in creating relationships based on the user's commands.

**Simple Test: Element Modification**

This test partly assesses how well the plugin fulfills E3 by testing its abilities to update model elements.

The user sends ten modification commands to test different aspects of element modification:

- Command 1: "Rename the block from 'OriginalBlock' to 'ModifiedBlock'"
- Command 2: "Update the requirement to have a 'Performance' stereotype"
- Command 3: "Move the modified block to the root model"
- Command 4: "Update the use case with both a new name ('UpdatedUseCase') and move it to the root model"
- Command 5: "Update the enumeration by adding a new literal 'Value3'"
- Command 6: "Update the value type 'Speed' to use 'kilometer_per_hour' as its unit"
- Command 7: "Update the constraint block to add a specification 'speed <= maxSpeed'"
- Command 8: "Add a port named 'newPort' to the interface block"
- Command 9: "Duplicate the actor with name 'SecondaryActor'"
- Command 10: "Copy the block 'ModifiedBlock' to the TestPackage with name 'CopiedBlock'"

The plugin was able to successfully modify 100% of the elements correctly for each command (10/10). This shows that the plugin can make simple modifications to elements in a SysML model with seemingly no problems fulfilling a part of the plugin's expectation E3 concerning its ability to update model elements successfully.

#### Explorative Test: Model Modification

Similarly to the previous explorative test, the LLM will be sent multiple messages through the plugin and the responses will be evaluated. In this test, more complex model modification capabilities with multiple steps will be evaluated to assess how well the plugin fulfills expectation E4. This testing will start with an empty model which will be modeled into a simple BTS monitoring subsystem by only utilizing the LLM's model modification capabilities. The responses will be assessed using an evaluation criteria, which will be detailed shortly. The reason for assessing the responses using evaluation criteria is to attempt to get a measurable approximation of the quality of the responses, similar to the previous explorative test. The following details the evaluation criteria used for scoring the responses, each of which will be given a score ranging from 0 to 5:

**Score 5 (Excellent)**: All model elements are created and named correctly. Relationships between elements are logical and have the correct type. Model hierarchy is organized well, using packages. Real BTS monitoring needs are represented accurately with properties, ports, and interfaces.

**Score 4 (Very Good)**: Almost all model elements are created correctly. Almost all relationships are correctly typed. A very good model in terms of organization and hierarchy. Almost all of the properties and interfaces are realistic.

**Score 3 (Good)**: Most model elements are created correctly. The main relationships are present but may not be correctly typed. Model organization is functional but could be improved. Basic properties and interfaces are present.

**Score 2 (Acceptable)**: Some elements are not present or they are incorrectly created. Many relationships are missing or incorrect. The model organization needs improvement. Properties and interfaces lack detail.

**Score 1 (Poor)**: Many elements are missing or incorrect. Most relationships are missing or wrong. Poor model organization. Missing crucial properties and interfaces.

**Score 0 (Completely Unacceptable)**: Failed to create a basic model structure. No meaningful relationships were created. No logical organization. Properties and interfaces are missing.

**The Testing Results:** The user sends 8 messages using the plugin instructing how the LLM should modify the model. The messages are enumerated from M1 to M8 and the answers from A1 to A8.

M1: "Let's model a BTS monitoring subsystem. Start by creating a package structure and initial block definition diagram showing the main components of the monitoring system."

A1 Score: 4

M2: "Let us next define the relationships between these blocks and add more detail to the SensorBlock since monitoring is heavily dependent on various types of sensors. Add the correct relationships from MonitoringSubsystem to the other blocks, and create specific types of sensor blocks for monitoring different BTS parameters like temperature, power, and RF signal strength. These should be connected to the SensorBlock with the correct relationships."

A2 Score: 4

M3: "Let's model the data flow aspects of the monitoring system. Add interface blocks for different types of monitoring data and value types for the measurements that each sensor provides. Then create the necessary ports and interfaces on the blocks to show how monitoring data flows through the system."

A3 Score: 4

M4: "Create an internal block diagram for the MonitoringSubsystem to show how the components we've defined interact with each other. Show the connections between the sensors, processor, and communication interface, using the interface blocks we defined for the data flow. Make sure to show how the monitoring data flows from sensors through processing to storage and external communication."

A4 Score: 4

M5: "Let's create an activity diagram to represent the basic monitoring workflow. Create an activity diagram owned by the MonitoringSubsystem that shows the main activities of collecting sensor data, processing measurements, storing data, and sending alerts. Add these as activity elements to the diagram."

A5 Score: 3

M6: "Let's define the key requirements for the monitoring system. Create a requirements diagram and add requirements covering monitoring accuracy, frequency of measurements, alarm response times, and data retention. Include any relevant relationships between requirements."

A6 Score: 4

M7: "Add constraint blocks to specify the mathematical and logical constraints that ensure our requirements are met. Create constraints for measurement accuracy calculations, timing requirements for data collection and alerts, and data retention checks. Then create a parametric diagram showing how these constraints apply to the monitoring system properties."

A7 Score: 4

M8: "Create a use case diagram to show how operations personnel interact with the monitoring system. Add use cases for configuring monitoring parameters, viewing monitoring data, responding to alerts, and managing data retention. Include an actor for the operations personnel and show their interactions with these use

cases."

A8 Score: 4

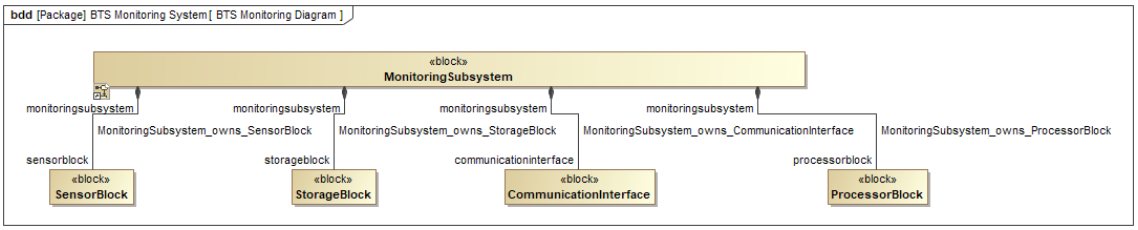


Figure 5.1: A Block-Definition Diagram created by the plugin during the test

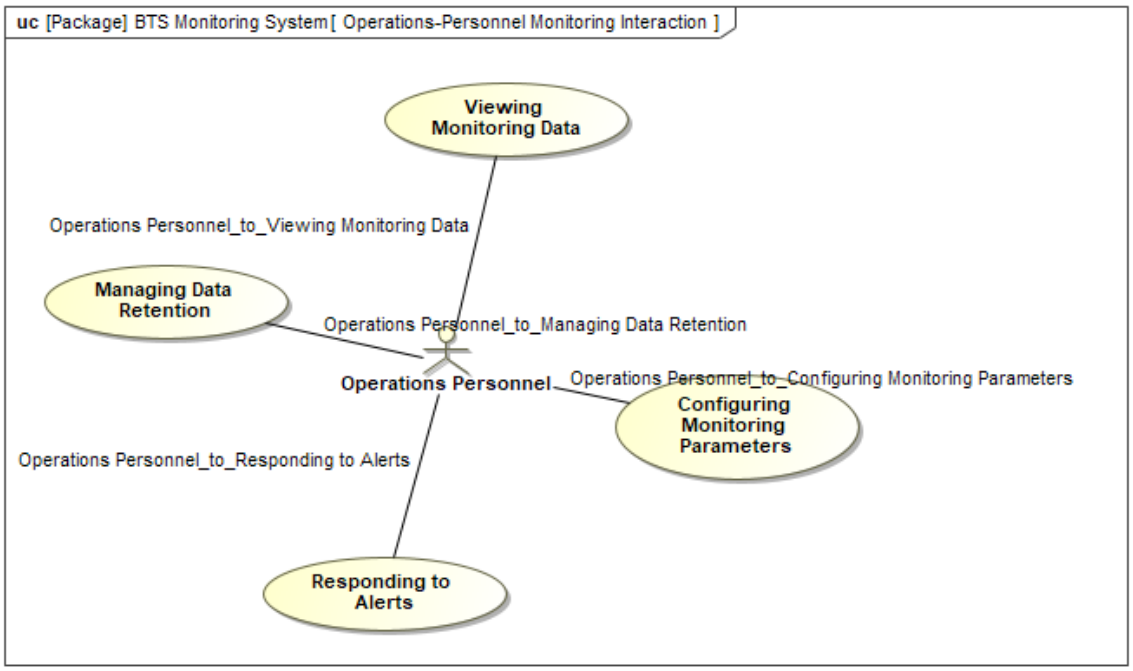


Figure 5.2: A Use Case Diagram created by the plugin during the test

The responses received an average score of 3.875 (77.5% of the maximum score), indicating very good performance in model modification. The score also indicates that the plugin fulfills expectation E4 well, correctly handling commands that are complex and involve many steps and elements based on the user's request. The responses seemed to perform the best in defining the basic structure, requirements, and operational interactions. The specifications and properties of the created ele-

ments could have been more detailed, which seemed to have been a slight weakness of the responses.

### 5.3.4 Performance and Token Usage Tests

The test will assess how well the plugin fulfills E5 and E6 by tracking the response times and token usage for models with varying amounts of elements. To test the performance of the speed of the model gathering process and the response time of the LLM as well as the token usage, models of different sizes have been made. The models consist of 25, 50, 100, 200, and 400 elements respectively, and the elements are simply SysML blocks with simple names and no other properties. This is done to get a lower limit for the amount of time responses and model representation gathering take as well as the amount of tokens consumed per model with a given number of elements.

The user sends the following message to the LLM to get the response time for a message that is not limited by the speed of text generation of the LLM: 'Respond to this message with just the word: "Hello"'. The time taken for gathering the model representation as well as the response time from the LLM is captured using timers inside the plugin code and the results are logged into the plugin's terminal. The plugin's model representation's token amount is also logged into the terminal when the model representation is gathered.

The token usage of the plugin's model representation is also compared to the token usage with an XMI representation of the model. Only the relevant part of the XMI representation for the model is captured and used as a comparison as if the whole exported XMI file of the model was used in the comparison, then all of the SysML and UML specification information would be included. This would make the comparison unfair in favor of the plugin's model representation, as the plugin's model representation does not include all of that information, and that would add

a lot of tokens to the model representation.

| Elements | Plugin tokens | XMI tokens | Representation gathering | Response time |
|---|---|---|---|---|
| 25 | 575 | 1346 | 31ms | 5898ms |
| 50 | 1101 | 2428 | 32ms | 4779ms |
| 100 | 2153 | 4592 | 35ms | 3414ms |
| 200 | 4257 | 8919 | 33ms | 9908ms |
| 400 | 8465 | 17575 | 52ms | 5779ms |

Table 5.3: Tokens used in the plugin's model representation, tokens used in an XMI representation, time taken for gathering the representation in the plugin, and the whole response time for getting a response from the LLM are presented for models with various element amounts

Looking at the results, models with higher element counts have higher token usage, which is expected as models with more elements will increase the amount of text that is needed to represent the model. The amount of tokens used by the model representation with XMI is over twice the amount that is used in the plugin's model representation, showcasing why the plugin's model representation is superior when it comes to minimizing token usage. The time taken for gathering the model representation inside the plugin seemed to only start increasing from 200 elements to 400 elements. For the various sized models, the time taken for gathering the model representation is very low, about 30-50ms, indicating good performance of the plugin's model representation gathering. The response times for the LLM do not seem to correspond linearly with the size of the model in the tests. However, it seems like the response time is between 3-10 seconds for sending a model representation and getting a response back. This response time will also be dependent on the amount of text the LLM generates, which was kept constant in these tests. Overall, receiving a response from the LLM does not take long, fulfilling the plugin's requirement R6: User Experience.

Based on this data, the average amount of tokens needed to represent an element in the plugin is about 21.8. This is about as low as the token usage can be per element, as in these model examples the elements only had a short name, path, and

type properties. This means that even with the more efficient model representation that the plugin uses might result in issues with the model representation being too large for the context window of the LLM. This means that with the maximum context window size of the GPT-4o LLM being 128,000 tokens, the maximum size of the model representation that it could handle is a model with approximately 5870 elements. In a typical model however, the LLM is likely unable to handle a model with even close to this amount of elements, as this assumed elements that only had a short name, path, and type properties, so the plugin might only be usable with smaller models. This means that the current proof-of-concept version of the plugin using the GPT-4o as the LLM should only be used in smaller models.

## 5.4 Summary of the results and the Potential of the Plugin

The Table 5.4 summarizes the results and findings of the tests.

Table 5.4: Summary of Plugin Testing Results

| **Test Category** | **Test Description** | **Related Expectations** | **Result** |
|---|---|---|---|
| Element Lookup | Testing ability to find paths and types of elements | E1 | 100% accuracy |
| Relationship Query | Testing identification of relationship types and directions | E1 | 95% accuracy |
| Intelligent Suggestion Generation | Comprehensive system engineering suggestions | E2 | 83.3% score |
| Simple Model Modification | Testing element creation, deletion, and relationship creation | E3 | 100% success |
| Complex Model Modification | Building BTS monitoring subsystem model | E4 | 77.5% score |
| Performance Testing | Response times for different model sizes | E5 | 3-10s response |
| Token Usage | Comparison with XMI representation | E6 | >50% reduction |

Overall, the plugin shows itself to fulfill all the expectations quite well. It fulfills the basic expectations E1 and E3 almost perfectly and it received very good scores in the testing of the expectations E2 and E4, indicating that it may be able to perform well as an assistant for complex, real-world systems engineering tasks. The response times also remained reasonable throughout all the testing as well as in the performance test and the token usage of the model state representation that the plugin uses is more compact than the XMI representation of the model.

The evaluation performed in this chapter has addressed several of this thesis' research questions. For RQ2, the evaluation demonstrated that the SysML model representation developed for the plugin is efficient and effective, consuming fewer tokens than the XMI representation while providing model understanding. This representation was understood by the GPT-4o model, as evidenced by the LLM's performance in element lookup (100%) and relationship query (95%) tests. Regarding RQ3, the results of the evaluation suggest that an LLM like GPT-4o can effectively assist in model understanding and modification when integrated within MagicDraw, achieving perfect accuracy in basic model modifications and a 77.5% score in complex modeling tasks. The testing also addressed RQ4 by showing the potential impact of LLM integration on MBSE adoption and productivity. The plugin's ability to give intelligent suggestions (83.3% score in intelligent suggestion generation) and handle complex modeling tasks that include many steps indicates its potential to address some of the key challenges related to MBSE adoption.

# 6 Conclusions and Future Research

## 6.1 Conclusion

This thesis has explored the transition from Document-Based Systems Engineering to Model-Based Systems Engineering in the context of Base Transceiver Station development. The thesis also investigated how Large Language Models can be integrated with modeling tools to improve adoption and productivity.

The research identified several findings regarding RQ1 on the benefits and challenges of transitioning from DBSE to MBSE for BTS development, including improved communication and shared understanding (B1), enhanced traceability (B2), and reduced time and cost (B4). Significant challenges were also identified, particularly substantial upfront investment, awareness, and change resistance (C1), method definition (C6), and integration with legacy systems (C12).

Chapter 4 of the thesis detailed a successful approach for answering RQ2, which concerned the creation of an LLM-interpretable SysML model representation. The implemented solution used in the thesis achieved more than a 50% reduction in token usage compared to the XMI representation of the model that MagicDraw uses. This representation of the SysML model was well understood by the LLM, as evidenced by the evaluation results in Chapter 5. The plugin implemented for this thesis, which integrates the LLM into MagicDraw and creates SysML model representation is comprised of over 5,800 lines of Java code. The majority of this codebase was

dedicated to creating the model state representation using MagicDraw's OpenAPI and implementing model modification capabilities through LLM interactions.

RQ3 on the capability of LLMs to assist in model understanding and modification was addressed in the evaluation performed in Chapter 5, which showed promísing results. The implemented plugin performed very well in basic tasks, achieving 100% accuracy in element lookup tests, 95% accuracy in relationship queries, and received high scores in both simple (100%) and complex (77.5%) model modification tasks. The results of the plugin show that LLMs can effectively understand and manipulate SysML models when given an appropriate model representation and clear instructions.

At last, the research suggests potential regarding RQ4 on the impact of LLM integration on MBSE adoption and productivity. The results of the evaluation of the plugin indicate that LLM assistance may help address the major challenge of the steep learning curve of SysML (SysML-C1) and tool dependence (SysML-C2). The plugin exhibited capabilities in providing intelligent suggestions and handling complex modeling tasks, which demonstrates its potential to accelerate the learning process and improve modeling productivity.

## 6.2 Limitations of the Study

### 6.2.1 Limitations of the Literature Review and the Case Study

While there was a good amount of literature on MBSE and its benefits and challenges, a large amount of the literature on the topic did not provide much if any empirical data. A lot of the papers included in my literature review seemed to cite the benefits of MBSE without supporting evidence. This limitation concerning the lack of empirical evidence made it more difficult to draw definitive conclusions about the expected benefits of transitioning to MBSE.

The case study analysis also had limitations, as the analysis was conducted during the initial stages of MBSE adoption, which means that the long-term impacts and benefits could not be assessed.

### 6.2.2 Limitations of the Plugin and its Evaluation

The plugin uses OpenAI's GPT-4o LLM as the assistant that the user communicates with. Using the assistant's API of OpenAI was a convenient way to integrate an LLM into the plugin, but it also came with some downsides. Sending requests and receiving a generated response back consumes tokens, which is not free. This means that using the plugin with OpenAI's LLM might get fairly expensive, especially with larger models. Additionally, OpenAI's LLMs are pretrained with massive amounts of data, and they have not been trained with data specifically from the plugin's model representations. This means that an LLM specifically trained with or fine-tuned with data from the plugin's model representations could perform better in being used as the LLM assistant for the plugin. Building and training a whole new model is however extremely resource-intensive and costly, which is why it would most likely not be worth the investment to build and train a model just to be used as an assistant in the plugin. [46] However, OpenAI does provide possibilities to fine-tune their LLMs using new data, which could be used to potentially improve an LLM's performance as an assistant in the plugin. This could be a worthwhile area for future development, as it would not take nearly as much time as building and training a new model from the beginning while still potentially improving performance.

If the plugin is to be used with larger models that have thousands of elements, it might be necessary to come up with different strategies to not exceed the context window of the LLM. One possibility would be to simply use a different LLM with a higher context window size, like Google's Gemini 1.5 Pro, which can handle up to 2,000,000 tokens in its context window. This would enable the usage of models in

the plugin that are about 15.6 times larger than with the current LLM, GPT-4o. However, the evaluation in this thesis was done on OpenAI's GPT-4o, so Gemini may perform differently compared to GPT-4o as an assistant in the plugin.

Another possibility would be to try to utilize RAG (Retrieval-Augmented Generation) to reduce the number of tokens consumed by retrieving only relevant information from the full model representation. [47] However, it would have to be evaluated whether this kind of approach would be able to retrieve all the necessary information to generate useful responses to the user's messages without retrieving too much information or too little information. A potential pitfall with using RAG for only retrieving a part of the model state might be that it might miss the context, which would be crucial for generating a useful response to the user's message.

A clear limitation of the explorative testing is that the evaluation and the scoring of the responses are ultimately subjective. The goal of the explorative testing was to evaluate the plugin in more complex situations, which do not have specific correct answers to the questions, unlike the simple tests. This could be tested by using the plugin in a real-world systems engineering project by comparing the results of using the plugin to not using the plugin or by simply having systems engineers use the plugin and get their impressions. However, in the case of this thesis, these approaches were unfortunately not possible.

Additionally, in its current state, the plugin only offers limited model modification functionality. This means it can only create specified elements and make specified changes to the element's properties in the model in MagicDraw. Because of this, only the implemented model modification functionality was tested. Implementing the plugin to be able to perform all the model modification functionality that human users can perform would have taken an unreasonable amount of time, which is why the testing was only limited to the model modification functionality that had been implemented in the plugin.

## 6.3 Future Research Directions

One clear opportunity for future research is utilizing the plugin in a real-world systems engineering project and evaluating the benefits and challenges associated with using it. This could provide empirical data on the benefits and challenges of transitioning to MBSE. A plugin such as the one created for this thesis could also be used in such a real-world MBSE transition project, providing concrete evidence of the effects and capabilities of using LLMs as assistants in systems engineering tasks.

Another opportunity for future research is using and evaluating different LLMs in the plugin and comparing the results. It also could be evaluated how well an LLM would perform as a systems engineering assistant in the plugin when it is given the user's message, the model state, and a relevant document, which would describe a part of the system, providing domain knowledge.

# References


[1] D. D. Walden *et al.*, "Systems engineering handbook: A guide for system life cycle processes and activities", 2015.

[2] E. R. Carroll and R. J. Malins, "Systematic literature review: How is model-based systems engineering justified?", Mar. 2016. DOI: 10.2172/1561164. [Online]. Available: https://www.osti.gov/biblio/1561164.

[3] M. Adedjouma, T. Thomas, C. Mraidha, S. Gerard, and G. Zeller, "From document-based to model-based system and software engineering: Experience report of a selective catalytic reduction system development", in *Joint Proceedings of EduSymp 2016 and OSS4MDE 2016*, Copyright © 2016 held by the author(s), CEA, LIST, Department of System and Software Engineering, Gif-sur-Yvette, France, 2016. [Online]. Available: https://ceur-ws.org/Vol-1835/paper05.pdf.

[4] P. Logan, D. Harvey, and D. Spencer, "Documents are an essential part of model-based systems engineering", in *INCOSE International Symposium 2012*, Published and used by INCOSE with permission. Copyright © 2012 by Paul Logan, David Harvey and Daniel Spencer, INCOSE, Eltham Vic and Fyshwick ACT, Australia, 2012. [Online]. Available: https://www.shoalgroup.com/wp-content/uploads/2017/06/Logan-et-al-2012-Documents-are-an-Essential-Part-of-Model-Based-Systems-Engineering-INCOSE-2012.pdf.

[5] L. E. Hart, *Introduction to model-based system engineering (mbse) and sysml*, Presented at the Delaware Valley INCOSE Chapter Meeting: Lockheed Martin, IS&GS, Jul. 2015.

[6] A. L. Ramos, J. V. Ferreira, and J. Barceló, "Model-based systems engineering: An emerging approach for modern systems", *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 1, pp. 101–111, Jan. 2012, ISSN: 1558-2442. DOI: 10.1109/TSMCC.2011.2106495.

[7] S. Friedenthal, R. Griego, and M. Sampson, "Incose model based systems engineering (mbse) initiative", Jan. 2009.

[8] P. M. Redondo, *Understanding the 10 key principles of model-based system engineering (mbse)*, Accessed: 2024-09-11, 2023. [Online]. Available: https://www.stariongroup.eu/understanding-10-key-principles-of-model-based-system-engineering-mbse/.

[9] D. Long and Z. Scott, *A primer for model-based systems engineering*. 2012.

[10] J. Towers, "The 5 principles of mbse", Feb. 2014.

[11] S. Friedenthal, A. Moore, and R. Steiner, *A Practical Guide to SysML: The Systems Modeling Language*, Third. Waltham, MA, USA: Morgan Kaufmann, 2015, ISBN: 978-0-12-800202-5.

[12] J. Ma, G. Wang, J. Lu, H. Vangheluwe, D. Kiritsis, and Y. Yan, "Systematic literature review of mbse tool-chains", *Applied Sciences*, vol. 12, no. 7, 2022, ISSN: 2076-3417. DOI: 10.3390/app12073431. [Online]. Available: https://www.mdpi.com/2076-3417/12/7/3431.

[13] M. Hause *et al.*, "The sysml modelling language", in *Fifteenth European systems engineering conference*, vol. 9, 2006, pp. 1–12.

[14] Object Management Group, *Omg system modeling language (sysml)*, 2024. [Online]. Available: https://www.omg.org/spec/SysML/.

[15] O. M. Group, "Omg systems modeling language (omg sysml™) version 1.6", Object Management Group (OMG), Tech. Rep. formal/19-11-01, Nov. 2019. [Online]. Available: https://www.omg.org/spec/SysML/1.6/.

[16] P. White, *Welcome to sysml, the language of mbse*, Presented at BAE/KIHOMAC: BAE/KIHOMAC, Oct. 2019.

[17] K. Henderson and A. Salado, "Value and benefits of model-based systems engineering (mbse): Evidence from the literature", *Systems Engineering*, vol. 24, no. 1, pp. 51–66, 2021. DOI: https://doi.org/10.1002/sys.21566. eprint: https://incose.onlinelibrary.wiley.com/doi/pdf/10.1002/sys.21566. [Online]. Available: https://incose.onlinelibrary.wiley.com/doi/abs/10.1002/sys.21566.

[18] A. Madni and S. Purohit, "Economic analysis of model-based systems engineering", *Systems*, vol. 7, p. 12, Feb. 2019. DOI: 10.3390/systems7010012.

[19] K. X. Campo, T. Teper, C. E. Eaton, A. M. Shipman, G. Bhatia, and B. Mesmer, "Model-based systems engineering: Evaluating perceived value, metrics, and evidence through literature", *Systems Engineering*, vol. 26, no. 1, pp. 104–129, 2023. DOI: https://doi.org/10.1002/sys.21644. eprint: https://incose.onlinelibrary.wiley.com/doi/pdf/10.1002/sys.21644. [Online]. Available: https://incose.onlinelibrary.wiley.com/doi/abs/10.1002/sys.21644.

[20] J. Maurandy, A. Helm, E. Gill, and R. Stalford, "11.5.3 cost-benefit analysis of sysml modelling for the atomic clock ensemble in space (aces) simulator", *INCOSE International Symposium*, vol. 22, no. 1, pp. 1726–1745, 2012. DOI: https://doi.org/10.1002/j.2334-5837.2012.tb01433.x. eprint: https://incose.onlinelibrary.wiley.com/doi/pdf/10.1002/j.2334-5837.

2012.tb01433.x. [Online]. Available: https://incose.onlinelibrary.wiley.com/doi/abs/10.1002/j.2334-5837.2012.tb01433.x.

[21] T. Bayer, "Is mbse helping? measuring value on europa clipper", in *2018 IEEE Aerospace Conference*, Mar. 2018, pp. 1–13. DOI: 10.1109/AERO.2018.8396379.

[22] E. B. Rogers III and S. W. Mitchell, "Mbse delivers significant return on investment in evolutionary development of complex sos", *Systems Engineering*, vol. 24, no. 6, pp. 385–408, 2021. DOI: https://doi.org/10.1002/sys.21592. eprint: https://incose.onlinelibrary.wiley.com/doi/pdf/10.1002/sys.21592. [Online]. Available: https://incose.onlinelibrary.wiley.com/doi/abs/10.1002/sys.21592.

[23] M. Chami and J.-M. Bruel, "A survey on mbse adoption challenges", in *INCOSE EMEA Sector Systems Engineering Conference (INCOSE EMEASEC 2018)*, INCOSE, Berlin, Germany, Nov. 2018. [Online]. Available: http://oatao.univ-toulouse.fr/226437.

[24] No Magic, Inc., *Magicdraw: Award-winning business process, architecture, software and system modeling tool with teamwork support*. [Online]. Available: https://www.3ds.com/products/catia/no-magic/magicdraw.

[25] No Magic, Inc., *Sysml plugin documentation*. [Online]. Available: https://docs.nomagic.com/display/SYSMLP2021xR2/SysML+Plugin+Documentation.

[26] No Magic, Inc., *Teamwork cloud: Providing a powerful collaborative environment for system engineers practicing mbse with no magic*. [Online]. Available: https://www.3ds.com/products/catia/no-magic/teamwork-cloud.

[27] No Magic, Inc., *Guidelines for working with large models*, MagicDraw 2024x Documentation, Dassault Systèmes, 2024. [Online]. Available: https://docs.

nomagic.com/display/MD2024x/Guidelines+for+Working+with+Large+Models.

[28] H. Naveed, A. U. Khan, S. Qiu, *et al.*, "A comprehensive overview of large language models", *arXiv preprint arXiv:2307.06435*, 2023.

[29] OpenAI, *How can i access gpt-4, gpt-4 turbo, gpt-4o, and gpt-4o mini?*, 2024. [Online]. Available: https://help.openai.com/en/articles/7102672-how-can-i-access-gpt-4-gpt-4-turbo-gpt-4o-and-gpt-4o-mini.

[30] Google, *Long context*, 2024. [Online]. Available: https://ai.google.dev/gemini-api/docs/long-context.

[31] J. Wei, Y. Tay, R. Bommasani, *et al.*, "Emergent abilities of large language models", *arXiv preprint arXiv:2206.07682*, 2022.

[32] M. Shanahan, "Talking about large language models", *Communications of the ACM*, vol. 67, no. 2, pp. 68–79, 2024.

[33] Z. Chu, S. Ni, Z. Wang, *et al.*, "History, development, and principles of large language models-an introductory survey", *arXiv preprint arXiv:2402.06853*, 2024.

[34] M. Chami, C. Zoghbi, and J.-M. Bruel, "A first step towards ai for mbse: Generating a part of sysml models from text using ai", *A First Step towards AI*, 2019.

[35] E. Bader, D. Vereno, and C. Neureiter, "Facilitating user-centric model-based systems engineering using generative ai.", in *MODELSWARD*, 2024, pp. 371–377.

[36] E. S. Crabb and M. T. Jones, "Accelerating model-based systems engineering by harnessing generative ai", in *2024 19th Annual System of Systems Engineering Conference (SoSE)*, IEEE, 2024, pp. 110–115.

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

[38] A. Patel, Y. Maheshwaran, and P. Santhya, "Easing adoption of model based system engineering with application of generative ai", in *2024 IEEE Space, Aerospace and Defence Conference (SPACE)*, IEEE, 2024, pp. 871–874.

[39] M. Chami, N. Abdoun, and J.-M. Bruel, "Artificial intelligence capabilities for effective model-based systems engineering: A vision paper", Jun. 2022. DOI: 10.1002/iis2.12988.

[40] INCOSE, "Incose systems engineering vision 2035", International Council on Systems Engineering, Technical Report, 2021. [Online]. Available: https://www.incose.org/products-and-publications/se-vision-2035.

[41] No Magic, Inc., *Magicdraw 2021x plugins*. [Online]. Available: https://docs.nomagic.com/display/MD2021x/Plugins.

[42] *Assistants api overview*. [Online]. Available: https://platform.openai.com/docs/assistants/overview.

[43] *Chat completions*. [Online]. Available: https://platform.openai.com/docs/guides/chat-completions.

[44] *Hello gpt-4o*. [Online]. Available: https://openai.com/index/hello-gpt-4o/.

[45] E. Y. Chang, "Prompting large language models with the socratic method", in *2023 IEEE 13th Annual Computing and Communication Workshop and Conference (CCWC)*, IEEE, 2023, pp. 0351–0360.

[46] W. contributors, *Large language model*, 2024. [Online]. Available: https://en.wikipedia.org/wiki/Large_language_model#Training_cost.

[47] P. Lewis, E. Perez, A. Piktus, *et al.*, *Retrieval-augmented generation for knowledge-intensive nlp tasks*, 2021. arXiv: 2005.11401. [Online]. Available: https://arxiv.org/abs/2005.11401.