Empirical Evaluation of Large Language Models for the Generation of Model-Based Systems Engineering Systems.

Nguyen Trinh, Jack Phong-Anh

Department of Aerospace Engineering, University of Bristol, Queen's Building, University Walk, Bristol. BS8 1TR. UK.

1. ABSTRACT/EXECUTIVE SUMMARY

Model-Based Systems Engineering (MBSE) is the gold standard for complex and interdisciplinary systems modelling. However, the process of creating these systems is complicated and involved, often having to build the model from the ground up for each use case. Variability modelling and conceptual design generation are vital for generalising MBSE frameworks across disciplines. Large Language Models (LLMs) show great potential for this task, with their utility being well documented across many tasks across various disciplines but never explored for systems engineering. This paper tests their effectiveness by empirically evaluating the responses' accuracy, scalability, and repeatability when prompted for system functions, modes, and architecture, given a set of system requirements. The evidence suggests that LLMs are very powerful for conceptual design space exploration. They can also scale up their effectiveness for complex systems. However, their lack of understanding of the nuances of systems engineering and the output variability makes them unsuitable as a standalone tool to automate the Systems Engineering process. Instead, they are best used as a tool in the hands of competent systems engineers who can validate their outputs and apply sound engineering judgement.

Keywords: Model-Based Systems Engineering, Large Language Models, Natural Language Processing, Systems Engineering, Similarity Analysis,

2. INTRODUCTION

Large Language Models (LLMs) have recently experienced unprecedented capability growth, leading to excellent performance in a variety of activities; for example, OpenAI's GPT-4 model matched, and even outperformed, human performance in a variety of professional and academic benchmarks [1]. The sub-field of Generative AI has seen some fascinating developments, from investigating Reinforcement Learning from Human Feedback to train AI systems to better understand human meaning [2], to creating models that have the ability to take text prompts and synthesise them into high-resolution art [3]. Looking at more engineering-focused applications, models such as Minerva have achieved state-of-the-art results on mathematical and quantitative reasoning tasks [4]. At the same time, AlphaCode can create novel solutions for programming tasks that require deeper reasoning [5].

Concurrently, investigations into the benefits of a Model-Based Systems Engineering (MBSE) approach over a Document-Based approach and its implementation into the context of functional avionics have shown certain application areas can be improved by MBSE [6]. However, Gregory, Berthoud, Tryfonas and Faure (2020) highlighted the difficulties of developing the Spacecraft Early Analysis Model (SEAM), an MBSE framework for space systems. The issue raised was that the versatility of SEAM relied on development by use case, meaning more cases are needed to generalise [7]. The creation of more MBSE models for a variety of use cases will be essential to the development of MBSE frameworks, but the issue of how difficult and involved it is to create these models remains. Timperley et al. partly tackled this issue by exploring the use of Generative Design with MBSE to improve Design Space Exploration [8]. MBSE tools lack good variability modelling and need additional code for design exploration, such as MATLAB or JavaScript. LLMs present a unique opportunity to bridge this gap.

LLMs can potentially be used in place of the complex toolkit used by Timperley et al. to generate MBSE systems. Using LLMs to generate MBSE systems requires no knowledge of coding languages or how LLMs work; instead, the user only needs to pass information about the systems to the LLM through

prompts. These prompts will be a list of requirements for the LLM to constrain the problem into a workable solution. This is extremely powerful as it could provide users with a straightforward way to create a range of systems, given a set of requirements, which should make MBSE more accessible.

The use of LLMs to complement MBSE workflows is beginning to enter the literature. One use case was using LLMs to standardise MBSE system requirements, as these are still set out in natural language (NL) and, therefore, contain ambiguities and inconsistencies and suffer from a lack of structure that hinders their direct translation into models [9]. Tikayat Ray et al. suggest a two-fold strategy to standardise Natural Language Processing (NLP) requirements into machine readable requirements. This is encouraging as it involves LLMs to aid MBSE system generation. However, there is still very little to no literature on using LLMs to generate MBSE models and evaluate *the quality* of these generated systems.

This paper aims to advance the literature on using LLMs to generate an MBSE system. It evaluates LLMs' text output's accuracy, interpretability, and scalability compared to the text equivalents derived from validated MBSE systems such as the University of Bristol PROVE Pathfinder Satellite Payload and the James Webb Space Telescope.

3. AIM AND OBJECTIVES

Aim:

This project aims to evaluate the use of Large Language Models (LLMs) to generate Model-Based Systems Engineering (MBSE) systems.

Objectives:

- Research the uses of LLMs in MBSE modelling.
- Evaluate similarity methods for comparing LLM-generated output.
- Evaluate the quality of LLM-generated systems.
- Evaluate the scalability of LLM-generated systems.
- Evaluate the repeatability of LLM-generated systems.
- Provide recommendations for the use of LLMs in the Systems Engineering process.

4. LITERATURE REVIEW

4.1 Model-Based Systems Engineering (MBSE)

An approach called MBSE is being used to formalise the application of modelling principles, methods, languages, and tools to the entire lifecycle of large, complex, interdisciplinary, socio-technical systems [10]. This approach challenges the traditional document-based approach with the aid of improving computer technology [11].

The central aspect of MBSE is a unified, coherent system model that mirrors the behaviour of its real-life equivalent. Currently, MBSE models use programming languages and software such as SysML or OPDs/OPL, represent their systems graphically, and store all information in one repository, so all information and analysis is self-contained in one place. The system is made by interconnected modelling elements (subsystems, components, …) representing its key aspects, such as requirements, structure, behaviour, and parameters [11].

4.2 Large Language Models (LLMs)

Understanding various Natural Language Processing techniques is essential to understanding how large language models work. Understanding methods such as neural networks, Recurrent Neural Networks

(RNNs), and Transformers is a basis for the methods used by the leading LLMs today: Generative Pre-Training (GPT) and Bidirectional Encoder Representation for Transformers (BERT).

Neural networks [12] form the basis of the machine learning models for NLP, allowing a machine to read Natural Language sentences. RNNs, proposed by Bahdanau et al., were the next breakthrough in NLP, allowing machines to read much longer sentences [13], enabling the processing of longer texts, hence allowing for better comprehension and contextual understanding. By eschewing the recurrence (sequential nature of the RNN), the Transformer model solely focused on self-attention mechanisms (a mechanism that mimics human cognitive attention). This allowed for much more efficient parallel processing, vastly improving training efficiency, training dataset size, and inference ability [14].

The ability to learn effectively from raw text is crucial to alleviating the dependence on supervised learning in NLP [15]. Radford and Narasimhan proposed GPT as an improvement to the Transformer model, as a way of improving the model's ability to learn from raw text data to tackle a wide range of NLP tasks such as textual entailment, question answering, semantic similarity assessment, and document classification. This is the base architecture for GPT-2, GPT-3 and now, GPT-4, which is the model behind ChatGPT, which is the LLM used in this paper.

Building on the Transformer model, Google AI researchers released BERT, which at the time returned state-of-the-art performance on a plethora of NLP tasks [16]. The model develops semantic understanding by pre-training bidirectional representations from unlabelled text and can easily be fine-tuned for the task at hand [16].

4.3 Similarity Measures for NLP

The primary objective of this study is to assess the viability of leveraging Large Language Models (LLMs) for the generation of Model-Based Systems Engineering (MBSE) systems, with particular emphasis on evaluating the quality of the resultant systems. It is important to note that the existing body of literature on utilising LLMs for MBSE system generation is notably limited, and there is a lack of research specifically addressing the evaluation of such systems. In contrast, the broader literature offers an abundance of methodologies for appraising the quality of MBSE systems. However, many of these approaches necessitate the extraction of LLM Natural Language (NL) output, the subsequent construction of a complete MBSE system, and external analysis.

Consequently, this paper seeks to explore alternative evaluation methods that do not entail external analyses, instead focusing on assessing the NL output of the LLM. The chosen approach for evaluating this data involves comparing it against a verified benchmark, specifically through a similarity analysis. Looking through the available literature, several potential similarity measures have been identified, with the chosen metric and rationale explained in the Methodology.

4.3.1 Euclidean Distance

Euclidean distance is a commonly used similarity metric for comparing the closeness of two vectors within the Euclidean geometry field [17]. The Euclidean distance $E(p, q)$ for vectors p & q is as follows:

$$E(p, q) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2} \quad (1)$$

4.3.2 Manhattan Distance

Also known as the taxicab metric, the Manhattan distance is very similar to the Euclidean distance but instead is the sum of the absolute differences in the Cartesian coordinates of two vectors [17].

4.3.3 Cosine Similarity

Cosine similarity is a normalised measure of the angle between two vectors and is a commonly used metric for similarity in text data [18]. For vectors A & B:

$$Cosine\ Similarity(A,B) = \frac{A.B}{|A||B|} \tag{2}$$

4.3.4 Jaccard Similarity Coefficient

The Jaccard coefficient, also known as the Jaccard similarity coefficient, is a measure used to calculate the similarity between two sets. It is often employed in various fields, including data mining, information retrieval, and NLP. The Jaccard coefficient is defined as the size of the intersection of the sets divided by the size of the union of the sets [19]. The coefficient is bounded by 0 and 1, with 0 being no similarity and 1 being identical. Mathematically, it is expressed as follows:

$$J(A,B) = \frac{A \cap B}{A \cup B} \tag{3}$$

4.3.5 Radial Basis Function (RBF) Kernel

The RBF Kernel computes the similarity between two samples X_1, X_2. It is heavily utilised in Support Vector Machines for large, non-linear datasets. It is widely used due to its similarity to the Gaussian Function. It is represented mathematically by:

$$K(X_1,X_2) = exp\left(\frac{\left||X_1 - X_2|\right|^2}{2\sigma^2}\right) \tag{4}$$

4.4 Test Cases

This section describes the space systems that will be used to validate the LLM outputs.

4.4.1 PROVE Pathfinder Subsystem

Pointable Radiometer for Observation of Volcanic Emissions (PROVE) Pathfinder is a CubeSat payload that detects and tracks volcanic ash clouds using novel sensing technology [20]. PROVE is a subsystem with clearly defined requirements, functions, modes, and components. This is useful as it will be used to test the LLM's use at various levels of abstraction within the system architecture. The system architecture breakdown is shown in Figure 14 in the Appendix.

4.4.2 James Webb Space Telescope System

The James Webb Space Telescope (JWST) is a space-based observatory launched in 2021 by NASA, ESA, and CSA. It uses infrared light to study the early universe, distant galaxies, exoplanets, and the solar system. Focusing on the Observatory as the test case, JWST's space segment comprises four main subsystems. First is the Optical Telescope Element (OTE), which contains mirrors and a backplane for collecting light. It includes a 6.5-meter primary mirror [21]. Second is the Integrated Science Instrument Module (ISIM) containing JWST's cameras and instruments, integrating four primary instruments and numerous subsystems into one payload [22]. Third is the sunshield, which consists of five layers of thin membranes that unfold after launch and shield the telescope from the Sun's heat and radiation [23]. Finally, the spacecraft bus provides power, propulsion, communication, and thermal control for the telescope. [24]. The JWST's complex nature and the components' uniqueness will make it a suitable test case to evaluate the effectiveness of LLMs in solving advanced systems engineering problems. The system architecture is shown in Figure 15 in the Appendix.

5. METHODOLOGY

This section aims to build a holistic, high-level framework for evaluating LLM-generated MBSE systems by comparing them against validated model test cases. Firstly, the system requirements are manually pre-processed and then used to prompt the LLM to return the functions, modes, and system architecture. The LLM NL output and validation test cases undergo a data wrangling process to clean the data and are subsequently converted into machine-readable numerical vectors. These vectors can then be compared using mathematical similarity metrics described in 4.3. An additional investigation into the repeatability of LLMs will be conducted by generating a large number of datasets and performing statistical analysis of the distribution of factors, such as the number of output sentences and the accuracy of the outputs. A summary of the similarity analysis framework is shown in Figure 1.

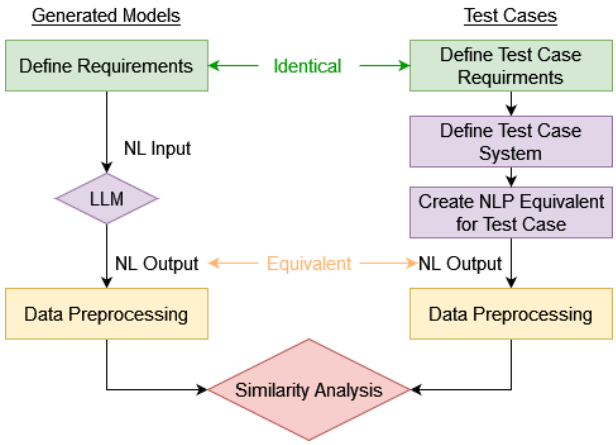

Figure 1: Framework for Empirically Evaluating LLM Outputs using Similarity Analysis

5.1 Text Data Processing

5.1.1 Pre-LLM Work and LLM Prompting

The initial phase in the system generation analysis involves establishing test cases. In the context of this study, the test cases are the PROVE subsystem test case and the James Webb Telescope System.

The following step employs the pre-defined mission and system requirements as the guiding framework for prompting the LLM to generate the systems. These requirements are relayed to the LLM, thus facilitating the assessment of the LLM's capacity to interpret and process human-authored requirements. The LLM is then prompted using a zero-shot learning prompt [25] to generate the structural elements of the mission's system, encompassing functions, modes, and components. Due to the potentially unlimited levels of abstraction of systems engineering and the token limits on the input and output of the LLM, the output system will be limited to a maximum of two layers for this paper, as shown in Figure 2.

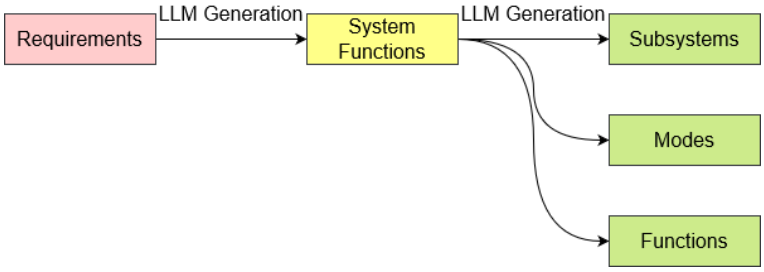

Figure 2: Systems Engineering Process for the Generation of Functions and subsequent System Architecture, Modes, and Sub Functions

The prompting of LLMs was a difficult process, especially when prompting for the design of the JWST. The requirements are over 10,000 words, roughly equivalent to 15,000 tokens (pieces of information). This nears the 16,000 tokens context limit (for prompt and response) for OpenAI's GPT 3.5-turbo model (the model available to all users), requiring the use of OpenAI's GPT-4-turbo (OpenAI's latest, most powerful model), which can be accessed using the Python API. Therefore, a custom script was produced accessing the OpenAI Python API, and the code for this task can be found in the GitHub repository provided in the Appendix.

5.1.2 LLM NL Output Processing

The LLM generates a textual corpus comprising the mission requirements, modes, and functions. This corpus serves as the basis for subsequent evaluation. The LLM offers flexibility in specifying the output format, such as employing lists or bullet points, for clarity and organisation. Upon examining the LLM-generated output, an equivalent version is constructed by aligning the functions and modes with those identified in the test cases.

The text corpora then undergo a pre-processing stage in preparation for similarity analysis, a procedure commonly referred to as data wrangling, as shown in Figure 3. This process encompasses several steps, including the conversion of the text to lowercase, elimination of HTML tags and URLs, contraction expansion, removal of special characters (e.g., punctuation), tokenisation (segmentation of the text into sentences), and the application of stemming (reducing words to pseudo-stems) or lemmatisation (reducing words to linguistically correct lemmas). Additionally, common stop words are excluded, which do not contribute substantially to the text's overall meaning. NLTK (Natural Language Toolkit), a Python library, provides comprehensive functionality for these data-wrangling tasks [26].

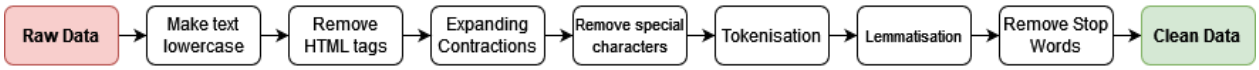

Figure 3: Data Wrangling Pipeline

5.2 Vectorisation and Analysis

5.2.1 Vectorisation using Sentence Embedding

Following the data cleaning process, the next step involves the conversion of the text into machine-readable tokens through vectorisation. Vectorisation transforms text data into numerical vectors, making it possible for machine processing [27]. Several methods exist to perform this step, such as one-hot encoding, bag-of-words, n-grams, and TF-IDF. These methods are frequency and statistical-based and are primitive compared to the modern-day alternative, word embeddings [28]. Choosing a suitable vectorisation method for text data was the first challenge. Two main methods exist: frequency-based and semantic. Frequency-based methods focus on word occurrence, while semantic methods use deep-learning NLP models to assign vectors based on context. This is crucial for complex systems where components are interconnected. For instance, the words IR, infrared, and light would yield distinct vectors in frequency-based methods but similar vectors in semantic methods due to contextual understanding. Therefore, semantic methods allow for more accurate similarity measurements between generated and validation texts. Word embeddings can be pre-trained with relevant literature for niche topics like space, enhancing semantic understanding. However, this requires significant time and computational power due to long training times.

Word embeddings are trained neural networks used to reconstruct the linguistic context of words [27]. Word embeddings have the advantage of being able to semantically evaluate words, allowing to make similarity comparisons between words that are syntactically close through semantic operations: e.g. "king" – "man" + "woman" = "queen" [29]. The only drawback is that they must be trained on large text corpora. Sentence embedding is the modification of these processes to embed sentences efficiently.

In this study, Sentence-BERT, a modification of BERT described in 4.2, was utilised to create semantic sentence embeddings for the NLP output. Many pre-trained models are available for Sentence-BERT, and there is the option to train the model manually. Due to time and hardware constraints, pre-trained models were selected for the embedding task. The model was chosen to evaluate its recognition of complex engineering jargon within the JWST requirements, extracting the semantic meaning and comparing it against simpler equivalent sentences (which have been manually simplified), shown in Table 1 and Table 6 in the Appendix. A cosine similarity analysis compares the fully complex (original) requirement against a simplified version and an even further simplified version (basic). Despite the simplification of the text, each corresponding sentence was carefully changed to ensure the underlying semantic meaning was preserved. The result of this analysis is discussed in 6.1, and 'allenai-specter', a model trained on academic material such as research papers, was chosen as the method to move forward with due to its superior performance in understanding complex engineering terminology.

5.2.2 Similarity Analysis

Two similarity analyses will be performed: corpus-to-corpus and sentence-wise comparisons. The entire text and individual elements will be converted into vectors. Various methods will assess vector similarity, as detailed in the Background section. The choice of metric depends heavily on the type of data. The output vectors of Sentence-BERT are high-dimensional vectors of fixed length. Cosine Similarity is robust in high dimensions and sensitive to vector length, which makes it ideal for our data. Meanwhile, distance-based metrics like Euclidean and Manhattan Distance struggle to capture high-dimensional data similarity. Methods that evaluate the intersection of sets, such as Jaccard and RBF, will find this data difficult to interpret. As explained in 7.1, the vectors of the same word may be slightly different, which would not count as an intersection (as they must be the same), which would skew the result. No other methods were considered due to implementation difficulty or lack of suitability. Cosine Similarity was a clear and obvious choice of metric for performing similarity analysis.

5.2.3 Repeatability Analysis

One feature of Generative AI is that the LLM may produce different outputs for the same input prompts. This is a double-edged sword for designers, as the variation in output causes inconsistencies and potential mistakes, or the variability of LLMs can be leveraged to produce a wide range of ideas.

The first task was to generate data to test the repeatability. This was performed using a call to the ChatGPT API, using both the GPT-3.5 and GPT-4.0 models, prompting them to follow the systems engineering process and generate the system architecture, modes, and functions. LLMs operate on a probabilistic basis, meaning their outputs have a degree of randomness. This randomness can be controlled using a parameter known as 'temperature'. Ranging from 0 to 1, the temperature parameter essentially adjusts the 'creativity' of the model. For instance, a temperature of 0 makes the model completely deterministic, producing the most likely output every time. Conversely, a temperature of 1 allows for a more diverse and creative output, broadening the range of possible outputs. The temperature used for the repeatability analysis was 0.7, while the similarity analysis was 0.3.

For each test case (PROVE Functions, PROVE Components, JWST System Architecture), the number of features (e.g. components or functions) and the similarity were computed and plotted into a histogram. Two main approaches were used to obtain a probability density distribution: Kernel Density Estimation and distribution fitting. Kernel Density Estimation (KDE) is a non-parametric way of producing a distribution from a histogram [30]. A KDE module in Sci-kit Learn Python Library [31], with two key parameters: bandwidth and kernel. The kernel chosen was a Gaussian kernel, and the bandwidth was found using Scott's method [32]. An alternative (and more robust) method would have been to use a K-fold cross-validation method [33], [34], but computational power and time constraints restricted its use. Parametric distribution fitting is the method of evaluating probability distributions against the data using the Residual Sum-of-Squares using a package called distfit [35].

6. RESULTS

6.1 Sentence Embedding Model Comparison

Four pre-trained models were tested: 'all-mpnet-base-V2' (the largest, supposedly most accurate general-purpose model), 'all-MiniLM-L6-V2' (the smallest and quickest general-purpose model), 'all-MiniLM-L12-V2 (a 'middle ground' model), and 'allenai-specter' (a specialised model trained on scientific journals). The four models were tested on how well they could recognise the similarity of the sentences in Table 6. Figure 16, Figure 17, and**Error! Reference source not found.** show the results of the different JWST requirements being compared to simpler equivalents. The runtime was also considered, as it has cost implications. The results show that 'allenai-specter' computed the most significant average similarity between the sentences, showing its superior ability to semantically identify complex engineering terminology. This came at the cost of a high runtime, but it was chosen as the sentence embedding model to be used in the framework.

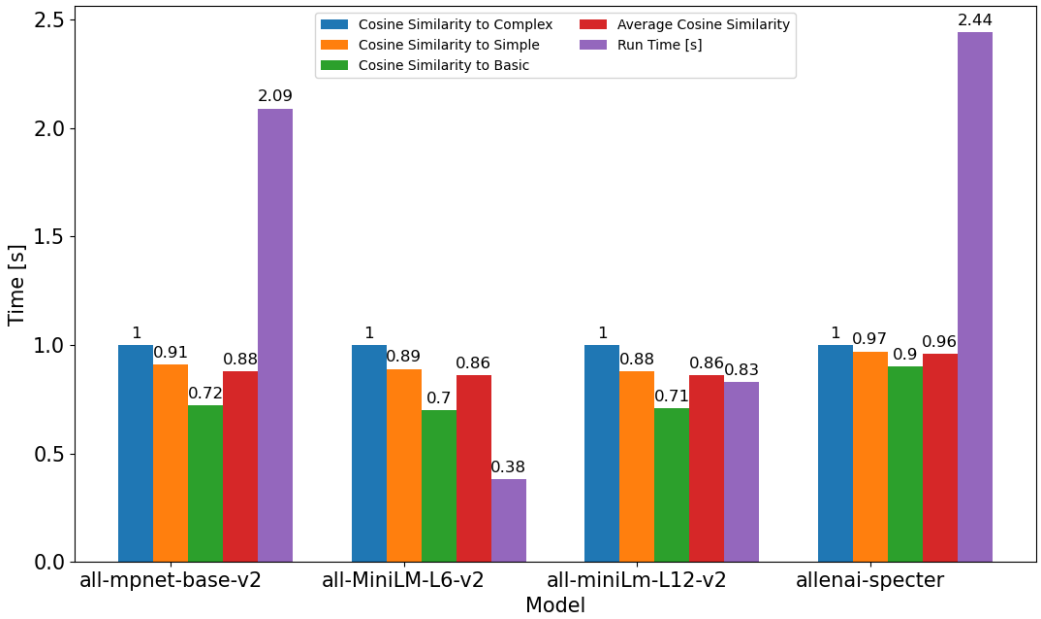

Figure 4: Sentence-BERT Pre-trained Model Comparison Results for JWST Requirement MR 271

6.2 Similarity Analysis

6.2.1 PROVE Corpus-to-corpus Similarity Analysis

The corpus-to-corpus analysis yielded high similarity for all three test cases, as shown in Table 1. This showed that the underlying semantic meaning of the entire response is very close to what the engineer is looking for, demonstrating the LLM's comprehension of the requirements and the systems engineering process. In all three test cases, the LLM included extra information that would have altered the vector, but since the underlying semantic meaning is the same, the similarity scores were still very high.

Table 1: Corpus-to-corpus Similarity Scores for the PROVE Test Case

PROVE Pathfinder	**System Functions**	**System Modes**	**System Components**
Full Corpus Cosine Similarity	0.7456	0.8056	0.8086

6.2.2 PROVE Sentence-to-sentence Similarity Analysis

The sentence-to-sentence similarity analysis revealed several themes. The average similarity was low due to the inclusion of non-corresponding sentences. A more accurate metric, computed by averaging

similarities between corresponding sentences, showed a higher similarity of around 0.5, though still less than the corpus-to-corpus values. Figure 5, Figure 6, and Figure 18 show the sentence-wise similarity scores for the functions, components, and modes, respectively, with Table 7, Table 8, and Table 9 in the Appendix with the complete sentences for both the generated and validation test cases.

Looking deeper into each of the sentences, some interesting points were found. The PROVE components included structural elements and interfacing parts, which are not components. Some of the outputs can be described as 'features' of a system, but not technically modes, functions, or components. Many of the generated PROVE functions were just repetitions of the requirements, not functions that are meant to be performed, such as 'achieve pointing accuracy of 1 degree or less' and 'maintain a total payload mass of under 2.66 kg'. Many corresponding sentences have a very low similarity despite describing the same thing, e.g., the pre-imaging mode has almost no similarity to its generated counterpart, as the phrasing is different. Still, the LLM has correctly identified an equivalent mode. The LLM traced each PROVE mode to a requirement but did not do this for the functions or components. Overall, the LLM seems to be overcomplicating the solution, adding potentially useful but incorrect or unnecessary information.

Table 2: Average Sentence-wise Similarity Scores for the PROVE Test Cases

PROVE Pathfinder	**System Functions**	**System Modes**	**System Components**
Average Sentence Similarity of Whole Corpus (Cosine)	0.1645	0.3032	0.1657
Average Sentence Similarity of Corresponding Sentences (Cosine)	0.4558	0.4280	0.5400

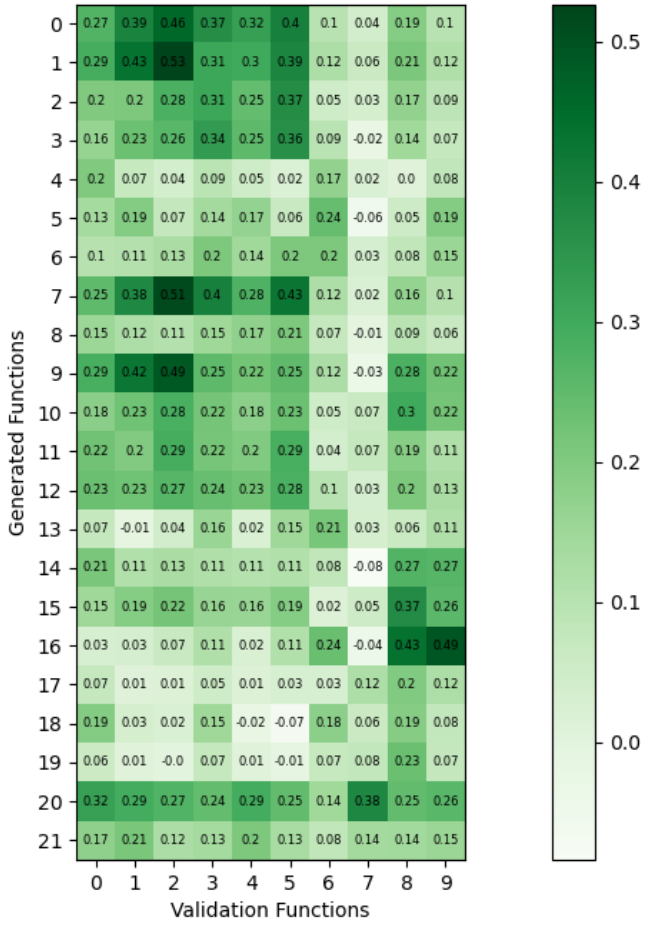

Figure 5: Sentence-wise Similarity Scores for PROVE Functions. Sentence IDs are described by Table 7 in the Appendix.

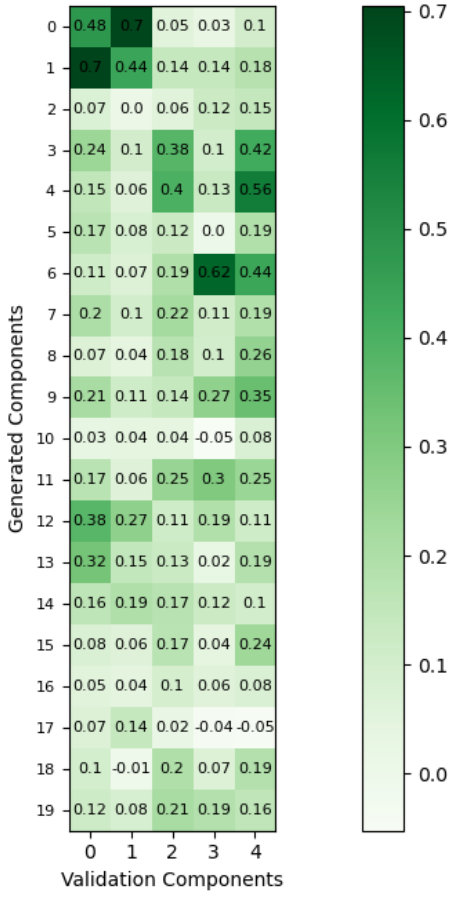

Figure 6: Sentence-wise Similarity Scores for PROVE Components. Sentence IDs are described by Table 9 in the Appendix.

6.2.3 JWST Corpus-to-corpus Similarity Analysis

Unexpectedly, the similarity values are even higher than those for PROVE, as shown in Table 3. This indicates that LLMs can perform well in systems engineering tasks for more complex systems. The System Architecture value is especially high, which can be attributed to the detailed set of input requirements and a large body of material available online regarding the JWST mission, with countless academic papers being written and a plethora of websites discussing the topic. The modes/phases response was longer than its validation counterpart, often breaking down the real-life phases into sub-phases. However, the underlying semantic meaning is still very similar, which explains the high similarity value. The only significant difference is that the phases included a lot of detail about the Ground Segment despite being prompted for the Observatory Segment modes, which lowers the similarity.

Table 3: Corpus-to-corpus Similarity Scores of the JWST Test Cases

JWST Observatory	**System Modes**	**System Architecture**
Full Corpus Cosine Similarity	0.8197	0.9487

6.2.4 JWST Sentence-to-sentence Similarity Analysis

Figure 7 shows the sentence-wise similarity scores for the JWST modes, with the complete sentences shown in Table 10 in the Appendix. Figure 8 is the sentence-wise similarity for the JWST system architecture, with the complete sentences in Table 11 in the Appendix. Figure 19, Figure 20, Figure 21, Figure 22, Figure 23, Figure 24, Figure 25, and Figure 26 show the sentence-wise similarity scores of the system architectures broken down into its subsystems, with TablesTable 12, Table 13, Table 14, Table 15, Table 16, Table 17, Table 18, and Table 19 in the Appendix containing the complete sentences for both the generated and validation test cases. The sentences are ordered as best as possible so that corresponding functions and components have as similar positions as possible. The similarity appears higher along the diagonal, with some high similarities away from the diagonal, showing that the LLM correctly identified many functions and components.

Once again, diving deeper into the corpus and looking at sentence-to-sentence analysis, a similar trend appears. The average sentence-to-sentence similarity is lower, but the value of corresponding component similarity is very high compared to the other test cases. Similarly, to PROVE, the LLM did include incorrect but useful information, such as including algorithms and software as components, for example, the inclusion of a sunshield algorithm. It also ‘overcomplicated’ the system phases, including many more phases. These additional phases seemed to be parts of the validation phases (e.g. the orbit transfer, commissioning, and deployment phases are combined into one phase for the real-life example).

Unlike the PROVE test case, however, the LLM failed to consider every aspect of the system architecture. Firstly, looking at the subsystems, the LLM provided a comprehensive description of a thermal subsystem but failed to describe a propulsive subsystem. This is a strange result but can be explained by the presence of a detailed set of thermal requirements but no propulsion requirements. It also failed to identify an omnidirectional antenna, a tertiary mirror, and many of the crucial actuators and mechanisms. Certain components, such as the batteries and the Control and Data handling computer, seemed to have relatively low similarities despite being correctly identified by the LLM. This can be explained by the fact that the LLM-generated versions of these components often included extra text, usually explaining how the component works, which changes the vectorisation, which decreases the similarity.

Table 4: Average Sentence-wise Similarity Scores for JWST Test Cases

JWST Observatory	**System Modes/Phases**	**System Architecture**
Average Sentence Cosine Similarity	0.3463	0.1576
Average Cosine Similarity of Corresponding Sentences	0.488	0.7162

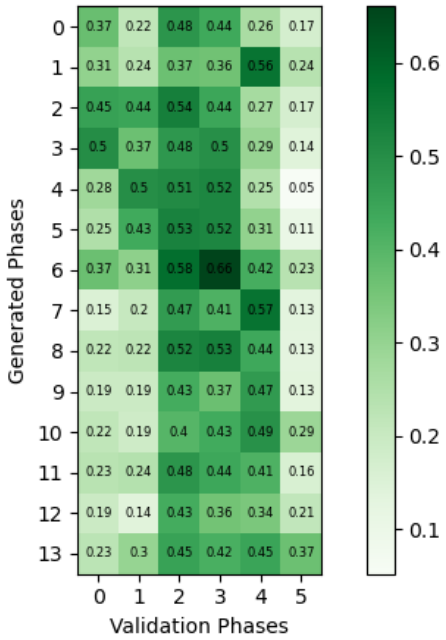

Figure 7: Sentence-wise Similarity Scores for the JWST Phases. Sentence IDs are described by Table 10 by the Appendix.

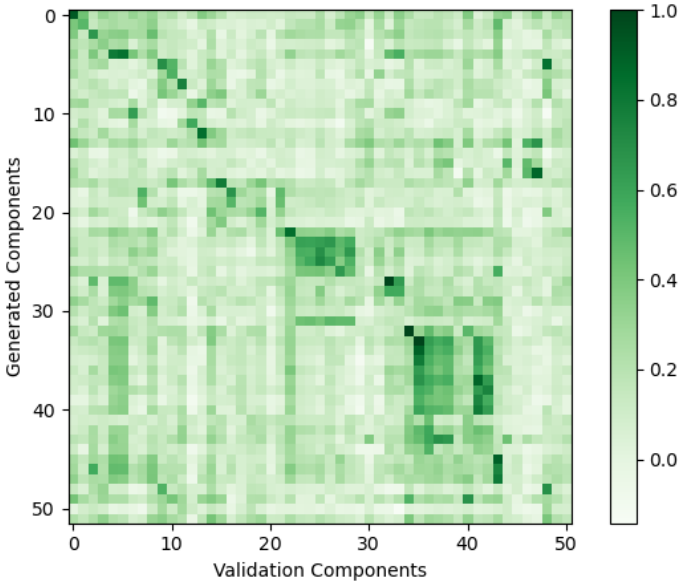

Figure 8: Sentence-wise Similarity Scores for the JWST System Architecture. Sentence IDs are described in Table 11 in the Appendix.

6.3 Repeatability Analysis

Three main test cases were investigated for similarity analysis: the PROVE functions, PROVE components, and JWST system architecture. 1096 data points were analysed for the PROVE functions, while the other two test cases were evaluated with 256 data points due to time constraints.

The PROVE functions followed a very interesting trend, which resulted in a bimodal distribution for both the number of outputs and the overall similarity to the validation dataset. Analysing the actual text revealed that the LLM outputs followed two general trends: verbose and concise. The verbose outputs usually contained more outputs, and each of the functions often came with additional descriptors, while the concise outputs were the opposite, with fewer outputs described using fewer words. Unsurprisingly, the longer outputs varied more in terms of number of outputs and similarity, as shown in Figure 9, where the right peaks were wider than the left peaks. This is reinforced by Figure 10, where the data was manually clustered into two separate groups: red represents the concise and less variable responses, while green corresponds to the more verbose and widely varying responses. Surprisingly, the more concise responses had a lower similarity, which indicates a lack of understanding about the system. The verbose responses had a slightly higher similarity, which may be attributed to there being more detail, some of which was semantically similar to the validation case.

A KDE using Scott's method to calculate bandwidth yielded a visually suitable distribution that captures the bimodality of the data. The parametric distributions were a Double Weibull distribution and a Beta distribution for the number of outputs and corpus similarity, respectively. The Double Weibull captures the bimodality, but it is a symmetrical distribution, which fails the skewness of the number of outputs. Conversely, the Beta fails to capture the bimodality but much more accurately captures the distribution at the extremes.

The PROVE components did not follow the same trend as the functions, but instead, the number of components showed a decaying distribution, while the similarity followed an opposing trend. The spread of the similarity is also greater, ranging from 64.1% to 88.5%. The number of outputs was fitted with a Lognormal distribution, while the similarity was fitted with a Log-gamma distribution. The parametric

distribution captured the data trend well for the number of outputs but failed to capture the separate peaks in the similarity, with the KDEs capturing this nuance in the distribution, as shown in Figure 11.

Inspecting the sets of components individually, all the sets of responses contained the validation components, which is a positive result. However, common additional components often include structural features (e.g. ‘Payload Structure Vibration Resistance System’, ‘Structural Support System’, and ‘Payload Structure Sine Sweep Resistance System’), which can be directly traced back to structural requirements. These are useful, but once again, they are not formal components but rather considerations for the design of the structure. The components also sometimes contained repetitions (e.g. ‘FLIR Tau IR Camera’ and ‘Infrared Camera’), which showed that the model failed always to understand they are the same thing. Finally, some of the sets of components included the Requirement IDs (e.g. CDH-Bus-02), which showed that the LLM misunderstood the ID as an acronym for a component. This unnecessary text explains the long right-handed tail of the output number distribution and left hand side of the similarity distribution.

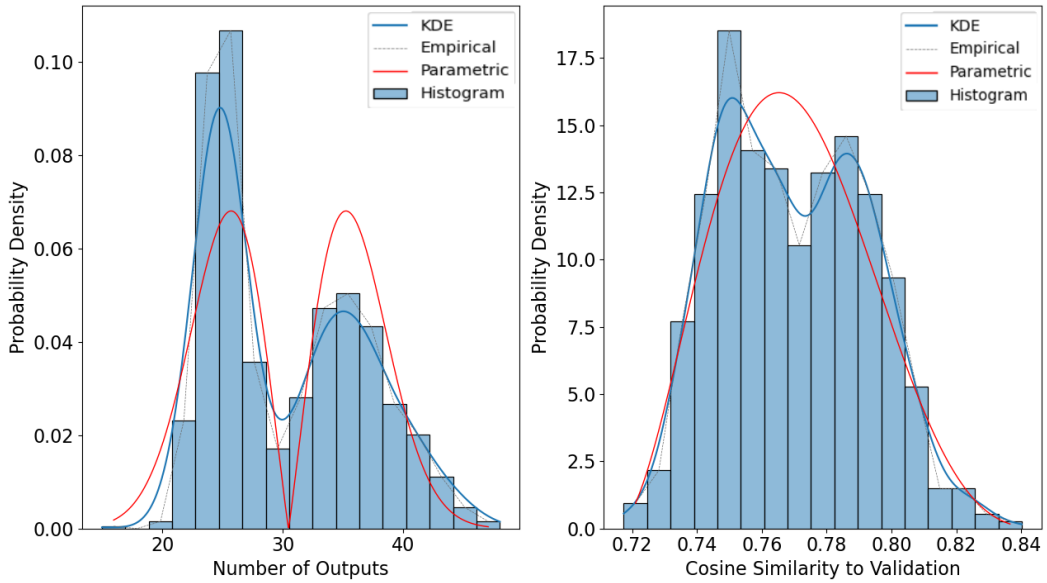

Figure 9: Repeatability Analysis Results for PROVE Functions.

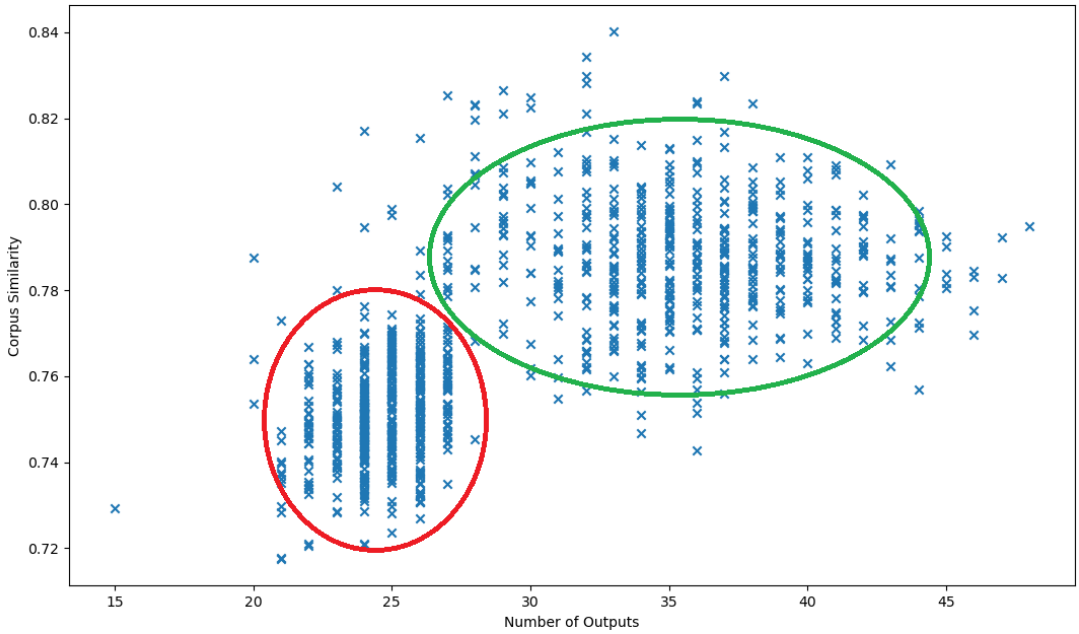

Figure 10: PROVE Functions' Similarity against the Number of Outputs

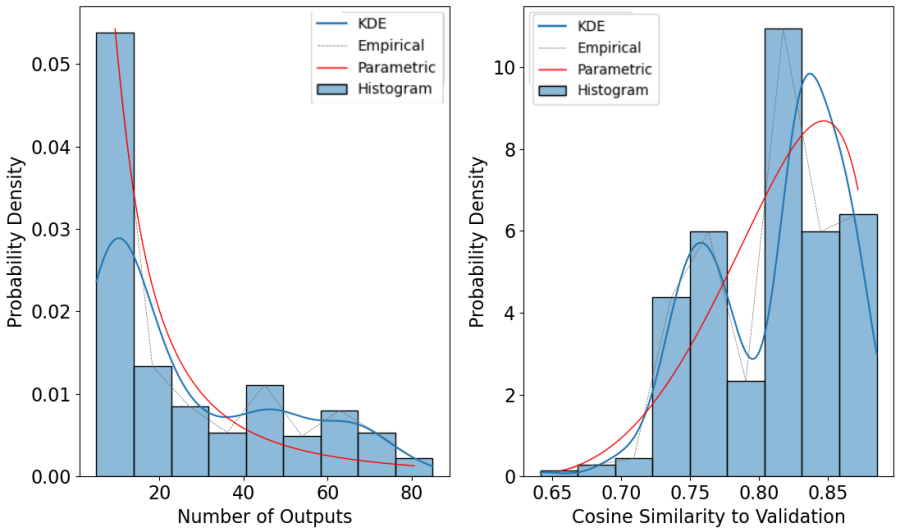

Figure 11: Repeatability Analysis Results for PROVE Components

When generating the dataset for the JWST system architecture, there was an issue with the API, and the data set had to be generated in two steps. However, there was a slight disparity in the prompts, which led to very different distributions, with the distributions in Figure 12 describing the responses when prompted for *components* and Figure 13 when prompted for *component names*. Apart from this disparity, every other part of the prompt and the LLM parameters were identical, which has created an interesting point to discuss. The first dataset will be referred to as dataset A, while the second dataset will be referred to as dataset B.

Looking at Figure 12, the number of outputs shows the same bimodality as the PROVE functions for dataset A. Similar to the PROVE Functions, there seems to be a clear peak for corpus similarity. Analysing the distributions in Figure 12. The two peaks represent two distinct types of output, one being a correct set of system components and the other being an incorrect mix of system components and functions. The left peak represents the list of correct components, where you see the highest similarity is around 50 inputs, which is the number of components in the real JWST Observatory. This also correlates with the peak in the similarity distribution. There was also a higher and lower spread for the number of outputs and similarity distribution, respectively. This is because, when prompting for components, some of the responses were a mix of both components (and a short description) and system functions.

When investigating the responses when asking for the component name, displayed in Figure 13, both the similarity and number of outputs are much greater at around 95% similarity and 50 outputs, respectively, and with a much smaller spread. The responses were much more accurate and concise, and the whole dataset followed the same trend, with many responses being identical.

Both parametric and non-parametric distributions described dataset A's number of outputs well, as shown in Figure 12. The KDE captures the variation more accurately, but the parametric methods capture the height of the peaks better. In dataset B, shown by Figure 13, the parametric distributions performed much better, as they did not include the outlier terms (high number of output and low similarity), but KDEs can be sensitive to outliers. The KDEs did, however, capture the peaks better than the parametric distributions.

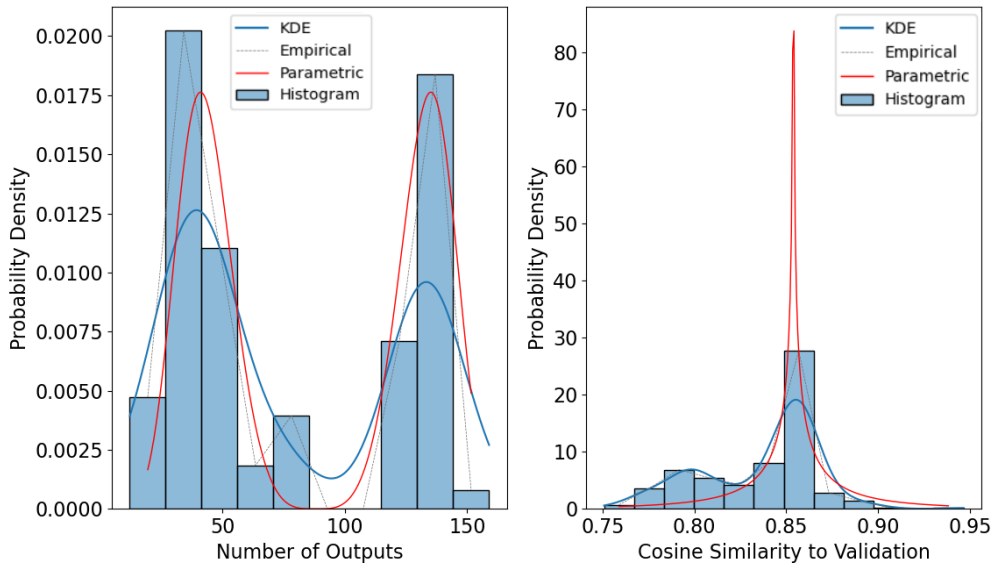

Figure 12: Repeatability Analysis Results for JWST System Architectures prompting for Components.

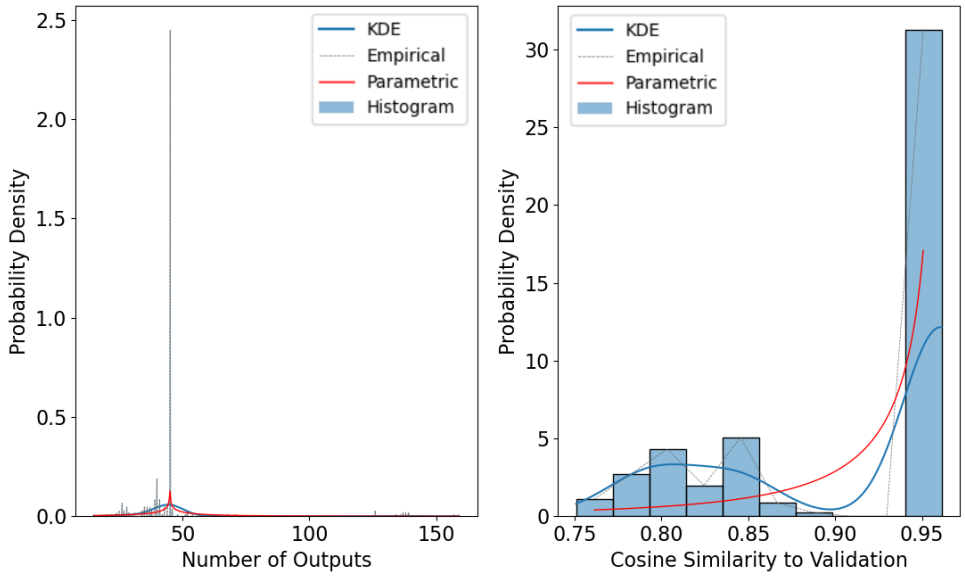

Figure 13: Repeatability Analysis Results for JWST System Architectures Prompting for Component Names

Table 5: SciPy Parametric Distributions Fitted to the Number of Outputs and Corpus Similarity for each Test Case

	PROVE Functions		**PROVE Components**		**JWST System Architecture (Components)**		**JWST System Architecture (Component Names)**	
	Number of Outputs	*Corpus Similarity*	*Number of Outputs*	*Corpus Similarity*	*Number of Outputs*	*Corpus Similarity*	*Number of Outputs*	*Corpus Similarity*
Distribution	Double Weibull	Beta	Lognormal	Log-gamma	Double Weibull	Weibull	Exponential	Log-gamma
loc	30.472	0.714	4.873	0.872	88.056	0.853	44.000	0.961
scale	6.498	0.137	11.384	0.012	49.626	0.014	3.097	0.000

7. DISCUSSION

7.1 Effectiveness of LLMs to Generate MBSE Systems

The results of this paper show that LLM-generated models are very semantically like real-life systems when comparing the whole output corpus. GPT-4 showed a 94.87% similarity when comparing the system architecture of the JWST and 81.97% when comparing the modes. The key performance indicator is the JWST system architecture, which is incredibly complex and has highly specialised components. This implies that LLMs can be effective in creating reliable and robust systems, just as a team of system engineers has done.

Looking deeper into individual functions, modes, or components; there seems to be a much lower similarity. For example, looking at the similarity of the corresponding sentence, the similarity of each of the JWST components is 71.62%. This is still significant but not as similar as corpus-to-corpus similarity. LLMs can produce alternative solutions, which are semantically like the validation cases but often with alternative wording and extra information, which lowers the similarity. To avoid this, a set of more equivalent validation cases could have been created, e.g. each function, mode, and component could have come with a description, but this also comes with the risk of the 'correct' responses having a lower similarity. Tailoring the validation for each of the responses could also lead to bias.

There seems to be a greater similarity when comparing the system architecture than when comparing modes and functions. The explanation for this is that components and subsystems are described by more 'fixed' language. Contrastingly, modes and functions are more abstract, which increases the possible combinations of words to describe them. Considering the probabilistic nature of LLMs, this leads to an output that has lower similarity but not necessarily a lower accuracy.

The requirements needed a lot of work before being inputted into the LLM (such as the removal of metadata and non-ASCII text), and certain formatting decisions may have changed the output of the LLM. The JWST requirements had to be manually rewritten from a PDF file, as many mathematical formulae and tables did not copy over directly. A lot of rewording was done throughout the process as well (e.g. $m2 \rightarrow m^2$). Also, the requirement IDs were left in the requirements to help with traceability, but on occasion, the LLM misunderstood the requirement IDs as acronyms for components, incorrectly identifying them as components. A particularly interesting observation is that the LLM created a thermal subsystem for JWST but did not include a propulsion subsystem. The black-box nature of LLMs meant that it is hard to inspect why it failed to include one but included another, but in this case, the reason lies with the systems engineer. The JWST mission requirements fail to mention anything about propulsion and manoeuvre, yet had a very detailed section on thermal protection, which highlighted a key warning to those using LLMs to aid the design process: *an AI system is only as powerful as the quality of the information which is provided and the competency of the user.*

In this case, the engineer is to blame for the 'incorrect' output, but engineers using LLMs must also be aware of hallucinations in machines [36], where LLMs produce objectively false outputs. An example of this was in the JWST system architecture, which repeatedly includes software and algorithms as components. The potential reason for this is that the requirements for both PROVE and JWST contained structural, electrical, and control requirements, which the LLM understood to be a need for a component or subsystem. This is not necessarily wrong, as these functions were derived from the mission requirements, but it showed a difference in the level of detail required and the nuance of what makes something a feature or an object in the system architecture. This is an example of a wider issue in using LLMs for systems engineering; LLMs do not follow the Systems Engineering process. The distinction between a system, subsystem, and component is essential in systems engineering, but LLMs do not 'understand' this concept. This emphasises the need for systems engineers who can validate the output to ensure the output is a correct systems architecture. This further reinforces the point that LLMs are useful tools for concept generation

and ensuring engineers do not miss useful information, but not as a standalone MBSE tool that comprehensively performs systems engineering tasks.

In an increasingly complex world, this paper demonstrates that LLMs can scale to solve complex problems. The study evaluated LLMs against a simple (PROVE) and a complex (JWST) system, showing that LLMs can perform systems engineering tasks on complex systems like JWST, even outperforming its own performance on simpler use cases. With system architecture and phase similarities of 94.87% and 81.97%, respectively. It is clear that LLMs can tackle complex problems. As research in NLP and LLMs grows and computational power advances, more powerful models will emerge to address humanity's challenges. The paper showed that JWST similarity scores were higher than PROVE values, due not only to LLM scalability but also to the systems engineering process of JWST. The extensive time and manpower invested in JWST led to more constrained requirements, reducing potential modes and system architectures in the design space, hence a higher similarity to the validation datasets.

The repeatability analysis revealed several key considerations when using LLMs for the systems engineering process. Firstly, it highlighted the importance of the temperature parameter. When set to a higher value, such as 0.7, the outputs were varied in both the number of outputs in the response as well as the content of the response itself. The variation of temperature did not change the fact that LLMs will output incorrect information, as this occurred at both a temperature of 0.3 (similarity analysis test cases) and 0.7 (repeatability analysis test cases). This showed that the LLMs have an inherent lack of understanding of the systems engineering process rather than it being an effect of their probabilistic nature. A useful investigation would have been to perform the analysis at a range of temperatures to understand its effect on the output.

However, another big takeaway from the analysis was the importance of prompting. A single-word adjustment to the prompt led to significantly different datasets: one data set was hugely varied with a significantly lower accuracy (77% average), and the other was a dataset with very high accuracy and low variation (96% average). The prompt change also led to the format of the output being varied, with one set being very verbose with a wide bimodal distribution in the number of outputs, while the other dataset contained around 50 components for nearly all responses, with the validation system architecture having 51 components. This has big implications for the use of LLMs, as improper prompting could lead to ineffective use, preventing engineers from maximising their usefulness.

Another interesting thing is that the distribution of the responses does not tend to normal distributions (due to the probabilistic nature of the LLMs) but instead often follows specific patterns/trends. This leads to very 'peaky' distributions. An example of this would be in the PROVE functions, where the responses fell into two categories: concise and verbose, with varying levels of detail and accuracy. Please be wary of explicitly prompting for a set of concise functions/modes/components, as this leads to the LLM missing information. One issue not addressed in this paper was how to understand semantically which responses were concise or verbose automatically because the format and content of each test case varied widely between outputs. Instead, every individual output had to be manually inspected to look for trends in the data. Also, a larger dataset could have returned a more representative distribution and using a higher number of bins for the histogram may have revealed smaller micro-trends in the data. Another improvement would have been to use a clustering algorithm to identify trends in the data, e.g. which output belonged to which trend, and that could have provided more useful insights.

The unsymmetrical peakiness of the data indicated that KDEs are the preferred method for fitting a distribution to the output variation, as parametric methods fail to capture this. The KDEs do, however, fail to capture the magnitude of the peaks. This can be fixed by increasing the bandwidth, but this can lead to overfitting. The challenge is finding the optimal kernel and bandwidth for the KDE. Scott's method was used, but a more robust method would be using a cross-validation method, such as K-fold. Just beware that this method is very computationally expensive for large datasets with a large number of folds. The main issue is that the LLM does not seem to be following any trend based on the prompt; there seems to be very

little correlation between the request (e.g. function or system architecture) and the output variation distribution. This makes it difficult to apply the generated distributions to predict the variability of the responses to different prompts and tasks.

If engineers want to follow the systems engineering process, getting the system modes, functions, and system architecture, using carefully constrained prompts with a lower temperature value, and then looking at the shorter outputs would return better responses. Opposing this, if the engineer wants to explore the design space and develop a deeper insight, then leaving the prompt open-ended and using a higher temperature would lead to a better range of design ideas. Some of the responses even traced back each output to their requirements, which demonstrates their usefulness. Either way, LLMs are extremely useful in the early design stages for the initial conceptual design phase.

The aerospace industry is notoriously stringent when it comes to proprietary information and data security, as the largest companies work as defence and government contractors. The possibility of data leaks and loss of proprietary information is a large risk when using LLMs and is a very important consideration when using them. The black-box nature of LLMs and the fact they are developed by external companies mean it is very hard to have transparency about how the information is handled. As explained earlier, LLMs should not be used as an automated process to replace systems engineers but as a complementary tool, not just because of the concerns of accuracy and systems engineering understanding but also for safety and ethical reasons. The decision-making process is vital in engineering, and this process needs the systems thinking process to ensure decisions are made and checked by considering the social and ethical implications, which a machine is still currently unable to do [37].

This paper used empirical data to evaluate the effectiveness of LLMs, which returned useful insights, but this required having the data available. This is not always available, as was the case with the JWST functions. Companies do not always follow the process formally, especially in large projects involving contractors, which leads to the unavailability of information. An even greater issue is that for novel systems, there may be no equivalent empirical data to work with, which makes the framework in this paper redundant for evaluating novel systems.

8. FUTURE WORK

Throughout this paper, GPT-3 and GPT-4 were used, with zero-shot learning prompts. The logical next step would be either to use few-shot learning [38] or to fine-tune GPT-4. Few-shot learning involves providing the LLM with a series of examples and definitions to aid the model's generalisation capabilities, while fine-tuning involves providing extra training data including academic papers, system documentation, and other text corpora on the topic of space systems and systems engineering. This would give the machine a greater understanding of how to break down requirements into functions, modes and components, providing greater performance. An investigation into the performance of other LLM models could also return useful insights.

LLM responses often contain unnecessary information when responding to prompts, which can be tackled by a variety of methods. One would be the formalisation of prompts through a prompt engineering investigation for the systems engineering process. It would hopefully constrain LLMs so the outputs are in the desired format. The other way would be to standardise the output itself by creating a process to transform the verbose outputs into exactly the required format. This task could utilise the latest NLP techniques, even LLMs themselves, in a two-step process to extract the primary semantic meaning of the output and reword it to the desired format. This way, engineers can perform a 'quality check' on LLM responses and ensure the output is correct and understandable.

This paper evaluates the effectiveness of LLMs by using pre-existing systems. This requires having information about the system. A framework to evaluate the performance of LLM outputs for novel systems without the need for empirical data is yet to be explored and is crucial to future uses of LLMs in systems

engineering. An example methodology would be ensuring outputs can be traced back to requirements through semantic similarity using a combination of graph theory and NLP.

Finally, the goal of this research is to investigate the feasibility of integrating LLMs into MBSE. The integration of LLMs into MBSE tools is crucial for their use in design, though not explored in-depth in this paper. LLMs output text data, requiring a pipeline to convert this into program-specific code. Due to their probabilistic nature, LLM outputs can vary, necessitating a standardisation process before integration into MBSE tools. Python-based SysML packages like Gaphor could be useful for integration, given LLMs' accessibility via a Python API.

9. CONCLUSION

From the results of this paper, several recommendations can be made for using LLMs for the generation of MBSE systems. In their current state, LLMs can be used to generate useful information about system functions, modes, and components from a set of requirements. The main takeaway is that the quality of the LLM's outputs is dependent on the quality of their inputs.

The quality of the requirements is the most important aspect. The outputs are solely derived from these requirements, so providing a set of comprehensive and constrained requirements is crucial. The quality of the prompt will have a huge effect on the output format, which drives the understandability and readability of the output. Theoretically speaking, if a perfect set of requirements is used in combination with a fully constrained prompt, LLM responses will produce a syntactically and semantically set of functions, modes, and components. Even with the perfect inputs, they may still provide extra and unnecessary information, which can hinder the systems engineering process but potentially offer insights into information missed by the user. This is more relevant for modes and functions, which are more syntactically 'open', while the subsystems and components of the system architecture are more 'fixed'. The temperature parameter is also important, and its value should depend on the goal of the user. If the user wants a single, most accurate answer, then a low temperature is to be used. If the user wishes to explore the design space, then generating a selection of outputs using a high temperature will yield varying responses, providing a range of possible solutions. The user should be aware that LLM outputs like to follow certain trends depending on the input prompt and parameters.

The general trend was that LLMs like to output 'system features', useful information that the engineer should consider as a part of their system, from which the engineer should use their engineering judgement and extract the optimal solution. This paper recommends that LLMs be utilised in the early conceptual design phase as LLMs are not yet suitable to be used to replace the systems engineering process, but instead, in the right hands, a powerful tool to supplement the design process.

10. REFERENCES

[1] OpenAI, 'GPT-4 Technical Report'. arXiv, Mar. 27, 2023. doi: 10.48550/arXiv.2303.08774.

[2] S. Casper *et al.*, 'Open Problems and Fundamental Limitations of Reinforcement Learning from Human Feedback'. arXiv, Sep. 11, 2023. doi: 10.48550/arXiv.2307.15217.

[3] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, 'High-Resolution Image Synthesis with Latent Diffusion Models'. arXiv, Apr. 13, 2022. doi: 10.48550/arXiv.2112.10752.

[4] A. Lewkowycz *et al.*, 'Solving Quantitative Reasoning Problems with Language Models'. arXiv, Jun. 30, 2022. doi: 10.48550/arXiv.2206.14858.

[5] Y. Li *et al.*, 'Competition-Level Code Generation with AlphaCode', *Science*, vol. 378, no. 6624, pp. 1092–1097, Dec. 2022, doi: 10.1126/science.abq1158.

[6] J. Gregory, L. Berthoud, T. Tryfonas, A. Rossignol, and L. Faure, 'The long and winding road: MBSE adoption for functional avionics of spacecraft', *J. Syst. Softw.*, vol. 160, p. 110453, Feb. 2020, doi: 10.1016/j.jss.2019.110453.

[7] J. Gregory, L. Berthoud, T. Tryfonas, and L. Faure, ‘There’s no “I” in SEAM - An Interim Report on the “Spacecraft Early Analysis Model”’, Mar. 2020. doi: 10.1109/AERO47225.2020.9172702.
[8] L. Timperley, L. Berthoud, C. Snider, T. Tryfonas, A. Prezzavento, and K. Palmer, ‘Towards Improving the Design Space Exploration Process Using Generative Design With MBSE’, Mar. 2023. doi: 10.1109/AERO55745.2023.10116019.
[9] A. Tikayat Ray, B. F. Cole, O. J. Pinon Fischer, A. P. Bhat, R. T. White, and D. N. Mavris, ‘Agile Methodology for the Standardization of Engineering Requirements Using Large Language Models’, *Systems*, vol. 11, no. 7, Art. no. 7, Jul. 2023, doi: 10.3390/systems11070352.
[10] A. L. Ramos, J. V. Ferreira, and J. Barceló, ‘Model-Based Systems Engineering: An Emerging Approach for Modern Systems’, *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 42, no. 1, pp. 101–111, Jan. 2012, doi: 10.1109/TSMCC.2011.2106495.
[11] C. J. H. Mann, ‘A Practical Guide to SysML: The Systems Modeling Language’, *Kybernetes*, vol. 38, no. 1/2, Feb. 2009, doi: 10.1108/k.2009.06738aae.004.
[12] N. Kalchbrenner and P. Blunsom, ‘Recurrent Continuous Translation Models’, 2014.
[13] D. Bahdanau, K. Cho, and Y. Bengio, ‘Neural Machine Translation by Jointly Learning to Align and Translate’, *ArXiv*, vol. 1409, Sep. 2014.
[14] A. Vaswani *et al.*, ‘Attention is All you Need’, 2017.
[15] A. Radford and K. Narasimhan, ‘Improving Language Understanding by Generative Pre-Training’, 2018. Accessed: Oct. 11, 2023. [Online]. Available: https://www.semanticscholar.org/paper/Improving-Language-Understanding-by-Generative-Radford-Narasimhan/cd18800a0fe0b668a1cc19f2ec95b5003d0a5035
[16] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, ‘BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding’. arXiv, May 24, 2019. doi: 10.48550/arXiv.1810.04805.
[17] T. W. Schoenharl and G. Madey, ‘Evaluation of Measurement Techniques for the Validation of Agent-Based Simulations Against Streaming Data’, in *Computational Science – ICCS 2008*, M. Bubak, G. D. van Albada, J. Dongarra, and P. M. A. Sloot, Eds., in Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2008, pp. 6–15. doi: 10.1007/978-3-540-69389-5_3.
[18] J. Han, M. Kamber, and J. Pei, ‘2 - Getting to Know Your Data’, in *Data Mining (Third Edition)*, J. Han, M. Kamber, and J. Pei, Eds., in The Morgan Kaufmann Series in Data Management Systems. , Boston: Morgan Kaufmann, 2012, pp. 39–82. doi: 10.1016/B978-0-12-381479-1.00002-2.
[19] N. C. Chung, B. Miasojedow, M. Startek, and A. Gambin, ‘Jaccard/Tanimoto similarity test and estimation methods’, *BMC Bioinformatics*, vol. 20, no. S15, p. 644, Dec. 2019, doi: 10.1186/s12859-019-3118-5.
[20] ‘PROVE-Pathfinder – University of Bristol Satellite’. Accessed: Dec. 16, 2023. [Online]. Available: https://uobsat.uk/prove-pathfinder/
[21] ‘Optical Telescope Element: James Webb Space Telescope’. Accessed: Dec. 16, 2023. [Online]. Available: https://webb.nasa.gov/content/observatory/ote/index.html
[22] ‘Instruments and ISIM (Integrated Science Instrument Module) Webb/NASA’. Accessed: Dec. 16, 2023. [Online]. Available: https://webb.nasa.gov/content/observatory/instruments/index.html
[23] ‘The Sunshield Webb/NASA’. Accessed: Dec. 16, 2023. [Online]. Available: https://webb.nasa.gov/content/observatory/sunshield.html
[24] ‘Spacecraft Bus Webb/NASA’. Accessed: Dec. 16, 2023. [Online]. Available: https://webb.nasa.gov/content/observatory/bus.html
[25] Y. Xian, C. H. Lampert, B. Schiele, and Z. Akata, ‘Zero-Shot Learning -- A Comprehensive Evaluation of the Good, the Bad and the Ugly’. arXiv, Sep. 23, 2020. doi: 10.48550/arXiv.1707.00600.
[26] E. Loper and S. Bird, ‘NLTK: The Natural Language Toolkit’. arXiv, May 17, 2002. doi: 10.48550/arXiv.cs/0205028.
[27] T. Mikolov, K. Chen, G. Corrado, and J. Dean, ‘Efficient Estimation of Word Representations in Vector Space’. arXiv, Sep. 06, 2013. Accessed: Nov. 08, 2023. [Online]. Available: http://arxiv.org/abs/1301.3781
[28] E. Rudkowsky, M. Haselmayer, M. Wastian, M. Jenny, Š. Emrich, and M. Sedlmair, ‘More than Bags of Words: Sentiment Analysis with Word Embeddings’, *Commun. Methods Meas.*, vol. 12, no. 2–3, pp. 140–157, Apr. 2018, doi: 10.1080/19312458.2018.1455817.

[29] Y. Shao, S. Taylor, N. Marshall, C. Morioka, and Q. Zeng-Treitler, 'Clinical Text Classification with Word Embedding Features vs. Bag-of-Words Features', in *2018 IEEE International Conference on Big Data (Big Data)*, Dec. 2018, pp. 2874–2878. doi: 10.1109/BigData.2018.8622345.
[30] Y.-C. Chen, 'Lecture 6: Density Estimation: Histogram and Kernel Density Estimator'.
[31] F. Pedregosa *et al.*, 'Scikit-learn: Machine Learning in Python', *J. Mach. Learn. Res.*, vol. 12, no. 85, pp. 2825–2830, 2011.
[32] N.-B. Heidenreich, A. Schindler, and S. Sperlich, 'Bandwidth selection for kernel density estimation: a review of fully automatic selectors', *AStA Adv. Stat. Anal.*, vol. 97, no. 4, pp. 403–433, Oct. 2013, doi: 10.1007/s10182-013-0216-y.
[33] A. Charpentier, 'Cross Validation for Kernel Density Estimation | R-bloggers'. Accessed: Apr. 10, 2024. [Online]. Available: https://www.r-bloggers.com/2014/10/cross-validation-for-kernel-density-estimation/
[34] V. K. Verma, K. Saxena, and U. Banodha, 'Analysis Effect of K Values Used in K Fold Cross Validation for Enhancing Performance of Machine Learning Model with Decision Tree', in *Advanced Computing*, D. Garg, J. J. P. C. Rodrigues, S. K. Gupta, X. Cheng, P. Sarao, and G. S. Patel, Eds., Cham: Springer Nature Switzerland, 2024, pp. 374–396. doi: 10.1007/978-3-031-56700-1_30.
[35] 'distfit/CITATION.cff at master · erdogant/distfit'. Accessed: Apr. 10, 2024. [Online]. Available: https://github.com/erdogant/distfit/blob/master/CITATION.cff
[36] L. Huang *et al.*, 'A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions'. arXiv, Nov. 09, 2023. doi: 10.48550/arXiv.2311.05232.
[37] R. Watkins, 'Guidance for researchers and peer-reviewers on the ethical use of Large Language Models (LLMs) in scientific research workflows', *AI Ethics*, pp. s43681-023-00294–5, May 2023, doi: 10.1007/s43681-023-00294-5.
[38] A. Parnami and M. Lee, 'Learning from Few Examples: A Summary of Approaches to Few-Shot Learning'. arXiv, Mar. 07, 2022. doi: 10.48550/arXiv.2203.04291.

11. ACKNOWLEDGEMENTS

I would like to express my upmost gratitude and appreciation for Professor Lucy Berthoud and Louis Timperly for inspiring me and guiding me throughout this journey, with their invaluable insights and expertise. Without their support, I would still be scratching my head and staring at a blank document. I would also like to thank Professor Hong Bui for being my both my harshest critic and biggest supporter, to whom I owe everything,

12. APPENDIX

GitHub Repository: https://github.com/jacknguyen0810/LLM_2_MBSE_RP3

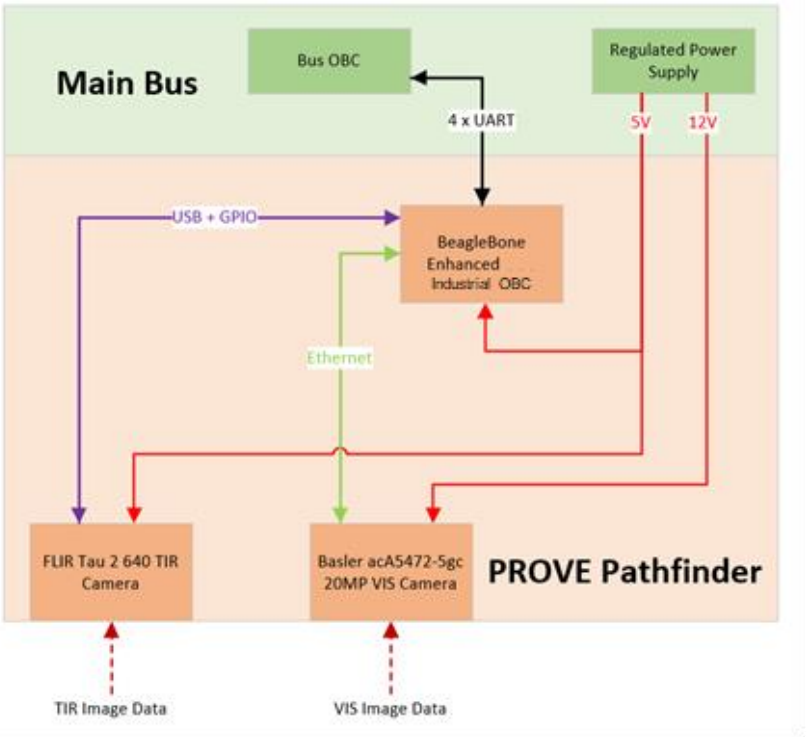

Figure 14: PROVE Pathfinder Satellite Payload System Architecture Schematic

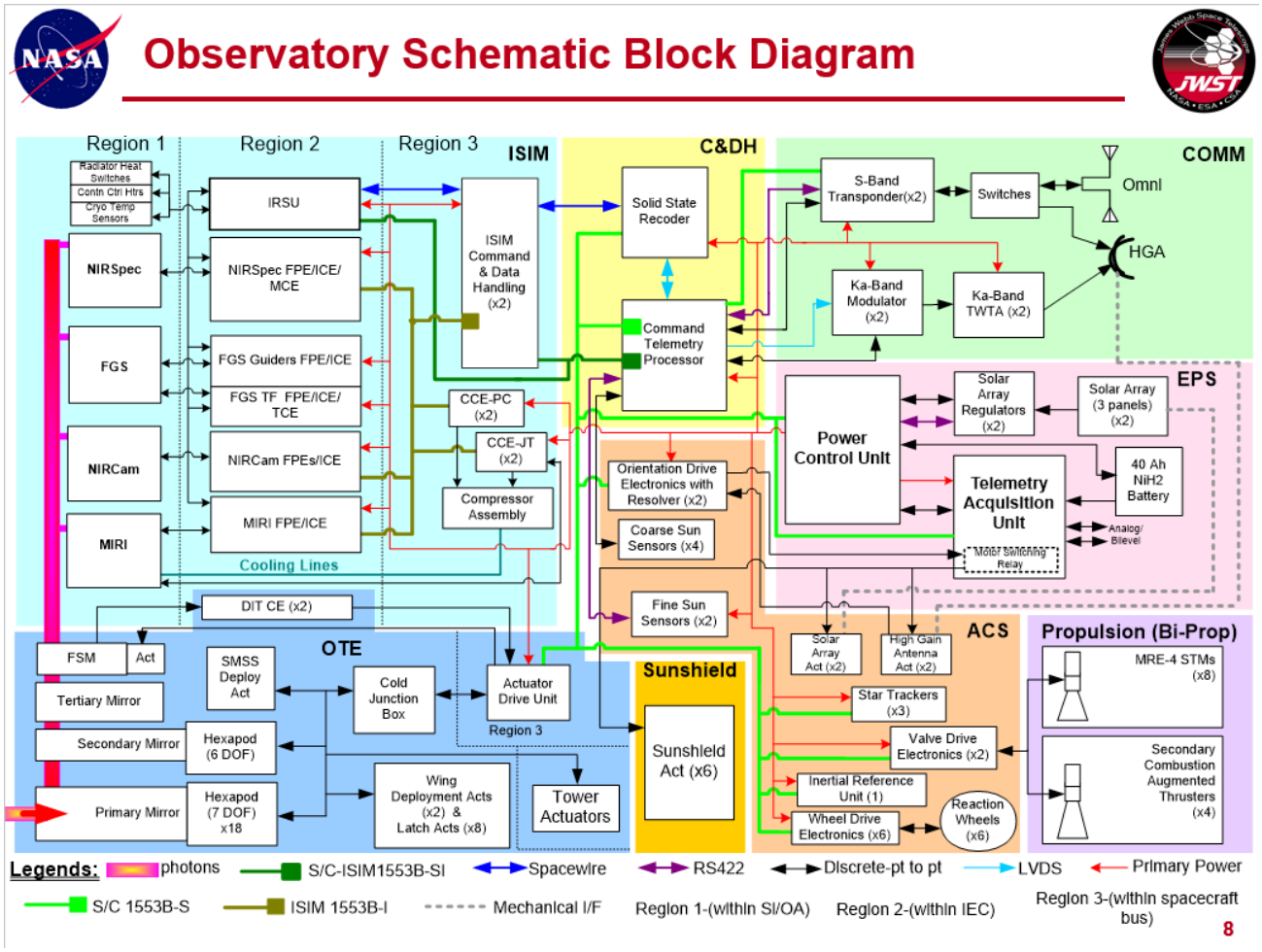

Figure 15: JWST Observatory System Architecture Schematic

Table 6: JWST Requirements for the Comparison of Sentence-BERT models, with corresponding simplified and basic versions.

Req ID:	*Full Complexity Example*	*Simplified Version*	*Basic Version*
MR 172	When commanded, the Observatory shall point the OTE boresight to an accuracy of better than or equal to 7 arcsec (1-sigma, per axis) without using the FGS or SIs. This is the maximum allowable difference in angle between the commanded pointing direction and the actual pointing direction in celestial coordinates. Boresight pointing axes are pitch and yaw. This requirement does not apply to roll around the boresight axis.	When commanded, the Observatory shall point the satellite's optical system to an accuracy of better than or equal to 7 arcsec, without using the FGS. This is the maximum allowable angle between input direction and actual direction in space coordinates. Boresight axes are pitch and yaw. This requirement does not apply to roll about boresight axis	When commanded, the Observatory shall point the camera and sensors accurately, without using scientific instruments. This is the max error in pointing direction. This applies to boresight pitch and yaw, and not roll.
MR 268	In order to assure their capability and reliability to support all JWST mission requirements, Observatory mechanisms shall have functional redundancy such that no single failure prevents the Observatory from meeting mission requirements or be designed, manufactured, integrated and tested to the requirements of the JWST Mechanisms Control Requirements (JWST-RQMT-004058).	The Observatory shall have redundancies, so that there is no single point of failure, preventing it from meeting mission needs.	The Observatory shall have backups in case of failure.

MR 271	The Observatory shall passively cool the Near-Infrared (NIR) Science Detectors to a temperature of less than or equal to 37K beginning at a time during commissioning that supports NIRCam and NIRSpec commissioning and continuing until the end of the science mission lifetime.	The Observatory shall passively cool the Near-Infrared Science Detectors to 37K from commission supporting NIRCam and NIRSepc till the end of the mission.	The Observatory shall passively cool the detectors throughout the life cycle.

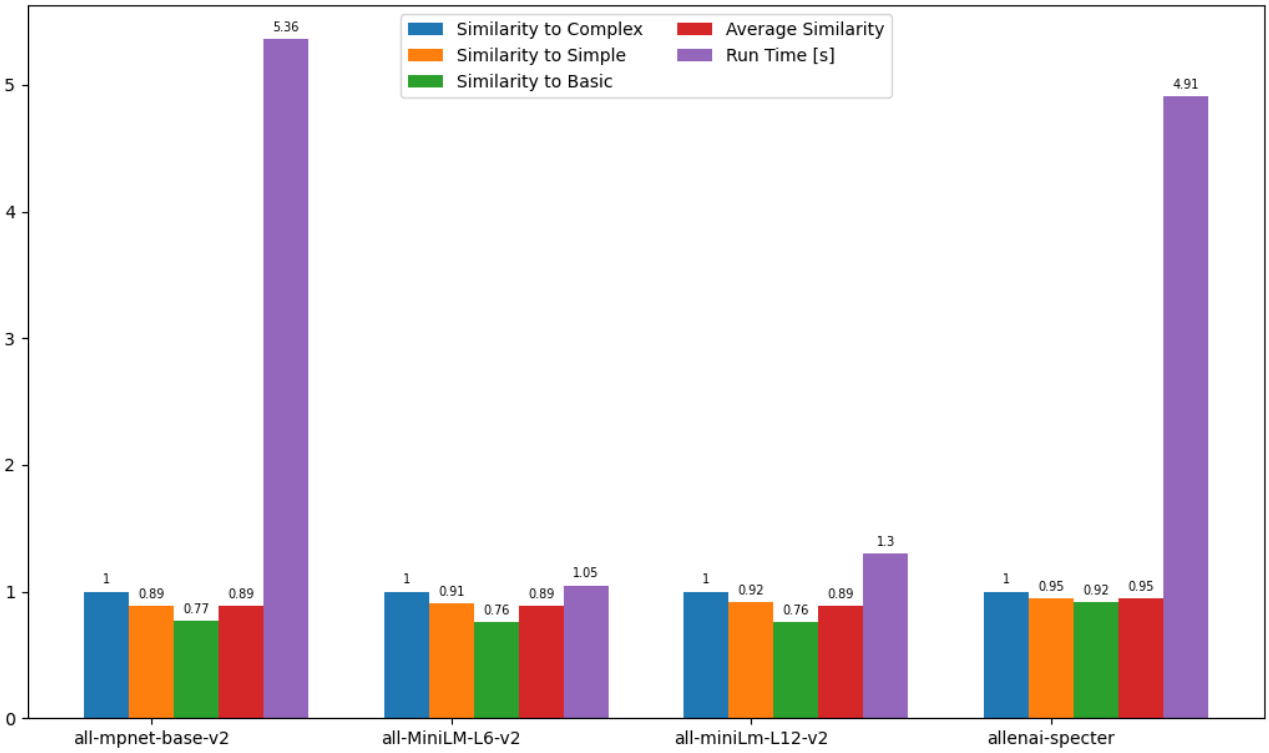

Figure 16: Sentence-BERT Pre-trained Model Comparison Results for JWST Requirement MR 172

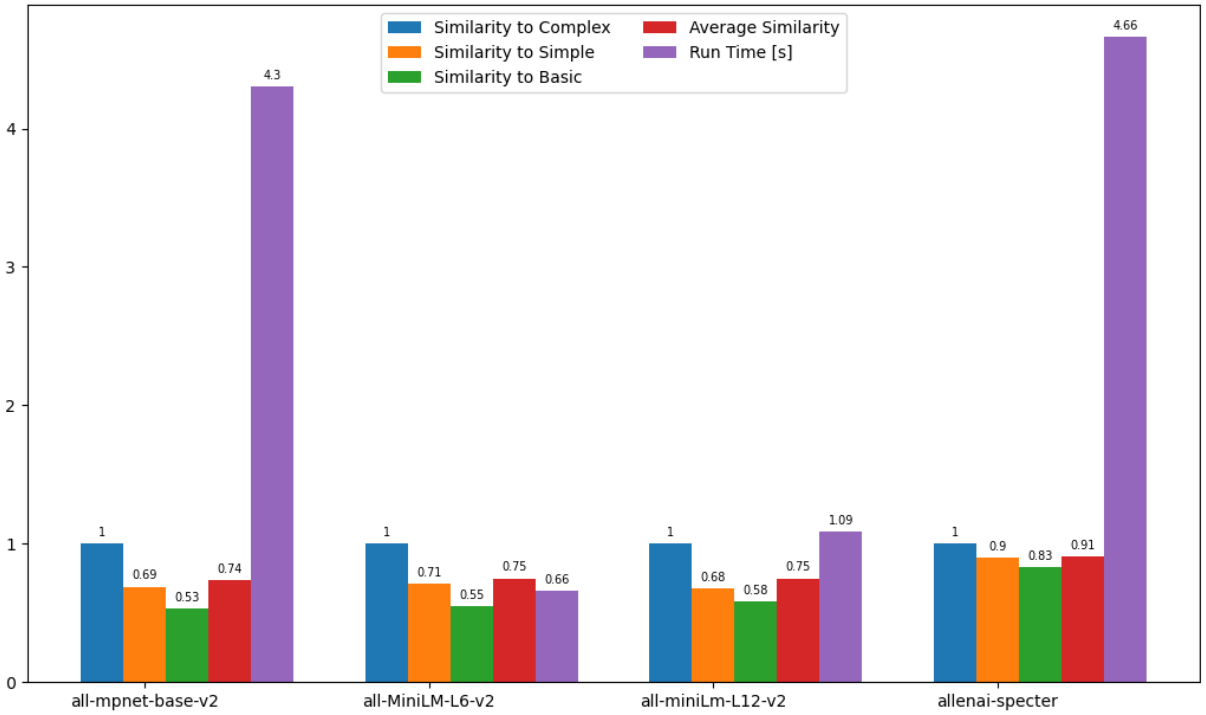

Figure 17: Sentence-BERT Pre-trained Model Comparison Results for JWST Requirement MR 268

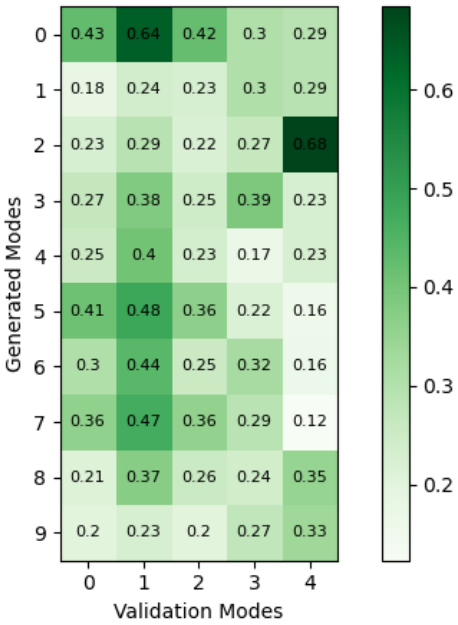

Figure 18: Sentence-wise Similarity Scores for PROVE Modes. Sentence IDs are described by Table 8 in the Appendix.

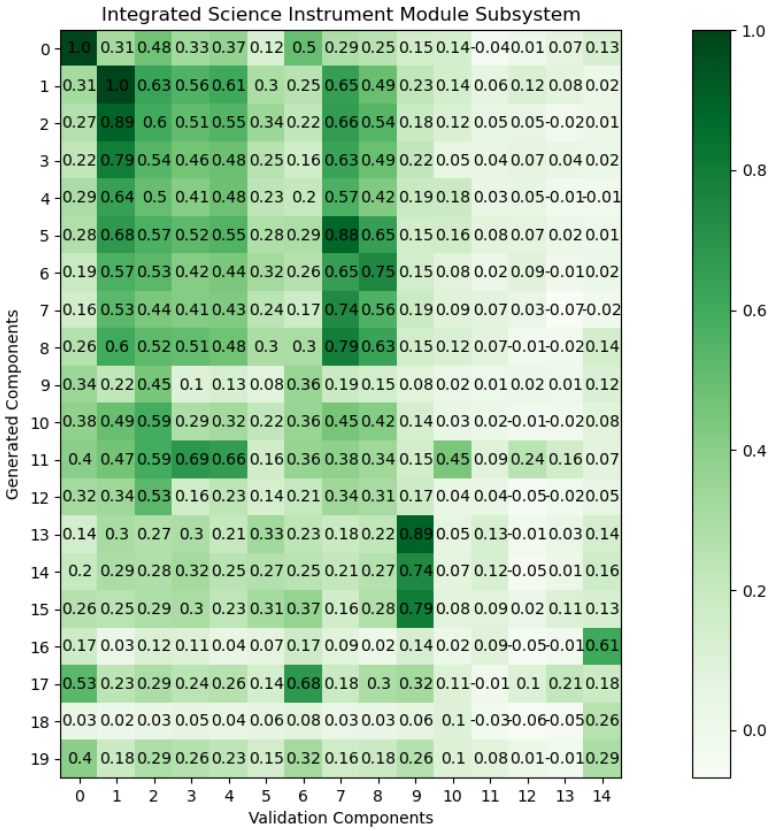

Figure 19: Sentence-wise Similarity Scores for JWST ISIM Subsystem. Sentence IDs are described by Table 12 in the Appendix.

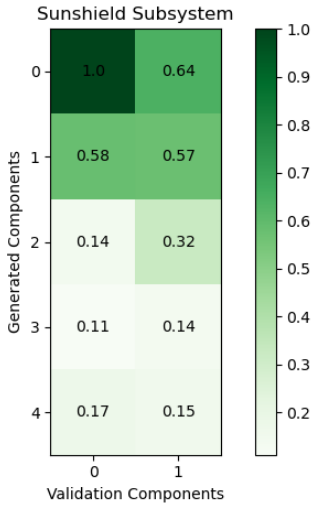

Figure 20: Sentence-wise Similarity Scores for JWST Sunshield Subsystem. Sentence IDs are described by Table 13 in the Appendix.

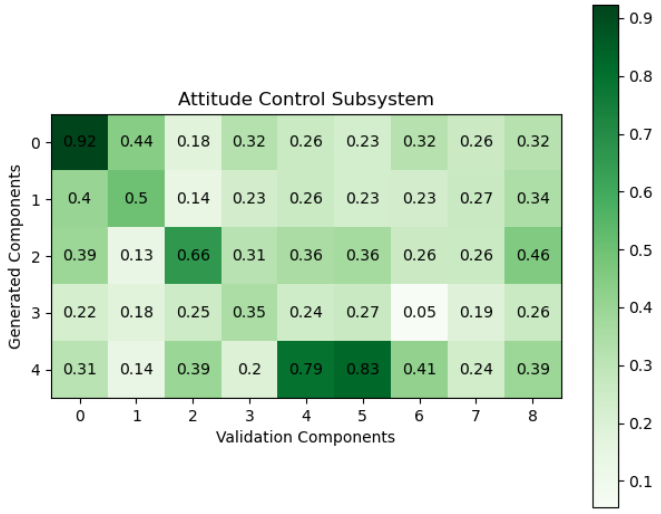

Figure 21: Sentence-wise Similarity Scores for JWST ACS Subsystem. Sentence IDs are described by Table 14 in the Appendix.

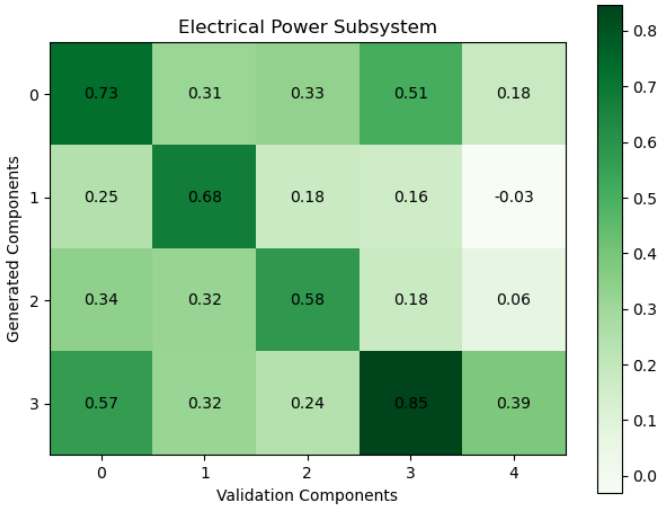

Figure 22: Sentence-wise Similarity Scores for JWST EPS Subsystem. Sentence IDs are described by Table 15 in the Appendix.

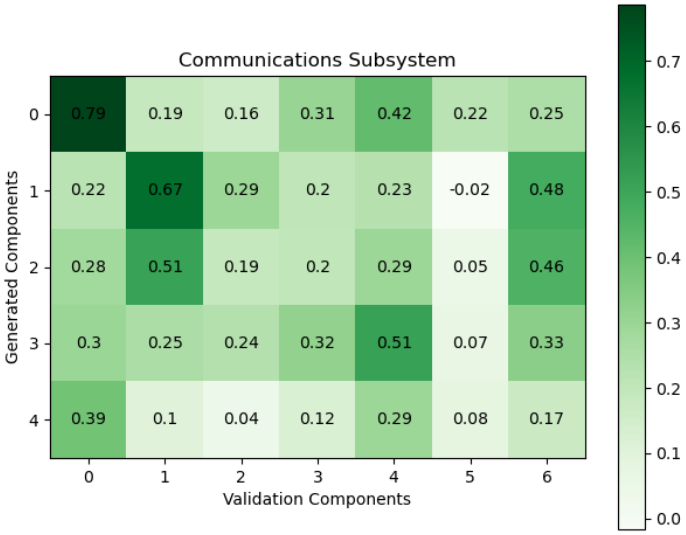

Figure 23: Sentence-wise Similarity Scores for JWST Comms Subsystem. Sentence IDs are described by Table 16Table 10 in the Appendix.

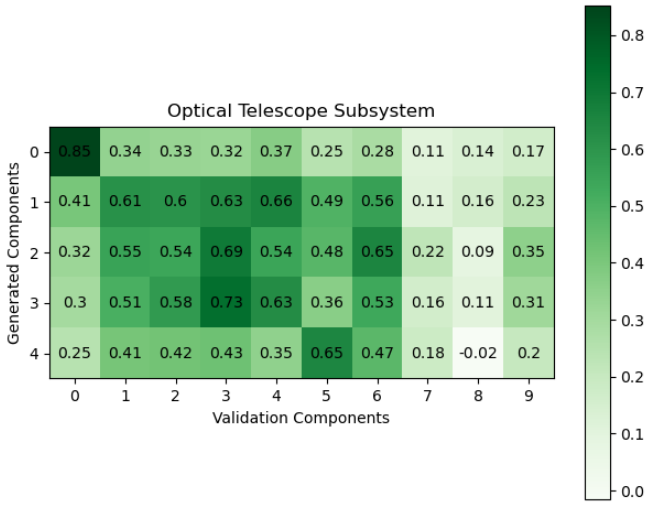

Figure 24: Sentence-wise Similarity Scores for JWST OTE Subsystem. Sentence IDs are described by Table 17 in the Appendix.

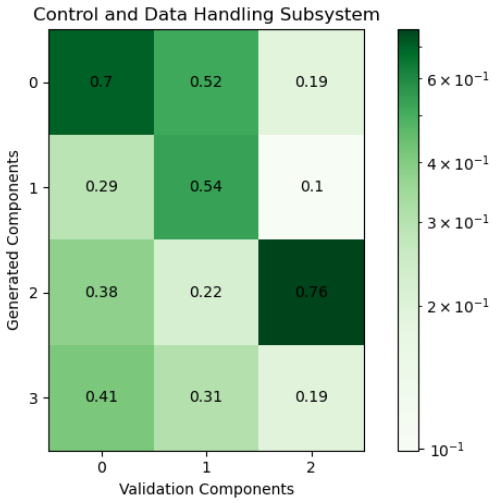

Figure 25: Sentence-wise Similarity Scores for JWST CDH Subsystem. Sentence IDs are described by Table 18 in the Appendix.

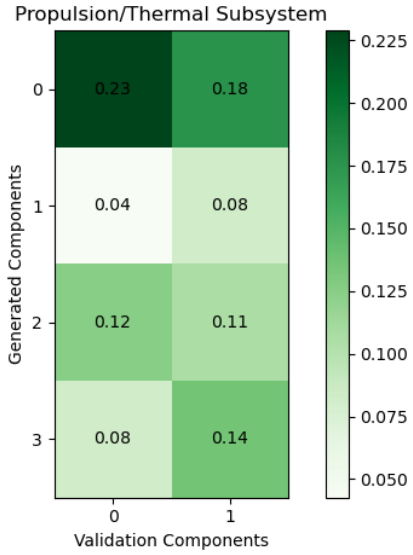

Figure 26: Sentence-wise Similarity Scores for JWST Generated Thermal and Validation Propulsion Subsystem. Sentence IDs are described by Table 19 in the Appendix.

Table 7: Sentence IDs for PROVE Functions

ID	**Generated Function**	**Validation Function**
0	- Provide IR and VIS images of a given primary target during a single primary observation pass	- Power Cycle Tau (Infrared Camera)
1	- Provide IR images of given secondary targets during a single secondary observation pass	- Initialise infrared camera interface and configure camera for use
2	- Provide multiple VIS images of a given tertiary target during a single tertiary observation pass	- Capture images from infrared camera

3	- Distribute images over the observation window to provide a range of viewing angles	- Power Visual Camera
4	- Perform observations for a threshold lifetime of 1 month with a goal of 12 months	- Initialise visual camera interface and configure camera for use
5	- Automate observations based on a predetermined schedule with the ability to uplink priority scenarios	- Capture images from visual camera
6	- Maintain a minimum elevation angle of 30 degrees for target observations	- Collect telemetry from current sensors and temperature sensors.
7	- Provide narrow-angle IR and VIS images for science observations	Enter safe mode
8	- Maintain an alignment of imager field of views within 12 degrees	- Write payload image data to SD card
9	- Provide IR images with a spatial sampling distance (SSD) at nadir of 300m (threshold), with a goal of 90m	- Write telemetry data to SD card
10	- Provide narrow-angle VIS images with a SSD at nadir of 50m (threshold), with a goal of 15m	
11	- Provide a minimum of 10 science images per primary or tertiary observation pass, with a goal of 50	
12	- Provide a minimum of 1 science image per target per secondary observation pass	
13	- Achieve a pointing accuracy of 1 degree	
14	- Downlink image data within a threshold of 1 month, with a goal of 7 days	
15	- Size onboard storage for a threshold of one primary target image set, with a goal of 10 sets	
16	- Transfer data to the bus for downlink or additional storage	
17	- Maintain total payload mass under 2.66kg	
18	- Ensure total payload power usage does not exceed 14W	
19	- Ensure total payload volume does not exceed 2U	
20	- Provide imaging, standby, safe modes, and other TBD non-nominal modes	
21	- Qualify mission hardware to levels specified in GSFC-STD-7000B	

Table 8: Sentence IDs for PROVE Modes

ID	**Generated Modes**	**Validation Modes**
0	- Imaging Mode: The system shall enter Imaging Mode during primary, secondary, and tertiary observation passes to capture IR and VIS images of targets as per MIS-001, MIS-002, and MIS-003.	- Pre-imaging mode
1	- Standby Mode: The system shall enter Standby Mode when not performing observations or data downlink, performing necessary housekeeping tasks as per CDH-OPS-04.	- Imaging mode

2	- Safe Mode: The system shall enter Safe Mode if faults are detected that cannot be automatically resolved, ensuring minimal system functionality to protect against further damage as per CDH-OPS-03.	- Post-imaging mode
3	- Data Downlink Mode: The system shall enter Data Downlink Mode to transfer image data and health telemetry to ground stations within the specified timeframes as per MIS-025.	- Transfer to bus mode
4	- Calibration Mode: The system shall enter Calibration Mode to perform radiometric calibration of the IR camera before target capture begins as per MIS-021.	- Safe mode
5	- Automatic Imaging Sequence Pass (AISP) Mode: The system shall perform automated observations based on a pre-determined schedule, capturing multiple images of standard targets as per CDH-OPS-01.	
6	- Manual Imaging Sequence Pass (MISP) Mode: The system shall allow for manual uplink of priority observation scenarios, overriding the automated observation schedule as per CDH-OPS-01.	
7	- Single Image Pass (SIP) Mode: The system shall have the capability to perform targeted single image captures of specific targets as needed, providing flexibility in observation operations as per CDH-OPS-01.	
8	- Health Monitoring Mode: The system shall continuously monitor the health of payload components, specifically the two cameras and computer, to ensure operational safety and efficiency as per ELE-HMS-00 and CDH-HMS-00.	
9	- Power Management Mode: The system shall manage power distribution to all components, ensuring sufficient power for each system mode and cutting power to components as necessary for safety as per ELE-POW-00, ELE-HMS-02, and ELE-HMS-03.	

Table 9: Sentence IDs for PROVE Components

ID	**Generated Components**	**Validation Components**
0	- FLIR Tau2 IR camera	- Basler Visual camera
1	- Basler acA640-100gm camera	- Tau Infrared camera
2	- Payload structure	- Enhanced Industrial On board computer
3	- Payload computer	- Power Supply
4	- Onboard storage system	- Bus On-board computer
5	- Data downlink system	
6	- Power supply system with 12V, 5V, and 3V buses	

7	- Health monitoring sensors for payload components	
8	- Electrical connections for data interfaces between payload components	
9	- Harwin Gecko connector for power and data connections to the main satellite bus	
10	- Command and Data Handling (CDH) subsystem	
11	- Temperature and power consumption monitoring system for payload components	
12	- Automatic power cut system for cameras	
13	- Position and attitude determination system for spacecraft	
14	- Radiation protection system for CDH	
15	- Single Event Upset (SEU) and Single Event Latch-up (SEL) protection system	
16	- Random vibration frequency spectrum support structure	
17	- Sine sweep and sine burst withstand structure	
18	- Launch vehicle compatibility structure	
19	- Thermal protection for payload components	

Table 10: Sentence IDs for JWST Phases

ID	**Generated Phases/Modes**	**Validation Phases/Modes**
0	- The JWST Observatory, including the Spacecraft, Optical Telescope Element (OTE), and Integrated Science Instrument Module (ISIM), is assembled and undergoes initial testing to ensure all components are functioning as per design specifications.	- Pre Launch Phase (manufacture, assembly, testing, launch-vehicle integration, and launch preparation)
1	- The Ground Segment, comprising the Science and Operations Center (S&OC), Institutional Systems, and Common Systems, is developed and tested for operational readiness to support the mission.	- Launch Phase (the payload is launched into orbit)
2	- The Launch Segment, including the Launch Vehicle, Payload Adapter, and Launch Site Services, is prepared, ensuring compatibility and readiness for the JWST Observatory's deployment into space.	- Deployment and Trajectory Correction Phase (Observatory separates, deploy sunshield, OTE, SMSs, antennae,)
3	- The JWST Observatory is integrated with the Launch Vehicle, followed by a series of pre-launch checks and rehearsals to confirm all systems are go for launch.	- Cruise and Commissioning Phase (Align OTE, initialise and calibrate instruments, insert into L2 orbit, target Acquisition,)
4	- The JWST is launched, and the Launch Segment executes the mission to place the Observatory on a trajectory towards its operational orbit around the Second Lagrange Point (L2).	- Normal Science Operations Phase (perform science operations, transmit data to ground for processing)
5	- Following launch, the JWST Observatory separates from the Launch Vehicle, and the operational orbit transfer is initiated to	- Decommissioning Phase

	navigate the Observatory to its designated orbit around L2.	
6	- The commissioning phase begins, involving the activation and calibration of the JWST Observatory systems, including the Spacecraft, OTE, and ISIM, ensuring all elements perform according to mission requirements.	
7	- The Ground Segment establishes and maintains continuous two-way communication with the JWST Observatory, facilitating real-time data exchange and command operations.	
8	- The Observatory performs Guide Star Acquisition and enters fine guidance mode, stabilizing the platform for scientific observations.	
9	- The science mission officially commences, with the JWST Observatory conducting observations and collecting data, while the Ground Segment ensures efficient data processing, archiving, and distribution to the scientific community.	
10	- Routine operations are conducted, including monitoring Observatory health, performing necessary adjustments or corrections, and planning observation schedules to maximize scientific return.	
11	- The JWST Observatory executes Event Driven Operations, systematically conducting scientific observations, data collection, and transmission based on the planned mission timeline.	
12	- Anomaly responses and emergency procedures are implemented as needed, ensuring the safety and operational integrity of the JWST Observatory throughout the mission.	
13	- The mission enters its final phase, with a comprehensive review of the scientific data collected, followed by a formal mission conclusion process, including the deactivation of the JWST Observatory systems and securing the Ground Segment infrastructure.	

Table 11: Sentence IDs for JWST System Architectures

ID	**Generated Components**	**Validation Components**
0	- Attitude Control System (ACS)	- Attitude Control Subsystem (ACS)
1	- Reaction wheels for fine pointing adjustments	- Reaction Wheels and Wheel Drive Electronics
2	- Star trackers for absolute attitude determination	- Star trackers
3	- Gyroscopes for rate sensing	- Inertial Reference Unit

4	- Sun sensors for coarse sun pointing	- Fine Sun Sensors
5	- Command and Data Handling System	- Coarse Sun Sensors
6	- Flight computers for processing and executing commands	- Solar Array Actuators
7	- Solid-state recorders for data storage	- High Gain antenna Actuators
8	- High-speed data buses for internal communication	- Orientation Drive Electronics with Resolver
9	- Electrical Power System	- Command and Data Handling Subsystem (CDH)
10	- Solar arrays for power generation	- Command Telemetry Processor
11	- Batteries for energy storage	- Solid State Recorder
12	- Power distribution units	- Electrical Power Subsystem
13	- Thermal Control System	- Solar Arrays
14	- Multi-layer insulation blankets	- Batteries
15	- Radiators for passive cooling	- Power Control and distribution units
16	- Heaters for temperature regulation	- Telemetry Acquisition Units
17	- Communication System	- Communication Subsystem
18	- High-gain antenna for communication with Earth	- High Gain antenna
19	- Low-gain antennas for emergency communication	- Ka-band Travelling-wave tube amplifier
20	- Transponders for data transmission and reception	- Ka-band modulator
21	- Deep Space Network compatibility for ground communication	- S-band Transponders
22	- Optical Telescope Element (OTE)	- Switches
23	- Primary Mirror: Segmented primary mirror assembly	- Omni antenna
24	- Primary Mirror: Actuators for mirror segment alignment	- Optical Telescope Element
25	- Secondary Mirror Deployable secondary mirror	- Primary Mirror and Hexapod
26	- Fine steering mirror for image stabilization	- Secondary Mirror and Hexapod
27	- Sunshield	- Secondary Mirror Structure Deployment Actuator
28	- Deployment mechanisms for sunshield layers	- Tertiary Mirror
29	- Wavefront Sensing and Control	- Fine Steering Mirror
30	- Sensors for wavefront error measurement	- Mirror Actuators
31	- Algorithms for mirror shape correction	- Tower Actuators
32	- Integrated Science Instrument Module (ISIM)	- Cold Junction Box
33	- Near Infrared Camera (NIRCam)	- Wing Deployment Actuators and Latch Actuators
34	- Near Infrared Camera (NIRCam): Detectors for near-infrared observations	- Sunshield
35	- Near Infrared Camera (NIRCam): Filters for wavelength selection	- Sunshield Actuators
36	- Near Infrared Camera (NIRCam): Coronagraph for exoplanet imaging	- Integrated Science Instrument Module (ISIM)
37	- Near Infrared Spectrograph (NIRSpec)	- Near Infrared Camera (NIRCam)
38	- Near Infrared Spectrograph (NIRSpec): Micro-shutter array for target selection	- MIRI: Mid Infrared Instrument

39	- Near Infrared Spectrograph (NIRSpec): Gratings for spectral dispersion	- Cryogenic Temperature Sensors for Mid-Infrared Instrument
40	- Near Infrared Spectrograph (NIRSpec): Detectors for spectroscopic data acquisition	- Cryogenic Cooling Environment for Mid-Infrared Instrument
41	- Mid-Infrared Instrument (MIRI)	- Focal Plane Electronics
42	- Mid-Infrared Instrument (MIRI): Combined camera and spectrograph for mid-infrared	- Instrument Control Electronics
43	- Mid-Infrared Instrument (MIRI): Cryocooler for detector cooling	- NIR Spec: Near Infrared Spectrograph
44	- Mid-Infrared Instrument (MIRI): Filters and coronagraph for diverse observations	- Micro-shutter Control Electronics for Near Infrared Spectrograph
45	- Fine Guidance Sensor (FGS)	- FGS: Fine Guidance System
46	- Fine Guidance Sensor (FGS): Sensors for guide star acquisition and tracking	- Compressor Assembly for Cryogenic Cooling
47	- Fine Guidance Sensor (FGS): Interface with spacecraft for pointing control	- Cooling Lines
48	- Data Handling	- Radiator Heat Switches
49	- Onboard software for instrument control	- Control Heaters
50	- Data compression algorithms for efficient storage	- Command and Data Handling
51	- Interfaces for data transfer to spacecraft bus	

Table 12: Sentence IDs for JWST ISIM

ID	**Generated Components**	**Validation Components**
0	- Integrated Science Instrument Module (ISIM)	- Integrated Science Instrument Module (ISIM)
1	- Near Infrared Camera (NIRCam)	- Near Infrared Camera (NIRCam)
2	- Near Infrared Camera (NIRCam): Detectors for near-infrared observations	- MIRI: Mid Infrared Instrument
3	- Near Infrared Camera (NIRCam): Filters for wavelength selection	- Cryogenic Temperature Sensors for Mid-Infrared Instrument
4	- Near Infrared Camera (NIRCam): Coronagraph for exoplanet imaging	- Cryogenic Cooling Environment for Mid-Infrared Instrument
5	- Near Infrared Spectrograph (NIRSpec)	- Focal Plane Electronics
6	- Near Infrared Spectrograph (NIRSpec): Micro-shutter array for target selection	- Instrument Control Electronics
7	- Near Infrared Spectrograph (NIRSpec): Gratings for spectral dispersion	- NIR Spec: Near Infrared Spectrograph
8	- Near Infrared Spectrograph (NIRSpec): Detectors for spectroscopic data acquisition	- Micro-shutter Control Electronics for Near Infrared Spectrograph
9	- Mid-Infrared Instrument (MIRI)	- FGS: Fine Guidance System
10	- Mid-Infrared Instrument (MIRI): Combined camera and spectrograph for mid-infrared	- Compressor Assembly for Cryogenic Cooling
11	- Mid-Infrared Instrument (MIRI): Cryocooler for detector cooling	- Cooling Lines
12	- Mid-Infrared Instrument (MIRI): Filters and coronagraph for diverse observations	- Radiator Heat Switches
13	- Fine Guidance Sensor (FGS)	- Control Heaters
14	- Fine Guidance Sensor (FGS): Sensors for guide star acquisition and tracking	- Command and Data Handling
15	- Fine Guidance Sensor (FGS): Interface with spacecraft for pointing control	

16	- Data Handling	
17	- Onboard software for instrument control	
18	- Data compression algorithms for efficient storage	
19	- Interfaces for data transfer to spacecraft bus	

Table 13: Sentence IDs for JWST Sunshield

ID	**Generated Components**	**Validation Components**
0	- Sunshield	- Sunshield
1	- Deployment mechanisms for sunshield layers	- Sunshield Actuators
2	- Wavefront Sensing and Control	
3	- Sensors for wavefront error measurement	
4	- Algorithms for mirror shape correction	

Table 14: Sentence IDs for JWST ACS

ID	**Generated Components**	**Validation Components**
0	- Attitude Control System (ACS)	- Attitude Control Subsystem (ACS)
1	- Reaction wheels for fine pointing adjustments	- Reaction Wheels and Wheel Drive Electronics
2	- Star trackers for absolute attitude determination	- Star trackers
3	- Gyroscopes for rate sensing	- Inertial Reference Unit
4	- Sun sensors for coarse sun pointing	- Fine Sun Sensors
5		- Coarse Sun Sensors
6		- Solar Array Actuators
7		- High Gain antenna Actuators
8		- Orientation Drive Electronics with Resolver

Table 15: Sentence IDs for JWST EPS

ID	**Generated Components**	**Validation Components**
0	- Electrical Power System	- Electrical Power Subsystem
1	- Solar arrays for power generation	- Solar Arrays
2	- Batteries for energy storage	- Batteries
3	- Power distribution units	- Power Control and distribution units
4		- Telemetry Acquisition Units

Table 16: Sentence IDs for JWST Comms Subsystem

ID	**Generated Components**	**Validation Components**
0	- Communication System	- Communication Subsystem
1	- High-gain antenna for communication with Earth	- High Gain antenna
2	- Low-gain antennas for emergency communication	- Ka-band Travelling-wave tube amplifier
3	- Transponders for data transmission and reception	- Ka-band modulator
4	- Deep Space Network compatibility for ground communication	- S-band Transponders
5		- Switches

6		- Omni antenna

Table 17: Sentence IDs for JWST OTE

ID	**Generated Components**	**Validation Components**
0	- Optical Telescope Element (OTE)	- Optical Telescope Element
1	- Primary Mirror: Segmented primary mirror assembly	- Primary Mirror and Hexapod
2	- Primary Mirror: Actuators for mirror segment alignment	- Secondary Mirror and Hexapod
3	- Secondary Mirror Deployable secondary mirror	- Secondary Mirror Structure Deployment Actuator
4	- Fine steering mirror for image stabilization	- Tertiary Mirror
5	- Sunshield	- Fine Steering Mirror
6	- Deployment mechanisms for sunshield layers	- Mirror Actuators
7	- Wavefront Sensing and Control	- Tower Actuators
8	- Sensors for wavefront error measurement	- Cold Junction Box
9	- Algorithms for mirror shape correction	- Wing Deployment Actuators and Latch Actuators

Table 18: Sentence IDs for JWST CDH

ID	**Generated Components**	**Validation Components**
0	- Command and Data Handling System	- Command and Data Handling Subsystem (CDH)
1	- Flight computers for processing and executing commands	- Command Telemetry Processor
2	- Solid-state recorders for data storage	- Solid State Recorder
3	- High-speed data buses for internal communication	

Table 19: Sentence IDs for JWST Thermal/Propulsion Subsystem

ID	**Generated Components**	**Validation Components**
0	- Thermal Control System	- Propulsion Subsystem
1	- Multi-layer insulation blankets	- Thrusters
2	- Radiators for passive cooling	
3	- Heaters for temperature regulation	