LEVERAGING LANGUAGE MODELS SEMANTIC SIMILARITY CAPABILITIES TO FACILITATE INFORMATION REUSE IN SYSTEM ENGINEERING

Paul Darm[a*], Francesco Marchetti[b], Gérald Garcia[c], Paloma Maestro Redondo[d], Annalisa Riccardi [e*], Alberto González Fernández[f],

[a] Department of Mechanical & Aerospace Engineering, University of Strathclyde, 16 Richmond St, Glasgow G1 1XQ, United Kingdom, paul.darm@strath.ac.uk

[b] Department of Mechanical & Aerospace Engineering, University of Strathclyde, 16 Richmond St, Glasgow G1 1XQ, United Kingdom, francesco.marchetti@strath.ac.uk

[c] Thales Alenia Space France, 5 All. des Gabians, Cannes, 06150, France, gerald.garcia@thalesaleniaspace.com

[d] RHEA Group, Jonckerweg 18, Noordwijk, 2201 DZ, The Netherlands, pm.redondo@rheagroup.com

[e] Department of Mechanical & Aerospace Engineering, University of Strathclyde, 16 Richmond St, Glasgow G1 1XQ, United Kingdom, annalisa.riccardi@strath.ac.uk

[f] European Space Agency (ESA) - ESTEC, Keplerlaan 1, Noordwijk, 2201 AZ, The Netherlands, alberto.gonzalez.fernandez@ext.esa.int

* Corresponding Author

Abstract

Model-Based Systems Engineering (MBSE) is a powerful approach for designing complex engineering systems, which also generates valuable data after each conducted study. However, currently there are few to no approaches for reusing this information in a systematic way. In this paper, we propose using state-of-the-art Natural Language Processing (NLP) methods and a graph database to analyze data from past missions and facilitate the design process of new missions. In particular, we firstly develop techniques for analysing a database of past-mission requirements. This include the ability to identify semantic similar requirements from past missions for a given new requirement. We also fine-tune a language model in order to analyse the logical traceability between two requirements. These methods are meant to enable engineers to more efficiently define the requirement space for a new spacecraft.Secondly, we also develop methods to analyse the physical and functional architectures of past missions. Based on an input for a new design, a graph database of past-mission design can be queried for similar design choices and functionalities by again leveraging the abilities of semantic similarity and a specialised breadth-first-search algorithm. Finally, we show how both the requirement and design analyses could in order to automatically verify if the provisions of a requirements are reflected in the physical architecture. For this analysis, a language model is used to extract core concepts from a requirement. Then, in a second step, the concepts from the requirement are mapped to nodes in the graph database. For the actual verification, a relevant extract of the graph together with the requirement are then used as input for a large language model, which is prompted to reason if the requirement is fulfilled or not. By leveraging NLP and graph search techniques, we believe that these approaches can lead to more efficient and effective design processes for complex engineering systems by reusing information from past designs. The proposed techniques have been developed and tested on real past-mission requirements and design architectures in collaboration with Thales Alenia Space, RHEA group, and the European Space Agency.

1. Introduction

1.1 *Background*

Model-Based Systems Engineering (MBSE) is defined as 'The formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases' [7]. MBSE is commonly presented as one of the main contributors to the future of systems engineering to overcome the challenges and needs of the discipline [8]. Its application has shown a positive Return on Investment with respect to the traditional document-based approach, when applied to complex systems [12]. MBSE can underpin the accomplishment of collaborative work [8]. Moreover, improved consistency, increasing system understanding, productivity, analysis capability, efficiency, and automation support are mentioned as the benefits of MBSE [5]. Compared to traditional systems engineering projects, MBSE projects require greater investment during the conceptual and preliminary design stages and substantially lower investment in the latter stages. This investment in MBSE is associated with some costs for 'evaluating methodologies, infrastructure implementation/extension, personnel training, model development and verification, model curation, and configuration management' [9].

Interoperability between MBSE solutions is often mentioned as one of the main issues for MBSE adoption. This is caused by differences in the ontology, modeling language and tools used by each solution [1, 10, 14]. The

MBSE Hub is one of the solutions proposed by the European Space Agency (ESA) to overcome the interoperability issue. It consists in a centralised hub to be used by different levels and disciplines of engineering, as well as throughout different phases of a space project, and it shall allow the stakeholders to exchange information using a common vocabulary described in the Space System Ontology (SSO) [6, 15].

1.2 *Purpose*

This paper presents different analysis modes for a digital assistant to support systems engineers in projects performed with MBSE. Leveraging Natural Language Processing (NLP) techniques, we aim to identify common concepts across multiple past missions, like requirements or components, and propose suggestions to the systems engineers. This will not only speed up the mission and spacecraft definition phases, but also reduce the cost by avoiding the repetition of previous mistakes. ESA's MBSE Hub is meant to act as the common database, where information from multiple past missions can be stored and the analysis modes can be applied to.

1.3 *Scope*

In particular, we showcase three exemplary analyses modes that can be applied to past missions as defined under the SSO. This includes an analysis of mission requirements by comparing semantic similarity between requirements as well a fine-tuned language model to predict logical dependencies between requirements. Secondly, we present a filtered-breadth-first search search algorithm to efficiently search through past-mission design spaces as defined by the SSO. Thirdly, we propose a possible combination of both analyses modes, where based on a requirement we extract a relevant part of the design space from the graph database and subsequently use a Large Language Model (LLM) as a reasoning engine to verify the model. [18, 19]

The second section provides a brief introduction into the theory and applied methods. It starts by introducing key concepts of NLP and the applied graph algorithm. The next section showcases the results of the three analyses modes. In the discussion, the results are evaluated and future directions are being suggested. The conclusion section summarizes the key findings of the paper.

2. Related Work

Singh [13] describes an approach to identify the similarity between requirements using NLP, semantic analysis, and graph theory to develop inter-related requirements. This is done to evaluate the similarity between requirements coming from different stakeholders and formulated using a different vocabulary. In doing so, the presented approach is also able to aid the engineers by highlighting fault-prone regions in the considered requirements. Their approach can also be used to find redundant or extraneous requirements. Another use of semantic similarity is presented in [2], where the authors compared different multilingual embedding models in English and German. They propose an approach to leverage the similarity between multilingual embeddings, evaluated using the cosine similarity, to find redundant requirements. They apply it only to previous projects and do not use it to suggest requirements for future ones. To conclude, Zhao et al. [20] present a survey about the use of NLP for requirements engineering. Their study showed great use of NLP technologies for requirements engineering, of which a minimal part was used in an industrial setting. Moreover, their study highlighted a great discrepancy between the current stat-of-the-art and the state of practice of NLP technologies for requirements engineering. The study we propose aims to partially address this gap by being performed in direct collaboration with the industrial project partner, Thales Alenia Space.

Model verification is a challenging field in MBSE. One proposed methodology is enriching the commonly used MBSE language SysML by a model checker like NuSMV. [4]. Wang et al. explored how this could be used to verify the design and state transitions of the Flap control system of an avionic assembly. The drawback is that the system and the requirements/rules to check against need to be modeled in the language of the tool in order to be able to analyse it. [17] Salado et al. found that implementation of text based requirements in MBSE can be very restrictive on the proposed design solutions. They show that by e.g. defining a requirement as a state machine diagram, another design solution might not be accepted in the verification check, although it also upheld the provisions in the requirement. Therefore, they argue that the semantics of most modeling languages need to be extended in order to have more flexibility when defining requirements in MBSE. [16] We take a different approach, by leveraging the capabilities of NLP and reason directly over requirements in text form.

3. Methodology and Theory

3.1 *Natural Language Processing*

3.1.1 *Supervised Learning*

Given an input text sequence $X = \{x_1, x_2, \ldots, x_N\}$, an encoder language model produces a sequence of output embeddings $H \in \mathbb{R}^{N \times D}$, where each $h_i \in \mathbb{R}^D$.[3]. For fine-tuning, a task-specific classification linear layer is added, which in the case of a binary classification has the following representation:

$$y = \sigma(W \cdot h_{\text{CLS}} + b) \quad [1]$$

Where y is the predicted probability, W is the weight matrix, h_{CLS} is the embedding of the special token prepended to the sequence, b is the bias, and σ is the sigmoid function.

In the case of a binary classification, the model parameters θ consist of the hidden layers of the language model (LM) and the hidden layer from the classification head (LL). These parameters are defined as

$$\theta = \theta_{\text{LM}} \cup \theta_{\text{LL}}$$

and are fine-tuned following the standard procedure in deep learning by calculating the cross-entropy loss for the predicted and true labels and then computing the gradients with respect to θ. The model parameters are updated as:

$$\theta^{\text{new}} = \theta^{\text{old}} - \alpha \frac{\partial L}{\partial \theta^{\text{old}}}$$

The binary cross-entropy loss for the predicted and true labels is given by:

$$L(y, \hat{y}) = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y}) \quad [2]$$

Where y is the true label and $\hat{y}$ is the predicted probability.

3.1.2 *Semantic similarity*

To analyse the semantic similarity between two text sequences we follow the standard methodology of using cosine similarity as expressed in Equation 3. [11] **a** and **b** represent the vectors containing the embeddings of the two considered requirements. The cosine similarity evaluates the angle θ between the two vectors and the closer the angle is to 0 (cosine similarity score close to 1), the more similar are the two requirements.

$$cos(\theta) = \frac{\mathbf{a} \cdot \mathbf{b}}{||\mathbf{a}||\ ||\mathbf{b}||} \quad [3]$$

3.2 *Graph Theory*

A graph database can be defined as

$$G = \{E, R, C, A\},$$

where E, R, C, A represent respectively the mutually disjoint sets of entities, relations, concepts and attributes in G.

We define a function f mapping each entity to a concept.

$$f : E \to C$$

To search a graph efficiently for specific entities and concepts we apply a filtered breadth-first search (filtered BFS) algorithm.

Given a starting entity $e_s \in E$ and a set of target concepts C_{target} , the goal of the filtered BFS is to find:

$$V \subseteq E$$

such that for every $e \in V$, we have $f(e) \in C_{target}$, and there exists a path from e_s to e in G that does not pass through any other entity w with $f(w) \equiv f(e)$.

For an entity e, let Visited(e) be a function that returns true if e has been visited, and false otherwise. We then initialise the algorithm with the starting entity and the distance "0". For each entity, the distance from the starting entity is mapped by the amount of entities that is passed in the search.

The Filtered BFS-algorithm can then be described by the iterative process:

Algorithm 1 Filtered BFS Algorithm

procedure FILTEREDBFS($G, s, \mathcal{C}_{\text{target}}$)
  results $\leftarrow \emptyset$
  $Q \leftarrow$ an empty queue
  $C_{\text{found}} \leftarrow \emptyset$
  Visited $\leftarrow \emptyset$
  CutOffDistance $\leftarrow$ map from C to integers, initialized with ∞ for all concepts.
  Add $(v_s, 0)$ to Q
  while Q is not empty **do**
    Pop a vertex-distance pair (v, d) from Q
    for each neighbor u of v that's not in Visited **do**
      Add u to Visited
      if $C_{\text{found}} \neq C_{\text{target}}$ **then**
        Push $(u, d + 1)$ onto Q
      if $f(v) \in C_{\text{target}}$ and $f(v) \notin C_{\text{found}}$ **then**
        Add $C(v)$ to C_{found}
        CutOffDistance$[C(v)] \leftarrow d + 1$
      if $f(v) \in C_{\text{target}}$ and $d =$ CutOffDistance$[f(v)]$ **then**
        Add v to results
  return results

The set V is therefore found by the following expression:

$$V = \bigcup_{t \in T_{\text{found}}} \{v \in E : f(v) = t \wedge d(v) = \text{CutOffDistance}[t]\}$$

4. Results

4.1 *Requirement analyses*

The requirement analyses included two separate analyses modes. The first one focused on detecting similar requirements from past missions with semantic similarity. The second one fine-tuned a language model to identify logical dependency ("traceability") between requirements.

4.1.1 *Past-mission requirement similarity analysis*

The workflow for comparing a new requirement to requirements from previous projects is the following. First, acronyms of typical concepts in spacecraft engineering are expended in the requirement's texts. Then, embeddings for each past-mission requirement text as well as the input requirement are computed by using the *allmpnet-base-v2* SentenceBERT-model * [11]. Subsequently, the input requirement is compared to all past-mission requirements using the cosine similarity metric, as described in 3.1.

Some qualitative results from this comparison can be seen in Table 1. Overall, the general-domain language model is able to estimate at an acceptable level, the similarity between two requirements. This could be further improved by domain-adaption, so the embeddings can

*https://huggingface.co/sentence-transformers/all-mpnet-base-v2

capture better the most important features for the domain when comparing requirements or by applying a re-ranking classifier afterward.

4.1.2 *Requirement traceability analysis*

For analysis if two requirements can be traced to each other a language model encoder had to be fine-tuned as described in Sectioon 3.1. An example of a logical dependency is the following:

1. Req 1: The Data-Processing Unit (DPU) shall allow on-board lossy compression of cloud contaminated data.

2. Req 2: It shall be possible to implement on-board real-time data reduction (e.g., cloud-contaminated data) in conjunction with compression.

In this example, the first requirement gives more details about how the provisions of the second one are fulfilled. For fine-tuning the data set described in Appendix B was used. The original data set only had positive examples, which meant that negative examples needed to be created. The requirement traceability data set included 332 positive traces between requirements, see Appendix B. For creating negative examples and balancing out the original data set the following simple algorithm was applied:

- For each positive pair of requirements in the data set, e.g. (req_1, req_2), retrieve the specification level of $req1$ and $req2$
- For $req1$ randomly select a requirement, say req_{r1}, with the same specification level.
- For req_2, randomly select a requirement, say req_{r2}, with the same specification level.
- Add the pairs (req_1, req_{r1}) and (req_{r2}, req_2) as negative examples to the data set.

This algorithm is based on the heuristic that only requirements from different specification levels can be traced with each other. Therefore, for creating negative examples, only requirements from the same specification as the positive example ones were picked. This balanced data set ensures the model genuinely compares both requirements before deciding if they're linked, rather than relying on shortcuts. In an unbalanced data set, the model might predict a trace as positive based on frequency, rather than the requirements semantic meanings.

For pre-processing, the data set was first shuffled and then split into training and validation with a ratio of 0.8. Then a BERT-like domain-specific LM was trained until the validation loss did not reduce any further. The results of the validation set can be seen in Table 2.

The scores are in general strong for both labels. With a perfect precision score (1) for the traced examples, this means that every example predicted as traced was indeed a traced example from the data set. On the other hand, the recall of the traced examples is still relatively high at 0.8,

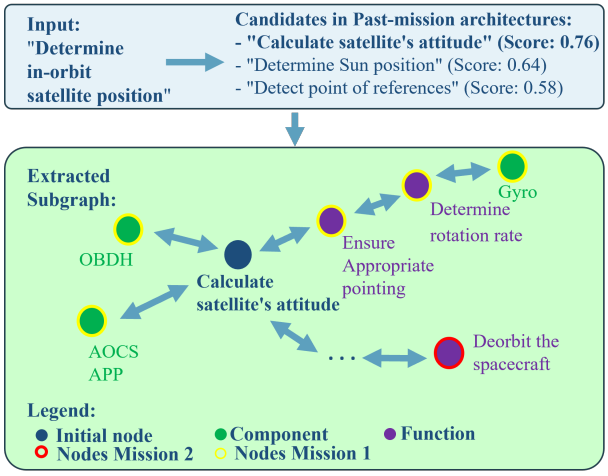

Fig. 1: Design analyses process. Two-step process of identifying and extracting relevant information based on past-mission designs

which means that of all traced examples in the data set 0.8 were correctly predicted. Overall, this configuration of trading off some recall for precision is to be preferred as it ensures that every predicted trace is a trace and does not introduce noise by incorrectly predicting traces. However, the prediction could be further improved for example by using more training samples or generating more ambiguous negative samples by using another heuristic that samples negative requirements based on a high semantic similarity between the samples.

4.2 *Design analysis*

This analysis focuses on expanding the architectural design of a new mission by incorporating components, functions, and physical parts derived from past missions. Our approach primarily relies on the integration of semantic similarity and the Filtered-BFS graph traversal algorithm described in Section 3.2.

The process is visualised in Figure 1. It starts by encoding the given textual input, which could either be the name of a component or a description of a functionality, using again the sentence embedding model *all-mpnet-base-v2*. Then, we compare this and identify the most entity in our database with the most semantically similar name in graph database of past missions. With this entity identified, we deploy the Filtered-BFS algorithm to traverse the graph, initiating from this entity. The traversal continues until we find a node representing each concept of interest, such as "PhysicalComponent" or "PhysicalArchitecture" for one specific or multiple missions in the database. Additionally, all nodes traversed during the search are also captured and returned.

Through this process, we can efficiently extract relevant information from the graph database based on the input. By providing what elements are related to a functionality / components from past-missions, or vice-versa, the engineer is assisted in defining or even enhancing of the architectural design for a new mission, thus saving time

Table 1: Similarities of examples input requirements paired with Past-Mission (PM) requirements

Input requirement	PM requirement	Similarity
When there is no ground station visibility, the RF transmission shall be disabled	It shall be possible to disable the RF radiation from the communication transmitters when not in view of the respective ground station in order to comply with the ITU regulations.	0.81
No single failure shall lead to the lost of the mission.	No single failure shall lead to catastrophic or critical consequences.	0.65
The absolute pointing accuracy shall be better than 0.01 deg	The pointing accuracy shall include all pointing error contributors (tolerances, backlash...) and statistically independent contributions may be added in a quadratic way (root mean square).	0.72

Table 2: Precision, Recall and F1-scores for Traceability Classification

Labels	Precision	Recall	F1-score
Not Traced	0.9	1.0	0.95
Traced	1.0	0.8	0.89
average	0.95	0.9	0.92

and reducing errors.

4.3 *Requirements verification check*

In this third analysis, we aim to automatically verify if a model design, as presented in the previous chapter, meets its specified requirements. We extract relevant entities and relations based on the requirement from a mission design and then apply an LLM as a reasoning engine to verify if the mission design upholds the implications of the requirement. To verify the following requirement "The DHS shall autonomously collect all science data and store it for 4 days.", one would need to check if a "Data-Handling Subsysten (DHS)" entity exists in the graph mission design model. Then, depending on the underlying ontology, the information if science data is collected and stored for 4 days might be encoded in diverse ways. For instance, "science_data" could be another entity with the attribute "stored for 4 days" connected via a relation "collect" to the "DHS". The information for storing science data could also be encoded just as an attribute "stores_science_data" to the "DHS" A rule-based system, relying on parsing the requirement into a machine readable form, might be susceptible to variations in requirement wording, the ontology structure, and the preference of the engineer defining the system. An LLM could bridge this gap by reasoning more robustly if a design either implicitly or explicitly contains the information described in a requirement. [18, 19]

To extract information from the mission design for the language model's analysis, first, concepts are extracted from the requirements, that could correspond to entities in the mission design graph, with an open-source concept recognition model [†]. Subsequently, these concepts undergo analysis to determine their semantic similarity with components present in the mission design graph. Once a component with similar attributes is identified within the graph, a graph search algorithm is employed. This search focuses on extracting neighboring components, as well as the relations that link them to the root entity and, when applicable, the attributes associated with each. For efficient representation, the N-triples format was selected, which is a common format to encode graph information.

This information is then used as input together with an instruction prompt to the language model. The prompt can be seen in Appendix A. The overall process is outlined in Figure 2 and one qualitative result can be seen in Table 3. In this example, the mass of the spacecraft is investigated and according to the requirement should be below 1000 kg. The difficulty is that the dry mass, as well as the wet mass, were extracted from the graph database, which means the model would need to decide which mass value to consider for this requirement. As seen in Table 3, the "out-of-the-box" reasoning results of GPT-4[‡], the largest model available at the moment, are quite strong. The results from an open-source model (Vicuna-13b)[§] and GPT-3.5 (ChatGPT)[¶] on the other hand, while lexical correct, show some flaws. In the example of Vicuna-13b, the model identifies correctly that the mass of the spacecraft should be below 1000kg, however, it considers the dry mass and not the wet mass for its evaluation. As the type of the mass is not specified in the requirement, the model should pick the larger quantity of the two, which is also the one that the spacecraft will have at launch and therefore the determining one to check if the requirement is upheld or not. On the other hand, in the response of GPT-3.5 both masses are mentioned correctly. However, the reasoning in this case has a logical flaw in that it determines "946 kg" to be above 1000 kg. So while the final answer is correct the reasoning is flawed. GPT-4 correctly understands that the dry mass of the spacecraft is below the required 1000 kg. Additionally, it further explains that the wet mass in contrast exceeds the given limit. It further

[†] https://huggingface.co/icelab/spacescibert_CR
[‡] https://openai.com/research/gpt-4
[§] https://huggingface.co/lmsys/vicuna-13b-v1.5
[¶] https://openai.com/blog/chatgpt

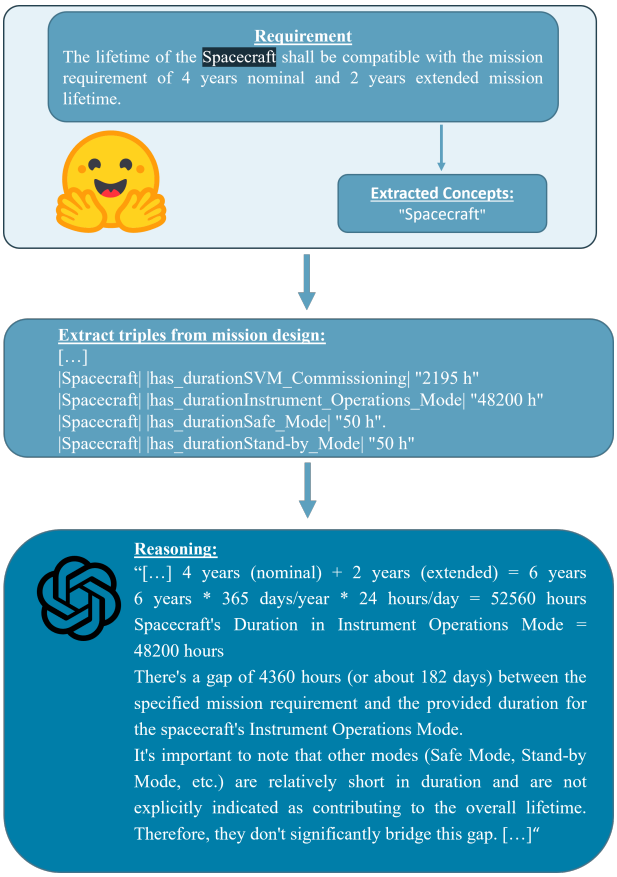

Fig. 2: Automatic Requirement Verification process. Three step process of extracting concepts of requirement, mapping it to information in graph database, and LLM reasoning example.

explains what "wet mass" means in the context of spacecraft engineering, and articulates that usually the wet mass is the deciding factor when specifying the constraints on a spacecraft. This response is very close to the desired output and correctly identifies the requirement as not fulfilled. More example results can be seen in Appendix C.

5. Discussion

The results highlight the potential of our analysis methods for assisting in the early stage design process of spacecraft. For identifying similar requirements, the results show that "off-the-shelf" sentence embeddings are able to generalise to requirements from spacecraft engineering. Together with the traceability classifier, these two methods enable sophisticated analysis modes e.g. identifying a similar requirement together with all its dependent lower-level requirements, thus enabling a detailed analysis of the requirement space for a new mission. These methods could be further extended in the future e.g. by a larger data set to train the traceability classifier.

The results for the second analysis showcase again the out-of-domain capabilities of modern sentence embeddings, but also of a graph database to explore past-mission design. However, these analyses could be more extensively verified by conducting test studies where system engineers are prompted to design a specific part with and without access to this tool, compare the final designs, and evaluate user experience in terms of time saved, and usefulness of the recommendations provided.

Thirdly, the automatic requirement verification analysis showcases how LLMs could be employed in the future to assist enginers in the verification process. While the results could benefit from a larger quantitative study, the proof of principle is presented and its effectiveness is demonstrated.

All the analysis methods here proposed aim to make the spacecraft design process more efficient and reduce the likelihood of errors, by automating some of the tasks of an engineer, such as requirement definition, past-mission analysis and requirements verification. This approach envisions a synergistic collaboration where an automated assistant partners with an engineer. However, a challenge could be a limited adoption of Model-Based Systems Engineering (MBSE) in the industry, leading to a scarcity of data, which e.g. is crucial for re-proposing past mission design solutions and on which our solution is heavily based on. Additionally, there's concern about the Technology Readiness Level (TRL) of using Large Language Models (LLMs) for reasoning tasks. Assessing the trustworthiness and resilience of these applications is still an ongoing active research field.

6. Conclusion

In this study, we presented three different analysis modes for exploring past-mission design and requirements by leveraging NLP techniques. It was demonstrated qualitatively that publicly available sentence embedding models can capture well similarity in the field of space system engineering. Furthermore, we curated a data set and fine-tuned a language model to identify logical traceability between two requirements. The second analysis mode focused on analysing and exploring the mission design of past missions. FOr that, the mission design information was formalised as a graph database and a filtered breadth-first search algorithm was deployed to query the graph based on a starting node for multiple different entity types. In the third analysis mode, a bridge between the previous two analysis modes was created by first identifying which relevant part of the mission design graph is relevant for the requirement, and then prompting a LLM to reason if the requirement is fulfilled or not. The results show that modern NLP solutions usually can generalize well to, in this case, spacecraft engineering, with very little in-domain training data needed. The development of these methodologies can be seen as an ongoing effort to create an assistant, which helps in the design process of space missions. Future studies could build on top of the developed analysis modes or go into developing test studies to evaluate the efficiency increase when using an assistant in the spacecraft design process.

Table 3: Example reasoning of GPT-4, GPT-3.5 and Vicuna-13b over requirement and graph database extract.

Requirement: The spacecraft shall have a mass <1000 kg.		
Graph triples: *|Spacecraft||has_durationStand-by_Mode|"52 h"* *|Spacecraft ||has_durationInstrument_Operations_Comms_Mode |"4.5 h"* *|Spacecraft ||has_durationSafe_Mode |"52 h" |Spacecraft ||has_durationInstrument_Operations_Mode|"48250 h"* *|Spacecraft||has_durationSVM_Commissioning_and_Decontamination_Mode|"2215 h"* *|Spacecraft||has_durationActive_Cooling_Mode|"12 h"* *|Spacecraft||has_durationLaunch_Mode|"2 h"* *|Spacecraft||has_durationSVM_Commissioning_and_Decontamination_Peak_Power|"9 h"* *|Spacecraft||has_durationSun_Acquisition_Mode|"4 h"* *|Spacecraft ||has_dry_mass|"946 kg"* *|Spacecraft||has_wet_mass |"1085 kg"*		
GPT-4	**GPT-3.5**	**Vicuna-13b**
[...] Step 3: From the provided engineering system, we have: • Dry mass of the spacecraft: 946 kg • Wet mass of the spacecraft: 1085 kg The dry mass of the spacecraft (946 kg) is below the required limit of 1000kg. However, the wet mass (1085 kg) exceeds the given limit. In the context of spacecraft, wet mass typically includes the mass of the spacecraft itself (dry mass) plus any consumables (like fuel). For most mission-related requirements, it is the wet mass that is often more pertinent since it represents the total mass at launch or start of the mission. FINAL ANSWER: NO. The provided engineering system does not satisfy the given goal requirement.	[...] Step 3: From the given engineering system, we can see that the spacecraft's dry mass is "946 kg" and its wet mass is "1085 kg." Both of these values are above 1000 kg, indicating that the spacecraft does not meet the requirement of having a mass below 1000 kg. FINAL ANSWER: NO. The provided engineering system does not satisfy the given requirement.	Step 1: The given engineering system describes the different modes and durations that the spacecraft can operate in, including stand-by mode, instrument operations mode, and various commissioning and decontamination modes, as well as the mass of the spacecraft. Step 2:The requirement states that the spacecraft shall have a mass less than 1000 kg. Step 3: From the given engineering system, the dry mass of the spacecraft is 946 kg, which is less than 1000 kg. Therefore, the spacecraft satisfies the given requirement. FINAL ANSWER: YES

References

[1] N. Badache and P. Roques. Capella to sysml bridge: A tooled-up methodology for mbse interoperability. In *9th European Congress on Embedded Real Time Software and Systems (ERTS 2018)*, 2018.

[2] U. Bisang, J. Brünnhäußer, P. Lünnemann, L. Kirsch, and K. Lindow. Evaluate Similarity of Requirements with Multilingual Natural Language Processing. *Proceedings of the Design Society*, 2:1511–1520, may 2022. ISSN 2732-527X. doi: 10.1017/pds.2022.153. URL https://www.cambridge.org/core/product/identifier/S2732527X22001535/type/journal{_}article.

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

[4] M. Hause et al. The sysml modelling language. In *Fifteenth European Systems Engineering Conference*, volume 9, pages 1–12, 2006.

[5] K. Henderson and A. Salado. Value and benefits of model-based systems engineering (mbse): Evidence from the literature, 2021. ISSN 15206858.

[6] T. Hoppe, T. Stoitsev, and C. Borrett. Model based engineering hub – a firm foundation for a new generation of mbse exchange. In *3rd Model Based Space Systems and Software Engineering workshop (MBSE2022)*, 2022.

[7] INCOSE. *Systems Engineering Vision 2020*. 2007.

[8] INCOSE. *INCOSE Systems Engineering Vision 2035. Engineering solutions for a better world*. 2022.

[9] A. M. Madni and S. Purohit. Economic analysis of model-based systems engineering. *Systems*, 7, 2019. ISSN 20798954. doi: 10.3390/systems7010012.

[10] C. Pessa, M. Cifaldi, E. Brusa, D. Ferretto, K. M. Malgieri, and N. Viola. Integration of different mbse approaches within the design of a control maintenance system applied to the aircraft fuel system. In *2016 IEEE international symposium on systems engineering (ISSE)*, pages 1–8. IEEE, 2016.

[11] N. Reimers and I. Gurevych. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. 8 2019. URL http://arxiv.org/abs/1908.10084.

[12] E. B. Rogers and S. W. Mitchell. Mbse delivers significant return on investment in evolutionary development of complex sos. *Systems Engineering*, 24, 2021. ISSN 15206858. doi: 10.1002/sys.21592.

[13] M. Singh. Using Natural Language Processing and Graph Mining to Explore Inter- Related Requirements in Software Artefacts. *ACM SIGSOFT Software Engineering Notes*, 44(1):37–42, mar 2019. ISSN 0163-5948. doi: 10.1145/3310013.3310018. URL https://dl.acm.org/doi/10.1145/3310013.3310018.

[14] L. Van Ruijven. Ontology for systems engineering as a base for mbse. In *INCOSE International Symposium*, volume 25, pages 250–265. Wiley Online Library, 2015.

[15] A. Vorobiev, K. Tiensuu, S. Gerené, S. Jahnke, L. Bitetti, and H. P. de Koning. Model based system engineering hub. In *3rd Model Based Space Systems and Software Engineering workshop (MBSE2022)*, 2022.

[16] P. Wach and A. Salado. The need for semantic extension of sysml to model the problem space. In A. M. Madni, B. Boehm, D. Erwin, M. Moghaddam, M. Sievers, and M. Wheaton, editors, *Recent Trends and Advances in Model Based Systems Engineering*, pages 279–289, Cham, 2022. Springer International Publishing. ISBN 978-3-030-82083-1.

[17] H. Wang, D. Zhong, T. Zhao, and F. Ren. Integrating model checking with sysml in complex system safety analysis. *IEEE Access*, 7:16561–16571, 2019. doi: 10.1109/ACCESS.2019.2892745.

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

[19] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

[20] L. Zhao, W. Alhoshan, A. Ferrari, K. J. Letsholo, M. A. Ajagbe, E.-V. Chioasca, and R. T. Batista-Navarro. Natural Language Processing for Requirements Engineering. *ACM Computing Surveys*, 54 (3):1–41, apr 2022. ISSN 0360-0300. doi: 10.1145/3444689. URL http://arxiv.org/abs/2004.01099https://dl.acm.org/doi/10.1145/3444689.

Acknowledgments

A. LLM prompt

> *I want you to act as an Spacecraft Engineer. I will provide an extract of an Engineering System in triple format as well as a requirement, and it will be your job to analyze if the requirement is satisfied by the engineering system or not. First try to extract the information (entities and relations) from the System that is potentially useful in your analysis. Next try to establish if that information either implicitly or explicitly fulfills the requirement. This could involve performing a gap analysis between the engineering system and the requirement. The Final step is to clearly state if the requirement is satisfied or not by stating FINAL ANSWER: YES or FINAL ANSWER: NO. My first engineering system and requirement are:*

B. Data

2.0.1 *Trace classifier dataset*

For the requirements analyses a dataset was curated to train a classifier to predict, if the first requirement satisfies the second. This dataset was curated manually at Thales Alenia Space. For training a model, 332 unique pairs of positive traces were therefore labelled.

2.0.2 *Graph spacecraft design database*

The curated model graph database consist of combined data models, which are supposed to follow the new SSO ontology. In our case the source, for data came from ESA's concurrent design models, which are modelled to follow another data model, which is described in the ECSS-TM-10-25A standard. This data model does not describe the functional architecture of a space mission. Therefore additionally, Capella models were manually mapped to the corresponding ECSS-model to represent the physical components, what functions they fulfill, and interactions between them in the spacecraft design.

C. Additional Requirement Verification examples

Table 4: Similarities of examples input requirements paired with Past-Mission (PM) requirements

Requirement: The communication subsystem shall be able to perform the following functions throughout all the mission phases after launch: • Receive and demodulate the uplink signal from the ground segment and transmit the telecommands (TC) to the data handling system • Receive a telemetry (TM) data stream from the data handling system and transmit this data to the ground segment"		
Graph triples: *|On-Board Computer with Mass Memory ||mission data (TM)||Communication Subsystem|* *|Communication Subsystem ||TC ||On-Board Computer with Mass Memory |* *|Acquire TC ||isDirectlyPerformedByEntity ||Communication Subsystem |* *|Decode TC ||isDirectlyPerformedByEntity ||Communication Subsystem |* *|Analyse TC ||directlyBelongsToActivity ||Analyse & excecute telecommands |* *|Analyse & execute telecommands ||isDirectlyPerformedByEntity ||On-Board Computer with Mass Memory|* *|On-Board Computer with Mass Memory||isDirectlyPerformedByEntity||Generate telemetry|* *|Communication Subsystem||isDirectlyPerformedByEntity||Transmit telemetry|* *|On-Board Computer with Mass Memory||isDirectlyPerformedByEntity||estimate satellite attitude|* *|On-Board Computer with Mass Memory||isDirectlyPerformedByEntity||Monitor zone temperature|* *|On-Board Computer with Mass Memory ||isDirectlyPerformedByEntity ||Monitor and control energy sources |* *|Transmit telemetry ||mission data (TM)||Receive and use mission data|* *|Receive and use mission data |isDirectlyPerformedByEntity |Ground Segment|*		
GPT-4	**GPT-3.5**	**Vicuna-13b**
[...] The requirement indicates two main functionalities the Communication Subsystem should fulfill: 1.Receive and demodulate the uplink signal from the ground segment and transmit the telecommands (TC) to the data handling system. 2.Receive a telemetry (TM) data stream from the data handling system and transmit this data to the ground segment. Step 3: Using the provided engineering system: 1.For the first function: - The Communication Subsystem can "Acquire TC," implying it can receive the telecommands. - It can "Decode TC," which might be equivalent to demodulating the uplink signal.. - The Communication Subsystem can send "TC" to the "On-Board Computer with Mass Memory," which appears to function as a data handling system. 2. For the second function: - The "On-Board Computer with Mass Memory" can "Generate telemetry." - The Communication Subsystem can "Transmit telemetry," and this telemetry (TM) data is subsequently received by the "Ground Segment" as implied by the "Receive and use mission data" being performed by the Ground Segment. Given the aforementioned, the Communication Subsystem seems to be performing both functions specified in the requirement. FINAL ANSWER: YES. The provided engineering system satisfies the given requirement.	[...] From the system description, we can see that the communication Subsystem can perform actions related to receiving and transmitting signals, including telecommands and telemetry. The "Transmit telemetry" activity and its relationships suggest that it can transmit telemetry data to the ground segment. However, the system lacks an explicit statement regarding the Communication Subsystem's capability to "Receive and demodulate the uplink signal from the ground segment and transmit the telecommands (TC) to the data handling system." While it's implied that the Communication Subsystem can receive and decode TCs, the exact connection between the uplink signal and the data handling system is not detailed. Therefore, while the system provides some functionalities related to the requirement, there is a gap in the system's explicit description of the process of receiving and demodulating the uplink signal and transmitting TCs to the data handling system. FINAL ANSWER: NO. The provided engineering system does not fully satisfy the given requirement due to the lack of explicit details about the process of receiving and demodulating the uplink signal and transmitting telecommands to the data handling system.	Step 1: The given engineering system describes the components and their relationships in the spacecraft, including the On-Board Computer with Mass Memory (OBK-MM), Communication Subsystem, and various functions performed by these entities throughout the mission phases after launch. Step 2: The requirement specifies that the Communication Subsystem shall be able to perform the following functions throughout all the mission phases after launch: 1. Receive and demodulate the uplink signal from the ground segment and transmit the telecommands (TC) to the data handling system. 2. Receive a telemetry (TM) data stream from the data handling system and transmit this data to the ground segment. Step 3: From the given engineering system, it is evident that the Communication Subsystem is capable of performing the following functions: 1. Receive and demodulate the uplink signal from the ground segment. 2. Transmit the telecommands (TC) to the data handling system. 3. Receive a telemetry (TM) data stream from the data handling system. 4. Transmit this data to the ground segment. Therefore, the given engineering system satisfies the given requirement. FINAL ANSWER: YES. The provided engineering system satisfies the given requirement.