Proceedings of the Design Society, Volume 5: ICED25
https://doi.org/10.1017/pds.2025.10059

AI-Based Management and Generation Of Non-Functional Requirements in Vehicle Development and Integration

Mahmoud Bazzal[1,2,✉], **Adriana Lungu**[2], **Benjamin Kruse**[2], **Ruslan Bernijazov**[1] and **Roman Dumitrescu**[1]

[1]Paderborn University, Paderborn, Germany, [2]AUDI AG, Ingolstadt, Germany

✉ mahmoud.bazzal@hni.uni-paderborn.de

ABSTRACT: As modern technical systems grow in complexity, ensuring the quality of these systems during early development phases becomes more challenging. This is particularly evident in the development of modern passenger vehicles, where non-functional requirements (NFRs) play a critical role in ensuring that a vehicle operates according to specified standards and expectations, especially across different vehicle configurations and environmental conditions. The introduction of Artificial Intelligence (AI) in automotive engineering has transformed the approach to vehicle system design and development. This paper presents a pipeline for analyzing and generating NFRs for vehicle systems using generative AI-based methods. The pipeline categorizes NFRs, explores their interdependencies with vehicle configurations and environmental conditions, and addresses the completeness of NFRs in relation to specific vehicle use cases. The paper focuses on selecting appropriate NFR types for various use cases, taking into account diverse configurations and environmental factors. Examples of NFRs with varying parameters are provided for an electric vehicle under development at a leading car manufacturer, illustrating the benefit as well as the challenges of applying generative AI to automotive engineering.

KEYWORDS: Vehicle Development, Systems Engineering, Non-Functional Requirements, Machine Learning, Artifcial Intelligence

1. Introduction

With the increasing functionality of modern technical systems, especially in the automotive industry, the complexity of these systems has grown rapidly. As a result, ensuring that these systems meet their quality and performance expectations—such as safety, reliability, performance, and scalability—has become an essential focus during their development (Ardagna and Bena, 2023). This focus on non-functional aspects is evident in the development of modern vehicles (Shankar et al., 2020), following the steps already established in the software engineering discipline (Olsina et al., 2024).
Non-Functional Requirements (NFRs) are defined as those that specify criteria that can be used to judge the operation of a system, rather than specific behaviors (Board, 2024). Due to their importance, the complete, consistent, and unambiguous definition and management of NFRs becomes critical in the early stages of vehicle development. These requirements must be thoroughly described during the concept phase, when the use cases of the vehicle are being determined. The definition of these requirements is considered a criterion for the success of development project (Saroja and Haseena, 2023). Defining NFRs at this stage ensures that the system will meet the desired standards across all intended operational conditions, thus preventing the system from operating incorrectly, while risking undiscovered issues due to failures in testing (Eckhardt et al., 2016). Therefore, ensuring that the entire spectrum of NFRs is

addressed at the conceptual level can stabilize system design and guide the integration of various components into a cohesive and vehicle system that performs up to the expected standards.

1.1. Problem Description

Manually documenting and analyzing all potential NFRs for a given use case across different system configurations and operational environments is a daunting task. As the number of system configurations and external factors (e.g., environmental conditions, network connectivity, and interactions with external systems) increases, the complexity of identifying and Defining NFRs escalates. Manual methods often struggle to capture all these aspects comprehensively and accurately, leading to incomplete or ambiguous requirements. Furthermore, given the dynamic nature of modern vehicle systems, these requirements must be continuously updated and validated, adding another layer of difficulty to the process.

1.2. Solution

Recent advances in Generative AI (GenAI) - referring to AI systems that create new content by learning patterns from large datasets - offer promising solutions in terms of the processing and analyzing natural language. It can also help with generating text that includes complex interconnected information (Vaswani et al., 2023; Brown et al., 2020).
Given the nature of non-functional requirements, it is possible to use GenAI to streamline and automate the management of NFRs in complex systems. GenAI can help accelerate the process of creating, refining, and validating non-functional descriptions of use cases. GenAI tools can also support the automatic identification of missing or ambiguous NFRs, ensuring greater completeness and consistency across different configurations and operational scenarios. In this paper, we show how leveraging GenAI techniques for the management of NFRs can help during the development process of modern passenger vehicles. We propose a GenAI-driven pipeline to (i) analyze and improve existing NFRs that describe vehicle use cases early in vehicle development process, and (ii) generate missing NFRs to establish a complete description of all relevant quality metrics of a vehicle use case. More concretely, this paper aims to answer the following research questions:

- How to address issues such as the validity of NFRs under external conditions and system configurations as well as their completeness
- How to apply these approaches specifically to vehicle development, which is yet to be addressed by existing work.

Our approach automates the generation and validation of NFR descriptions for diverse system configurations and environments, improving both the quality and completeness of the NFR management process. Through a series of experiments, we demonstrate the potential of our approach in a real-world development process at a leading automotive manufacturer, providing a novel solution to an increasingly critical challenge in automotive engineering.
The remainder of this paper is structured as follows: section 2 discusses existing literature that addresses the analysis of NFRs. Section 3 discusses the main contribution of this paper, namely, NFR management using GenAI techniques. Section 4 discusses the implementation of the pipeline using various prompting techniques. Section 5 discusses the results of applying the approach presented in this paper in the development of a vehicle at a leading automobile manufacturer. Finally, Section 6 presents the conclusion and results, and possible future contributions that build on it.

2. Related work

Given the recent advances in Machine Learning and GenAI, specifically the approaches of Natural Language processing (NLP) and Large Language Models (LLMs), there have been a number of works in recent years that focus on applying these technologies to requirements engineering, including Non-Functional Requirements (Habibullah et al., 2023).
One prominent approach for the AI-enabled NFR analysis involves using machine learning models to classify and prioritize NFRs. For example, (Li and Nong, 2022) used embedding models to classify NFRs out of a given corpus of text by developing a neural network model called NFRNet. Other works

have used supervised learning (Kurtanovic´ and Maalej, 2017), semi-supervised learning (Casamayor et al., 2010), transfer learning Hey et al., (2020) approaches for the identification, classification, and clustering of NFRs (Baker et al., 2019).
Additionally, there have been a variety of attempts to standardize NFRs with approaches such as structured english (for example with the help of LLMs (Norheim and Rebentisch, 2024)) or structured syntax such as EARS (Mavin et al., 2009).
However, most of the existing work on NFRs concerns the identification and classification of NFRs, and is mainly focused on applications in the software engineering discipline (Lopez-Hernandez et al., 2021).
In this paper, all the described processing steps are performed with the use of prompting an LLM. These prompts are designed to simulate a conversation between a user and the LLM while stepping through the process of managing requirement for the system under development (Vaswani et al., 2023). This methodology has already been used in the literature, particularly for tasks such as data analysis, summarizing information, and even writing assistance (Brown et al., 2020).

3. Non-Functional Requirement Management

This section provides a concise overview of the proposed pipeline and then details each step in depth. The pipeline begins by processing and classifying the given requirements into functional and non-functional ones. It then maps the identified non-functional requirements to relevant use cases, vehicle configura-tions, and environmental conditions. Finally, it ensures that all relevant quality categories are covered by generating missing NFRs and formalizes each requirement for easier verification and validation.

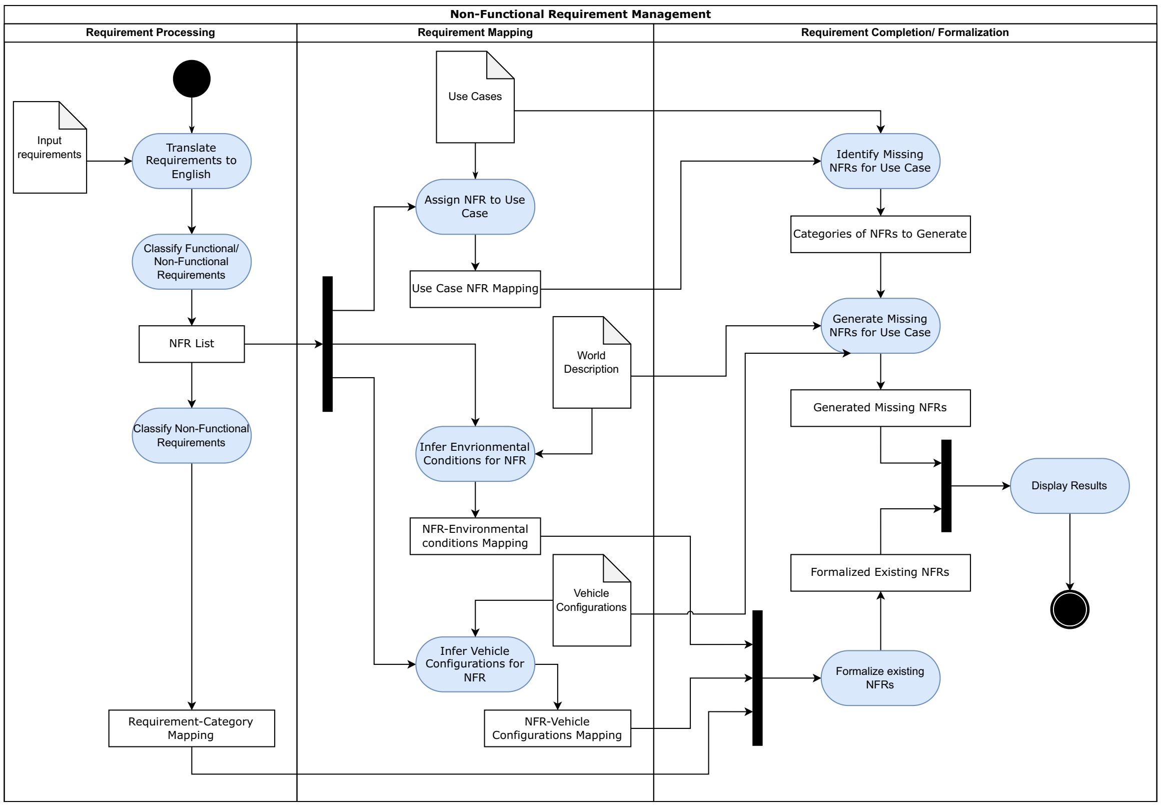

Non-Functional Requirement Management
Requirement Processing
Requirement Mapping
Requirement Completion/ Formalization
Input requirements
Translate Requirements to English
Classify Functional/ Non-Functional Requirements
NFR List
Classify Non-Functional Requirements
Requirement-Category Mapping
Use Cases
Assign NFR to Use Case
Use Case NFR Mapping
World Description
Infer Envrionmental Conditions for NFR
NFR-Environmental conditions Mapping
Vehicle Configurations
Infer Vehicle Configurations for NFR
NFR-Vehicle Configurations Mapping
Identify Missing NFRs for Use Case
Categories of NFRs to Generate
Generate Missing NFRs for Use Case
Generated Missing NFRs
Display Results
Formalized Existing NFRs
Formalize existing NFRs

Figure 1. An activity diagram describing the Non-Functional Requirement processing pipeline, including requirement processing, mapping, and completion/formalization steps

Figure 1 provides an overview of the overall processing pipeline. It takes a Use Case List, and Input Requirements as input, and is divided into three main phases: Requirement Processing, Requirement Mapping, and Requirement Completion and Formalization. Moreover, it uses Vehicle Configurations and Environmental Conditions as configuration parameters. The output is a consistently structured and

Table 1. NFR Categories considered in this paper

NFR Category		Description
Performance	(Speed)	Describes the speed or acceleration capability of the vehicle.
	(Response Time)	Quantifes the time required for an action to be executed.
	(Throughput)	Describes the amount of executions of a given action per time unit.
Effciency	(Range)	Describes the system's power usage with regard to the vehicle's range.
	(Compute)	Describes the system's computational resource usage (e.g., processing power, memory).
Reliability	(Success)	Quantifes the probability of the system producing the correct output given correct input.
Stability		Describes the system's ability to execute actions consistently for a defined duration under extreme or varying conditions.

testable set of NFRs, complemented by automatically generated NFRs to ensure the completeness of the non-functional description for each use case.
In the following we describe the input parameters of the pipeline in Section 3.1, the processing steps in Section 3.2, and provide an application example in Section 3.3 that also illustrates the final pipeline output.

3.1. Input Parameters

The pipeline's input consists of three main components. First, a Use Case List describing the intended operational scenarios of the vehicle is provided. These use cases define the context in which the requirements will be evaluated, such as "Route Calculation" or "Visualize Map." Second, Input Requirements represent initial system specifications that may include functional and non-functional aspects, potentially in various languages and formats. For instance, a given requirement might state: "Die Berechnungszeit der Route zum Ziel bis zu 1000 km mit Ladepausen im Offline-Modus muss kleiner oder gleich 0,2 Sekun-den sein." (German text. Translation: The calculation time of the route to the destination up to 1000 km with charging stops offline must be less than or equal to 0.2 seconds.)
Additionally, Vehicle Configurations detail known or assumed configurations such as battery size, vehicle class, or drive type. If explicit configurations or environmental conditions are missing, default assumptions are applied. Similarly, a set of representative environmental conditions in which the vehicle is supposed to operate are defined.
The input parameters can thus be text documents or spreadsheets containing these requirements, use cases, vehicle configurations, and environmental conditions. While formats may vary, the pipeline assumes textual input and infers context and granularity from the provided data.

3.2. Processing Steps

3.2.1. Requirement Processing

The first phase of the pipeline focuses on translating and classifying requirements. Initially, the pipeline translates all Input Requirements into English to ensure a uniform linguistic basis for further analysis. It then performs an FR-NFR classification step, separating functional requirements (which describe what the system should do) from non-functional requirements (which describe how well the system shoul perform under certain conditions.)
Once NFRs are identified, the pipeline classifies each NFR into one of several predefined categories. Table 1 shows the categories used in this paper. The Performance category covers aspects such as speed, response time, and throughput. The Efficiency category addresses aspects like range and computational resource usage. The Reliability category focuses on success probabilities, and the Stability category ensures consistent operation under varying or extreme conditions. By the end of the Requirement Processing phase, each NFR is clearly identified, assigned a category, and associated with an appropriate NFR level, forming a robust foundation for subsequent steps.

3.2.2. Requirement Mapping

After classifying the requirements, the second phase maps each NFR to a specific use case, ensuring that the identified non-functional qualities are tied to a realistic operational context. For instance,a

Table 2. Environmental conditions that could apply to given NFRs

Environmental Condition	Possible Conditions	Default Condition
Road Tilt	[−25%, +25%]	0%
Road Conditions	Off Road, Highway, Dry, Snow	Dry
Weather Conditions	Rain, Heavy Rain, Fog, Wind, Heat, Extreme Heat, Cold, Extreme Cold, Normal	Normal
Network Conditions	5G, 4G/LTE, GSM, No Service	5G
Road Infrastructure	Electronic Road Signs, Smart Traffic Lights, Highway with Construction Works	Electronic Road Signs

Table 3. Pre-defined vehicle configurations that could apply to given NFRs

Vehicle Configuration	Possible Configurations	Default Configuration
Vehicle Class	Basic, Main Class, Sport, Offroad	Main Class
Vehicle Trim	Basic Edition, Limited Edition	Basic Edition
Battery Size	80,95,110 KWh	80 KWh
Drive Profile	Normal, Eco, Performance, Performance+	Normal
Drive Type	Rear Wheel Drive, All Wheel Drive	Rear Wheel Drive
Wheel Size	18,19,21,22 inch	18 inch

performance-related NFR mentioning route calculation (either implicitly or explicitly) is mapped to the "Route Calculation" use case. This ensures that each NFR is not only defined in the abstract but it is grounded in a scenario the vehicle must handle and can be tested and validated under this scenario. In addition to linking NFRs to use cases, this phase infers environmental conditions and vehicle configurations required for testing or validating the NFR. If a requirement does not specify certain conditions—such as weather or road type—default conditions from Table 2 are applied. Similarly, if no vehicle configuration is given, a default configuration from Table 3 is chosen (e.g., "Main Class" vehicle, "Normal" drive profile). By the end of the Requirement Mapping phase, each NFR is fully contextualized with both operational scenarios and technical parameters, reflecting the complexity and variability of the vehicle's operating environment. Note that the configurations and environmental conditions used here will also be used to quantify requirements generated to describe extreme vehicle configurations and environmental conditions.

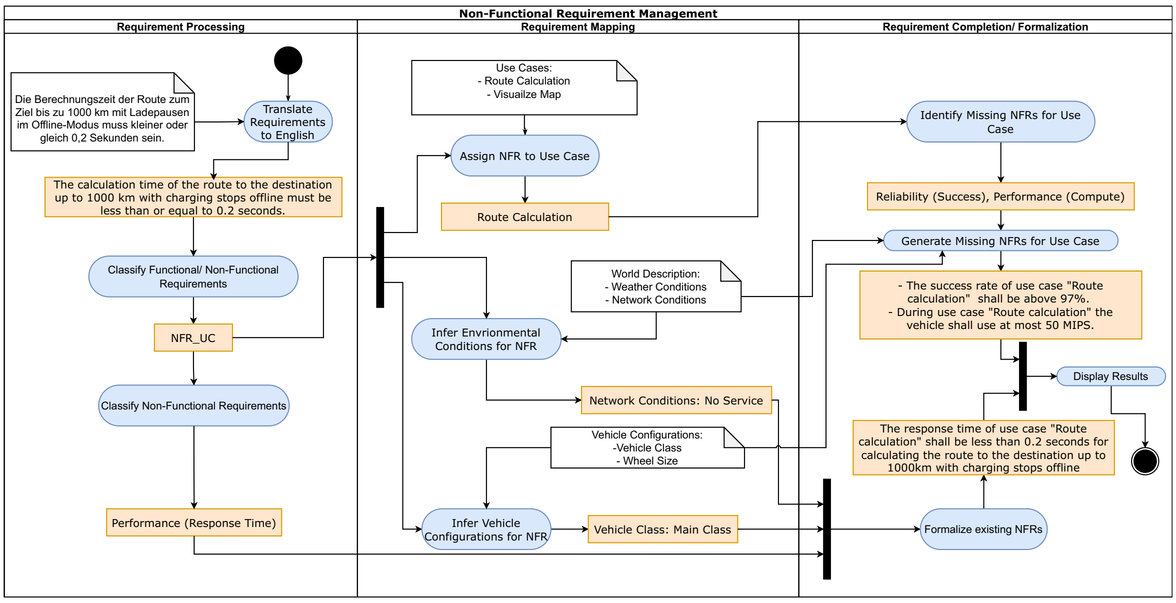
3.2.3. Requirement Completion and Formalization

In the final phase, the pipeline ensures completeness and clarity of the NFR set. It checks for missing NFR categories for each use case scenario. If some categories are absent—for example, if only performance requirements are defined but no reliability or efficiency requirements exist—the pipeline generates additional NFRs using a generative AI approach. These newly created requirements ensure that all critical quality dimensions are addressed, resulting in a comprehensive and complete requirements specification. For example, the "Generate Missing NFRs for Use Case" step in Figure 2 shows an example of two generated NFRs for the use case "route calculation," that are generated to cover the "reliability (success)" and "performance (compute)" NFR categories.

Once the complete set of NFRs is established, the pipeline formalizes all requirements using standardized templates, similar to the EARS syntax (Mavin et al., 2009). This step ensures consistency and testability, allowing for straightforward integration into verification and validation activities. Each formalized NFR includes references to its category, its assigned NFR level, and its associated environmental conditions and vehicle configurations, providing a structured and easily interpretable representation of the non-functional qualities that the vehicle must fulfill.

3.3. Application Example with Integrated Pipeline Output

In figure 2 we consider an initial requirement and a set of use cases being used as the inputs to the pipeline. Furthermore, we provide a simplified version of the world model and vehicle configurations as described in Section 3.2.2. The figure shows how the pipeline processes these given inputs and configura-tion parameters by translating the requirement to english, classifying it as a use-case related requirement, and also as a Performance (Response Time) requirement, then mapping it to the use case "Route Calculation" which is more relevant than "Visualize Map". The pipeline then infers that the requirement applies when there is no network service (text mentions offline). The default configuration for vehicle configurations is then inferred and the requirements is then formalized according to the given template. Finally, the pipeline assumes the further need for "Reliability (Success)" and "Performance (Compute)" requirements, that are then generated.



Non-Functional Requirement Management
Requirement Processing
Requirement Mapping
Requirement Completion/ Formalization
Die Berechnungszeit der Route zum Ziel bis zu 1000 km mit Ladepausen im Offline-Modus muss kleiner oder gleich 0,2 Sekunden sein.
Translate Requirements to English
The calculation time of the route to the destination up to 1000 km with charging stops offline must be less than or equal to 0.2 seconds.
Classify Functional/ Non-Functional Requirements
NFR_UC
Classify Non-Functional Requirements
Performance (Response Time)
Use Cases: - Route Calculation - Visuailze Map
Assign NFR to Use Case
Route Calculation
World Description: - Weather Conditions - Network Conditions
Infer Envrionmental Conditions for NFR
Network Conditions: No Service
Vehicle Configurations: -Vehicle Class - Wheel Size
Infer Vehicle Configurations for NFR
Vehicle Class: Main Class
Identify Missing NFRs for Use Case
Reliability (Success), Performance (Compute)
Generate Missing NFRs for Use Case
- The success rate of use case "Route calculation" shall be above 97%. - During use case "Route calculation" the vehicle shall use at most 50 MIPS.
Display Results
The response time of use case "Route calculation" shall be less than 0.2 seconds for calculating the route to the destination up to 1000km with charging stops offline
Formalize existing NFRs

Figure 2. An activity diagram illustrating an example of the pipeline being applied for a given input requirement, and a list of use cases. Note that the elements colored as yellow are the outputs of each processing steps

The Pipeline Output for this example would be a finalized set of NFRs that not only includes the original performance-oriented requirement, but also newly generated NFRs that ensure a balanced coverage of all relevant quality categories. Each resulting NFR is expressed in a standardized, testable form, similar to the EARS requirement syntax, referencing its applicable category, NFR level, environmental conditions, and vehicle configurations. This structured output can then be used in subsequent verification and validation processes, ensuring a comprehensive and context-aware quality specification for the vehicle's operational scenarios.

4. Implementation

The solution presented in this paper is implemented through a series of chained and structured prompts, orchestrated within Azure Promptflow, utilizing a company-specific deployment of the GPT-4o Large Language Model (LLM).
A prompt typically comprises three main parts: the system, the assistant, and the user. The system serves as the framework or context in which the AI operates. It defines the environment, including the rules, guidelines, and limitations that the assistant must follow. The assistant interprets the user's input and acts according to the parameters defined by the system. The user is the individual who interacts with the AI system, providing input in the form of questions, requests, or instructions.
The implementation follows a standardized prompt design, with each prompt adhering to a predefined template. The template for each prompt includes the following components:

- Input Definition: This section specifies the data or context provided to the LLM, establishing the basis for generating relevant and appropriate responses.
- Requirement Engineering Task: This component defines the specific task or objective to be performed by the LLM, guiding it in executing the desired actions based on the provided input.

- Output Definition: This describes the expected structure and content of the LLM's output, providing a clear framework for the desired result.
- Examples: For prompts using single-shot (Kojima et al., 2023) or few-shot (Brown et al., 2020) prompting techniques, examples of inputs and corresponding outputs are provided. These examples demonstrate the expected approach the LLM should follow, facilitating effective task execution.

The following listing shows the the prompt used to map a each NFR in a given list to one or multiple use cases as described in Section 3.2.2:

```
system:
[Input Definition]
    You are a requirements Engineer working in the automobile industry.
    You are given a list of non-functional requirements.
    Each non-functional requirement describes the quality, performance,
    reliability, stability, or throughput of the system in a certain
    use case context.
    You are also given a list of use cases which describe the functional
    capabilities of the vehicle system.
[Requirement Engineering Task]
    You will associate each non-functional requirement to a use case from
    the provided use case list.
[Output Definition]
    You shall produce your output as a Json array with the following format:
    [Format omitted for brevity]
    Where "requirement_text" is the requirement being considered, and
    "associated_use_case" is the english name of a use case from the use case
    list you decided to assign to this requirement.
You will always translate the use case name to english.
It is possible to associate a requirement with multiple use cases.
If you cannot find an appropriate use case for a given requirement, you will
assign it to "N/A"
You will produce the output similar to the following example:
    [Example omitted for brevity]
You shall ONLY produce a valid Json.
user:
Perform the described requirements engineering task on the following
information:
non-functional requirements: {{filtered_nfr_list}} use cases:
{{use_case_list}}
```

Given the choice that all prompts utilize the GPT-4o LLM, a key parameter influencing the behavior of the LLM is the temperature setting, which controls the randomness and creativity of the model's outputs. The temperature parameter ranges from 0 to 2. A low temperature (e.g., 0.2) generates more deterministic outputs, suitable for tasks requiring precision, consistency, and factual accuracy, such as data retrieval, summarization, or structured tasks Vaswani et al. (2023); Brown et al. (2020). On the contrary, a high temperature (e.g., 1.5) encourages greater randomness and diversity in the outputs, which is suitable for tasks that benefit from varied responses, such as ideation, content generation, or problem-solving.
Table 4 outlines the specific prompting techniques and LLM temperature settings used in the implementation.
As shown in the table, we have used the default temperature value for most prompts, which have been observed to keep the the generation of undesired tokens to a minimum. In addition, we observed that tasks depending on the standard knowledge of the LLM have performed well with zero-shot prompting without the need to provide examples. Which is in line with the findings in (Kojima et al., 2023), while prompts requiring a specific knowledge about the form of the given requirements and use cases performed better with few-shot and single-shot prompting. Note that the assignment of NFRs to use cases

Table 4. The configurations and prompting techniques of prompts implementing the pipelineshown in figure 1

Processing Step	Prompting Technique	LLM Temperature
Translate Requirements to English	Zero-Shot	1.0
Classify Functional/ Non-Functional Requirements	Zero-Shot	1.0
Classify Non-Functional Requirements	Few-Shot	1.0
Assign NFR to Use Case	Single-Shot	1.2
Infer Environmental Conditions for NFR	Single-Shot	1.0
Infer Vehicle Configurations for NFR	Single-Shot	1.0
Identify Missing NFRs for Use Case	Single-Shot	1.0
Generate Missing NFRs for Use Case	Zero-Shot	1.0
Formalize Existing NFRs	Zero-Shot	1.2

as well as NFR formalization are performed while adjusting the LLM temperature to 1.2 to allow for more diversity in the formalized NFR text, as well as ensuring that NFRs are assigned to the correct use case, even if the requirement text doesn't explicitly mention the use case name, or the use case lacks a proper description.

5. Evaluation

The evaluation of the NFR management pipeline presented in this paper was conducted by generating outputs based on a set of 442 input requirements, which were divided into two distinct sets. The first set comprised relatively simple requirements that had already been partially formalized, while the second set consisted of more sophisticated and heterogeneous requirements. These complex requirements described a comprehensive range of vehicle configurations and environmental conditions, relevant to the development of a fictional vehicle.

The inputs used for the evaluation describe use cases corresponding to four distinct features of the vehicle: navigation, sound, digital assistance (all three belonging to the first requirement set), and acceleration (belonging to the second requirement set.) These features were specifically selected to reflect the relevant aspects of the pipeline that are being validated, ensuring that the pipeline could handle a variety of requirements with varying levels of complexity.

The outputs were subjected to an evaluation process conducted by a panel of six experts in vehicle functional and system architecture, of the evaluated vehicle features at a major car manufacturer. These experts assessed the relevance, coherence, and technical soundness of the generated outputs, by either accepting or rejecting each part of the output for each NFR in their corresponding area of expertise. Their feedback focused on the alignment of the generated responses with the associated vehicle features, as well as their practical applicability to the development process. For example, the NFRs generated by the requirement formalization step have been evaluated on their conformance with an internal set of requirement templates for the respective NFR category.

Based on the experts' assessment, the accuracy of each step in the pipeline was calculated as shown in Table 5. Accuracy in this context is considered as the ratio of results accepted by the experts responsible for a given feature to the overall number of results produced by the pipeline.

The results in table 5 reveal that the NFR classification step performed better with the requirement set 2. This is because the complex set had requirements at a consistent level of granularity, aligning well with the pipeline's capabilities to process sophisticated vehicle configurations and environmental conditions. In contrast, the simple set (requirement set 1) contained many vague or imprecise requirements, such as those describing features as "comfortable" to passengers, which the pipeline struggled to handle due to their lack of clear definition.

Other steps in the pipeline performed better in the requirement set 1 because many of the included requirements resembled the provided templates, which the pipeline could more easily process. However, miscalculation of NFRs in the early stages led to inaccuracies in later steps, particularly during NFR formalization and vehicle configuration/environmental condition mapping. These misclassified requirements resulted in incorrect outputs, highlighting an area for improvement in the pipeline's initial classification process.

Table 5. Individual Prompt accuracy

Prompt	Total Results	Rejected Results	Accuracy
Requirement Set 1: Navigation, Sound, Digital Assistant			
FR-NFR Classification	328	53	84%
NFR Classification	50	4	92%
Use Case Mapping	50	0	100%
Environmental Conditions Mapping	50	1	98%
Vehicle Configurations Mapping	50	1	98%
Existing NFR Formalization	50	3	96%
New NFR Generation	40	2	95%
Requirement Set 2: Acceleration			
FR-NFR Classification	114	9	92%
NFR Classification	12	3	75%
Use Case Mapping	12	0	100%
Environmental Conditions Mapping	12	1	92%
Vehicle Configurations Mapping	12	0	100%
Existing NFR Formalization	12	6	50%
New NFR Generation	28	7	75%

Additionally, the generation of new NFRs worked better for the simpler use cases in navigation, sound, and digital assistance, as these involved fewer vehicle configurations and environmental conditions. This simplicity allowed the pipeline to more easily generate relevant NFRs. In contrast, the complex use cases introduced greater variability, challenging the pipeline's ability to produce reasonable combinations of vehicle configurations and environmental conditions for new requirements that reflect both nominal and corner cases in terms of the use case being considered.
Finally, while the pipeline automates the entire process described in this paper, preparing the inputs is still done manually. For example, describing the environmental conditions and possible vehicle configurations is currently done manually, and could be automated.

6. Conclusion and Outlook

In this paper, we presented a generative AI-based pipeline for analyzing and generating non-functional requirements (NFRs) for vehicle systems, particularly focusing on their interdependencies with vehicle configurations and environmental conditions. By automating the categorization and generation of NFRs, our approach addresses a critical challenge in modern automotive engineering, ensuring that complex vehicle systems meet quality standards across diverse use cases and operational environments. Through a series of examples, we demonstrated the practical benefits of applying this pipeline to an vehicle under development, illustrating both its potential and the challenges of adapting generative AI to this domain. The results show that generative AI can significantly improve the completeness of the NFR management process, providing a structured and scalable approach to handling the increasing complexity of modern vehicle systems during early development phases. However, the application of AI in this context also reveals challenges, particularly in accurately identifying and dealing with more heterogeneous or ambiguous sets of NFRs, as well as analyzing requirements that concern different levels of granularity in the system. As such, there remains considerable potential for refi nement in the pipeline, particularly in enhancing the accuracy of NFR generation for more dynamic and complex configurations. Additionally, the discussed results' evaluation by the vehicle architecture experts showed a general acceptance of this pipeline as a tool to improve the effciency of NFR management, where all the existing information regarding NFRs were structured in an efficient manner. All interviewed experts showed high acceptance for integrating this pipeline in their workflows.
Looking forward, the integration of generative AI into the development of vehicle systems extend beyond NFR management to include using these NFRs to support decisions taken during the system architecture phase of development as well as automated testing. Future work will focus on improving the pipeline's ability to handle a wider variety of vehicle configurations and environmental factors, as well as incorporating feedback mechanisms that allow continuous learning and adaptation of the AI models.
As a next step, we plan to further use the results of this pipeline when specifying the logical and physical vehicle architecture, in order to enhance the overall quality and safety of modern vehicles.

Acknowledgements

The authors would like to thank AUDI AG for supporting and funding the research project.

References

Ardagna, C. A. and Bena, N. (2023). *2. non-functional certification of modern distributed systems: A research manifesto*.

Baker, C., Deng, L., Chakraborty, S., and Dehlinger, J. (2019). Automatic multi-class non-functional software requirements classification using neural networks. *In 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, volume 2, pages 610–615.

Board, S. E. (2024). The guide to the systems engineering body of knowledge (sebok), v. 2.11. *Hoboken, NJ: The Trustees of the Stevens Institute of Technology*. Accessed 11.12.2024. www.sebokwiki.org.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). *Language models are few-shot learners*.

Casamayor, A., Godoy, D., and Campo, M. (2010). Identification of non-functional requirements in textual specifications: A semi-supervised learning approach. *Information and Software Technology*, 52(4):436–445.

Eckhardt, J., Vogelsang, A., and Fernández, D. M. (2016). Are "non-functional" requirements really nonfunctional? an investigation of non-functional requirements in practice. *In Proceedings of the 38th International Conference on Software Engineering, ICSE '16*, page 832–842, New York, NY, USA. Association for Computing Machinery.

Habibullah, K., Gay, G., and Horkoff, J. (2023). Non-functional requirements for machine learning: understanding current use and challenges among practitioners. *Requirements Engineering*, 28.

Hey, T., Keim, J., Koziolek, A., and Tichy, W. F. (2020). Norbert: Transfer learning for requirements classification.*In 2020 IEEE 28th International Requirements Engineering Conference (RE)*, pages 169–179.

Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., and Iwasawa, Y. (2023). *Large language models are zero-shot reasoners*.

Kurtanovic´, Z. and Maalej, W. (2017). Automatically classifying functional and non-functional requirements using supervised machine learning. *In 2017 IEEE 25th International Requirements Engineering Conference (RE)*, pages 490–495.

Li, B. and Nong, X. (2022). Automatically classifying non-functional requirements using deep neural network.*Pattern Recognition*, 132:108948.

Lopez-Hernandez, D. A., Octavio Ocharan-Hernandez, J., Mezura-Montes, E., and Sanchez-Garcia, A. J. (2021). Automatic classification of software requirements using artifcial neural networks: A systematic literature review. *In 2021 9th International Conference in Software Engineering Research and Innovation (CONISOFT)*,pages 152–160.

Mavin, A., Wilkinson, P., Harwood, A., and Novak, M. (2009). Easy approach to requirements syntax (ears). *In 2009 17th IEEE International Requirements Engineering Conference*, pages 317–322.

Norheim, J. J. and Rebentisch, E. (2024). Structuring natural language requirements with large language models.*In 2024 IEEE 32nd International Requirements Engineering Conference Workshops (REW)*, pages 68–71.

Olsina, L., Becker, P., Papa, M. F., and Rossi, G. (2024). Concepts and applicability of non-functional requirements for particular and universal things. *In 2024 L Latin American Computer Conference (CLEI)*,pages 1–10.

Saroja, S. and Haseena, S. (2023). *Functional and non-functional requirements in agile software development.*

Shankar, P., Morkos, B., Yadav, D., and Summers, J. D. (2020). 2. towards the formalization of non-functional requirements in conceptual design. *Research in Engineering Design*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2023). *Attention is all you need*.