Low-Level Hardware Requirement Classification Using Large Language Models: Challenges, Insights, and Future Directions for Embedded Control Systems

Ekrem Bilgehan Uyar[1,2*], Ali Ergin Gürsoy[2], Cemil Gökçe[2], Tuğba Taşkaya Temizel[1]

[1] *Graduate School of Informatics, Middle East Technical University, Ankara 06800, Türkiye*
[2] *Roketsan Inc., Ankara 06780, Türkiye*

Abstract
Automated Requirements Engineering (RE) activities can streamline development processes, reduce errors, and facilitate informed decision-making, particularly for low-level hardware requirements where modifications are costly. Classification is a widely studied automated RE activity for software requirements. Yet, its applicability remains underexplored due to the lack of structured datasets. This study adapts and evaluates software requirement classification techniques for hardware by extracting low-level requirements from open-source hardware design artifacts of Embedded Control Systems. We evaluate two classification methods: fine-tuning a BERT-based model and zero-shot prompting with a quantized LLM (Qwen2.5). While fine-tuning achieved high accuracy, zero-shot classification with specific prompts outperformed it in overall performance, achieving an average F1-score of up to 90% on the hold-out test set. Our findings suggest that automating downstream RE activities for low-level hardware requirements may not require large, task-specific datasets; however, classification performance can be further improved and can serve as an enabler for advanced tasks.

Keywords
Requirements Classification, Large Language Models, Embedded Control Systems.

1. Introduction

Embedded Control Systems (ECS) are widely employed in automotive, aerospace, and consumer electronics. Their hybrid hardware-software nature and ubiquity make them an intriguing subject for Requirements Engineering (RE) research [1]. Managing their hardware requirements is critical, as changes impact physical design and production, often incurring high costs and delays [2,3]. Requirement classification can support this process, enabling downstream activities such as class-based test planning or physical feasibility analysis.

While classification is well-studied for software requirements, hardware requirements differ due to factors such as interdependencies and quantitative constraints. They often involve strict quantitative constraints (e.g., minimum and maximum values) rather than the abstract definitions found in software. In addition, interdependence between requirements [4] adds complexity, where altering one can necessitate additional changes or create new constraints. Despite these differences, software requirement classification techniques can be adapted if suitable datasets are available. However, open-access low-level hardware datasets are scarce, as most requirement documents are proprietary or embedded within technical documentation.

In: A. Hess, A. Susi, E. C. Groen, M. Ruiz, M. Abbas, F. B. Aydemir, M. Daneva, R. Guizzardi, J. Gulden, A. Herrmann, J. Horkoff, S. Kopczyńska, P. Mennig, M. Oriol Hilari, E. Paja, A. Perini, A. Rachmann, K. Schneider, L. Semini, P. Spoletini, A. Vogelsang. Joint Proceedings of REFSQ-2025 Workshops, Doctoral Symposium, Posters \& Tools Track, and Education and Training Track. Co-located with REFSQ 2025. Barcelona, Spain, April 7, 2025.

* Corresponding author.

lastname.secondname@metu.edu.tr (E.B. Uyar); secondname.lastname@roketsan.com.tr (A. E. Gürsoy); name.lastname@roketsan.com.tr (C. Gökçe); ttemizel@metu.edu.tr (T. T. Temizel)
0000-0001-8154-332X (E. B. Uyar); 0000-0001-7387-8621 (T. T. Temizel)

To address this, we collaborated with domain experts (DEs) to create and annotate a dataset derived from open-source hardware design materials, refining inferred constraints to align with established dataset-creation frameworks. Then we experimented with two baseline classification methods—fine-tuning a BERT-based model and applying a quantized large language model (Qwen2.5) with zero-shot prompting—evaluating their performance using precision, recall, and F1-score. Our findings suggest that in highly specialized domains, leveraging larger models with well-crafted prompts is a more efficient alternative to constructing extensive datasets for fine-tuning. Furthermore, our analysis of misclassified requirements underscores the critical role of context in classification accuracy. These insights form a basis for downstream RE tasks and more advanced applications, including the integration of Large Language Model guardrails for industrial deployment.

The remainder of this paper is organized as follows: Section 2 reviews automated RE tasks for natural language requirements. Section 3 details the methodology and dataset creation. Section 4 presents the classification results, while Section 5 outlines assumptions, limitations, and exclusions. Section 6 discusses the findings, and Section 7 summarizes key insights and future directions for applying RE tasks in the ECS domain.

2. Related Work

Automated RE tasks have been extensively explored using Natural Language Processing (NLP) and machine learning in the embedded systems domain. Applications include, but are not limited to, requirements modeling [5], ontology-based specification [3], and Named Entity Recognition (NER) for requirement analysis [6]. Although the application of automated requirements classification tasks in the context of embedded systems is not particularly popular, it is widely studied in the field of software engineering [7]. Recently, transfer-learning approaches have shown their effectiveness both through approaches such as fine-tuning [8,9] and through approaches such as prompting [10,11] for software requirements classification. However, these methods rely on structured requirement datasets, limiting their direct applicability to ECS and hardware requirements due to the numerical and physical constraints that complicate direct adaptation.

In adjacent domains like chip design [12,13], Large Language Models (LLMs) have been employed for design validation and documentation synthesis, demonstrating their potential in processing technical constraints. Their applications extend across various industries, including aerospace [14] and automotive [15], underscoring the versatility of LLMs in solving complex engineering problems in different domains. However, such applications typically require domain-specific fine-tuning, a challenge given the lack of open datasets for hardware RE. A significant barrier to their broader application is the lack of open datasets tailored to domain-specific tasks. The performance of LLMs is highly dependent on the quality and relevance of the datasets used for training and evaluation [16].

Although zero-shot learning allows LLMs to operate in the absence of domain-specific datasets, its limitations become evident as even advanced models can struggle in such settings [17]. Therefore, the creation of high-quality datasets is indispensable. This study addresses these gaps by adapting NLP-driven RE techniques to a dataset derived from hardware design sources, exploring LLM-based classification within the constraints of ECS hardware requirements. To this end, annotation development frameworks, such as the MATTER cycle [18], along with dataset documentation frameworks [19-22], are essential for preparing datasets that support the effective application of LLMs in new domains.

3. Research Design

3.1. Research Questions & Scope

This study investigates the feasibility of adapting requirement classification techniques widely used in software engineering to ECS hardware requirements. To structure our investigation, we define the following research questions:

- RQ1: What are the challenges in creating a representative annotated requirements corpus from hardware design sources to enable automated classification?
- RQ2: How effective are fine-tuning and zero-shot approaches with large language models (LLMs) at accurately classifying ECS hardware requirements?

RQ1 addresses the methodological challenges by investigating how to extract, annotate, and validate hardware requirements in a structured manner. RQ2 evaluates the ability of existing classification techniques, adapted from software RE, to perform effectively in the hardware domain.

3.2. Dataset Creation Process & Its Practical Implications

In our case, hardware requirements are embedded within circuit diagrams, technical documentation, and component datasheets, unlike common software RE datasets compiled from textual specifications. Therefore, we developed a hardware requirements dataset by reverse-engineering design artifacts from open-source ECS projects. This approach introduced several challenges addressing RQ1:

- Implicit vs. Explicit Requirements: Unlike explicitly defined requirements, low-level ECS hardware requirements need to be inferred from design constraints and physical elements.
- Interdependencies: Hardware requirements can exhibit a higher degree of interconnectivity and interdependency; a single modification in one requirement often cascades into multiple downstream constraints.
- Quantitative Features: Unlike software requirements, which frequently describe system functionalities, hardware requirements predominantly involve numerical constraints (e.g., voltage ratings, timing requirements, and temperature tolerances).
- Size: To ensure effective model training, the dataset should be sufficiently large, similar to established datasets like PROMISE [23], which has been used in similar software requirement classification tasks with comparable complexity.

To overcome these challenges, we adopted an iterative annotation framework with DEs. We systematically managed both the requirement writing and annotation processes using guidelines developed according to the MATTER cycle [18]. This annotation development cycle, recognized within the RE community [24,25], employs an iterative and incremental approach to ensure accuracy and reliability throughout the dataset creation process. The details of the process are presented in the following subsections.

3.2.1. Domain Expert Selection

The selection criteria for the contributing DEs were critical to the creation and annotation of the ECS requirements corpus. As emphasized by [26], experts must demonstrate significant skills, knowledge, and experience. Addressing these, three electrical and electronics engineers with 10 to 15 years of experience in the R&D departments of ECS/embedded systems contributed voluntarily to this study. Bayerl and Paul [27] recommend that annotators possess comparable domain knowledge and receive appropriate training. Accordingly, comprehensive training sessions on RE and NLP tasks were provided to the DEs.

3.2.2. Classification Problem

The DEs identified five basic functionality categories by analyzing the functional blocks of the ECS, which form the basis of the annotation problem. Our labeling structure was designed to imitate the general architecture of feedback control systems. While alternative taxonomy approaches, such as categorization based on design expertise (e.g., analog, digital, or power circuit design), could have been applied, we determined that a more pragmatic, exploratory classification was appropriate since no clear guideline exists in the ECS domain to support it yet. Finally, we aimed to make the categories as inclusive as possible while ensuring broad applicability across different ECS application domains. The identified categories and their definitions are presented in Table 1.

Table 1
The five functional categories

Label	Category	Definition
F	Feedback	Any requirement defining a specification for sensing the controlled physical entity. Typically, they may refer to current delivered to the controlled load, position information according to the controlled motion of an actuator, pressure, humidity, or temperature.
C	Controller	Any requirement defining a specification for the "smart" functionalities regarding the processing cores of the system. Typically, any requirement consisting of any digital or analog control specification including processing or memory capabilities relates to this category.
I	Interface	Any requirement defining a specification for the connections of the system and the external world, as well as physical user interfaces such as buttons, switches, status LEDs, screens, etc.
P	Power	Any requirement defining a specification for managing the power delivered to or from the system. These include conversion, limits, fluctuation, EMI/C specifications, as well as power lines of the system peripherals such as sensors.
D	Driver	Any requirement defining a specification for driving the controlled load. Typically, the load may refer to electric motors, heaters, fans, pumps, or any type of actuator.

3.2.3. Open-Source Hardware ECS Project Selection

[18] emphasizes that to prevent bias in model training, the over-representation of features rarely encountered in real-world scenarios must be avoided. To represent a product inventory emulating an industrial setting that can be used for AI training, we selected source projects based on the following criteria: (1) permissive licensing, (2) representation of real-world ECS applications, (3) diversity in products and functional blocks to enhance dataset representativeness, and (4) availability of structured design documentation.

Based on these inclusion and exclusion criteria, eight open-source ECS projects were selected from the project list maintained by the Open-Source Hardware Association[2] or published in the open-source hardware journal HardwareX[3]. These projects were identified through targeted keyword searches focusing on representing a diverse range of physical entities, including pressure, weight, and velocity, and involve control instruments such as pumps and fans. Each selected project underwent review and consensus by the DEs to ensure alignment with the four mentioned criteria. Table 2 presents the complete list of projects along with their IDs and functional domains. The functional domain descriptions here aim to provide insight into system complexity, component dependencies, and diversity across projects. For example, domains like robotics and power electronics feature tightly integrated components, while food processing and agricultural automation rely on more modular architectures. This also reflects the variety and number of components, as different domains inherently require distinct sets of sensors, actuators, and control mechanisms, ensuring a representative dataset of diverse functional blocks.

---

[2] https://certification.oshwa.org/list.html
[3] ISSN: 2468-0672. Available online: https://www.sciencedirect.com/journal/hardwarex

Table 2
List of selected open-source ECS projects

ID	Source	Functional Domain
P001	[28]	Generic Embedded Control Systems (Flexible IoT control board for various applications)
P002	[29]	Robotics (Hydraulic quadruped robot control system)
P003	[30]	Additive Manufacturing (Multifunctional extruder with sensing and monitoring capabilities)
P004	[31]	Food Processing & Preservation (Automated dry-aging system for beef)
P005	[32]	Actuator Control (High-current driver for magnetic actuators)
P006	[33]	Agricultural Automation (Automated sensing and control for irrigation scheduling)
P007	[34]	Environmental Monitoring (Autonomous ocean profiler for hazardous area measurements)
P008	[35]	Power Electronics (Modular control platform for voltage source inverter)

3.2.4. Requirement Extraction and Annotation

Our reverse-engineering approach for extracting and annotating requirements from hardware projects has two key aspects: (1) Expert-Guided Requirement Identification, where DEs defined extraction guidelines, manually extracted requirements from design documents and categorized them; (2) Iterative Annotation and Refinement, where DEs refined guidelines, requirements, and dataset through consensus-based adjudication. For example, DE review meetings identified ambiguities, especially in multi-board projects and connector requirements. The guidelines were then revised to focus on electronic circuit structures while excluding mechanical and environmental constraints, as they had minimal impact on functional categorization.

The process resulted in 366 requirements from eight projects over 191 DE hours spanning 184 days. After ensuring, through consensus, that the requirements extracted by the DEs for each project comply with the guidelines, they were all randomly shuffled and reassigned to the DEs as a list for annotation. The resulting annotation consistency for the complete set was assessed using recommended [36,37] Inter-Annotator Agreement (IAA) metrics: Fleiss' Kappa [37] (0.76) and Cohen's Kappa [38] (0.66–0.74), indicating strong agreement. Based on these scores, labeling repetition was unnecessary. However, the adjudication process aimed to establish a gold standard by achieving consensus among annotators, ensuring high data quality through error correction and consistency checks.

During adjudication meetings, non-consensus requirements were reviewed through discussions where DEs explained their classification rationale. If disagreements stemmed from requirement quality issues, such as violations of agreed-upon guidelines or inconsistencies with the design artifacts, necessary corrections were made to align them with the established criteria. Requirements that failed to meet technical sufficiency standards were either revised or removed, ensuring coherence and reliability in the final dataset. As a result, 28 out of 366 requirements were removed due to insufficient technical details, while 24 were modified to resolve ambiguities. These refinements addressed violations of predefined guidelines but did not introduce a significant shift in distribution or category representation.

The final gold standard revealed category imbalance, reflecting real-world distributions. Project-level requirement distributions varied based on complexity, and differences among the three DEs' annotations (DE Distribution) were influenced by their distinct writing styles. An example of a feedback requirement can be seen in Table 3, while the finalized dataset's statistical distribution is presented in Table 4.

Table 3
DE writing styles across four original *Feedback* requirements targeting the same functionality.

Req ID	DE ID	Requirement
R107	DE001	The system shall be capable of measuring relative humidity and temperature with accuracy 2.0% and 0.5C respectively that communicate over SPI with the sensor.
R057	DE002	The system shall measure relative humidity (RH) with an accuracy level of ±2.0% and temperature with an accuracy level of ±0.5% via sensor with digital output.
R051	DE003	The system shall read relative humidity sensor with accuracy ± 2.0%
R044	DE003	The system shall read a temperature sensor with accuracy ± 0.5 °C

Table 4
Statistical Distributions of the Adjudicated ECS Dataset.

		Count	Percentage (%)
Label Distribution	I	99	29.29
	P	77	22.78
	F	77	22.78
	D	55	16.27
	C	30	8.88
Source Project Distribution	P006	59	17.51
	P005	48	14.24
	P002	47	13.95
	P007	44	13.06
	P001	37	10.98
	P003	36	10.68
	P004	35	10.39
	P008	31	9.20
DE Distribution	DE003	125	36.98
	DE002	110	32.54
	DE001	103	30.47

4. Automated Classification

4.1. Baseline Model: A Fine-tuned Approach with BERT

We utilized NoRBERT [8] as a baseline model for classifying ECS requirements, , which has reported F1 scores of up to 94% in multi-class software requirement classification performance. However, due to differences in classification schemes and domain-specific terminology, the pre-trained NoRBERT model could not be directly applied in this study. Instead, we adopted NoRBERT's methodology and fine-tuned it for classifying ECS requirements.

4.2. Zero-shot ECS Requirement Classification with LLMs

Given the limited availability of datasets, prompt-based classification with LLMs is a viable alternative to dataset-heavy fine-tuning methods. Our zero-shot prompting approach consisted of two main steps: Prompt Design and Model Selection/Setting.

In Prompt Design, we adapted the strategy and methodology from [10], who applied similar techniques in the RE domain to develop prompts that leverage the pre-trained capabilities of LLMs for multiclass requirement classification. Based on their recommendations, we considered four prompt patterns, which were shown to be effective for the binary classification of software requirements. Table 5 demonstrates how we adapted these patterns to the ECS low-level requirements classification problem.

We considered the model selection criteria based on potential needs that may arise in industrial use, as ECS design teams typically prioritize solutions that align with their operational and technical constraints. These include (1) **non-commercial solutions** to address data privacy and control concerns, (2) **scalable solutions** that do not require significant local computation investments, and (3) **less restrictive licenses** for broader adaptability. Based on these criteria, we selected a quantized variant of Qwen2.5-72B-Instruct. These models can be locally hosted on a single machine using popular open-source libraries, such as *llama.cpp* and *GPT4All* [15].

Table 5
Designed Zero Shot Prompt Patterns for ECS Requirements Classification

Pattern (ID)	Prompt
Cognitive Verifier (1)	Classify the given hardware requirement into one of the five functional labels. These labels are feedback (labelled as F), driver (labelled as D), interface (labelled as I), power (labelled as P) and controller (labelled as C). The label definitions are as follows: (...) Ask me questions if needed to break the given task into smaller subtasks. All the outputs to the smaller subtasks must be combined before you generate the final output. The requirement: (...)
Context Manager (2)	Classify the given hardware requirement into one of the five functional labels. These labels are feedback (labelled as F), driver (labelled as D), interface (labelled as I), power (labelled as P) and controller (labelled as C). The label definitions are as follows: (...) When you provide an answer, please explain the reasoning and assumptions behind your response. If possible, address any potential ambiguities or limitations in your answer, to provide a more complete and accurate response. The requirement: (...)
Persona (3)	Act as a requirements engineering domain expert in embedded control systems and classify the given hardware requirement into one of the five functional labels. These labels are feedback (labelled as F), driver (labelled as D), interface (labelled as I), power (labelled as P) and controller (labelled as C). The label definitions are as follows: (...) The requirement: (...)
Question Refinement (4)	Classify the given hardware requirement into one of the five functional labels. These labels are feedback (labelled as F), driver (labelled as D), interface (labelled as I), power (labelled as P) and controller (labelled as C). The label definitions are as follows: (...) Ask me questions if needed to break the given task into smaller subtasks. All the outputs to the smaller subtasks must be combined before you generate the final output. If needed, suggest a better version of the question to use that incorporates information specific to this task and ask me if I would like to use your question instead. The requirement: (...)

4.3. Experimental Setup

Various methods exist for splitting datasets into training, validation, and test sets, such as random or stratified sampling. However, rather than using cross-validation approaches, this study held out the requirements from two entire projects as the test set to prevent data leakage and ensure evaluation on a distinct dataset. The DEs selected these projects by consensus, choosing one that closely resembled the others in terms of diversity and complexity and another that was least similar. This strategy enabled a more comprehensive assessment of the model's generalizability under strict time and computational cost constraints. The test set was strictly excluded from all stages of the study, including training, fine-tuning, and model selection experiments.

Our BERT-based baseline model was fine-tuned using the remaining six projects, with hyperparameters optimized via Grid Search. For Qwen2.5-72B-Instruct, the llama.cpp library was used on a local machine. Each requirement was processed individually to prevent cross-influence, and results were logged separately. To ensure inference independence, the model was reloaded for each requirement. For handling anomalies in prompting results, we treated the following cases as misclassifications: when the model ignored requirement details, generated outputs outside the five predefined categories, or returned null.

4.4. Results

This section presents the results of the classification of ECS low-level hardware requirements, addressing RQ2 through the evaluation of two models: fine-tuned BERT and zero-shot Qwen-2.5, applied with four distinct prompt patterns. Table 6 summarizes the performance of the four prompt patterns: **(1) Cognitive Verifier**, **(2) Context Manager**, **(3) Persona**, and **(4) Question Refinement**, evaluated using Qwen-2.5 (N) and fine-tuned BERT (T-0) on a hold-out test set. The results are presented for each category with precision (P), recall (R), and F1-score (F1) metrics, along with average accuracy (A) and weighted F1-score (w.F) for each model.

Table 6
The experiment results for the hold-out test set.

Category \ Classifier		N-1	N-2	N-3	N-4	T-0
Feedback (Support: 24)	P	0.89	0.89	0.91	0.90	0.82
	R	0.71	0.71	0.83	0.75	0.75
	F1	0.79	0.79	0.87	0.82	0.78
Interface (Support: 32)	P	0.78	0.77	0.83	0.81	0.81
	R	0.91	0.94	0.94	0.94	0.94
	F1	0.84	0.85	0.88	0.87	0.87
Controller (Support: 7)	P	0.60	0.75	0.86	0.55	1.00
	R	0.86	0.86	0.86	0.86	0.86
	F1	0.71	0.80	0.86	0.67	0.92
Driver (Support: 13)	P	0.92	0.92	1.00	1.00	1.00
	R	0.85	0.85	0.85	0.69	0.62
	F1	0.88	0.88	0.92	0.82	0.76
Power (Support: 16)	P	1.00	1.00	1.00	1.00	0.74
	R	0.88	0.88	1.00	0.94	0.88
	F1	0.93	0.93	1.00	0.97	0.80
A		0.84	0.85	**0.90**	0.85	0.83
w.F		0.84	0.85	**0.90**	0.85	0.82

The results indicate that the Qwen-2.5 model with the Persona prompt pattern (N-3) generally achieves the most balanced performance among the evaluated configurations. Analysis of the confusion matrices reveals that N-3 reduces certain types of misclassifications compared to the fine-tuned BERT model (T-0). For instance, N-3 does not misclassify *Power* as *Feedback*, whereas T-0 records two such errors. Additionally, N-3 demonstrates improved accuracy in distinguishing between *Driver* and *Power*. For the test set, *Power* and *Driver* categories exhibit perfect precision, while *Controller* has the highest variance across most configurations. Since zero-shot settings eliminate the need for a test set, we also evaluated the Qwen-2.5 model using the full dataset, comprising all 338 requirements, under the same configuration. The results, as summarized in Table 7, demonstrate similar performance compared to the hold-out set, with category-level consistency largely maintained. However, confusion between *Interface/Feedback* and *Controller/Interface* persists across both models for individual requirements, indicating a common challenge in these categories: errors tend to occur when requirements involve communication or control specifications between sub-circuits or feedback sensors.

Table 7
The experiment results for the complete dataset.

Label \ Classifier		N-1	N-2	N-3	N-4
Feedback (Support: 77)	P	0.85	0.92	0.92	0.88
	R	0.78	0.77	0.84	0.82
	F1	0.81	0.84	0.88	0.85
Interface (Support: 99)	P	0.85	0.80	0.84	0.86
	R	0.88	0.95	0.94	0.87
	F1	0.87	0.87	0.89	0.86
Controller (Support: 30)	P	0.68	0.79	0.77	0.63
	R	0.90	0.87	0.77	0.90
	F1	0.77	0.83	0.77	0.74
Driver (Support: 55)	P	0.94	0.92	0.94	0.98
	R	0.87	0.87	0.89	0.85
	F1	0.91	0.90	0.92	0.91
Power (Support: 77)	P	0.92	0.93	0.93	0.91
	R	0.87	0.86	0.88	0.82
	F1	0.89	0.89	0.91	0.86
A		0.86	0.87	**0.88**	0.85
w.F		0.86	0.87	**0.88**	0.86

5. Threats to Validity

Our dataset is based on open-source ECS projects, which may not fully represent industrial requirements. Since contributions were voluntary, only three non-native (but proficient) DEs participated, potentially limiting annotation consistency and introducing subjective variations. Similarly, to balance computational efficiency and generalization assessment, we manually selected the test set instead of applying cross-validation, which would have required retraining the model multiple times at a high computational cost. To minimize potential bias, the selection was made through the DE consensus. While the DEs were highly skilled in technical domains, they lacked prior expertise in computational linguistics or annotation processes. Additionally, differences between LLM and library versions may significantly impact the obtained results. Contextual ambiguity also remains a challenge, as hardware requirements were classified without explicit system-wide references. To mitigate these issues, training sessions were held for the DEs, annotation guidelines were refined to minimize ambiguity, the adjudication process was applied at each iteration, and misclassifications were thoroughly analyzed to assess model errors.

While the dataset and code remain unpublished due to ongoing PhD research, we provide a detailed methodology and analytical framework to ensure transparency and support reproducibility within the constraints of our current research phase.

6. Discussion

This study explored how software requirement classification techniques can be adapted for low-level ECS hardware requirements, addressing dataset construction challenges (RQ1) and classification performance (RQ2).

Low-level hardware requirements were inherently implicit in our case, derived from design artifacts rather than explicitly stated. We experienced firsthand the labor-intensive process, a difficulty widely acknowledged in the literature [16]. However, the framework outlined in [18] provided a structured approach to managing [19-22] this process. Despite adhering to established guidelines, the domain's complexity led DEs to perceive strong interdependencies and a critical need for contextual understanding when evaluating individual requirements. Furthermore, constructing a dataset comparable in size to established software datasets like PROMISE [23] demanded considerable effort, and in our case, we could only achieve approximately half its size.

Our findings on the classification performance indicate that the fine-tuning approach performed well on structured hardware requirements, but zero-shot classification with prompt engineering outperformed the fine-tuned approach in certain cases, reducing reliance on dataset curation. Aligning with the most related work in the software domain [8-11], fine-tuning remains effective but demands high-quality datasets, technical expertise, and computational resources. However, contrary to [17], we observed that the zero-shot approach yielded better results for our case, considering the domain and dataset size. However, interpreting the impact of specific performance metrics, e.g., precision and recall, depends on the downstream task to which classification is applied.

Hardware requirements introduce strong interdependencies that make classification sensitive to contextual ambiguity. Some feedback-related requirements were classified as *Interface*, likely due to internal communication specifications, while some *Driver* specifications (e.g., DACs) were misclassified as *Controller* due to references to signal processing. *Driver* and *Power* categories tend to be quantitative constraints heavy while being less dependent on context, leading to higher precision. In contrast, *Feedback* and especially *Controller* involve significant interdependencies—internal data flow and control-related inter-circuit interfaces require either project context or additional requirements for accurate classification, making isolated evaluation more challenging.

Misclassified cases, particularly within the *Interface* category, often stemmed from pre-consensus disagreements among the DEs. A review of adjudication meeting notes revealed that ambiguous cases frequently involved multi-board projects, where internal and external interfaces were not always distinguishable without context. This suggests that for low-level hardware requirements, context is crucial for classification accuracy. Providing the full set of requirements or key system attributes as a context could significantly improve performance. Hybrid methods and prompt engineering could mitigate these challenges.

7. Conclusion and Future Work

This study assessed the feasibility of applying software requirement classification techniques to ECS hardware requirements. Dataset construction (RQ1) highlighted challenges in extracting requirements from design artifacts and developing a structured classification protocol, demanding domain expertise and iterative refinement. Model evaluation (RQ2) found that well-crafted zero-shot LLM prompts, particularly the Persona-based prompt, could outperform fine-tuned BERT models, eliminating the need for extensive dataset curation and fine-tuning. The findings highlight inherent ambiguity due to the interdependencies in low-level hardware requirements, particularly when analyzed individually versus holistically.

While classification currently serves as an auxiliary activity in our case, it can enable systematic efforts in ECS design for downstream tasks like analysis, verification, validation, testing, etc. Future work will explore dataset expansion and its open-access release as well as confidence-aware systems integrating human-in-the-loop approaches to enhance classification reliability. Additionally, understanding prompt efficiency can help reduce classification errors and improve requirement quality. Beyond the ECS, these advancements could enable cross-domain adaptability, allowing requirement classification techniques to be transferred to complex hardware-intensive domains, where structured but implicit requirements play a crucial role.

Finally, an important direction for future work is exploring embedded system co-design, where subsystem-level requirements are systematically decomposed into hardware and software components. This process could further enable automated test and validation procedures, ensuring seamless integration between software and hardware functionalities.

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT and Grammarly to: Grammar and spelling check, Paraphrase, and reword. After using these services, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] Fariha, A., Alwidian, S., & Azim, A. (2023). Towards Requirements Specification Collaboration Forum for Embedded Software Systems. *2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*, 312–317. https://doi.org/10.1109/MODELS-C59198.2023.00061

[2] Aceituna, D. (2013). Survey of Concerns in Embedded Systems Requirements Engineering. *SAE International Journal of Passenger Cars - Electronic and Electrical Systems*, *7*(1), 1–13. https://doi.org/10.4271/2013-01-2403

[3] Sousa, A., Couto, T., Agra, C., & Alencar, F. (2016). Use of Ontologies in Embedded Systems: A Systematic Mapping. *2016 10th International Conference on the Quality of Information and Communications Technology (QUATIC)*, 1–8.

[4] Aalund, R., & Philip Paglioni, V. (2025). Enhancing Reliability in Embedded Systems Hardware: A Literature Survey. *IEEE Access*, *13*, 17285–17302. IEEE Access.

[5] Ruan, K., Chen, X., & Jin, Z. (2023). Requirements Modeling Aided by ChatGPT: An Experience in Embedded Systems. *2023 IEEE 31st International Requirements Engineering Conference Workshops (REW)*, 170–177.

[6] Chow, M. Y. (2023). Analysis of Embedded System's Functional Requirement using BERT-based Name Entity Recognition for Extracting IO Entities. *Journal of Information Processing*, *31*(0), 143–153. https://doi.org/10.2197/ipsjjip.31.143

[7] Lopez-Hernandez, D. A., Octavio Ocharan-Hernandez, J., Mezura-Montes, E., & Sanchez-Garcia, A. J. (2021). Automatic Classification of Software Requirements using Artificial Neural Networks: A Systematic Literature Review. *2021 9th International Conference in Software Engineering Research and Innovation (CONISOFT)*, 152–160.

[8] Hey, T., Keim, J., Koziolek, A., & Tichy, W. F. (2020). NoRBERT: Transfer Learning for Requirements Classification. *2020 IEEE 28th International Requirements Engineering Conference (RE)*, 169–179. https://doi.org/10.1109/RE48521.2020.00028

[9] Kici, D., Malik, G., Cevik, M., Parikh, D., & Başar, A. (2021). A BERT-based transfer learning approach to text classification on software requirements specifications. *Proceedings of the Canadian Conference on Artificial Intelligence.*

[10] Ronanki, K., Cabrero-Daniel, B., Horkoff, J., & Berger, C. (2024). Requirements engineering using generative AI: Prompts and prompting patterns. In *Generative AI for Effective Software Development* (pp. 109-127). Cham: Springer Nature Switzerland.

[11] Arvidsson, S., & Axell, J. (2023). *Prompt engineering guidelines for LLMs in Requirements Engineering*. https://gupea.ub.gu.se/handle/2077/77967

[12] Meng, X., Srivastava, A., Arunachalam, A., Ray, A., Silva, P. H., Psiakis, R., Makris, Y., & Basu, K. (2023). *Unlocking Hardware Security Assurance: The Potential of LLMs* (No. arXiv:2308.11042). arXiv. https://doi.org/10.48550/arXiv.2308.11042

[13] Liu, M., Ene, T., Kirby, R., Cheng, C., Pinckney, N., Liang, R., Alben, J., Anand, H., Banerjee, S., Bayraktaroglu, I., Bhaskaran, B., Catanzaro, B., Chaudhuri, A., Clay, S., Dally, B., Dang, L., Deshpande, P., Dhodhi, S., Halepete, S., … Ren, H. (2023). *ChipNeMo: Domain-Adapted LLMs for Chip Design* (No. arXiv:2311.00176). arXiv. https://doi.org/10.48550/arXiv.2311.00176

[14] Tikayat Ray, A., Cole, B. F., Pinon Fischer, O. J., White, R. T., & Mavris, D. N. (2023). aeroBERT-Classifier: Classification of Aerospace Requirements Using BERT. *Aerospace*, *10*(3), 279.

[15] Uygun, Y., & Momodu, V. (2024). Local large language models to simplify requirement engineering documents in the automotive industry. Production & Manufacturing Research, 12(1), 2375296.

[16] Sambasivan, N., Kapania, S., Highfill, H., Akrong, D., Paritosh, P., & Aroyo, L. M. (2021, May). "Everyone wants to do the model work, not the data work": Data Cascades in High-Stakes AI. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (pp. 1-15).

[17] Murthy, R., Kumar, P., Venkateswaran, P., & Contractor, D. (2024). Evaluating the Instruction-following Abilities of Language Models using Knowledge Tasks. *arXiv preprint arXiv:2410.12972.*

[18] Pustejovsky, J., & Stubbs, A. (2012). Natural Language Annotation for Machine Learning: A guide to corpus-building for applications. " O'Reilly Media, Inc.".

[19] Gebru, T., Morgenstern, J., Vecchione, B., Vaughan, J. W., Wallach, H., Iii, H. D., & Crawford, K. (2021). Datasheets for datasets. *Communications of the ACM*, *64*(12), 86-92.

[20] Holland, S., Hosny, A., Newman, S., Joseph, J., & Chmielinski, K. (2020). The dataset nutrition label. *Data Protection and Privacy*, *12*(12), 1.

[21] Bender, E. M., & Friedman, B. (2018). Data statements for natural language processing: Toward mitigating system bias and enabling better science. *Transactions of the Association for Computational Linguistics*, *6*, 587-604.

[22] Hutchinson, B., Smart, A., Hanna, A., Denton, E., Greer, C., Kjartansson, O., ... & Mitchell, M. (2021, March). Towards accountability for machine learning datasets: Practices from software engineering and infrastructure. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency* (pp. 560-575).

[23] Cleland-Huang, J., Mazrouee, S., Liguo, H & Port, D. (2007). nfr [Data set]. Zenodo. Available: http://doi.org/10.5281/zenodo.268542

[24] Fischbach, J., Frattini, J., Spaans, A., Kummeth, M., Vogelsang, A., Mendez, D., & Unterkalmsteiner, M. (2021). Automatic detection of causality in requirement artifacts: the cira approach. In *Requirements Engineering: Foundation for Software Quality: 27th International Working Conference, REFSQ 2021, Essen, Germany, April 12–15, 2021, Proceedings 27* (pp. 19-36). Springer International Publishing.

[25] Noh, Y., Kim, K., Lee, M., Heo, C., Jeong, Y., Jeong, Y., ... & Choi, K. S. (2020, October). Enhancing quality of corpus annotation: Construction of the multi-layer corpus annotation and simplified validation of the corpus annotation. In *Proceedings of the 34th Pacific Asia Conference on Language, Information and Computation* (pp. 216-224).

[26] Hopkins, P., & Unger, M. (2017). What is a'subject-matter expert'? *Journal of Pipeline Engineering*, *16*(4).

[27] Bayerl, P. S., & Paul, K. I. (2011). What determines inter-coder agreement in manual annotations? A meta-analytic investigation. *Computational Linguistics*, *37*(4), 699–725.

[28] *Anthilla/AnthC*. (2024). [HTML]. Anthilla. https://github.com/Anthilla/AnthC

[29] Fang, L., Zhang, J., Zong, H., Wang, X., Zhang, K., Shen, J., & Lu, Z. (2023). Open-source lower controller for twelve degrees of freedom hydraulic quadruped robot with distributed control scheme. *HardwareX*, *13*, e00393.

[30] Klar, V., Pearce, J. M., Kärki, P., & Kuosmanen, P. (2019). Ystruder: Open source multifunction extruder with sensing and monitoring capabilities. *HardwareX*, *6*, e00080.

[31] Lau, S. K., Ribeiro, F. A., Subbiah, J., & Calkins, C. R. (2019). Agenator: An open source computer-controlled dry aging system for beef. *HardwareX*, *6*, e00086.

[32] Bujnowicz, Ł., & Sarewicz, M. (2022). Multichannel pulse high-current driver of magnetic actuator. *HardwareX*, *11*, e00286.

[33] Abbas, N. S., Salim, M. S., & Sabri, N. (2024). ASCD: Automatic sensing and control device for crop irrigation scheduling. *HardwareX*, *18*, e00523.

[34] Poulsen, E., Eggertsen, M., Jepsen, E. H., Melvad, C., & Rysgaard, S. (2022). Lightweight drone-deployed autonomous ocean profiler for repeated measurements in hazardous areas–Example from glacier fronts in NE Greenland. *HardwareX*, *11*, e00313.

[35] Lezcano, H., Rodas, J., Pacher, J., Ayala, M., & Romero, C. (2023). Design and validation of a modular control platform for a voltage source inverter. *HardwareX*, *13*, e00390.

[36] Klie, J. C., Castilho, R. E. D., & Gurevych, I. (2024). Analyzing dataset annotation quality management in the wild. *Computational Linguistics*, *50*(3), 817-866.

[37] Kim, M., Qiu, X., & Wang, Y. (Arthur). (2024). Interrater agreement in genre analysis: A methodological review and a comparison of three measures. *Research Methods in Applied Linguistics, 3(1)*, 100097. https://doi.org/10.1016/j.rmal.2024.100097

[38] Fleiss, J. L. (1971). Measuring nominal scale agreement among many raters. *Psychological bulletin*, *76*(5), 378.

[39] Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and psychological measurement*, *20*(1), 37-46.