Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 9, No. 3, 856-863
2025
Publisher: Learning Gate
DOI: 10.55214/25768484.v9i3.5357
© 2025 by the author; licensee Learning Gate

Using Large Language Models in Software Requirements Analysis

Han Seong Son*[1]
[1]Division of Software Engineering, Joongbu University, 305 Donghun-ro, Goyang-si, Gyeonggi-do, 10279, Republic of Korea; hsson@joongbu.ac.kr (H.S.S.).

Abstract: Improperly specified requirements can cause hazardous situations when they are designed and implemented into software. For example, missing requirements may lead to deaths and injuries. Software requirements analysis, which involves identifying and correcting inadequately defined requirements, plays a crucial role in reducing the possibility of hazardous situations caused by safety-critical software. This work proposes a requirements analysis method for detecting missing requirements and minimizing sets of requirements. The proposed method utilizes an LLM and a prompt pattern. The feasibility and effectiveness of the proposed method have been demonstrated by applying it to the existing requirements.

Keywords: *Hazard Analysis Risk Assessment, Large Language Models, Recipe Pattern, AI tools, Software requirements analysis (SRA).*

1. Introduction

The importance of software requirements analysis (SRA) in the application of software to safety-critical systems cannot be overstated. Software requirements analysis involves identifying and correcting these inadequately defined requirements. Improperly specified requirements can cause hazardous situations when they are designed and implemented into software. For example, missing requirements may cause deaths and injuries [1]. Extensive researches have been conducted to identify missing requirements efficiently. Cantone and Adonsou proposed to use Karnaugh Maps (K-Maps) in software requirements analysis to identify absent requirements and reduce redundant related requirements [2]. They applied the analysis method to actual requirements for the feasible responses of International Space Station to avoid the collision with earth debris and solar debris. Through the application, they demonstrated that K-Maps are useful to identify absent requirements and reduce redundant related requirements.

Ensuring the software safety frequently depends on iterative activities involving natural language (NL), including requirements safety analysis. Thus, tools based on Artificial Intelligence (AI) that can handle NL have the potential to enhance the efficiency and speed of these iterative activities. Large Language Models (LLMs) are among the most promising AI tools for ensuring the software safety with the consideration of these aspects. Ali Nouri et al. suggested a LLM-based Hazard Analysis Risk Assessment (HARA), which describes a HARA task to an LLM through prompt engineering [3]. To design the pipeline of the prompts for LLM-based HARA, they used a few prompt patterns which are suggested in White, et al. [4]. They structured the prompts and the pipeline to automatically generate HARA results based on the function description of any automotive feature.

This work proposes a requirements analysis method for detecting missing requirements and minimizing sets of requirements. The proposed method utilizes an LLM and a prompt pattern suggested inWhite, et al. [4]. The feasibility and effectiveness of the proposed method has been manifested by being applied to the existing requirements, the same requirements as those in Anthony

© 2025 by the author; licensee Learning Gate
History: Received: 8 January 2025; Revised: 28 February 2025; Accepted: 28 February 2025; Published: 11 March 2025
* Correspondence: hsson@joongbu.ac.kr

and Adonsou [2]. The author organized the rest of this paper as follows: Section 2 presents a comprehensive literature review on the related works. The proposed requirements analysis method is described in Section 3. The application results are presented in Section 4. Finally, Section 5 summarizes the findings from the application and proposes potential future works.

2. Related Works

This literature review examines research studies, methodologies, and tools addressing the missing and duplicated requirements issues, highlighting advancements in requirement engineering.

2.1. Missing Requirements

Missing requirements often arise due to incomplete stakeholder elicitation, ambiguous documentation, and inadequate domain knowledge. Literature suggests that missing requirements can lead to costly post-deployment fixes, system failures, and reduced customer satisfaction [5]. Researchers have developed Natural Language Processing (NLP) techniques to identify missing functional and non-functional requirements by analysing requirement patterns and gaps [6]. Ontologies help formalize domain knowledge, allowing automated reasoning tools to detect incomplete specifications [7]. Model-driven approaches use graphical and formal models to verify completeness, ensuring all essential requirements are captured [8].

2.2. Redundant Requirements

Redundant requirements typically emerge from multiple sources contributing similar specifications, lack of standardized documentation, or poor traceability mechanisms. Studies indicate that redundancy increases development and maintenance costs while leading to inconsistencies [9]. Text mining and clustering techniques can detect similar requirement statements, helping analysts eliminate redundant elements [10]. Traceability techniques track requirement dependencies, identifying redundant specifications that overlap with existing requirements [11]. As machine learning approaches, supervised and unsupervised learning methods help identify redundant patterns in large requirement repositories, improving efficiency [12].

2.3. Tools and Lessons Learned

Requirements engineering process produces requirements artefacts that can be managed by software engineering tools [13-15]. Modern software engineering tools integrate AI-driven analytics, automated requirement validation, and model-based verification techniques. Tools such as IBM Rational DOORS, RequisitePro, and JIRA provide functionalities to detect missing and redundant requirements through automated analysis and stakeholder collaboration. The literature suggests that addressing missing and redundant requirements requires a combination of NLP, ontology-driven models, and AI-based techniques [16]. As software systems grow in complexity, leveraging automated tools and structured methodologies in SRA will be essential for improving requirement completeness and efficiency. Future research should explore hybrid approaches that combine multiple techniques to enhance requirement accuracy further.

3. Proposed Requirements Analysis Method

As mentioned inAnthony and Adonsou [2] software requirements analysis is the process of ensuring that software requirements accurately break down the system requirements. During the process, it is checked if the software requirements are complete, unambiguous, correct, verifiable, concise, consistent, feasible, traceable and necessary. Absent, duplicated or inconsistent requirements are frequent problems in requirements specifications, thus a review to determine if they exist in the specifications is an essential activity of the requirements analysis. For the review, this work proposes an efficient method to find out missing requirements and lessen the possibility of duplicated or contradictory requirements through minimizing requirements. The proposed method utilizes an LLM and prompt engineering. The author has taken notice that prompt patterns play a crucial role in successful prompt engineering. Thus the author investigated all the prompt patterns suggested in White, et al. [4] and reached a conclusion that the Recipe Pattern is suitable to the purpose of the proposed requirements analysis method.

3.1. The Recipe Pattern

According to White, et al. [4] the Recipe Pattern sets limitations to produce a structured sequence of steps using given partial "ingredients," which need to be organized systematically to reach a specified objective. Fundamental contextual statements of this pattern are as follows:

- I would like to achieve X
- I know that I need to perform steps A, B, C
- Provide a complete sequence of steps for me
- Fill in any missing requirements
- Identify any unnecessary steps

The first statement helps the LLM to focus on the overall goal. The second statement presents a partial set of steps that the user intends to incorporate into the complete recipe. The third statement informs the LLM that the objective is to generate a fully ordered sequence of steps. The next statement “Fill in any missing steps” prompts the LLM to complete the recipe independently by making necessary choices for missing steps, rather than merely identifying additional required information. Lastly, the final statement helps identify inaccuracies in the user's original request, ensuring the final recipe is optimized and efficient.

3.2. Logical Use of the Recipe Pattern

While the Recipe Pattern focuses on finding and filling missing steps of the recipe, detecting and filling logically missing requirements of the requirements specification shall be involved in the requirements analysis. Therefore, in order to deal with the incompleteness and redundancy of requirements, fundamental contextual statements of the Recipe Pattern are modified as follows:

- I would like to achieve logically complete set of requirements
- I know the requirements X, Y, Z
- Give me a comprehensive list of requirements
- Complete any missing requirements
- Point out any redundant requirements

The first statement shows that the overall goal is to achieve logically complete set of requirements. It should be noted that the word steps in the Recipe Pattern has been changed into the word requirements.

4. Experimental Application

The feasibility and effectiveness of the proposed method has been manifested through a comparative study. The proposed method has been applied to the existing requirements, the same requirements as those in [2]. The requirements, which describe the feasible responses of International Space Station to avoid the collision with earth debris and solar debris, are as follows:

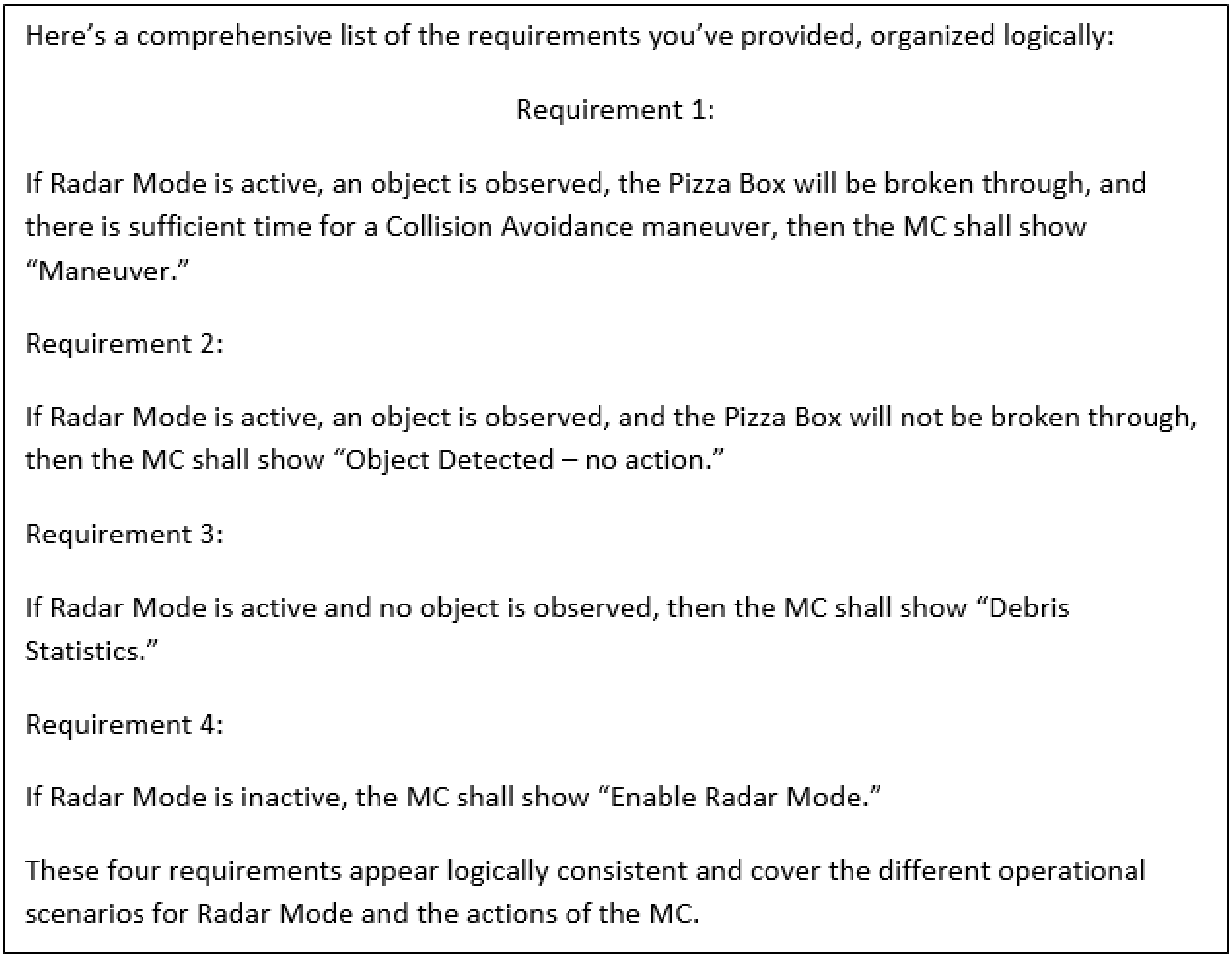
- If Radar Mode is active, an object is observed, the Pizza Box will be broken through, and there is sufficient time for a Collision Avoidance maneuver, then MC shall show “Maneuver.”

- If Radar Mode is active, an object is observed, Pizza Box will not be broken through, then MC shall show “Object Detected – no action.”
- If Radar Mode is active, no object is observed, then MC shall show Debris Statistics.
- If Radar Mode is inactive, MC shall show “Enable Radar Mode.”

Using the modified Recipe Pattern, the following prompt was firstly provided to the LLM, ChatGPT:

I would like to achieve logically complete set of requirements. I know the first requirement is that If Radar Mode is active, an object is observed, the Pizza Box will be broken through, and there is sufficient time for a Collision Avoidance maneuver, then MC shall show “Maneuver.” The second requirement is that If Radar Mode is active, an object is observed, Pizza Box will not be broken through, then MC shall show “Object Detected – no action.” And the third requirement is that If Radar Mode is active, no object is observed, then MC shall show Debris Statistics. The fourth requirement is that If Radar Mode is inactive, MC shall show “Enable Radar Mode.” Give me a comprehensive list of requirements.

This prompt used the first three statements of the pattern. ChatGPT provided the output as shown in Figure 1.



Here’s a comprehensive list of the requirements you’ve provided, organized logically:

Requirement 1:

If Radar Mode is active, an object is observed, the Pizza Box will be broken through, and there is sufficient time for a Collision Avoidance maneuver, then the MC shall show “Maneuver.”

Requirement 2:

If Radar Mode is active, an object is observed, and the Pizza Box will not be broken through, then the MC shall show “Object Detected – no action.”

Requirement 3:

If Radar Mode is active and no object is observed, then the MC shall show “Debris Statistics.”

Requirement 4:

If Radar Mode is inactive, the MC shall show “Enable Radar Mode.”

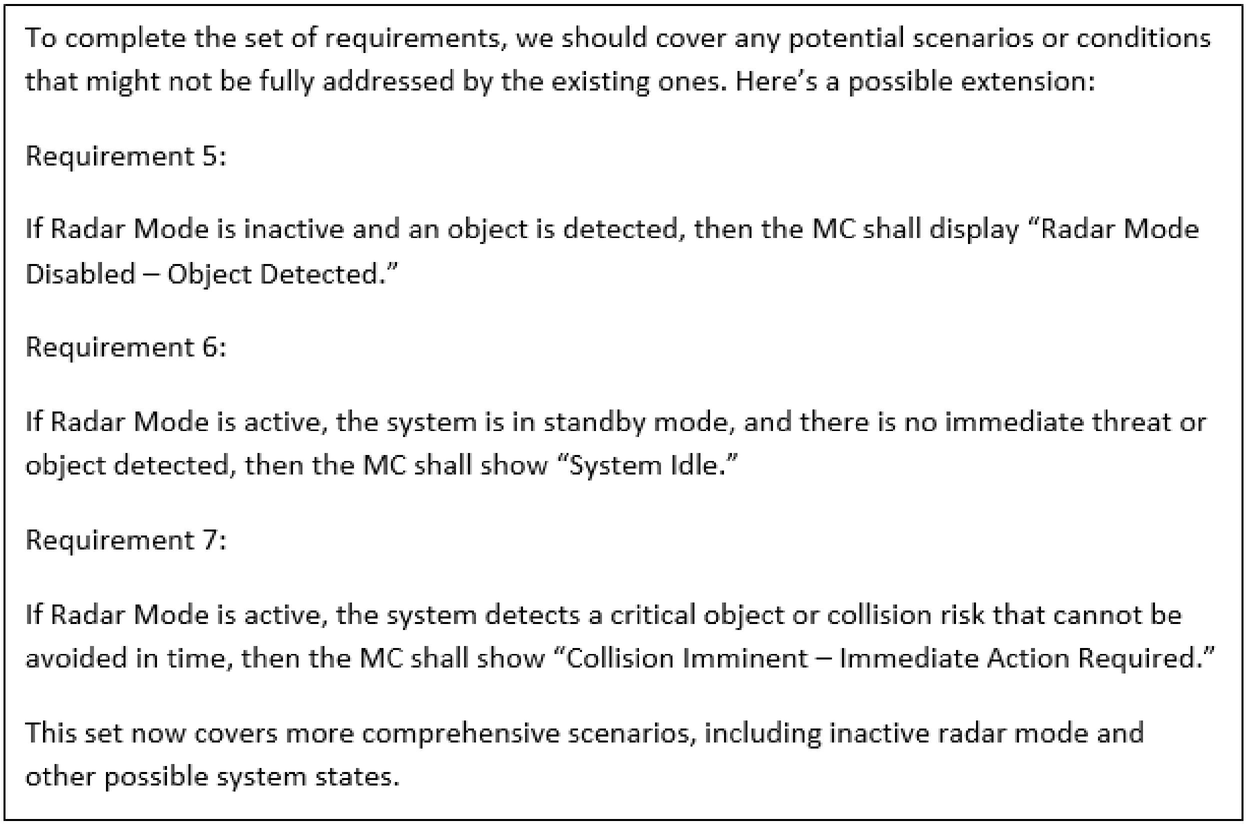
These four requirements appear logically consistent and cover the different operational scenarios for Radar Mode and the actions of the MC.

Figure 1.
The first output from ChatGPT.

The second prompt, which corresponds to the fourth statement of the pattern, was provided as follows:

Complete any missing requirements.

For this prompt, ChatGPT provided the output as shown in Figure 2. According to the output, all logical conditions are covered with the requirements, in which missing requirements are filled, ensuring that the MC displays appropriate messages for every possible scenario.



To complete the set of requirements, we should cover any potential scenarios or conditions that might not be fully addressed by the existing ones. Here's a possible extension:

Requirement 5:

If Radar Mode is inactive and an object is detected, then the MC shall display "Radar Mode Disabled – Object Detected."

Requirement 6:

If Radar Mode is active, the system is in standby mode, and there is no immediate threat or object detected, then the MC shall show "System Idle."

Requirement 7:

If Radar Mode is active, the system detects a critical object or collision risk that cannot be avoided in time, then the MC shall show "Collision Imminent – Immediate Action Required."

This set now covers more comprehensive scenarios, including inactive radar mode and other possible system states.

Figure 2.
The second output from ChatGPT.

The third prompt, which corresponds to the final statement of the pattern, was provided as follows:

Point out any redundant requirements.

For this prompt, ChatGPT provided the output as shown in Figure 3. ChatGPT successfully identified redundant requirements and streamlined the set of requirements. The resulted set of requirements are shown in Figure 4 and is the same as that of Anthony and Adonsou [2] which results from the K-Map based requirements analysis method.

5. Concluding Remarks

In this work, the author has shown that LLMs can be utilized to identify missing requirements and assist in reducing the set of requirements. Prompt engineering, especially using prompt patterns, makes it easy and efficient to use LLMs for the requirements analysis. Finding out that the Recipe Pattern is suitable to the requirements analysis process, the Recipe Pattern is modified for the purpose of the proposed requirements analysis method. The feasibility and efficiency of the proposed requirements analysis method has been demonstrated by the experimental application to the requirements, which describe the feasible responses of International Space Station to avoid the collision with earth debris and solar debris. ChatGPT successfully identified missing requirements and reduced redundant requirements. Furthermore it also streamlined the set of requirements successfully. Since incorrect and inconsistent requirements may cause safety critical failure of software system, future research is recommended to develop a requirements analysis method to detect incorrect and inconsistent requirements using LLMs.

Transparency:

The author confirms that the manuscript is an honest, accurate, and transparent account of the study; that no vital features of the study have been omitted; and that any discrepancies from the study as planned have been explained. This study followed all ethical practices during writing.

Acknowledgments:

This paper was supported by Joongbu University Research & Development Fund, in 2024.

Copyright:

© 2025 by the authors. This open-access article is distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

References

[1] L. Williams, "Two deaths, multiple infections linked to sterility failure of sublingual sensors," *Journal of Clinical Engineering*, vol. 30, no. 1, pp. 12-13, 2005. https://doi.org/10.1097/00004669-200501000-00020

[2] C. Anthony and Y. Adonsou, "Using karnaugh maps in software requirements analysis," *Journal of System Safety*, vol. 54, no. 1, pp. 22-39, 2018. https://doi.org/10.56094/jss.v54i1.82

[3] Nouri, "Welcome your new ai teammate: On safety analysis by leashing large language models," presented at the Conference on AI Engineering Software Engineering for AI (CAIN 2024), April 14–15, 2024, Lisbon, Portugal, 2024.

[4] J. White *et al.*, "A prompt pattern catalog to enhance prompt engineering with chatgpt," *arXiv preprint arXiv:2302.11382*, 2023. https://doi.org/10.48550/arXiv.2302.11382, 2023

[5] I. Sommerville, *Software engineering*, 9th ed. Pearson, 2011.

[6] A. Ferrari *et al.*, "Detecting requirements defects with NLP patterns: An industrial experience in the railway domain," *Empirical Software Engineering*, vol. 23, no. 6, pp. 3684-3733, 2018.

[7] J. Li, W. Zhang, and Y. Wang, "Ontology-based requirement completeness checking," *Requirements Engineering Journal*, 2019.

[8] B. Kovitz, *Practical software requirements: A manual of content and style*. Addison-Wesley, 2015.

[9] K. Pohl, *Requirements engineering: Fundamentals, principles, and techniques*. New York: Springer, 2010.

[10] E. Knauss, D. Damian, and K. Schmid, "Detecting duplicate and inconsistent requirements using clustering techniques," *Empirical Software Engineering*, 2017.

[11] M. Glinz, "A glossary of requirements engineering terminology," *Standard Glossary of the Certified Professional for Requirements Engineering (CPRE) Studies and Exam, Version*, vol. 1, p. 56, 2011.

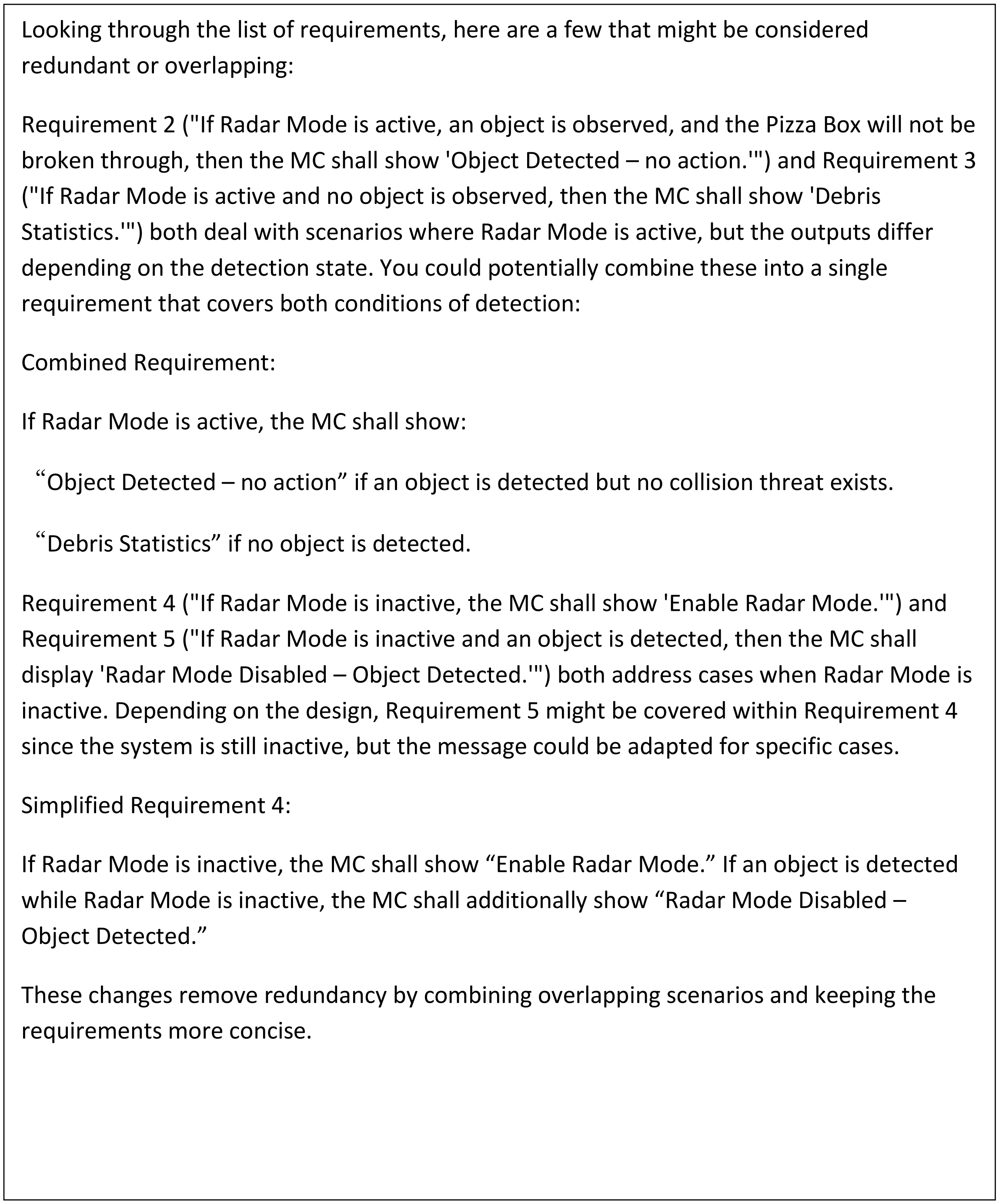
[12] M. Abadeer and M. Sabetzadeh, "Machine learning-based estimation of story points in agile development: Industrial experience and lessons learned," presented at the In 2021 IEEE 29th International Requirements Engineering Conference Workshops (REW) (pp. 106-115). IEEE, 2021.

[13] D. Zowghi and V. Gervasi, "The three dimensions of requirements engineering: A framework and its applications," *Information and Software Technology*, 2003.

[14] B. H. Cheng and J. M. Atlee, "Research directions in requirements engineering," *Future of Software Engineering*, pp. 285-303, 2007. https://doi.org/10.1109/fose.2007.17

[15] J. Cleland-Huang, O. Gotel, and A. Zisman, *Software and systems traceability*. Springer, 2012.

[16] S. Çevikol and F. B. Aydemir, "Detecting inconsistencies of natural language requirements in satellite ground segment domain," Retrieved: https://ceur-ws.org/Vol-2376/NLP4RE19_paper15.pdf, 2019.



Looking through the list of requirements, here are a few that might be considered redundant or overlapping:

Requirement 2 ("If Radar Mode is active, an object is observed, and the Pizza Box will not be broken through, then the MC shall show 'Object Detected – no action.'") and Requirement 3 ("If Radar Mode is active and no object is observed, then the MC shall show 'Debris Statistics.'") both deal with scenarios where Radar Mode is active, but the outputs differ depending on the detection state. You could potentially combine these into a single requirement that covers both conditions of detection:

Combined Requirement:

If Radar Mode is active, the MC shall show:

“Object Detected – no action” if an object is detected but no collision threat exists.

“Debris Statistics” if no object is detected.

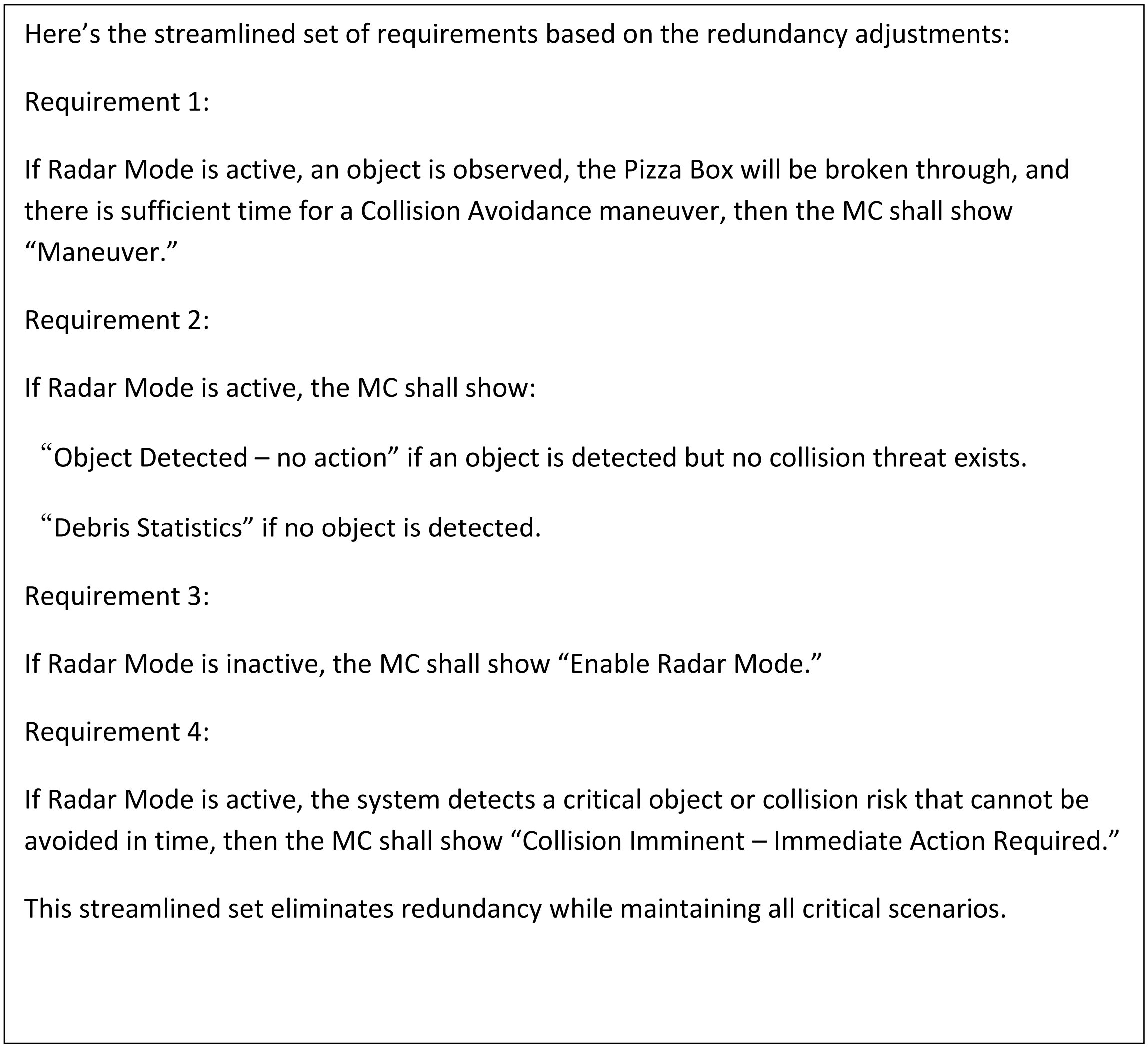
Requirement 4 ("If Radar Mode is inactive, the MC shall show 'Enable Radar Mode.'") and Requirement 5 ("If Radar Mode is inactive and an object is detected, then the MC shall display 'Radar Mode Disabled – Object Detected.'") both address cases when Radar Mode is inactive. Depending on the design, Requirement 5 might be covered within Requirement 4 since the system is still inactive, but the message could be adapted for specific cases.

Simplified Requirement 4:

If Radar Mode is inactive, the MC shall show “Enable Radar Mode.” If an object is detected while Radar Mode is inactive, the MC shall additionally show “Radar Mode Disabled – Object Detected.”

These changes remove redundancy by combining overlapping scenarios and keeping the requirements more concise.

Figure 3. The third output from ChatGPT



Here’s the streamlined set of requirements based on the redundancy adjustments:

Requirement 1:

If Radar Mode is active, an object is observed, the Pizza Box will be broken through, and there is sufficient time for a Collision Avoidance maneuver, then the MC shall show “Maneuver.”

Requirement 2:

If Radar Mode is active, the MC shall show:

“Object Detected – no action” if an object is detected but no collision threat exists.

“Debris Statistics” if no object is detected.

Requirement 3:

If Radar Mode is inactive, the MC shall show “Enable Radar Mode.”

Requirement 4:

If Radar Mode is active, the system detects a critical object or collision risk that cannot be avoided in time, then the MC shall show “Collision Imminent – Immediate Action Required.”

This streamlined set eliminates redundancy while maintaining all critical scenarios.

Figure 4. The set of requirements streamlined by ChatGPT