


Towards an automatic contradiction detection in requirements engineering

Alexander Elenka Gärtner  and Dietmar Göhlich

Technische Universität Berlin, Germany

 a.e.gaertner@outlook.de

Abstract

This paper presents a novel method for automatic contradiction detection in requirements engineering using a hybrid approach combining formal logic with Large Language Models (LLMs), specifically GPT-3. Our three-phase process detects contradictions by identifying conditionals and pseudo-grammatical elements, and employing LLMs for nuanced contradiction detection. Tested extensively, including on a real-world electric bus project, our method achieved 99% accuracy and 60% recall. This approach significantly reduces manual effort, enhances quality, and is scalable for future advancements.

Keywords: *requirements management, artificial intelligence (AI), design automation*

1. Introduction

In product development, it is essential to define a complete, unambiguous, and contradiction-free target system (Bender and Gericke 2021). However, the related documents often contain errors in the form of contradictions. Their occurrence can be attributed to the inherent complexity of the requirements specification process, encompassing several thousand individual requirements derived through interdisciplinary collaboration. Furthermore, these requirements are articulated in natural language (Luisa et al. 2004), which introduces an additional layer of complexity due to potential ambiguities and inconsistencies arising from the diverse perspectives and interpretations of multiple stakeholders involved (Babcock 2007).

Identifying and rectifying these contradictions manually presents considerable challenges and drawbacks. The manual correction process is time-consuming and incurs high costs due to the extensive effort required to scrutinize each requirement to detect potential contradictions. Moreover, the implications of not promptly addressing these contradictions can give rise to complications in subsequent stages of product development. Failure to resolve contradictory requirements can lead to misunderstandings, conflicts, and inefficiencies during the design, implementation, and testing phases, ultimately hindering the successful realization of the intended product or system (Ehrlenspiel and Meerkamm 2017).

Thus, the detection and resolution of contradictions in complex requirements documents emerge as a critical area of research and development within the field of product development. By devising automated methods and tools to identify and address these contradictions, it becomes possible to mitigate the costs, complexities, and risks associated with manual error correction. Additionally, automating this process enables the timely identification of contradictions, allowing for their resolution during the early stages of product development, thereby preventing potential complications and improving the overall efficiency and effectiveness of the development process.

This paper presents a comprehensive framework for automatic contradiction detection in requirements engineering, comprising an exploration of various categories of contradictions, the identification of conditionals in the requirement definition, and a software implementation of the method based on formal logic and Large Language Models. Furthermore, we discuss the integration of automated identification into the development processes.

2. Related work

Most studies that combine requirements engineering (RE) with natural language processing (NLP) consist of solution proposals evaluated through laboratory experiments or example applications (Zhao et al. (2021)). In contrast, only a small percentage (7%) of these studies undergo evaluation in industrial settings.

Marneffe et al. (2008) suggest a definition for contradictions in language. However, their emphasis is on implicit contradictions rather than explicit ones. Sentences like 'Sally sold a boat to John' and 'John sold a boat to Sally' are labeled as contradictory. However, in requirements engineering, this scenario does not necessarily indicate a contradiction, as it is possible for Sally to sell a boat to John while John simultaneously sells another boat to Sally.

Li et al. (2017) state that traditional context-based word embedding learning algorithms are ineffective for mapping contrasting words. They developed a neural network to learn contradiction-specific word embeddings that can separate antonyms to address this issue. However, their emphasis is not on explicit contradictions, e.g.: 'Some people and vehicles are on a crowded street' and 'Some people and vehicles are on an empty street' (Li et al. 2017). While these sentences contrast with each other, they might still simultaneously be correct.

Heitmeyer et al. (1996) introduce a technique called consistency checking for detecting errors in requirements specifications using formal logic. Their paper analyzes formal requirements, called SCR, and addresses issues like type errors, non-determinism, missing cases, and circular definitions. However, the requirements are not written using everyday language. Instead, they are expressed in a highly structured manner, where each requirement component has a predetermined category that it must be placed in during the writing process. Our method aims at requirements written more intuitively and naturally while still being formal.

Gervasi and Zowghi (2005) explore using formal logic and natural language parsing techniques to identify inconsistencies in requirements. The authors propose a method that automatically discovers and addresses these logical contradictions using theorem-proving and model-checking techniques. By translating the requirement into a set of logic formulae, they can detect contradictions such as α and its negation $\neg \alpha$. Hunter and Nuseibeh (1998) suggest utilizing formal logic, which records and tracks the information, enabling the tracking of inconsistent information by propagating labels and their associated data. While they do not only focus on negating assumptions like Gervasi et al., Hunter et al. can only detect conflicts if at least a part of the argument is negated, such as α and $\neg \alpha$. Therefore, Gervasi et al. and Hunter et al. cover an essential part of the issue addressed in our work, namely contradictories (see Figure 1). However, they do not consider contraries and subalterns, such as 'the car must be red' and 'the car must be blue' or 'it must be under 20' and 'it must be under 10'.

3. Datasets

Our study employed three distinct datasets (Gärtner and Göhlich 2023). Dataset 1, exclusively curated for this study, played a pivotal role as an initial testing suite during the development of our methodology. This dataset encompassed a comprehensive range of requirement pairs, including contradicting and non-contradicting ones, as a robust foundation for our research.

Datasets 2 and 3, on the other hand, were sourced from a recent real-world electric bus project, which aimed to establish a modular system to replace conventional bus powertrains with climate-friendly electric alternatives. Details about the vehicle can be found online¹. The importance of requirements in the context of electric bus development is extensively discussed in the publication 'Design of urban

¹ <https://www.iav.com/iavcars/elcty/>

electric bus systems' by Göhlich et al. (2018). Typically, real-world specification sheets remain confidential, precluding public access. Nonetheless, we have taken diligent steps to anonymize 210 requirement pairs and have made them accessible online².

Dataset 2 comprises a collection of 1071 requirement pairs, thoroughly verified and categorized for contradictions. Consequently, it was used to validate our proposed method. Meanwhile, Dataset 3 encompasses a substantial 3916 requirement pairs, which, while not manually checked, served a dual purpose: to demonstrate the scalability and effectiveness of our methodology with large datasets and provide a basis for comparing our hybrid approach and the GPT-3 model.

4. Method

Our method is structured into three phases. The initial phase involves an in-depth exploration of various categories of contradictions that may manifest within requirements documents. Subsequently, the second phase introduces a rule-based methodology to identify conditionals and pseudo-grammatical elements embedded within the requirements. Finally, the third phase integrates Large Language Models (LLMs) into the process, leveraging the combined strengths of formal logic and LLMs. While formal logic excels in discerning conditions and their implications within requirements, LLMs demonstrate prowess in deciphering the subtleties of natural language and identifying nuanced contradictions that may not be immediately apparent. This synergistic fusion of two methodologies forms a robust and automated tool for identifying contradictions within requirements. Crucially, the approach is underpinned by the theoretical groundwork established in the initial phase, which enables a systematic approach to identifying and addressing various types of contradictions, each with varying degrees of criticality. We adopted a frozen model approach using GPT3. Utilizing this version ensured that no updates would be applied to the model during our experiments, providing consistency in its behavior.

4.1. Syntax and semantics of requirements and taxonomy of contradicting requirements

The initial phase of our approach introduces a formal logic-based methodology designed for identifying and categorizing contradictions within requirements documents. Distinguishing itself from existing literature, our approach does not rely on classification for a specific, limited dataset; instead, it adheres to a widely accepted and rigorously tested systematic model. Therefore, we draw upon the logical philosophy of Aristoteles (1986), as elaborated in our previous paper (Gärtner et al. 2022). Aristotle's logic, known as term logic, traditional logic, or formal logic, centers on the law of non-contradiction (LNC) (Horn 2018), forming the basis for our understanding. Figure 1 illustrates various types of contradictions relevant to RE.

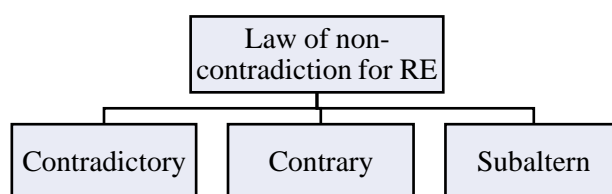


Figure 1. Contradictions relevant to requirements engineering (Gärtner et al. 2022)

Subsequently, a customized classification system explicitly tailored for requirements engineering was devised to pinpoint distinct types of contradictions: Contradictories, Contraries, and Subalterns, each further subdivided into Simplex, Idem, and Alius subcategories. Contradictory opposites are mutually exhaustive and mutually inconsistent. Contrary opposites are also mutually inconsistent but not exhaustive. The statement 'some people are sick' is the subaltern of 'everybody is sick' (superaltern). Simplex (lat. = simple): without conditions. Idem (lat. = same): with identical conditions. Alius (lat. = different): with different conditions.

² <https://swarm-engineer.me/2023/contradiction-detection/>

According to [Karlova-Bourbonus \(2019\)](#), it is essential to acknowledge that, in addition to contradictories, contraries must also be considered. Contraries, though often categorized as contradictions, are distinct and should be differentiated. Contradictory opposites, such as "he is sick" and "he is not sick," are all-encompassing and mutually exclusive. One of these statements must be true, and the other false, and vice versa. They cannot both be simultaneously true or false. On the other hand, contrary opposites, like "it is black" and "it is white," are also inconsistent with each other, but not exhaustively so. While they cannot be simultaneously true, they can be false concurrently.

The identification of the specific category necessitates the answering of a set of seven questions.

1. Question 1: This question determines whether the variables of effect one and effect two are identical or if one set of variables is a subset of the other, establishing the potential for an LNC contradiction. Formal logic was used to answer this question.
2. Question 2: This question examines the inclusivity between actions, determining if one condition subsumes the other. However, this question was not processed in the implementation due to limitations in the mathematical abilities of the current language models.
3. Question 3: This question identifies mutual exclusivity between two actions, detecting contradictory opposing actions. Formal logic and GPT-3 were used to assess similarity and determine if the actions were mutually exclusive.
4. Question 4: The fourth question assesses mutual inconsistency between two actions, identifying contrary opposing actions. GPT-3 was utilized to determine if the actions have the potential to contradict each other.
5. Question 5: This question determines the existence of a condition, aiming to eliminate potential Simplex-contradictions. The model of the second phase was employed for this condition detection.
6. Question 6: The sixth question identifies if the first condition is equivalent to the second one to detect Idem-contradictions requiring the same condition. Formal logic and string comparison were utilized to assess similarity.
7. Question 7: The seventh question determines if the first and second conditions can co-occur. GPT-3 assessed the theoretical possibility of both conditions happening simultaneously.

4.2. Automation of the detection of syntax, semantics, and taxonomy

The second phase of our approach is dedicated to the automated detection of sentence constituents, which is crucial for dissecting the intricate structures within requirements: specifically, conditions and effects as well as actions and variables. This step is instrumental in parsing these elements, forming the backbone of our methodology to address all 7 questions posed by our model.

For instance, in the requirement 'If the threshold is reached, the controller must limit the speed decay,' 'If the threshold is reached' and 'the controller must limit the speed decay' illustrate the condition and effect, respectively. 'The threshold' (variable) and 'is reached' (action) are identified within the condition, while 'the controller' (variable) and 'must limit the speed decay' (action) within the effect, demonstrating how variables and actions are integral to understanding requirements. Essentially, the variable signifies the central entity involved, while the action signifies the expected outcome.

It is imperative to emphasize that conditional statements, as encountered in requirements, do not inherently imply causality. To illustrate, consider the example of a car's requirement: the presence of wheels is a condition for its movement, while excessive torque applied to those wheels constitutes the cause. The resulting movement itself is referred to as the effect.

Our previous study ([Gärtner et al. 2023](#)) found that grammatical models incorporating grammatical rules, trigger words, and part-of-speech tags exhibit better suitability for identifying conditions than machine learning methods. To validate our findings, we evaluated a sample of 1,861 requirements. We compared the performance of a grammatical model against two machine learning models. Our dataset comprised confidential requirements sourced from an electric bus project and publicly available requirements from Fischbach ([Fischbach et al. 2020](#)). The results of this analysis indicate that the grammatical model outperforms machine learning models in identifying conditional requirements and their constituents.

4.3. Automated detection of contradicting requirements

Based on the above findings, we integrated formal logic and Large Language Models (LLMs) to achieve fully automated contradiction detection. This comprehensive approach is called 'ALICE,' which stands for Automated Logic for Identifying Contradictions in Engineering.

ALICE involves the practice of prompt engineering, a process that encompasses the meticulous design and fine-tuning of prompts aimed at eliciting specific behaviors from LLMs, notably the identification of contradictions. This seven-step procedure is scalable and adaptable to accommodate future advancements in the field, offering a modular framework for assessing the coherence of arguments. ALICE was developed as a Python code structured in 3 parts, as shown in Figure 2.

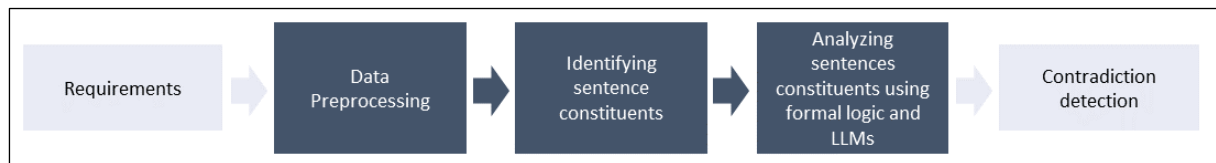


Figure 2. Code structure

5. Results

We conducted a comparative analysis with a purely LLM-based method to validate ALICE's performance. We used a dataset of 1,071 requirements pairs obtained from a recent real-world electric bus project. The pairs were manually checked and labeled for contradictions in order to conduct a validation of both methods.

The hybrid method achieved an accuracy of 99% and a recall of 60%. On the other hand, the LLM-only method achieved an accuracy of 97% but had a recall of 0%. Due to unbalanced data, the accuracy alone may not be a highly relevant metric (Powers 2011). The recall, which measures the proportion of correctly identified positives, provides a better understanding of the results. Accordingly, the hybrid method detected 60% of all contradictions, whereas the LLM method did not detect any. The confusion matrices are shown in Table 1.

Table 1. Confusion matrices for the first dataset

hybrid	0 expected	1 expected
0 calculated	1044	9
1 calculated	1	17

GPT3	0 expected	1 expected
0 calculated	1041	26
1 calculated	4	0

A case where ALICE failed to accurately detect a contradiction between two requirements because of incorrectly interpreting the variables involved is highlighted below:

1. 'If the heater stage CbnHeatg_StgOfHeatg > 0, then the PTC activation signal CbnHeatg_ActvnFlgOfHeatr = 1.'
2. 'The signal CbnHeatg_ActvnFlgOfHeatr is overridden by the parameter CbnHeatg_SubValForActvnFlgOfHeatr_C, if the parameter CbnHeatg_SubValForActvnFlgOfHeatrToSwt_C == 1.'

These requirements contradict each other because they mandate different actions for the same variable (CbnHeatg_ActvnFlgOfHeatr = 1 versus CbnHeatg_ActvnFlgOfHeatr equaling a parameter) under potentially overlapping conditions. The contradiction was not detected due to the models failure to recognize that both requirements referred to the same variable (CbnHeatg_ActvnFlgOfHeatr). ALICE

misinterpreted the variable in the first requirement as 'PTC' and did not realize that both requirements discussed the same signal.

Additionally, we analyzed a similar dataset comprising 3,916 requirements combinations, which took approximately 4 hours to complete. The results were subjected to random spot-checks, rendering the presentation of a confusion matrix impractical. It is important to note that our primary focus was not on speed or code optimization during this analysis. Techniques such as parallel processing were not leveraged, which presents a significant opportunity for efficiency improvements in future iterations. Despite this, the analysis effectively demonstrated the method's capacity to handle large datasets and highlighted the hybrid method's superiority over the exclusive use of GPT-3.

This study also sheds light on the limitations of using LLMs and symbolic AI for detecting contradictions in technical documents. LLMs may generate errors when confronted with insufficient data for precise requirement analysis, while factors like filler words, comma placement errors, and neologisms can mislead formal logic. It is advisable to employ active sentence structures and established terminology in technical writing to enhance precision and minimize ambiguity in results.

In summary, our method, consisting of seven questions, can detect formal contradictions between requirements. It is scalable and can be adapted to future advancements in the field, providing a modular way to evaluate the consistency requirements specification.

6. Process integration

This section discusses the potential integration of ALICE, the Automated Logic for the Identification of Contradictions in Engineering, into development processes.

According to [Bender and Gericke \(2021\)](#), in product development, three primary targets are crucial: deadlines, cost constraints, and product-related objectives. While this paper exclusively focuses on product-related objectives, it is conceivable that future methods, shaped by the insights gained here, could extend their horizon to evaluate cost- and schedule-related targets.

Numerous design process models are available to address these objectives to assist engineers in planning, documenting, finding solutions, and making decisions in their work. Despite variations in terminology and specifics, these models typically share common stages in the design process ([Gericke and Blessing 2012](#); [Wynn and Clarkson 2018](#)).

One critical phase in these models is the initial development stage, wherein the design task is clarified, resulting in a requirements list, sometimes referred to as a requirements specification ([Eisenbart et al. 2011](#); [Gericke and Blessing 2012](#)). This list encompasses essential functionalities, influences, constraints, and dependencies derived from stakeholder demands, market conditions, and other factors ([Eisenbart et al. 2011](#)). While various methods and checklists assist in identifying requirements, they often prioritize completeness over consistency, posing challenges in evaluating coherence and conflicting design objectives due to limited solution principles or details ([Göhlich et al. 2021](#)).

Many design process models suggest that this task is considered complete once the initial requirements list is established, despite emphasizing the importance of continuous revision and refinement. Requirements are dynamic and evolve alongside the understanding of the design problem, requiring continuous engagement with the requirements list throughout the design process ([Maher and Poon 1996](#); [Gericke et al. 2013](#)).

The revised VDI 2221 guideline Design of technical products and systems provides a generic process model representing a wide range of mechanical and mechatronic products (VDI 2221 Blatt 1:2019-11). One of the main aspects of this model is its intended flexibility, which addresses the need to adapt high-level procedural process models to the specific context of a company/design team. The model, shown in Figure 3, emphasizes that establishing and managing requirements is a continuous process through all four phases of product design: Task Clarification, Conceptual Design, Embodiment Design, and Detail Design.

The project's objectives or issues are delineated in the Task Clarification phase, which marks the initial stage of the design process. However, the requirements often lack detail and thorough elaboration during this early stage. They may be broad, general, or even described solely through use cases. This implies that while there is a foundational understanding of what needs to be achieved, the specifics, nuances, or detailed criteria may not have been defined yet, posing challenges to the application of ALICE.

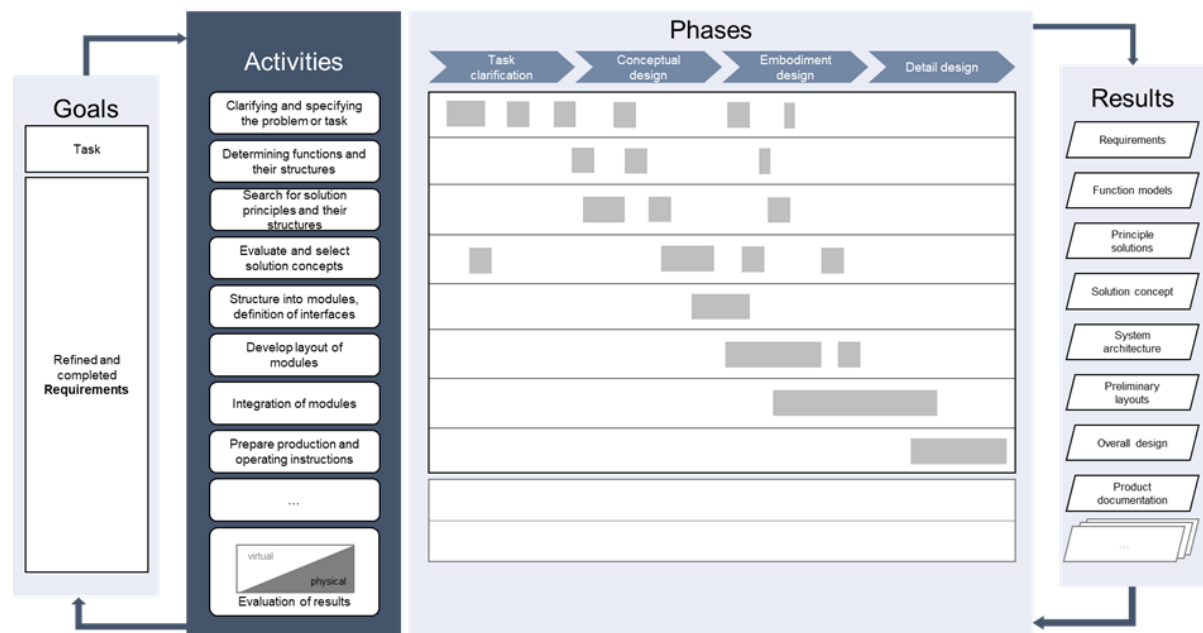


Figure 3. Process model according to VDI 2221 (Göhlich et al. 2021)

The outcomes of the Conceptual Design phase and, to some extent, the Embodiment Design phase can be summarized as the initial target system. This target system is a benchmark for assessing the project's success for all stakeholders in all development phases. Evaluating consistency and conflicting design objectives can be challenging at this stage due to the absence of solution principles or detailed information. While a general contradiction detection tool would be beneficial here, ALICE primarily focuses on identifying contradictions within technical specifications.

The Detailed Design phase delves into the product's technical specifications, materials, components, and manufacturing processes (DIN ISO 9000:2015). While our tool cannot enhance drawings and schematics, it can uncover contradictions in the specifications, potentially reducing the need for manual intervention. Furthermore, this phase can be categorized into two sub-processes, 'Developing Requirements' (also referred to as requirements engineering) and 'Working with Requirements' (also referred to as requirements management), each of which can be further delineated into specific activities, as depicted in Figure 4, according to Bender and Gericke (2021).

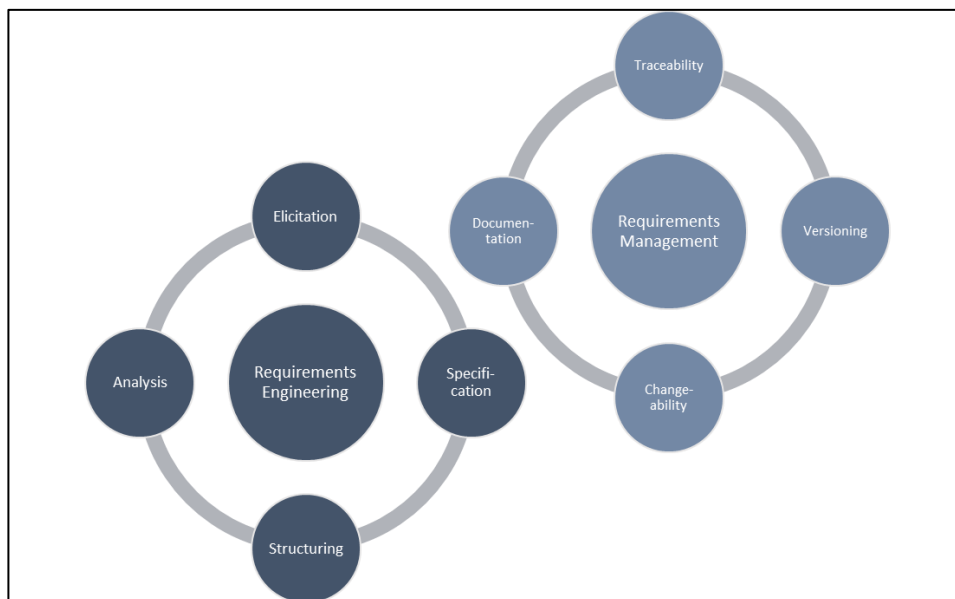


Figure 4. Tasks when developing requirements (based on Bender and Gericke (2021))

In the following, we will delve into the four constituent activities and their relevance to our conceptual tool:

- **Elicitation:** The tool could augment the requirement elicitation process by proactively identifying contradictions and inconsistencies early in development. This proactive detection aids in more precise and comprehensive requirement elicitation. However, findings obtained during the elicitation are often documented informally and not in clean requirements. It would, therefore, be necessary to check the elicitation results before deciding on the application of the tool.
- **Specification:** By identifying and resolving contradictions, the tool promises to improve requirement specification, ensuring that requirements are unambiguous, concise, and devoid of conflicts, thereby enhancing their quality. Nevertheless, specifying requirements is often a dynamic process in which they are subject to constant correction and editing. Since the specification continues to undergo regular changes, the tool's implementation might be premature at this stage.
- **Structuring:** Applying the tool in this context could facilitate more effective requirements structuring. By clarifying conflicting facets and supporting their reorganization into a coherent structure, the tool could assist in achieving improved requirement organization. However, the tool's results would not directly lead to automated structural insights. Therefore, it is not immediately applicable in this step. Further research might be valuable.
- **Analysis:** Among the various activities in requirement development, the 'Analysis' phase emerges as a natural and highly relevant domain for the application of the tool. This alignment is especially evident as it involves a detailed analysis of requirements to detect contradictions. The tool's analytical capabilities are specifically designed to uncover contradictions within requirements. Examining the requirement set offers a systematic means to pinpoint these issues in the development process. This proactive identification serves as the foundation for resolving contradictions, ensuring that the requirement set remains coherent, feasible, and aligned with project objectives.

Furthermore, complex development projects are mainly carried out in interdisciplinary collaboration within cooperation networks. Typically, the requirements for each level are managed in different documents for the overall product in *product specification books* and the subsystems in *component specification books* (Göhlich and Fay 2021). The partial results must be combined according to their logical and temporal dependencies to form the solution. "For complex systems in particular, this process involves a risk of error with regard to the consistency of the goals among themselves and thus also for the consistency [...] of the overall solution." (Bender and Gericke 2021). Our contradiction-detection tool can facilitate this process by identifying potential conflicts across multiple specifications and assisting engineers in recognizing cross-references spanning various domains, thereby promoting effective

7. Limitations

Regarding limitations, the methodology is tailored to handle controlled natural language requirements. It requires adherence to the principle of atomicity, meaning that requirements cannot be further broken down into smaller components. Filler words in requirements can introduce errors, and passive sentence constructions should be avoided for clarity. Proper comma placement is crucial to prevent erroneous condition detection. The use of standard sentence templates is recommended to maintain consistency in the application of the methodology.

Regarding threats to validity, further investigations and comparisons with additional datasets are necessary to account for various contextual factors such as the domain, development process, requirements engineering techniques, and technologies used. There is a concern about potential overfitting due to the methodology's initial development on a limited dataset. To address this, future research should strive for a more diverse and representative dataset to ensure the broader applicability and validity of the methodology.

8. Conclusion and outlook

This work contributes significantly towards automatic contradiction detection in requirements engineering, but further exploration and improvements are necessary.

A direction for future research could be to refine the hybrid approach, which combines formal logic and LLMs for contradiction detection. Specifically, exploring different machine learning models, such as neural networks or decision trees, in conjunction with formal logic and LLMs could lead to even higher accuracy and recall rates in contradiction detection, which would be highly beneficial for ensuring the consistency and correctness of requirements documents. Also, testing newer versions of GPT, such as GPT4, could yield other results.

Another area for future exploration is the development of custom models for specific domains or requirements. By training machine learning models on a specific domain (e. g., automotive), researchers could create highly tailored models optimized for detecting contradictions and other issues specific to that domain. This would significantly enhance the accuracy and effectiveness of automated requirement analysis in these fields.

Overall, combining different machine learning models and developing custom models is promising to improve the efficiency and effectiveness of requirements engineering. Further research in these areas is essential for realizing the full potential of automated requirement analysis.

Acknowledgments

We thank IAV GmbH for providing us with the requirements specifications for electric buses.

References

- Aristoteles (1986). *Metaphysik*. Schriften zur Ersten Philosophie. Übertr. u. hrsg. v. Franz F. Schwarz. Reclam.
- Babcock, Jonathan (2007). GOOD REQUIREMENTS ARE MORE THAN JUST ACCURATE. Available online at <https://practicalanalyst.com/good-requirements-are-more-than-just-accurate/> (accessed 5/20/2022).
- Bender, Beate/Gericke, Kilian (2021). Entwickeln der Anforderungsbasis: Requirements Engineering. In: Beate Bender/Kilian Gericke (Eds.). *Pahl/Beitz Konstruktionslehre*. Berlin, Heidelberg, Springer Berlin Heidelberg, 169–209.
- DIN ISO 9000:2015. *Qualitätsmanagementsysteme*, 2015.
- Ehrlenspiel, Klaus/Meerkamm, Harald (2017). *Integrierte Produktentwicklung*. Denkabläufe, Methodeneinsatz, Zusammenarbeit. 6th ed. München, Carl Hanser Verlag.
- Eisenbart, B./Gericke, K./Blessing, L. (2011). A framework for comparing design modelling approaches across disciplines. In: Culley, S. J, Hicks, B. J, et al. (Ed.). *Proceedings of the 18th International Conference on Engineering Design (ICED11)*, pp. 344–355.
- Fischbach, Jannik/Hauptmann, Benedikt/Konwitschny, Lukas/Spies, Dominik/Vogelsang, Andreas (2020). Towards Causality Extraction from Requirements. <https://doi.org/10.48550/arXiv.2006.15871>.
- Gärtner, A. E./Göhlich, D./Fay, T.-A. (2023). Automated Condition Detection in Requirements Engineering. In: *ICED23 Proceedings*, 707–716.
- Gärtner, Alexander Elenga/Fay, Tu-Anh/Göhlich, Dietmar (2022). Fundamental Research on Detecting Contradictions in Requirements: Taxonomy and Semi-Automated Approach. *Applied Sciences* 12 (15), 7628. <https://doi.org/10.3390/app12157628>.
- Gärtner, Alexander Elenga/Göhlich, Dietmar (2023). Contribution to an Automated Contradiction Detection in Complex Specification Sheets. <https://doi.org/10.21203/rs.3.rs-3384770/v1>.
- Gericke, Kilian/Blessing, L. (2012). An analysis of design process models across disciplines. In: D. Marjanovic/M. Storga/N. Pavkovic et al. (Eds.). *DESIGN 2012*. Proceedings of the 12th International Design Conference, May 21 - 24, 2012, Dubrovnik, Croatia. Zagreb, Fac. of Mechanical Engineering and Naval Architecture, pp. 171–180.
- Gericke, Kilian/Qureshi, A. J./Blessing, Lucienne (2013). Analyzing Transdisciplinary Design Processes in Industry: An Overview. In: Volume 5: 25th International Conference on Design Theory and Methodology; ASME 2013 Power Transmission and Gearing Conference, ASME 2013 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Portland, Oregon, USA, 04.08.2013 - 07.08.2013. American Society of Mechanical Engineers.
- Gervasi, Vincenzo/Zowghi, Didar (2005). Reasoning about inconsistencies in natural language requirements. *ACM Transactions on Software Engineering and Methodology* 14 (3), 277–330. <https://doi.org/10.1145/1072997.1072999>.

- Göhlich, Dietmar/Bender, Beate/Fay, Tu-Anh/Gericke, Kilian (2021). Product requirements specification process in product development. *Proceedings of the Design Society* 1, 2459–2470. <https://doi.org/10.1017/pds.2021.507>.
- Göhlich, Dietmar/Fay, Tu-Anh (2021). Arbeiten mit Anforderungen: Requirements Management. In: Beate Bender/Kilian Gericke (Eds.). *Pahl/Beitz Konstruktionslehre. Methoden und Anwendung erfolgreicher Produktentwicklung*, 9th ed. Berlin/Heidelberg, Springer Vieweg, 211–229.
- Göhlich, Dietmar/Fay, Tu-Anh/Jefferies, Dominic/Lauth, Enrico/Kunith, Alexander/Zhang, Xudong (2018). Design of urban electric bus systems. *Design Science* 4. <https://doi.org/10.1017/dsj.2018.10>.
- Heitmeyer, Constance L./Jeffords, Ralph D./Labaw, Bruce G. (1996). Automated consistency checking of requirements specifications. *ACM Transactions on Software Engineering and Methodology* 5 (3), 231–261. <https://doi.org/10.1145/234426.234431>.
- Horn, Laurence R. (2018). Contradiction. The Metaphysics Research Lab. Available online at <https://plato.stanford.edu/archives/win2018/entries/contradiction/> (accessed 4/21/2022).
- Hunter, Anthony/Nuseibeh, Bashir (1998). Managing inconsistent specifications. *ACM Transactions on Software Engineering and Methodology* 7 (4), 335–367. <https://doi.org/10.1145/292182.292187>.
- Karova-Bourbonus, Natali (2019). Automatic detection of contradictions in texts. Gießen, Universitätsbibliothek.
- Li, Luyang/Qin, Bing/Liu, Ting (2017). Contradiction Detection with Contradiction-Specific Word Embedding. *Algorithms* 10 (2), 59. <https://doi.org/10.3390/a10020059>.
- Luisa, Mich/Mariangela, Franch/Pierluigi, Novi Inverardi (2004). Market research for requirements analysis using linguistic tools. *Requirements Engineering* 9 (1), 40–56. <https://doi.org/10.1007/s00766-003-0179-8>.
- Maher, Mary Lou/Poon, Josiah (1996). Modeling Design Exploration as Co-Evolution. *Computer-Aided Civil and Infrastructure Engineering* 11 (3), 195–209. <https://doi.org/10.1111/j.1467-8667.1996.tb00323.x>.
- Marneffe, Rafferty, Manning (2008). Finding Contradictions in Text. USA. Available online at <https://nlp.stanford.edu/pubs/contradiction-acl08.pdf> (accessed 4/13/2022).
- Powers, David M. W. (2011). Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. *International Journal of Machine Learning Technology* 2:1, pp.37–63. <https://doi.org/10.48550/arXiv.2010.16061>.
- Wynn, David C./Clarkson, P. John (2018). Process models in design and development. *Research in Engineering Design* 29 (2), 161–202. <https://doi.org/10.1007/s00163-017-0262-7>.
- Zhao, Liping/Alhoshan, Waad/Ferrari, Alessio/Letsholo, Keletso J./Ajagbe, Muideen A./Chioasca, Erol-Valeriu/Batista-Navarro, Riza T. (2021). Natural Language Processing for Requirements Engineering. *ACM Computing Surveys* 54 (3), 1–41. <https://doi.org/10.1145/3444689>.