System Requirement Conflict Identification and Resolution using LLMs

Trung Kien Nguyen, Phong Tran, Quoc Co Luc
Swinburne University of Technology, Hawthorn, Victoria, Australia
Email: {104053642, 104334842, 103830572}@student.swin.edu.au

***Abstract*—System requirements often emerge from different stakeholders with diverse goals, priorities, challenges, motivations—leading to inconsistencies and conflicts that slow down system design. This project presents an automated framework for identifying and resolving conflicts using Large Language Models (LLMs). By utilizing the natural language understanding and reasoning capabilities of LLMs, the system extracts detailed requirements and usage contexts from multiple perspectives, detects contradictions across them, and proposes resolutions through guided strategies. The pipeline supports both functional and non-functional requirements and incorporates structured reasoning, persona-based simulation, and conflict identification analysis. As no existing solution offers fully automated conflict identification and resolution in requirements engineering, this research demonstrates a significant advancement toward intelligent, scalable, and inclusive software specification.**

***Index Terms*—Requirements Engineering, Requirements Clustering, Conflict Identification and Resolution,**

I. Introduction

In requirements engineering, one of the most persistent and complex challenges lies in reconciling conflicting needs and expectations from diverse stakeholders. As modern systems increasingly cater to heterogeneous user bases, it becomes critical to ensure that system requirements represent each user's perspective accurately, while avoiding internal inconsistencies and contradictions. This report investigates an automated framework for detecting and resolving such requirement conflicts—particularly at the user story level, using the capabilities of the State-of-the-art (SOTA) Large Language Models (LLMs).

To contextualize our study, we explore the ALFRED system—a virtual assistant platform designed to support older adults in living independently while enabling caregivers, medical staff, and developers to engage meaningfully with its ecosystem. ALFRED's documentation includes detailed user personas and use cases, which serve as an ideal foundation for investigating requirement-level conflicts across different user groups. These personas often reflect contrasting values: for example, one older adult may prioritize privacy and minimal interaction, while another may demand constant motivational feedback from the system. Such divergences can lead to requirement conflicts that, if left unresolved, hinder system integrity and usability [4].

Early iterations of our pipeline revealed a major limitation in using LLMs like ChatGPT for conflict identification. The model frequently misclassified the relationship between user stories—often labeling two unrelated or mutually supportive user stories as "contradictory," simply due to superficial differences in wording or inferred intent. This issue, raised during Weeks 6 and 7 of client consultations, underscored the opacity of LLM reasoning and the risk of over-interpreting vague language patterns as conflicts. It highlighted the need for a more structured conflict detection approach that could distinguish between genuine contradictions and contextually compatible expectations.

This report aims to answer the following research questions:

- **RQ1:** How can we generate user stories that are specific, unambiguous, highlighting the persona's singularities?
- **RQ2:** How can we reliably identify subtle or indirect requirement conflicts between user stories—especially when there is no obvious contradiction?
- **RQ3:** What resolution strategies are most suitable for addressing different types of conflicts identified between user stories?

Ultimately, this study contributes a methodology for automated system requirement conflict identification and resolution using LLMs, while grounding the approach in a realistic system (ALFRED). The framework is extensible and applicable to other domains where persona-driven requirement modeling is essential.

Structure. Section **??** provides the background of our study.

II. Background

A. Related Work / Literature Review

User stories have become a fundamental artifact in requirements engineering, serving as concise, user-centric descriptions of desired system outcomes. They emphasize the end goal from the perspective of software users, facilitating collaboration among diverse stakeholder groups and encouraging creative

problem-solving [4]. Deriving user stories from personas and use cases ensures that system functionalities reflect real user needs and contexts.

Requirements can broadly be categorized into functional and non-functional types. Functional requirements specify concrete system behaviors, while non-functional requirements express quality attributes such as security, performance, and usability. The ALFRED system organizes requirements into six pillars, including four functional pillars, a general requirements pillar, and a developer core pillar [4].

Conflict identification and classification among requirements is a well-studied challenge in software engineering. Aldekhail et al. [1] surveyed eight prominent conflict detection techniques and assessed their suitability for various requirement types and engineering pipelines. Table I summarizes key approaches relevant to this study.

TABLE I
SUMMARY OF CONFLICT CLASSIFICATION TECHNIQUES

Technique	Summary	Suitable for
Poort and de With's technique [5]	Groups functional requirements based on non-functional requirements. Distinguishes conflicts as grouping or in-group conflicts.	Functional requirements classification (used in clustering)
Sadana and Liu [6]	Analyzes conflicts within non-functional requirements, defining mutually exclusive and partial conflicts.	Non-functional requirements
Heng and Ming [**?**]	Uses multi-coordinated views to classify incomplete, overlapping, or disjoint requirements.	Functional and non-functional requirements
Butt et al. [**?**]	Classifies conflicts based on requirement importance categories: mandatory, essential, optional.	Functional and non-functional requirements
Kim et al. [**?**]	Proposes comprehensive classification covering both requirement types.	Functional and non-functional requirements
Moser et al. [**?**]	Automates detection of complex semantic conflicts between requirements.	Functional and non-functional requirements
Chentouf [3]	Focuses on detecting conflicts in both functional and non-functional requirements using feature interaction analysis.	Functional and non-functional requirements
Mairiza and Zowghi [**?**]	Specializes in non-functional requirement conflict classification.	Non-functional requirements

B. Foundational Concepts

a) User Stories and Personas: User stories represent specific, goal-oriented requirements articulated from a user's perspective. They are typically concise and non-technical, enabling cross-functional teams to understand user needs. Personas embody archetypal users with defined characteristics, goals, and challenges, serving as anchors for generating relevant user stories.

b) Functional vs. Non-Functional Requirements: Functional requirements describe what the system should do, while non-functional requirements capture how the system should behave, encompassing aspects like performance, security, and usability. Both are essential to system quality but often differ in clarity and conflict types.

c) Requirement Conflicts: Conflicts arise when two or more requirements impose incompatible constraints or expectations. They can be categorized into mutually exclusive conflicts, partial conflicts, and semantic conflicts. Detecting and resolving these conflicts early is crucial to avoid design flaws and implementation issues.

C. Contextualizing the ALFRED System

The ALFRED system is a virtual assistant platform targeting older adults and their caregivers, medical staff, and developers. It embodies a complex socio-technical system with diverse user groups and multifaceted requirements spanning physical, cognitive, and social domains [4].

Its requirements are structured around six pillars, addressing functional capabilities such as interaction assistance and personalized care, alongside non-functional concerns like privacy and accessibility. The system's diversity necessitates sophisticated requirement engineering techniques capable of managing conflicting needs and priorities across stakeholder groups.

This study leverages ALFRED's comprehensive user personas and use case documentation as a testbed for automated requirement generation, conflict identification, and resolution, demonstrating applicability to similarly complex systems.

REFERENCES

[1] Maysoon Aldekhail, Azzedine Chikh, and Djamal Ziani. Software requirements conflict identification: Review and recommendations. *International Journal of Advanced Computer Science and Applications (IJACSA)*, 7(10):325–335, 2016.

[2] Manal Binkhonain and Liping Zhao. A review of machine learning algorithms for identification and classification of non-functional requirements. *Expert Systems with Applications: X*, 1:100001, 2019.

[3] Zohair Chentouf. Detecting oam&p requirement conflicts using a feature interaction approach. In *International Journal of Network Management*, volume 22, pages 95–103. Wiley Online Library, 2012.

[4] ALFRED Consortium. D2.3 – User Stories and Requirement Analysis. Technical Report Version 1.4, ALFRED Project, March 2022.

[5] Eltjo R. Poort and Peter H. N. de With. Resolving requirement conflicts through non-functional decomposition. In *Proceedings of the Fourth Working IEEE/IFIP Conference on Software Architecture (WICSA'04)*, pages 12–21. IEEE, 2004.

[6] Vishal Sadana and Xiaoqing Frank Liu. Analysis of conflicts among non-functional requirements using integrated analysis of functional and non-functional requirements. *Proceedings of the 31st Annual International Computer Software and Applications Conference (COMPSAC 2007)*, pages 31–38, 2007.

III. METHODOLOGY

Fig. 1 provides an overview of our approach.

A. Overall Pipeline

The methodology employed in this project follows a structured and iterative pipeline designed to ensure comprehensive and consistent generation of system requirements driven by user personas and use cases. The overall pipeline comprises the following key steps:

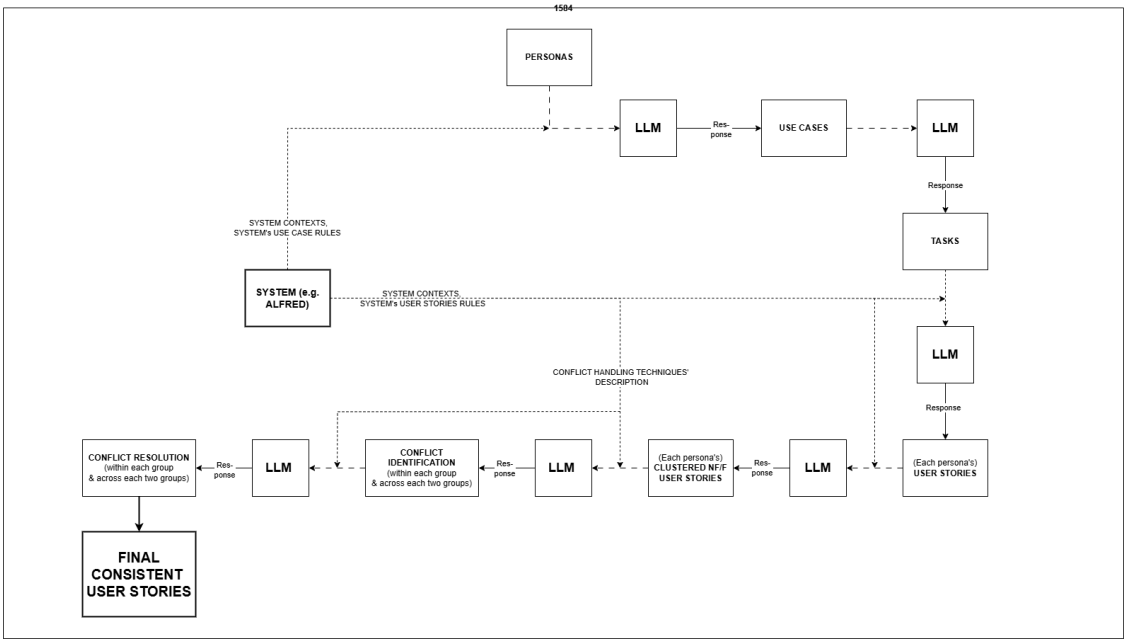

Fig. 1. Pipeline Overview - Summary version

1) **Persona and System-specific Contexts Collection:** This initial step involves collecting detailed user personas along with system-specific contexts that capture the environment, constraints, and user persona (stakeholder) perspectives. These contexts provide essential background to ground subsequent use case and requirement generation, validation, and analysis.
2) **Requirement (User Story) Generation:** Building upon the collected personas and contexts, use case generation and task extraction are performed as intermediate steps. Use cases describe goal-directed interactions between users and the system, while task extraction breaks these scenarios down into specific user tasks. These form the basis for generating detailed requirements, also referred to as user stories, which reflect concrete system functionalities and constraints.
3) **Requirement Clustering (Categorization)** The pipeline clusters similar requirements into thematic groups based on their functional or non-functional nature. This categorization facilitates easier management and understanding of requirement domains.
4) **Conflict Identification:** An essential phase involves detecting conflicts and inconsistencies among requirements, especially those arising from different user groups or conflicting stakeholder needs. The system leverages automated analysis to highlight potential requirement clashes.
5) **Conflict Resolution and Negotiation:** Following conflict identification, the pipeline supports the resolution process by proposing reconciled or unified requirements that balance competing needs, ensuring system coherence. The final set of system requirements, in which no conflicts or inconsistencies arise, forms a solid basis for system design and implementation.

This pipeline is iterative and modular, allowing continuous refinement as new personas, use cases, or system constraints emerge. The integration of automated LLM-based generation and conflict resolution tools significantly accelerates the requirements engineering process while maintaining high quality and relevance.

B. Data in This Project

Fig. 2 illustrates the initial data inputs, including system guidelines and persona groupings. This project relies on a rich and structured set of input data that forms the foundation for automated requirement generation, conflict detection, and resolution. The data can be broadly categorized into system-specific guidelines and user personas, which reflect diverse stakeholder perspectives.

a) System-Related Guidelines: These consist of high-level descriptions and formalized rules defining the system context, fundamental pillars, and the structure and format of use cases and user stories. For example, the ALFRED system acts as a reference case, with its documented system summary and core pillars guiding the generation and validation of requirements. Additional use case-related rules specify definitions, expected structures, scenario examples, and task extraction methodologies. Likewise, user story-related rules provide guidance on the user story format, examples, and crucially, clustering techniques that group requirements by thematic or functional similarity.

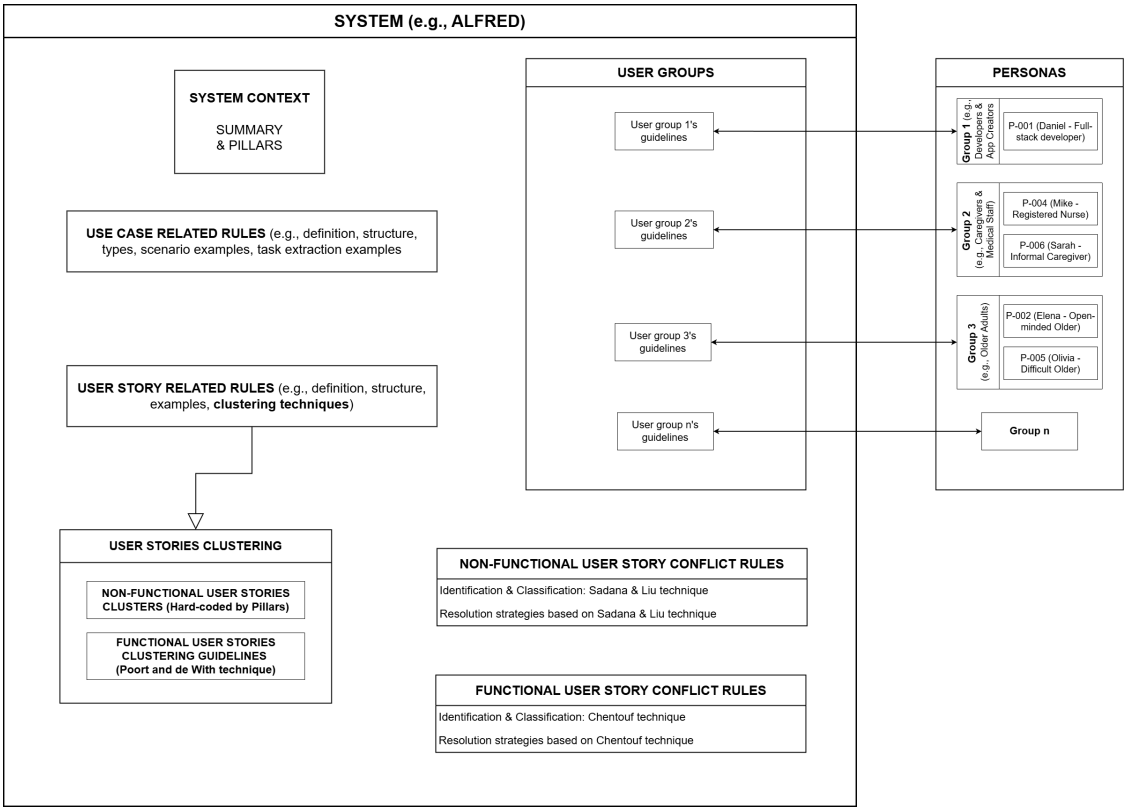

Fig. 2. Overview of Initial Data Inputs, including system guidelines and persona groupings.

b) User Personas and Stakeholder Groups: Personas represent archetypal users or stakeholders, each characterized by detailed attributes, goals, challenges, and unique singularities. These personas are organized into distinct user groups reflecting their roles within the system ecosystem—for example, developers and app creators, medical staff and caregivers, and older adult users. Each group is governed by tailored guidelines to capture their specific needs and constraints. The personas provide the human-centric perspective essential for personalized and context-aware requirement engineering.

By integrating these data—the system guidelines and personas—our approach ensures that automated user story generation and conflict analysis are both comprehensive and contextually relevant, faithfully reflecting the diverse viewpoints of all stakeholders involved.

C. User Story Generation

Fig. 3 shows the detailed pipeline for user story generation and clustering. User story generation in this project is a multi-phase process that transforms raw user personas and use cases into detailed, persona-driven user stories. The process consists of three key sub-steps: use case generation, use case task extraction, and user story creation.

a) Use Case Generation: Initially, use cases are generated from persona groups and system context by sampling and combining personas and use case types according to predefined configuration rules. These skeletal use cases include identifiers, associated personas, user groups, pillars, and use case types. To enrich these skeletons, a detailed use case name and description are generated using Large Language Models (LLMs) via carefully constructed prompts. These prompts incorporate system summaries, user group guidelines, persona information, use case definitions, and previously generated use cases to avoid duplication. The LLM is instructed to produce concise, unique names and contextual descriptions aligned with the system and personas.

b) Use Case Task Extraction: Once use cases have scenarios, the system extracts persona-specific tasks from these narrative scenarios. Task extraction employs LLMs prompted with the full use case scenario, persona details, and system context. The prompt emphasizes extracting both functional (goal-directed actions or interactions) and non-functional (expectations, emotional responses, usability constraints) tasks per persona, ensuring task uniqueness and alignment with the persona's characteristics. Extracted tasks are output as structured JSON grouped by persona.

c) User Story Generation: From the extracted tasks, skeleton user stories are created for each persona, capturing individual tasks linked to use cases. These skeletons are then enriched by generating de-

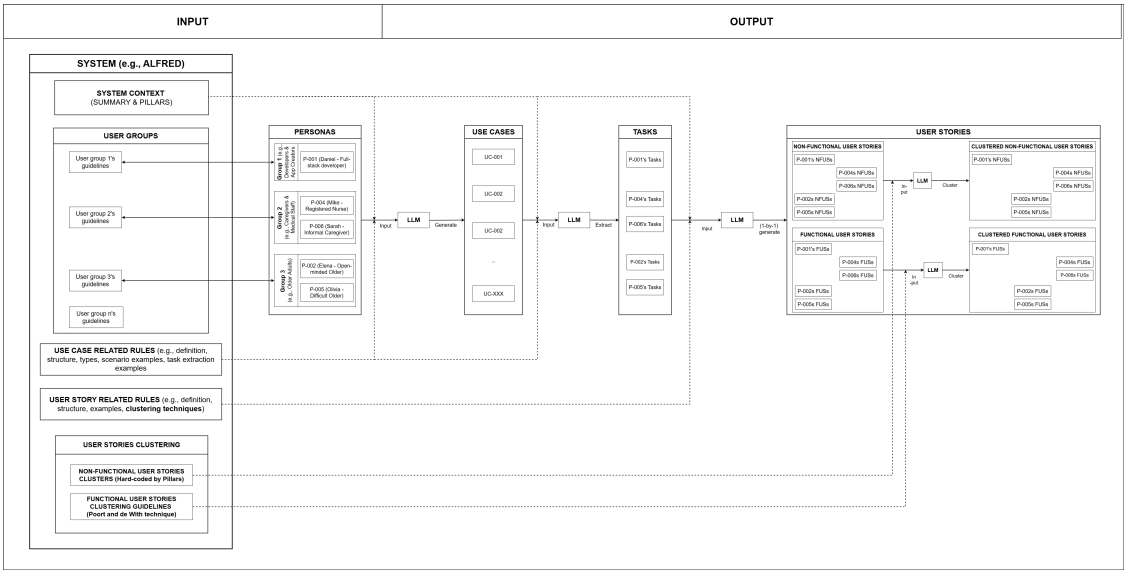

Fig. 3. Detailed pipeline for User Story Generation and Clustering, illustrating inputs, LLM-based generation steps, and clustering processes.

tailed user stories via LLMs that fill in missing fields such as title, summary, priority, and the most relevant system pillar. The prompt for this step integrates the system overview, user story guidelines, persona and use case details, and the raw task text to guide the LLM in producing personalized, realistic user stories that prioritize the persona's unique needs and possible conflicts with system assumptions.

Throughout these steps, the prompt engineering approach is critical. The prompts are long, context-rich, and structured with multiple sections that separately describe system context, user groups, personas, use case details, prior examples, and specific task or story instructions. This structure ensures the LLM's output is highly contextualized, coherent, and consistent with the personas and system requirements, while avoiding repetition or unrealistic smoothing of conflicting user needs.

This modular pipeline enables scalable, automated user story generation tailored to multiple personas and system contexts, serving as a foundation for subsequent analysis, clustering, and conflict resolution steps.

D. User Story Clustering

User story clustering is a critical step in organizing and analyzing the large volume of generated user stories. It facilitates systematic identification of thematic groups, which supports conflict detection, prioritization, and further refinement.

a) Non-functional User Stories Clustering: Initially, non-functional requirements (user stories) are clustered using a hard-coded taxonomy inspired by established research in requirements engineering, notably the comprehensive review by Binkhonain and Zhao [2]. Since their taxonomy is designed for general software systems, it requires adaptation to the specific context of the ALFRED system.

Based on insights from the literature and the team's domain expertise, the following broad clusters were established for general software systems:

- **Security-related clusters (8)**: Security, Integrity, Confidentiality, Authentication & Authorization, Access Control, Privacy, Cryptography/Encryption, Accountability.
- **Performance-related clusters (14)**: Performance, Availability, Reliability, Scalability, Efficiency, Time behavior, Resource utilization, Maintainability, Portability, Supportability, Accuracy, Fault tolerance, Lifecycle, Deployability.
- **Usability-related clusters (8)**: Usability, Look and Feel, Operationality, System Interface, Accessibility, Understandability, Attractiveness, Flexibility.
- **Other/Mixed categories (7)**: Functionality, Legal & Licensing, Interoperability, Modifiability, Testability, Traceability, Verifiability.

Following this initial classification, the team leveraged Large Language Models (LLMs) to rescope, update, specify, and prune the clusters, tailoring them to the ALFRED system's unique pillars. The refined non-functional user story clusters for ALFRED are:

- **General Requirements Clusters (2)**:
  - *Security, Privacy & Reliability*: Consolidates system-wide requirements for secure data handling, privacy protection, reliable system behavior, high availability, failure resilience, and consistent performance.

  - *Accessibility & Physical Usability*: Encompasses physical and cognitive accessibility requirements alongside ergonomic considerations, including adaptive interfaces and hardware usability for users with impairments.
- **Pillar I: User-Driven Interaction Assistant (1)**:
  - Encompasses non-functional expectations for voice-based controls, personalized interfaces, multimodal interactions, and adaptive UI support, ensuring usability for diverse user capabilities.
- **Pillar II: Personalized Social Inclusion (1)**:
  - Focuses on requirements facilitating social engagement, event recommendations, and communication support to mitigate social isolation.
- **Pillar III: Effective & Personalized Care (1)**:
  - Includes requirements related to health monitoring integration, data access for caregivers, and support for timely interventions.
- **Pillar IV: Physical & Cognitive Impairments Prevention (1)**:
  - Covers stimulation through games, challenges, and guided activities tailored to individual physical and cognitive abilities.
- **Developer Core Clusters (3)**:
  - *API Integration & Development Support*: Developer tools, stable APIs, documentation, testing, and debugging support.
  - *Marketplace & Interface Experience*: Infrastructure and UI design for the ALFRED marketplace, including app submission, discoverability, and minimal caregiver interfaces.

b) Functional User Stories Clustering: For functional user stories, the clustering approach is driven by their semantic relevance to the non-functional clusters, following the technique proposed by Poort and de With [5]. Specifically, each non-functional user story acts as a thematic anchor; all functional user stories from across personas that relate to that non-functional story are grouped into a cluster named after the non-functional user story.

IV. Evaluation

This section discusses the empirical evaluation of our approach.

V. Conclusion

In this paper, we ...

Acknowledgement. XXX

References

[1] Maysoon Aldekhail, Azzedine Chikh, and Djamal Ziani. Software requirements conflict identification: Review and recommendations. *International Journal of Advanced Computer Science and Applications (IJACSA)*, 7(10):325–335, 2016.

[2] Manal Binkhonain and Liping Zhao. A review of machine learning algorithms for identification and classification of non-functional requirements. *Expert Systems with Applications: X*, 1:100001, 2019.

[3] Zohair Chentouf. Detecting oam&p requirement conflicts using a feature interaction approach. In *International Journal of Network Management*, volume 22, pages 95–103. Wiley Online Library, 2012.

[4] ALFRED Consortium. D2.3 – User Stories and Requirement Analysis. Technical Report Version 1.4, ALFRED Project, March 2022.

[5] Eltjo R. Poort and Peter H. N. de With. Resolving requirement conflicts through non-functional decomposition. In *Proceedings of the Fourth Working IEEE/IFIP Conference on Software Architecture (WICSA'04)*, pages 12–21. IEEE, 2004.

[6] Vishal Sadana and Xiaoqing Frank Liu. Analysis of conflicts among non-functional requirements using integrated analysis of functional and non-functional requirements. *Proceedings of the 31st Annual International Computer Software and Applications Conference (COMPSAC 2007)*, pages 31–38, 2007.