From Complaints to Requirements: Using NLP and QFD to Derive Use Cases for University Academic Management System

José Pedro de Santana Neto
Department of Informatics, Federal University of Paraná
Curitiba, Brazil
jose.santana@ufpr.br

Cleiton Almeida dos Santos
Department of Informatics, Federal University of Paraná
Curitiba, Brazil
cleitonsantos@ufpr.br

Jaison Pisa Rezine
Department of Informatics, Federal University of Paraná
Curitiba, Brazil
jaison.rezine@ufpr.br

Marco Antonio Zanata Alves
Department of Informatics, Federal University of Paraná
Curitiba, Brazil
mazalves@inf.ufpr.br

Simone Dominico
Department of Informatics, Federal University of Paraná
Curitiba, Brazil
sdominico@inf.ufpr.br

Leticia Mara Peres
Department of Informatics, Federal University of Paraná
Curitiba, Brazil
lmperes@inf.ufpr.br

Eduardo Todt
Department of Informatics, Federal University of Paraná
Curitiba, Brazil
todt@inf.ufpr.br

ABSTRACT

This study introduces a hybrid methodology combining Natural Language Processing (NLP) and Quality Function Deployment (QFD) to enhance the quality and responsiveness of institutional academic systems. The approach systematically transforms unstructured user complaints into functional software requirements, thereby improving documentation and maintenance in legacy systems. By leveraging NLP to extract and prioritize user needs based on dissatisfaction severity and frequency, and translating these into technical requirements via QFD, the method creates a scalable, user-centered, and traceable requirements engineering process, ideal for legacy systems with limited documentation. Analyzing 4,357 service tickets, our results identify critical improvement areas, including authentication, enrollment, and bureaucratic workflows. Although steps like technical mapping and modeling remain manual, the methodology boosts prioritization, transparency, and alignment with essential software quality attributes. While applied to this specific context, the framework shows potential for transferability to other institutional or commercial sectors. These findings demonstrate the value of integrating AI with software engineering principles to modernize complex institutional platforms and increase user satisfaction.

KEYWORDS

Requirements Engineering (RE), Natural Language Processing (NLP), Quality Function Deployment (QFD), User-Centered Design (UCD), Academic Management Systems (AMS)

1 Introduction

Artificial Intelligence (AI) has increasingly driven the modernization of higher education systems, particularly in areas involving administrative efficiency. In Brazilian Higher Education Institutions, AI technologies have been recognized as strategic tools for addressing persistent challenges in system efficiency, usability, and user support service quality [2, 9, 10, 24, 35].

At the University[1], the Academic Management System (AMS) supports academic coordination, administrative workflows, and institutional record management. However, despite its importance, the AMS has struggled to keep pace with evolving user demands. The Post-Graduation module, in particular, has received frequent complaints related to usability, authentication, and bureaucratic workflows. These recurring complaints underscore a critical need for systematic improvements aligned with actual user expectations.

Additionally, this system lacks documentation, which makes maintenance and future development difficult. This challenge is compounded by the absence of a formalized and systematic process to translate user feedback into actionable requirements within the institution. Historically, improvements were managed on an ad-hoc basis. This context makes a direct comparative analysis a challenge. Also, it underscores the critical need for the development of a structured framework, such as the one proposed in this study.

A key aspect to ensure software effectiveness in this context lies in the domain of Software Quality (SQ), which encompasses both internal attributes (e.g., maintainability, modularity) and external qualities perceived by users (e.g., usability, reliability, and performance). The latter is especially relevant for institutional AMS, where user satisfaction and operational reliability are essential for daily academic operations [18]. Achieving high external quality depends on a robust Requirements Engineering (RE) process, which ensures that the system evolves by real needs and stakeholder priorities [31, 32].

However, capturing high-quality requirements in legacy systems is particularly challenging when institutional processes are complex, dynamic, and poorly documented. In such contexts, user

[1]The university is intentionally not identified to protect the privacy of the users and maintain the anonymity of the research.

complaints represent a valuable but underutilized source of information. Additionally, traditional RE approaches, while rigorous, often face challenges when there is a need to process large volumes of unstructured feedback. Bridging this gap requires a methodology that is systematic, traceable, and capable of integrating data-driven insights from real-world usage scenarios.

To address this problem, we propose a hybrid methodology that combines the principles of Quality Function Deployment (QFD) [1, 5] with Natural Language Processing (NLP) techniques [21, 28] for automated complaint analysis. This approach establishes a data-driven framework that systematically analyzes large volumes of user data to identify patterns and chronic problems. By combining complaint frequency and sentiment analysis, the methodology enables a more objective prioritization of user needs, moving beyond ad-hoc ticket analysis. The process aims to generate clear artifacts, such as thematic clusters and a QFD matrix, to build an explicit knowledge base. This is fundamental for maintaining poorly documented legacy systems. Thus, shifting from reactive problem-solving to a proactive strategy for improving SQ and user satisfaction.

To the best of our knowledge, this is the first study to integrate NLP and QFD in a comprehensive pipeline for deriving use cases directly from service ticket data within a university setting, thereby impacting the alignment of legacy system improvements with quality standards and user expectations. While applied to this specific context, the data-driven principles of this methodology hold promise for transferability to other institutional or commercial sectors.

The main contributions of this paper are to provide: (a) a hybrid methodology that integrates NLP and QFD to extract and prioritize software requirements directly from unstructured user complaints; (b) an empirical analysis of 4,357 service tickets from the AMS, resulting in the identification of four critical categories of user needs: data updates, authentication, enrollment, and defense workflows, and; (c) a data-driven prioritization framework that quantifies user dissatisfaction using sentiment analysis and translates needs into technical requirements through a structured QFD matrix.

This paper is organized as follows. Section 2 outlines the theoretical foundations that support our approach, covering essential concepts from RE, NLP, and QFD. Section 3 details the proposed methodology, describing each step of the workflow adopted to analyze user complaints and derive functional requirements. Section 4 presents the results of applying this methodology to a dataset of 4,357 service tickets recorded between 2023 and 2024. Section 5 discusses challenges and outlines directions for future work. Finally, Section 6 presents the conclusions of the study, summarizing the contributions, practical implications, and potential for broader adoption of the proposed framework.

2 Theoretical Background

This section presents the theoretical foundations that support the proposed methodology, based on three interrelated domains: NLP, QFD, and RE in the context of SQ. The integration of these concepts enables a structured, data-driven approach to extract functional requirements from user-generated complaints in complex systems.

2.1 Software Quality and Requirements Engineering

SQ is a fundamental component of software engineering, directly influencing user satisfaction, operational efficiency, and long-term system viability. It is commonly divided into two dimensions: internal and external. Internal quality refers to attributes such as code readability, modularity, and maintainability, which are essential for developers and system maintainers. External quality, which is the primary focus of this study, relates to the user experience and encompasses aspects like usability, reliability, performance, and security [18, 32]. These factors are particularly critical in institutional AMS, where fragmented workflows, legacy constraints, and scarce documentation demand user-centered improvements to ensure effectiveness and adoption.

A robust RE process is crucial to ensure external SQ throughout the system lifecycle. This discipline, which involves elicitation, modeling, and analysis of requirements, focuses on identifying and documenting system functionalities [31]. This study emphasizes the elicitation stage, which is key to capture user expectations and translating them into actionable development artifacts. In this context, a systematic approach to requirements elicitation based on user feedback and supported by practices like automated testing [32], provides a practical and scalable strategy to improve SQ in institutional systems. Identifying issues early and aligning the system more closely with user needs.

2.2 Natural Language Processing for Requirements Elicitation

NLP provides computational techniques to process and interpret human language, making it highly applicable to extracting insights from unstructured user feedback. In the context of helpdesk systems and service management, NLP has been used to identify patterns in complaint tickets, analyze user sentiment, and improve support workflows [3, 21, 37].

Bahad et al. [3] highlight the potential of sentiment classification to prioritize user issues, while other works use clustering and embedding techniques to group similar issues and automate requirements elicitation [37]. These approaches offer scalable mechanisms to filter noise and extract meaningful user intents from large datasets.

Despite this growing interest, the majority of NLP applications have focused on classification, prioritization, or traceability. Few studies have attempted to systematically bridge these insights into artifacts to support formal modeling practices such as use case generation.

2.3 Quality Function Deployment in Software Engineering

QFD is a structured technique originally developed in industrial engineering to transform customer needs into design specifications [1]. It is widely recognized for its use of the House of Quality matrix, in which rows represent user needs/expectations (**WHATs**), columns represent technical features/responses (**HOWs**) and the strength of the relationship between them is quantified using weighted values (strong = 9, moderate = 3, weak = 1, none = 0) [1, 5].

Table 1: Summary of the Methodological Workflow

Step	Goal	Tools
1. Identify and Collect Customer Needs	Extract user complaints	Clustering
2. Prioritize Customer Needs	Rank by impact	Sentiment analysis
3. Technical Mapping	Translate needs into system features	Manual analysis
4. Construct the House of Quality	Relate needs to features	QFD matrix
5. Analyze the Matrix	Identify key features	Matrix weights
6. Derive Functional Requirements	Write specifications	Functional requirements templates
7. Create Use Cases	Model system behavior	Textual use cases

In software engineering, QFD has been adapted to support requirements prioritization and traceability, helping teams align technical specifications with stakeholder value [12, 22, 27, 34]. Its emphasis on traceable, quantifiable mappings makes it particularly suitable for institutional settings, where user feedback is abundant but not structured.

2.4 Integrating NLP and QFD for Requirements Modeling

The integration of NLP and QFD in this study constitutes a novel contribution. Previous works have applied NLP for clustering, classification, or sentiment analysis of complaint data [3, 21, 37], and QFD has been employed for structured requirements translation [27]. However, this is the first time these approaches are combined into a unified framework for automated use case derivation from real-world user complaints.

Our method operationalizes this integration by using NLP techniques to identify and cluster recurrent issues, assess dissatisfaction via sentiment scores, and assign weights to each cluster. These weighted needs are then structured within a QFD matrix, allowing for the prioritization of technical responses. Finally, these responses are mapped into formal functional requirements and use case specifications. By combining the analytical depth of NLP with the systematic rigor of QFD, this approach supports a scalable, user-centered methodology for RE. It is useful, especially in cases of legacy and complex systems with limited documentation.

3 Methodology: Data-Driven Derivation of Functional Requirements from User Feedback

We propose a structured, seven-step methodology based on QFD principles [1, 5] to systematically transform user complaints into functional software requirements. The methodology was developed in close collaboration with the development team of the Information Technology Department of the university.

Briefly, the process begins with the collection of real user complaints from the university institutional ticketing system. These inputs are processed using NLP-based clustering algorithms to uncover recurrent and high-impact issues. Sentiment analysis is then applied to prioritize user needs based on dissatisfaction levels. These needs are mapped to technical requirements and organized into a QFD matrix that supports strategic development decisions. This ensures that software improvements remain aligned with both user expectations and institutional objectives. Table 1 summarizes the methodology, which is detailed in the following subsections.

3.1 Step 1 – Identify and Collect Customer Needs

User complaints were extracted from the GLPI (Gestion Libre de Parc Informatique) [2] system, which is used by the university to manage support tickets. The dataset was composed of 4,357 service tickets recorded between 2023 and 2024. The selection criteria included both *in-progress* and *concluded* tickets. This decision was based on an initial analysis showing that many of the *concluded* tickets represented temporary workarounds or palliative solutions, rather than definitive fixes addressing the root causes of issues. This resulted in similar issues recurring in new tickets opened by other users. Therefore, the methodology examined *in-progress* tickets to capture current demands and *concluded* tickets to uncover the origins of chronic problems, enabling the identification of persistent structural issues. By leveraging this historical base, the approach facilitated proactive software enhancements aimed at reducing future tickets, rather than merely reacting to the most recent complaints.

The raw textual data were processed using the Python programming language (version 3.11.13), taking advantage of its extensive ecosystem of libraries for text preprocessing and NLP tasks (Numpy 2.0.2, NLTK 3.9.1, Pandas 2.2.2, Scikit-learn 1.6.1, Spacy 3.8.7, Transformers 4.39.1). The preprocessing phase involved cleaning and preparing the data for NLP. This included the removal of duplicates, empty values, symbols, excessive whitespace, accented and special characters, punctuation, and conversion of all text to lowercase. The NLP pipeline, implemented using the SpaCy and NLTK libraries [15, 23], consisted of stop-word removal, non-numeric filtering, and lemmatization to reduce words to their base forms [28].

To categorize the complaints into meaningful thematic groups, a text clustering analysis was performed. First, dimensionality reduction was applied using the Truncated SVD method [17] to extract the most relevant semantic features of the text. Then, the K-Means algorithm was used to group the complaints based on recurring patterns [33]. To support the interpretation and validation of the clusters, the t-SNE (t-Distributed Stochastic Neighbor Embedding) technique was employed to visualize the groupings in a two-dimensional space [6]. The optimal number of clusters for the K-Means algorithm was determined through the elbow method, which evaluates intra-cluster variance across different group counts.

[2]Available at: https://glpi-project.org/

The inflection point of the elbow curve was used to identify the ideal number of clusters [33, 36].

3.2 Step 2 – Prioritize Customer Needs

To rank the extracted user needs, sentiment analysis was applied to each complaint using a Transformer-based model (BERT) fine-tuned for multilingual text classification, including Portuguese [8, 16, 26]. The model pipeline includes automatic truncation for long texts, ensuring scalability for large volumes of data and robustness.

The user dissatisfaction was mapped in five levels, ranging from **Level 1** (L1, mild concerns) to **Level 5** (L5, strong frustration or severe complaints). Each level is associated with a continuous sentiment score between 0 and 1, which captures the intensity of the expressed sentiment within that level. This approach enables a more refined analysis, facilitating the identification of critical patterns in user feedback, complaints, or interactions [10]. To prioritize user needs, we calculated the importance weights(W_k) based on the level of dissatisfaction and its frequency. The weight reflects the importance of each user need in terms of emotional intensity and volume of complaints. The importance weight for each group was calculated as shown in Equation 1, which assigns a weight W_k to each cluster k based on the proportion of complaints classified in its most frequent sentiment level and the severity of that sentiment. Specifically, N_{max} represents the number of complaints in the most frequent sentiment level within the cluster, N_t is the total number of complaints in the dataset, and S_l denotes the sentiment level score (ranging from L1 to L5) corresponding to that level.

$$W_k = \left(\frac{N_{max}}{N_t} \right) \cdot 100 \cdot S_l \tag{1}$$

To standardize the scale and enable fair comparisons across clusters, the weights (Equation 1) were normalized to a target range from $W^*_{\text{min}} = 1$ to $W^*_{\text{max}} = 5$ using the min-max normalization method [14], as shown in Equation 2, where W_{kn} is the normalized weight, W_k is the original (non-normalized) weight, W^*_{min} and W^*_{max} are the minimum and maximum values of the target range, respectively. This formulation ensures that the smallest original weight maps to W^*_{min}, the largest maps to W^*_{max}, and all other values are scaled proportionally in between.

$$W_{kn} = W^*_{\text{min}} + (W^*_{\text{max}} - W^*_{\text{min}}) \cdot \frac{W_k - \min(W_k)}{\max(W_k) - \min(W_k)} \tag{2}$$

3.3 Step 3 – Technical Mapping

Once the user needs were identified and ranked, the next step comprised the analysis and translation into corresponding technical characteristics of the system. This stage involved structured brainstorming sessions with the core development team responsible for the AMS Post-Graduation module. The team consisted of full and senior software engineers and analysts from the IT department of the institution, who were directly responsible for the system under analysis. These professionals were selected based on their accumulated practical knowledge of the system architecture and business rules, which was essential for autonomously proposing solutions and ensuring that each user need could be addressed through feasible system features or architectural improvements. This results in actionable system elements.

Detailing this process flow is crucial, as it enhances methodological clarity, value, and replicability while opening avenues for future work. This enhancement is supported by ISO/IEC/IEEE 29148:2018 [19] on RE processes and ISO/IEC/IEEE 42010:2022 [20] on architecture description, along with software engineering literature that recommends structured practices, iterative feedback cycles, and continuous validation [7, 13]. Additionally, it can be integrated with innovative practices, such as brainstorming approaches based on mental models and systemic methodologies to foster team convergence, collaborative innovation, and holistic decision-making in complex environments [4, 11, 25, 29, 30]. The combination of such approaches, along with engineering best practices, contributes to reliability, interdisciplinary collaboration, and process quality. Considering our experience, it is possible to conduct this step as follows:

1. **Present Requirement**: a user need (**WHAT**) is selected from the prioritized list, and it is set up with the meaning, context, and constraints;
2. **Generate Ideas Individually**: each analyst/engineer sketches or notes possible solutions (**HOWs**) silently;
3. **Share and Discuss**: with equal airtime, everyone presents their idea briefly, considering pros/cons, feasibility, risks, and alternatives;
4. **Decide and Document**: the best **HOW** is selected via consensus and the QFD matrix is updated;
5. **Loop or Close**: if more requirements remain, a new requirement is presented; otherwise, the technical mapping is finalized.

3.4 Step 4 – Construct the House of Quality

To relate user needs to system-level responses, a *QFD matrix* (House of Quality) was built following [1, 5]. In this matrix, user needs were represented by the rows, and technical characteristics by the columns. The relationship between each user need and each technical characteristic was quantified with a numerical value, in which: a strong relationship is represented by a value of 9, a moderate relationship is represented by a value of 3, a weak relationship is represented by a value of 1, and no relationship is represented by a value of 0. This step enabled a structured mapping between user complaints and their corresponding technical responses, supporting traceability and impact analysis.

3.5 Step 5 – Analyze the Matrix

The matrix was analyzed using weighting techniques to calculate the relative importance of each technical feature. The weights were derived by multiplying the importance score of each need by its respective relation strength in the matrix.

To perform these computations, let n denote the number of user needs (**WHATs**), m be the number of technical characteristics (**HOWs**), W_i be the weight assigned to user need i, where $i \in 1, 2, \ldots, n$ and, R_{ij} be the relationship strength between need i and technical characteristic j, with typical values ranging from 0 to

9 [1, 5]. The priority score S_j of each technical characteristic j is calculated by:

$$S_j = \sum_{i=1}^{n} W_i \cdot R_{ij} \tag{3}$$

This equation aggregates the weighted contributions of each user need to a given technical solution. The resulting score S_j indicates the relative importance of implementing the technical characteristic j to satisfy user priorities. Higher values of S_j suggest that the corresponding feature addresses more critical user needs and should be prioritized during development.

This step ensures that the design decisions are directly traceable to validated user feedback extracted from complaint data, thus supporting a data-driven and user-centered RE process.

3.6 Step 6 – Derive Functional Requirements

From the most relevant technical characteristics, functional requirements were formulated. These were written in a structured format, using templates aligned with software engineering standards.

3.7 Step 7 – Create Use Cases

In this step, use case specifications are defined for the key requirements previously identified. Each specification outlines essential components, including actors, preconditions, main and alternative flows, exceptions, and postconditions. Although this stage represents a critical aspect of the RE process, it is not the primary focus of this study. It is worth mentioning that the comprehensive modeling of use cases, an area already well supported by established methods in the literature, is beyond the scope of this work and is left for future research and development.

4 Evaluation

The research analyzed 4,357 service tickets recorded between 2023-2024, using advanced NLP clustering and sentiment analysis techniques to map and classify user demands. Based on the cluster analysis and user complaints, we were able to identify four main clusters, revealing the main problem categories related to the system, which are:

Cluster 0: Data Changes and Updates
Key terms in this cluster include the words *year*, *base*, *student*, *change*, *modification*, and *date*. These terms are associated with requests to modify the student records database, encompassing updates to student information, adjustments to dates, and changes within the academic database. This cluster may include requests like transcript corrections, enrollment amendments, or updates to registration details.

Cluster 1: Course and Access Management
Key terms in this cluster include the words *ams*, *course*, *student*, *instructor*, *class*, *process*, *access*. This cluster primarily refers to issues related to course management, academic classes, and administrative processes, including system access and permission-related issues. These terms indicate frequent requests involving authentication, user registration, and permission control within the system.

Cluster 2: Enrollments and Courses
Key terms in this cluster include the words *course*, *class*, *enrollment*, *CPF* [3], *manage*. This cluster focuses mainly on demands regarding course enrollments, issues with viewing classes, and operational difficulties, such as how to "manage" actions in the system. It includes cases where students are not able to enroll or view courses correctly.

Cluster 3: Defense Processes and Documentation
Key terms in this cluster include the words *minutes*, *sign*, *defense*, *password*, *committee*, *signature*. This cluster indicates a focus on bureaucratic processes related to academic defenses (specialization, master's, doctorate), including generating minutes, digital signatures, and instructor access. Password and authentication issues are also common in this context.

As it can be observed in Figure 1, the user dissatisfaction is categorized into five levels, denoted as L1 through L5. Regarding it, L1 represents the lowest degree of dissatisfaction (e.g., mild concerns). While L5 is the highest level (e.g., strong frustration or severe complaints). The stacked bars for each cluster illustrate the frequency of occurrences at these levels. The gradient shading, from lighter (L1) to darker (L5) tones, emphasizes the progression in intensity.

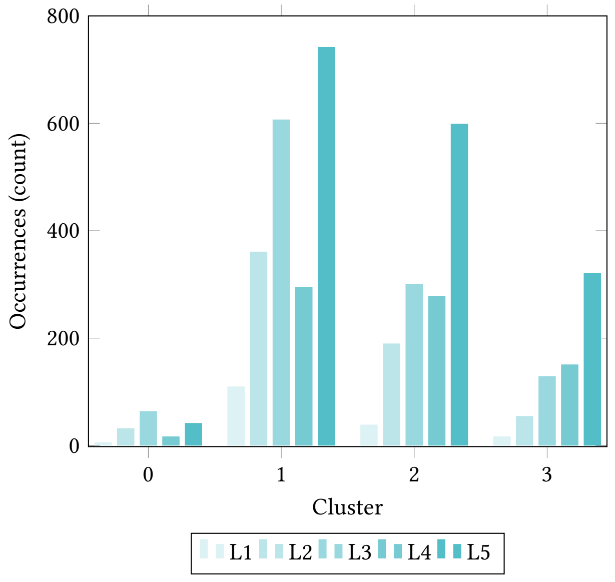

Figure 1: User Dissatisfaction level grouped by Cluster. The Levels are categorized as L1 (mild concerns) to L5 (strong frustration or severe complaints). Stacked bars show occurrence frequencies per cluster, with gradient shading from lighter (L1) to darker (L5), highlighting intensity progression.

A significant number of tickets (39.1%) indicate extreme dissatisfaction, underscoring the urgency for targeted improvements (Figure 1). Cluster 1 (system access) dominates in extreme dissatisfaction cases, followed by Clusters 2 and 3. In contrast, Cluster 0 exhibits relatively fewer instances of dissatisfaction, which may suggest that its associated issues are either less critical or being addressed more effectively. From the above analyses, four main need categories (**Step 1**) are presented in Table 2.

[3]Brazilian individual taxpayer registry

Table 2: Identified User Needs (WHATs)

ID	User Need (WHAT)
N1	Update and correct academic data easily
N2	Access the system without authentication failures
N3	Complete enrollment and view courses correctly
N4	Manage defense processes and signatures with less bureaucracy

These needs provide a focused lens on the users priorities for system improvement. This analysis guided the prioritization of user needs in **Step 2**, which were computed according to Equation 1, and the outcomes are presented in Table 3, emphasizing that authentication and enrollment are key areas for intervention.

Table 3: Prioritization of User Needs

ID	User Need (WHAT)	Dissatisfaction Level	Normalized Weight (W_{kn})
N1	Academic data updates	Low	1.0
N2	System access and authentication	High	5.0
N3	Course enrollment and management	High	4.2
N4	Thesis defense processes	Moderate	2.6

Based on the Table 3, we conducted the next **Step 3** by deriving the corresponding technical responses, as shown in Table 4. The mapping indicates a strategic focus on authentication robustness, enrollment automation, and user notifications, which align with the highest dissatisfaction areas.

Table 4: Identified Technical Responses

ID	Technical Characteristic (HOW)
T1	Robust authentication with password recovery
T2	Automatic validations for enrollment and courses
T3	Interface for editing and correcting registration data
T4	Management panel for thesis defenses with status and signatures
T5	Real-time notifications of errors or pending issues

Through the House of Quality matrix (Table 5), in **Step 4**, it is possible to verify the strong relationships between critical user needs and technical features. For instance, the robust authentication (T1) directly addresses the highest priority need (N2), while enrollment validations (T2) strongly support course management needs (N3).

In **Step 5**, by multiplying each row weight by the matrix values according to Equation (3) to prioritize the technical features, we have the final scores as presented in Table 6. It confirms the top priorities for technical effort, with authentication (T1) and enrollment validation (T2) scoring highest, followed by error notifications (T5). These results provide a clear, data-driven guide for system development priorities.

Table 5: House of Quality Matrix: Relationship between User Needs (WHATs) and Technical Characteristics (HOWs)

WHAT / HOW	T1	T2	T3	T4	T5
N1: Data update	1	3	9	1	1
N2: System access	9	1	1	1	3
N3: Enrollment/course	1	9	3	1	3
N4: Defense and signatures	1	1	1	9	3

Table 6: Calculation of Final Scores for Technical Characteristics (HOWs)

HOW (T)	Score Calculation	Final Score
T1	(1.0×1)+(5.0×9)+(4.2×1)+(2.6×1)	52.8
T2	(1.0×3)+(5.0×1)+(4.2×9)+(2.6×1)	48.4
T3	(1.0×9)+(5.0×1)+(4.2×3)+(2.6×1)	29.2
T4	(1.0×1)+(5.0×1)+(4.2×1)+(2.6×9)	33.6
T5	(1.0×1)+(5.0×3)+(4.2×3)+(2.6×3)	36.4

As a result, we have the orderly prioritized technical requirements (**Step 6**) in Table 7. It emphasizes improvements that directly respond to user dissatisfaction. For example, implementing functional password recovery and clear error messages (T1) can improve system accessibility and reduce user dissatisfaction.

Finally, the results obtained provide a solid foundation for designing targeted use cases (**Step 7**), ensuring that the development efforts of the system are aligned with the most critical and user-centered priorities. Based on the identified needs and their respective technical responses, the following use cases are proposed:

- Login with password recovery;
- Assisted enrollment process;
- Real-time alert and notification system;
- Management dashboard for academic defenses and digital signatures;
- Editing and updating registration data.

In summary, the integrated analysis of user complaints, levels of dissatisfaction, and technical mappings provides a comprehensive framework to prioritize improvements that are likely to maximize user satisfaction and system effectiveness. Future work may include a detailed description of use cases, which lies beyond the scope of this study.

4.1 Worked Example

To illustrate the application of the proposed methodology, this subsection presents a concrete example in which a single user complaint is transformed into a structured development artifact, passing through all six steps of the workflow (see Table 1). It is depicted as follows:

Table 7: Prioritized Technical Requirements

Priority	Technical Requirement	Description
1	T1 – Improve Authentication	Implement functional password recovery, secure authentication, and clear error messages
2	T2 – Enrollment Validations	Automate validations for classes, prerequisites, and course availability
3	T5 – Error Notifications	Display real-time alerts and notifications for pending actions and process failures
4	T4 – Defense Manager	Create a panel to control committee status, meeting minutes, signatures, and deadlines
5	T3 – Data Editing Interface	Create a screen for correcting/editing records with a history of changes

(1) **Ticket (Raw Complaint)**: this complaint *"I cannot log in to the system with my credentials. Every time I try, it says my password is incorrect. I already reset it twice, but it still does not work."* was extracted from the institutional GLPI helpdesk dataset and classified into **Cluster 1 – Course and Access Management**, which concentrates on issues related to authentication and access;
(2) **Identified User Need (WHAT)**: from clustering and thematic analysis, this ticket is mapped to **N2: Access the system without authentication failures** (see Table 2);
(3) **Prioritization through Sentiment Analysis**: Sentiment analysis classified the complaint as Level 5 (L5), corresponding to strong frustration or severe dissatisfaction. Given its high recurrence in the dataset, the normalized weight for this need was **5.0** (see Table 3), establishing N2 as a top-priority user demand;
(4) **Technical Response (HOW)**: during the technical mapping sessions with senior IT staff, the user need N2 was translated into the technical response **T1: Robust authentication with password recovery** (see Table 4);
(5) **Weighted Impact Analysis**: from T1, the relation between user needs and technical responses is quantified to compute their relative impact and prioritize implementation. For N2, this results in a strong correlation with T1 (see Table 6);
(6) **Prioritized Technical Requirements**: from the impact analysis, a functional requirement is formalized and ranked according to its priority (see Table 7). In this case, it has the highest priority (1).

This example demonstrates how the methodology transforms a single raw complaint into a requirement specification, ensuring transparency, traceability, and alignment between user dissatisfaction and technical improvements.

5 Challenges and Prospects

This section discusses the main challenges encountered in applying the methodology and outlines prospects for its evolution and broader adoption within academic software systems.

The proposed methodology is highly transferable, and its core principles extend beyond the university academic management context. The automated NLP pipeline (Steps 1-2) is domain-agnostic and highly transferable, making it applicable to any domain with user feedback, such as helpdesk tickets or support forums. However, its success depends on domain experts, such as senior software engineers or business analysts, who have deep knowledge of the target system architecture and business rules. As such, the framework acts as a powerful, data-driven support tool for RE, allowing the transformation of user needs into viable technical solutions.

The results highlight recurring patterns of user dissatisfaction, revealing persistent challenges related to usability, data management, authentication, and bureaucratic workflows. These findings underscore the importance of adopting data-driven practices for system improvements. One of the most significant contributions of this study lies in the ability to operate in legacy system environments characterized by complexity, evolving institutional processes, and chronic documentation gaps.

However, several challenges remain. A key limitation is the manual nature of critical stages, such as the technical mapping (Step 3). While the knowledge of the expert team is vital for ensuring technical feasibility, this approach may introduce subjectivity. For example, the proposed technical solutions may be influenced by the experiences and architectural preferences of the team, potentially overlooking other alternatives. Future iterations should consider other stakeholders, like business analysts and key users. It will enrich the solution space and reduce potential bias.

Furthermore, we acknowledge that the formal validation of the derived requirements by RE specialists and end-users is essential. This step must involve techniques such as expert reviews, user interviews, focus groups, participatory validation sessions, or usability testing. However, the contextual challenges of the legacy system environment made it complex to execute this phase within the scope of this study. As future directions, we plan to implement the prioritized use cases and integrate structured interactions with stakeholders to consolidate the insights generated automatically. Thereby, reinforcing the legitimacy of the produced artifacts. This feedback loop is fundamental to refining the requirements, aligning the evolution of the system with user expectations, and strengthening the overall reliability and robustness of the methodology.

Additionally, the lack of a formalized baseline for RE made it unfeasible to conduct a direct comparison against previous approaches. Thus, the reliance on reactive and undocumented problem-solving was the motivation for this work. Therefore, the main contribution is the establishment of a novel, systematic, and traceable process.

Despite these limitations, the approach already offers concrete benefits. It accelerates and structures the prioritization of system improvements, enhances communication within development teams, and promotes alignment with user demands. The integration of AI tools facilitated the analysis of large-scale feedback, supporting the process of requirements elicitation. This reinforces the need for continuous and structured communication between users and development teams. Thus, reaching the balance between user satisfaction and operational constraints, such as legal aspects and internal processes.

6 Conclusions

This study proposed an integrated methodology that combines NLP and QFD to systematically transform unstructured user complaints into functional software requirements. Applied to AMS, the approach demonstrated its ability to prioritize user needs and support informed technical decisions, even in the context of legacy platforms with limited documentation.

Although some steps still rely on manual analysis, the methodology significantly enhances the speed, transparency, and objectivity of the requirements engineering process. Its practical benefits, coupled with its alignment with key dimensions of software quality, suggest strong potential for adoption in other institutional and commercial systems. As such, it provides a scalable and user-centered framework for digital transformation, particularly effective in contexts where user complaints are abundant but technical documentation is limited. Using real user feedback, the approach enables faster, more transparent, and aligned system improvements.

ARTIFACT AVAILABILITY

The artifacts supporting the findings of this study, including the dataset of 4,357 service tickets and the analysis scripts, are not publicly available. This restriction is due to confidentiality issues and to protect the privacy of the institution and its users, as mentioned in the article.

ACKNOWLEDGMENTS

The authors thank the University and their Information Technology Department for support in the development of this research. Collaboration and access to institutional data were essential for the successful execution of the study. This work was carried out with the support of the CAPES through the Academic Excellence Program PROEX.

REFERENCES

[1] Yoji Akao. 2004. *Quality Function Deployment: Integrating Customer Requirements into Product Design* (1st ed.). Productivity Press. 392 pages.
[2] Fernando Almaraz-Menéndez, Alexander Maz-Machado, Carmen López-Esteban, and Cristina Almaraz-López. 2022. *Strategy, Policy, Practice, and Governance for AI in Higher Education Institutions.* IGI Global. 334 pages.
[3] Pritika Bahad, Dipti Chauhan, and Diksha Bharawa. 2022. Sentiment Analysis of Helpdesk Calls: Enhancing Customer Support through Natural Language Processing. Preprint.
[4] Yiding Cao, Yingjun Dong, Minjun Kim, Neil G. MacLaren, Sriniwas Pandey, Shelley D. Dionne, Francis J. Yammarino, and Hiroki Sayama. 2023. Visualizing Collective Idea Generation and Innovation Processes in Social Networks. *IEEE Transactions on Computational Social Systems* 10, 5 (2023), 2234–2243. doi:10.1109/TCSS.2022.3191885
[5] Lai-Kow Chan and Ming-Lu Wu. 2002. Quality function deployment: A literature review. *European Journal of Operational Research* 143, 3 (2002), 463–497. doi:10.1016/S0377-2217(02)00178-9
[6] Yichen Cheng, Xinlei Wang, and Yusen Xia. 2021. Supervised t-Distributed Stochastic Neighbor Embedding for Data Visualization and Classification. *INFORMS Journal on Computing* 33, 2 (2021), 419–835.
[7] Mike Cohn. 2004. *User Stories Applied: For Agile Software Development.* Addison-Wesley Professional, Boston, MA.
[8] Kia Dashtipour, Soujanya Poria, Amir Hussain, Erik Cambria, Ahmad Y. A. Hawalah, Alexander Gelbukh, and Qiang Zhou. 2016. Multilingual Sentiment Analysis: State of the Art and Independent Comparison of Techniques. *Cognitive Computation* 8 (2016), 757–771.
[9] Andreia de Bem Machado, Maria José Sousa, Francesca Dal Mas, Silvana Secinaro, and Davide Calandra. 2024. *Digital Transformation in Higher Education Institutions.* Springer Nature. 232 pages.
[10] José Pedro de Santana Neto, Jaison Pisa Rezine, Cleiton Almeida dos Santos, Marcelo Massarelli Maitan, and Felipe Sanches Bueno. 2025. Mapeamento de Desafios Operacionais no SIGA-UFPR: Uma Análise usando Inteligência Artificial. In *WTICIFES 2025 – Workshop de Tecnologia da Informação e Comunicação das Instituições Federais de Ensino Superior* (Santarém, PA, Brasil).
[11] Shelley D. Dionne, Hiroki Sayama, Chih-Wei Hao, and Benjamin J. Bush. 2010. The Role of Leadership in Shared Mental Model Convergence and Team Performance Improvement: An Agent-Based Computational Model. *The Leadership Quarterly* 21, 6 (dec 2010), 1035–1049. doi:10.1016/j.leaqua.2010.10.007
[12] I. Erikkson and F. McFadden. 1993. Quality function deployment: a tool to improve software quality. *Information and Software Technology* 35, 9 (1993), 491–498. doi:10.1016/0950-5849(93)90016-V
[13] George H. Fairbanks. 2010. *Just Enough Software Architecture: A Risk-Driven Approach.* Marshall Brainerd.
[14] T. Hastie, R. Tibshirani, and J.H. Friedman. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer. https://books.google.com.br/books?id=eBSgoAEACAAJ
[15] Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-Strength Natural Language Processing in Python. *Zenodo* (2020). doi:10.5281/zenodo.1212303
[16] HuggingFace. 2024. Multilingual Sentiment Analysis. https://huggingface.co/tabularisai/multilingual-sentiment-analysis Accessed: 4 Apr. 2025.
[17] Asmaul Husna, Habibur Rahman, and Emrana Kabir Hashi. 2019. Statistical Approach for Classifying Sentiment Reviews by Reducing Dimension Using Truncated Singular Value Decomposition. In *1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT).* IEEE Xplore. doi:10.1109/ICASERT.2019.8934507 Accessed: 2025.
[18] International Organization for Standardization. 2011. ISO/IEC 25010:2011 Systems and Software Engineering – Systems and Software Quality Requirements and Evaluation (SQuaRE) – System and Software Quality Models. https://www.iso.org/standard/35733.html. Accessed: 2025-06-27.
[19] ISO/IEC/IEEE. 2018. *ISO/IEC/IEEE 29148:2018. Systems and Software Engineering — Life Cycle Processes — Requirements Engineering.* Technical Report. International Organization for Standardization, Geneva, Switzerland. https://www.iso.org/standard/72089.html
[20] ISO/IEC/IEEE. 2022. *ISO/IEC/IEEE 42010:2022. Software, Systems and Enterprise — Architecture Description.* Technical Report. International Organization for Standardization, Geneva, Switzerland. https://www.iso.org/standard/74393.html
[21] Jayusman and Riyanto Jayadi. 2024. Sentiment Analysis of Digital Customer Service Helpdesk Tickets: A Comparison between BERT and SVM. In *2024 International Conference on Computer Engineering, Network, and Intelligent Multimedia (CENIM).* 1–6.
[22] X. F. Liu. 2000. Software quality function deployment. *IEEE Potentials* 19, 5 (Dec. 2000), 14–16. doi:10.1109/45.890072
[23] Edward Loper and Steven Bird. 2002. NLTK: The Natural Language Toolkit. http://arxiv.org/abs/cs/0205028
[24] Miltiadis D. Lytras, Afnan Alkhaldi, Sawsan Malik, Andreea Claudia Şerban, and Tahani Aldosemani. 2024. *The Evolution of Artificial Intelligence in Higher Education: Challenges, Risks, and Ethical Considerations.* Emerald Group Publishing. 321 pages.
[25] Sara A. McComb. 2007. Mental Model Convergence: The Shift from Being an Individual to Being a Team Member. In *Research in Multi-Level Issues.* Vol. 6. Emerald Group Publishing Limited, 95–147. doi:10.1016/S1475-9140(07)06009-9
[26] El Mahdi Mercha, Houda Benbrahim, and Mohammed Erradi. 2024. Heterogeneous Text Graph for Comprehensive Multilingual Sentiment Analysis: Capturing Short- and Long-Distance Semantics. *PeerJ Computer Science* 10 (2024), e1876.
[27] Okechukwu Emmanuel Okonta, Adimabua Arnold Ojugo, Wemembu Uchenna Raphael, and Dele Ajani. 2013. Embedding Quality Function Deployment In Software Development: A Novel Approach. *West African Journal of Industrial & Academic Research* 6, 1 (March 2013).
[28] Simone Perazzoli, Aastha Joshi, Sayana Ajayan, and José Pedro de Santana Neto. 2022. Evaluating Environmental, Social, and Governance (ESG) from a Systemic Perspective: An Analysis Supported by Natural Language Processing. *SSRN*

(2022). doi:10.2139/ssrn.4244534

[29] Simone Perazzoli, Wagner Santos, and Jose Pedro de Santana Neto. 2022. Towards a Systemic-Based Automated Platform Designed to Foster Collaborative and Co-Responsibility Environment. doi:10.2139/ssrn.4249435

[30] Simone Perazzoli, Wagner Sousa Santos, and Jose Pedro de Santana Neto. 2022. CONSTELADEV: An Innovative Platform Based on Systemic Methodologies to Support Teams and Project Management. Poster presentation.

[31] Klaus Pohl and Chris Rupp. 2022. *Requirements Engineering: Fundamentals, 2nd Edition.* IREB Foundation Level Study Guide.

[32] Roger S. Pressman. 2010. *Software Engineering: A Practitioner's Approach* (7 ed.). McGraw-Hill Education.

[33] Kristina P. Sinaga and Miin-Shen Yang. 2020. Unsupervised K-Means Clustering Algorithm. *IEEE Access: Practical Innovations, Open Solutions* 8 (2020), 80716–80727.

[34] Fernando Antonio Sonda, José Luis D. Ribeiro, and Márcia Elisa Echeveste. 2000. A aplicação do QFD no desenvolvimento de software: um estudo de caso. *Production* 10, 1 (2000), 51–75. doi:10.1590/S0103-65132000000100004

[35] François Staring. 2022. Digital Higher Education: Emerging Quality Standards, Practices and Supports. doi:10.1787/f622f257-en

[36] Douglas Steinley and Michael J. Brusco. 2011. Choosing the Number of Clusters in K-Means Clustering. *Psychological Methods* 16, 3 (2011), 285–297.

[37] Dumitru-Tudor Tolciu, Christian Săcărea, and Cristian Matei. 2023. Analysis of Patterns and Similarities in Service Tickets Using Natural Language Processing. *Journal of Communications Software and Systems* 17, 1 (2023). https://jcoms.fesb.unist.hr/pdfs/v17n1_1024_tolciu.pdf