Article

A Review of Non-Functional Requirements Analysis Throughout the SDLC

Cyrille Dongmo

Computer Science Department, School of Computing, College of Science, Engineering and Technology (CSET), Science Campus, University of South Africa (UNISA), Florida Park, Roodepoort 1709, South Africa; dongmc@unisa.ac.za

Abstract: To date, unquestionable efforts have been made, both in academia and industry, to facilitate the development of functional requirements (FRs) throughout the different phases of the software development life cycle (SDLC). Functional requirements are understood to mean the users' needs pertaining to the services to be rendered by a software system. For example, semi-formal or graphically based approaches such as UML, and mathematically based or formal approaches such as Z and related tools have all been developed with the intention of addressing FRs. In the same vein, most of the proposed software methodologies, for instance, agile software development and model-driven software development, primarily target functional requirements. Considering the importance and even the criticality of non-functional requirements describing the quality of software systems and the constraints upon them, similar progress would be expected for their development. However, it appears that making headway with NFRs has been more challenging due to the complexity of the requirements. In this regard, the main purpose of this work is to unveil (from the academic perspective) the current state of development of NFRs through the review of publications carefully selected from five online databases.

Keywords: non-functional requirements; NFRs analysis; NFRs development; software engineering; SDLC; requirements engineering; systematized literature review; SLR

check for updates

Citation: Dongmo, C. A Review of Non-Functional Requirements Analysis Throughout the SDLC. *Computers* **2024**, *13*, 308. https://doi.org/10.3390/computers13120308

Academic Editor: Paolo Bellavista

Received: 13 September 2024
Revised: 7 October 2024
Accepted: 9 October 2024
Published: 23 November 2024

1. Introduction

Software development is known to be a complex task comprising various development phases, each of which requires some level of specific expertise [1,2]. Thus, far, various methodologies, techniques and tools have been developed to further the engineering of software systems; to reduce the cost and time of production, especially for complex systems; and/or to improve the quality of the final software product. This brings forth the two fundamental and complementary building blocks of software engineering: the functional requirements (FRs) and non-functional requirements (NFRs).

Functional requirements describe the services to be rendered by the software (in construction), while NFRs define the quality of the service, as well as the user interface and constraints on how the services are to be produced and rendered [3,4]. So, in service-level agreement contract documents, quality of service (QoS) parameters, also known as non-functional parameters, are used to specify the minimum quality of service level that must be guaranteed by a system. To date, the major progress in software engineering is (said to be) largely related to FRs. For instance, the unified modeling language (UML), one of the popular semi-formal (graphical) modeling approaches of the object-oriented software development methodology, does not inherently integrate the mechanisms for NFRs. The same is true with other methods such as agile [5], model-driven software development (MDSD) [6], and the established formal methods (FMs) such as the Z notation [7]. Nevertheless, for the past few decades, efforts have been made to further the development of NFRs. For example, the goal-oriented requirements engineering (GORE) methods [8–10], starting with the genesis of the NFR framework [11], have revolutionized the engineering of NFRs by bringing together the analysis of FRs and NFRs within the same (goal) model. Approaches extending the existing methods for FRs or combining them to model and analyze NFRs have also been derived. It is noted in the first place, the enrichment of UML with new concepts using UML profiles to develop new techniques such as SysML and MARTE [12]. As a UML, the systems modeling language (SysML) is an OMG standard that has been used to support systems engineers in various fields including production, aerospace, mechatronics, automotive, energy, defense, safety and robotics [13]. On the other side, MARTE, which is another OMG-standardized modeling language, is more dedicated to the modeling and analysis of real-time and embedded systems.

Nonetheless, in many publications on NFRs, statements indicating that NFRs have been neglected, overlooked or poorly developed because of their complexity, are still frequently encountered.

Thus, the intention of this article is to establish the current state-of-the-art in the analysis/development of NFRs with the goal of pinpointing the progress made so far and developing a better idea of what is still to be achieved. The main research question this work seeks to answer is therefore the following:

What is the current state of non-functional requirements development throughout the software development life cycle? (RQ)

From the main question, the following two sub-questions are derived:

- What approaches or mechanisms are there for the processing of NFRs? (RQ1)
- Can a process be derived from existing NFRs approaches that span part or the entirety of SDLC? (RQ2)

The main contributions of this article are as follows:

1. The establishment of the current level of analysis and development of non-functional requirements throughout the SDLC. By so doing, this article sheds light on the research gap in the field of NFRs analysis and hence provides means to derive future work, including the possibility of master's and doctoral research topics.
2. This article has equally endeavored to unpack the methods and techniques that have been used to develop various means for NFRs analysis as well as the roles played by such techniques and methods in the development of NFRs.
3. This work has also examined the different phases of the software development life cycle for which approaches have been proposed for the analysis and development of NFRs. For each SDLC phase, the activities of such approaches have been identified and discussed.
4. This work has enabled us to elicit a number of future works to further research in the area of NFRs analysis and development.

Following the above introduction with the research questions, the research methods used in this article are presented in Section 2, including the research process in Section 2.1, the search string with the selected online databases in Section 2.2, the inclusion/exclusion criteria in Section 2.3 and the search results presented in Section 2.4. Section 3 uncovers and examines the role of the existing methods and techniques used in the approaches proposed for the analysis of non-functional requirements. Meanwhile, the current state of NFR development within the software development life cycle (SDLC) is discussed in Section 4, with the focus on the requirements engineering activities in Section 4.1, as well as the software design in Section 4.2 and the implementation in Section 4.3. Thereafter, Section 5 presents the research results, followed by the discussion, the options for future work, and the threats to validity in Section 6. Finally, Section 7 concludes the article and is followed by a list of references.

2. The Research Method

In line with the research onion proposed by Saunders et al. [14], which is illustrated in Figure 1 and unpacked from the outer layer to the inner one and from left to right, this work is more in line with interpretations of research philosophy, owing to the qualitative analysis of the carefully selected research publications. The research approach is inductive with the mono method, with qualitative analysis as the main research methodology, as in this article, a systematized literature review is conducted to identify the publications from which appropriate data are extracted and qualitatively analyzed to seek answers to the research questions.

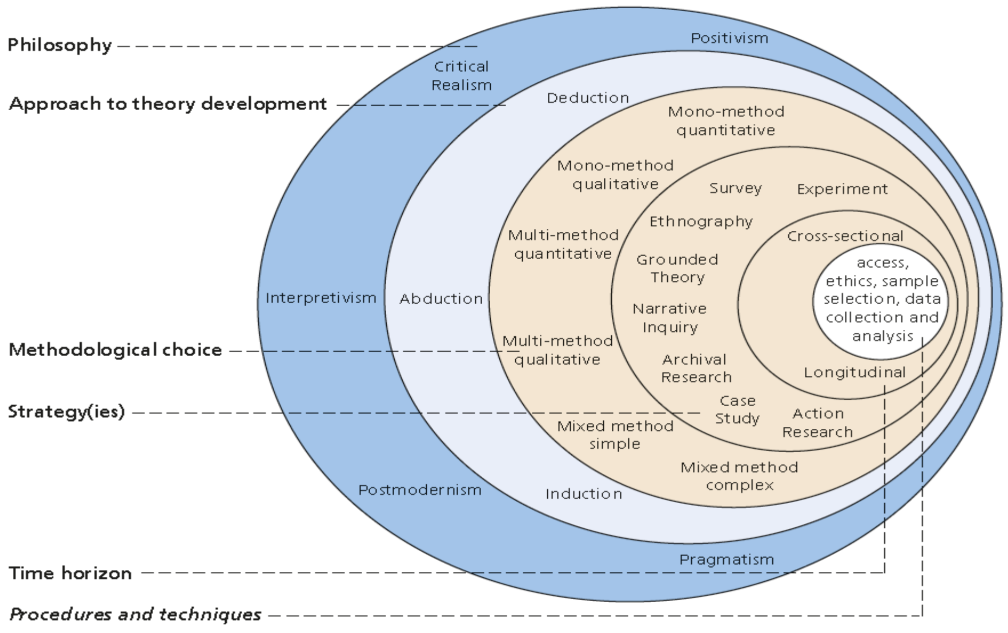

Figure 1. The research onion [14].

The data manipulated in this work were extracted from carefully selected journal articles and conference papers on non-functional requirements and software development activities. The literature review is therefore an important part of the research process. The documents considered are those published or available in the specified time period as this piece of work is intended to be a cross-sectional study.

2.1. The Research Process

To answer the research questions, a systematized literature search of publications in five selected online databases was conducted (as listed in Table 1). These databases are amongst the most frequently solicited by researchers in the field of computer science and information technology. The search string used to query the databases was formed by combining the keywords generated from the research questions. Due to the particularities and the way each database interprets a query, the string was modified slightly to suit the requirements of the respective databases. Depending on the results of the trials, the final selection was performed on the article's title and/or the abstract and/or its keywords. The selected articles were filtered using the inclusion/exclusion criteria. The remaining publications, those that explicitly addressed the development of non-functional requirements, were synthesized to answer the research questions.

In addition, the query used in each online database was also used to query both the Google Scholar and Springer Link databases to ensure the comprehensiveness of the search. The main purpose is to investigate the extent to which the development of NFRs has been integrated into the traditional software development life cycle (SDLC).

Table 1. List of the selected databases.

Database	Web Address via the University's Library
ACM Digital Library	https://0-dl-acm-org.oasis.unisa.ac.za/dl.cfm (accessed on 11 April 2023)
ScienceDirect	https://0-www-sciencedirect-com.oasis.unisa.ac.za/ (accessed on 11 April 2023)
Scopus	https://0-www-scopus-com.oasis.unisa.ac.za/ (accessed on 11 April 2023)
IEEE Xplore	https://0-ieeexplore-ieee-org.oasis.unisa.ac.za/Xplore/home.jsp (accessed on 11 April 2023)
Web of Science	https://0-webofknowledge-com.oasis.unisa.ac.za/WOS (accessed on 11 April 2023)

2.2. The Search String and the Online Databases

After a number of trials, the two keywords that were retained for non-requirements are "Non-functional requirements" and NFRs. Initially, other keywords such as "quality attributes", "quality requirements", and "non-functional properties" were added to the list, but they did not yield any positive contributions during the trial phase. The second set of keywords are specification, analysis, and modeling. The two sets of keywords were combined to form the following generic search string:

("non-functional requirement*" OR NFR*) AND (specification OR analysis OR model*).

As indicated earlier, depending on the database, this string was adapted to suit the requirements of the database. The databases were chosen amongst those mainly encountered in the literature reviews within the discipline of computer science and information systems. The list of databases chosen are presented in Table 1.

The databases were accessed via the online library of the University of South Africa and the links used are those given in the second column of Table 1.

2.3. The Inclusion and Exclusion Criteria

Due to time constraints for this study, only articles published between January 2013 and April 2023 were included. Although this time range could affect the number of relevant works, the final list of 34 relevant works was considered acceptable for the purposes of this study. More importantly, only publications that propose guidelines, a framework, a model and/or any other approach to develop or evaluate NFRs in software development or to integrate the analysis of NFRs into existing methods or techniques, were retained for the study. Non-English-language posters and publications were excluded, as well as those tangential to the analysis of NFRs. The search results and the selection process for the publications included are summarized in the next section.

2.4. The Search Results

Firstly, an automated search of the selected computer science and software engineering databases was undertaken. Secondly, the Google Scholar and Springer Link databases were used as a means to ensure the comprehensiveness of the first search. The number of publications obtained from each database, as well as the date and time of the search, are presented in Table 2. A total of 270 publications were extracted during the first search.

Table 2. Search Results per Database.

Database	Date Searched	Publications
ACM Digital Library	11 April 2023, 12:09:14	48
ScienceDirect	11 April 2023, 12:30:15	85
Scopus	11 April 2023, 14:46:57	51
IEEE Xplore	11 April 2023, 20:35:21	41
Web of Science	11 April 2023, 21:47:33	45
Total publications		270

The inclusion and exclusion criteria were progressively applied to filter the initial list of publications. For each search, a bibtex document with the search result was generated and exported to a specific folder in Mendeley Desktop Version 1.19.8. As illustrated in Figure 2, the filtration process was undertaken in Mendeley starting with the selection based on the articles' titles and abstracts which resulted in a reduction from the initial list of publications to 67. This was followed by the screening of the content of each article, with a focus on the methodology, discussion, and conclusions that led to the exclusion of 33 publications. A total of 34 publications were therefore retained for the study. This comprises 21 (representing 61.8%) studies, which were published in conference proceedings, 13 (38.2% of 34) of which were journal articles.

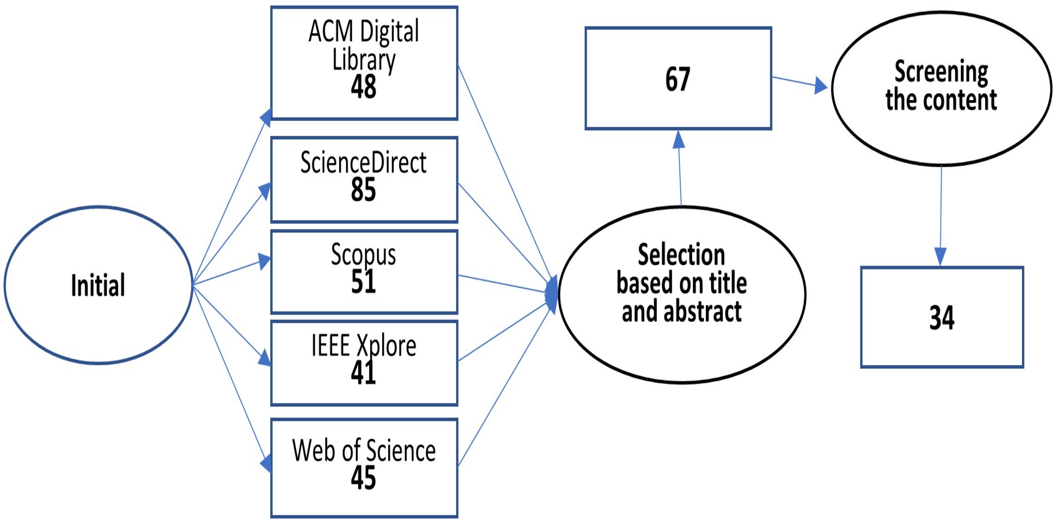

Figure 2. The selection process.

The number of publications per year and per type of publication are shown in Figure 3. The blue line represents the number of papers published in conference proceedings, and the red line represents journal articles.

The majority of the conference papers, (8 in total representing 38.1% of all the conference papers), were published in 2013. The remaining 13 were published from 2014 to 2020 at a rate of about 1 or 2 papers per year, with no publications from 2021 to 2023. From 2013 to 2022 (excluding 2016 and 2017), the number of journal articles published per year is approximately 0 or 1. The highest number, 5, representing about 38% of the total number of the articles, occurred in 2023, followed by 2 articles published per year in 2016 and 2017.

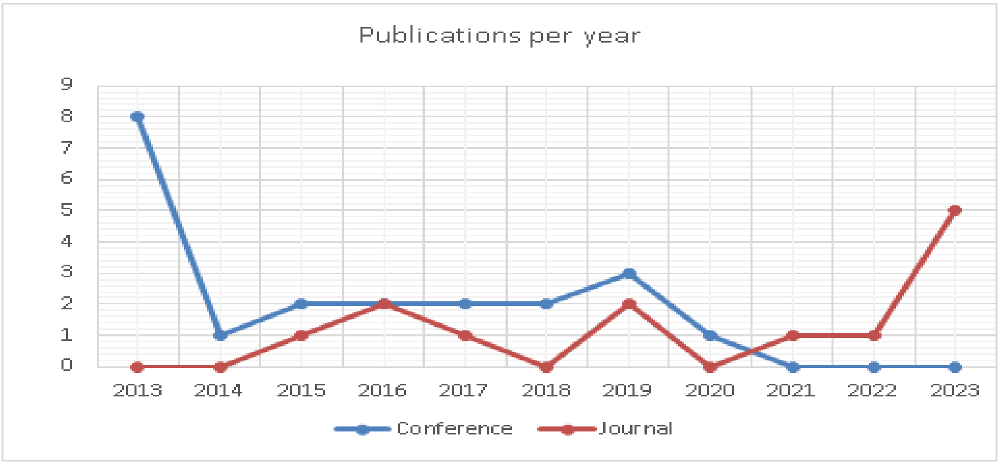

Figure 3. The publications per year from 2013 to 2023.

Importantly the study of the selected publications revealed that techniques and methods currently employed in software engineering are frequently used or adopted to analyze NFRs. Amongst others, these include, UML, XML, machine learning/deep learning algorithms and mathematically based approaches to software engineering, etc. Thus, the next section elaborates further on how these techniques are used and what aspects of NFR development they address.

3. Methods and Techniques Used in the Analysis of Non-Functional Requirements (NFRs)

The selected software engineering approaches used or adopted for the development of NFRs include the GORE methods, unified modeling language (UML), mathematically based approaches [15], extensible markup language (XML) [16], and machine learning. Algorithms and available datasets are also included. The chart in Figure 4 depicts the number of publications per technique.

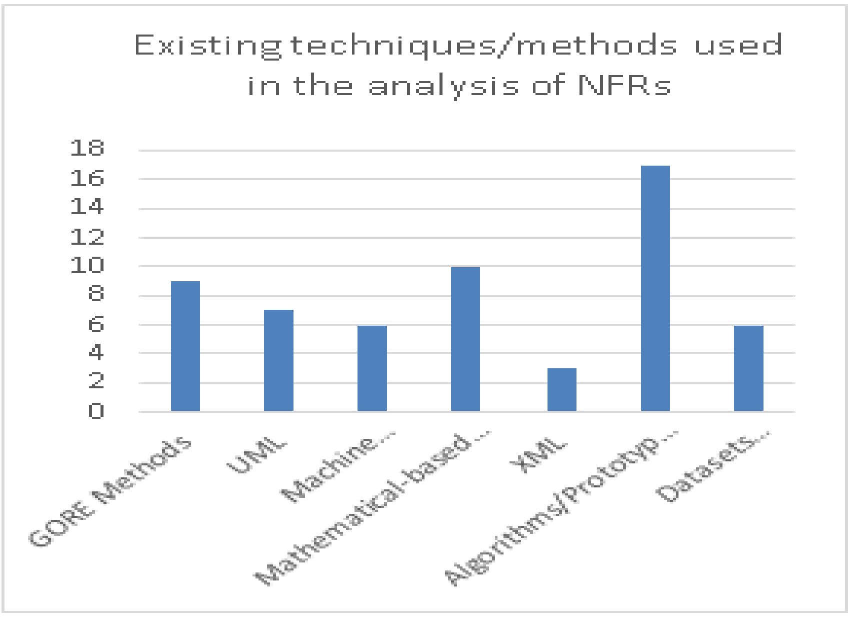

Figure 4. Statistics on techniques re-used in NFRs development.

3.1. GORE Methods

Goal-oriented requirements engineering (GORE) methods are the revolutionary requirements engineering frameworks that combine the analysis of goals describing functional requirements and soft goals describing non-functional requirements [17]. These include, among others, the non-functional requirement (NFR) framework [11], knowledge acquisition in automated space (KAOS) framework [18], istar (or i*) framework [19], Tropos/Secure-Tropos [20,21], goal-oriented requirement language (GRL) [22], etc. The central concept is that of the soft goal interdependency graph (SIG). A SIG encapsulates both the goals and soft goals refinement and their operation. A refinement decomposes intentional elements (goal, soft goal, etc.) into more detailed ones, using the And/Or decomposition concept. Whereas operation provides the means for satisfying goals and soft goals, it also uses the concept of links to establish the relationships between the different elements in the graph, helping to model and propagate the impact of a given solution on other intentional elements (goal, soft goal, actor, etc.). Another important aspect is the possibility of applying an evaluation (or propagation) algorithm to the SIG to determine the level of (soft) goal satisfaction for a given strategy.

As shown in the chart in Figure 4, 9 of the 34 selected publications for this study (26.5%) used a GORE method in their treatment of NFRs, most of which either consider the SIG model as a direct input to their proposed method or first transform or extend it with new concepts before using it.

In that regard, it is noted that the extension of the NFR framework into a directed graph optimizes the operation of soft goals to select the best implementation strategy [23]. DeVries et al. [24] proposed a three-step NFR analysis tool, namely SOTTER, that introduces new concepts for soft goal mitigation to the existing goal model (KAOS) to facilitate the detection of soft goal violation and their mitigation. Goncalves and Krishna in [25] proposed an extension of the NFR framework model with the weighting of soft goals and their operations to facilitate the generation of software agents in a model-driven agent development executed in a dynamic environment. Arif et al. in [26] suggested a method for the development of FRs modeled with UML, NFRs and the NFR framework where they applied fuzzy logic to the SIG model to accommodate the inherently vague and imprecise nature of SIG links. Conversely, Mehta et al. in [27] further the dependency link analysis in the SIG model to improve the effective strategy selection to satisfy soft goals. Based on the 14 privacy modeling capabilities as proposed by Peixoto et al. [28], these authors demonstrated that by comparing i*, the NFR framework, and Secure-Tropos, these languages do not fully support the modeling of privacy requirements. Lastly, Dermeval et al. in [29] borrowed NFR framework concepts to propose a framework for documenting architectural decisions.

By comparing the 4 selected GORE methods using the proposed features required for the specification of safety-critical systems (SCSs), Vilela et al. [30] demonstrated that GORE methods are not all equally rich in concepts necessary for the specification of certain specific systems (such as safety-critical systems). Although in [28] it is recommended to use a GORE method for the specification of privacy requirements, the conceptualization of the 14 features (for the meta-model) required for the specification of such requirements is performed with a UML class model. Similarly, in [29], the authors use the UML class diagram to conceptualize a meta-model for documenting the architectural design decisions based on i* model of NFRs.

3.2. Using UML Modeling for NFRs

Unified modeling language (UML) [31,32] is a standardized software modeling language that has gained considerable popularity during the past few decades. Although very rich in concepts pertaining to the modeling and design of the systems' functionalities, it does not inherently address NFRs. However, UML profiles [33] allow for the extension of existing models with stereotypes that have largely been used, amongst others, to support NFRs.

As depicted in Figure 4, 7 of the 34 selected publications (20.6%) utilized UML in their proposed NFRs analysis approach. These include publications in which EMF Ecore models or meta-models are used. Such models can be considered sub-models or refined models of UML.

In [34], the authors favored the use of a UML profile to model NFRs, models to which they applied fuzzy logic for NFRs trade-off analysis pertaining to the optimal satisfaction of NFRs. On the other hand, in [26], an alternative approach is suggested to associate NFRs from an NFR framework with UML use case diagrams modeling FRs. In the same vein, Kaur and Sharma in [35] believe in associating NFRs to use cases describing the functionalities of the system and in so doing, extending UML use case diagrams to integrate NFRs. Martinez et al. [36] suggested an approach to detect conflicts and dependencies in NFRs using UML scenarios and use cases describing the functionalities of the system. In [37], Khalique et al. proposed EMF Ecore meta-models to generate domain NFRs from product line NFRs using the MDSD transformation approach. Although not utilized to represent or analyze NFRs, in [28], a UML class diagram was used to model a privacy conceptual foundation to enable a comparison between i*, NFR framework and Secure-Tropos to determine which of the three was more appropriate to model privacy requirements. Similarly, a UML meta-model is proposed in [29] to enable the documentation of the architectural design decisions relating to the i* model of NFRs.

3.3. The Use of Machine Learning (ML) Techniques for the Analysis of NFRs

In regard to machine learning, this study has considered all articles that utilize ML algorithms [38], deep learning (DL) algorithms [39], and/or any natural language processing (NLP) tools. These algorithms have the merit of enabling computers to learn from data, without being explicitly programmed in order to make better predictions or decisions. Rooted in conventional neural networks, deep learning (DL) (a subset of ML) has outperformed traditional ML relying on statistical models [40] in various applications including NLP tasks: extraction of information from large documents, text classification, etc.

One of the challenges with NFR analysis is to extract such requirements from an often-large requirement specification document, written in natural language, and classify them into different categories. To overcome the manually intensive efforts and its inherent high probability of errors, the use of ML/DL algorithms has been solicited. Rashwan et al. [41] proposed an ontology annotation of NFR types and the ML algorithms known as a support vector machine (SVM) classifier to automatically categorize NFR sentences into their ontology classes. In [42], an improved DL approach, namely BERT-CNN, is proposed for the extraction and classification of NFRs. Similarly, in [43], it is the recurrent neural network (RNN) that is solicited for the classification of NFRs. On the other hand, Yahya et al. [44] preferred a hybrid deep learning model that combines an RNN model and 2 LSML to identify and classify NFRs solicited from user comments for mobile apps. Alashqa, in [45], opted to study the commonalities, mappings and relationships between non-functional requirements, by analyzing the results of 5 ML algorithms applied to an existing requirements dataset. The 5 ML models are k-nearest neighbor (KNN), logistic regression (LR), support vector machine (SVM), multinomial naive Bayes (MNB) and random forest (RF). Lastly, Singh et al. [46] proposed a rule-based system for automated classification of non-functional requirements from requirement specifications, using various NLP tools.

Thus, 6 out of the 34 publications (17.6%) selected for this study have suggested the use of ML algorithms and NLP tools to extract and classify NFRs from requirements documents.

3.4. Using Mathematically Based Methods to Analyze NFRs

Although ML algorithms are essentially based on mathematical models, they have been treated separately since they constitute well-established techniques in software engineering. Under the mathematically based methods, all the articles that include mathematical concepts, other than those of ML/DL, to analyze NFRs, have been included. Amongst others, these models or formulas are based on ontology, fuzzy logic, set theory, formal logic, and so forth. These approaches are mainly utilized for trade-off analysis of conflicting NFRs, their prioritization, and the selection of the strategies being those that best realize or satisfy the soft goals describing NFRs.

As shown in the graph in Figure 4, 10 of the 34 (29.4%) selected publications utilized some mathematics in their methods. To manage conflicts between NFRs, in [47], a framework based on the ontology model, namely sureCM, is proposed to specify the system's requirements and the parameters necessary to characterize the conflicts and facilitate decision making. In [23], Affleck et al. suggested an extension of the NFR framework's SIG into a directed graph to allow the application of set theory operations and arithmetic formulas for calculations to select operations that optimize NFRs satisfaction. Rashwan et al. [41] constructed an ontological model of NFRs to which they applied ML classifier algorithms to classify the NFRs. Saadatmand and Tahvili [34] applied fuzzy logic to an NFR model constructed with a UML profile for a trade-off analysis to help identify different design alternatives that lead to higher overall satisfaction of NFRs in the system. Conversely, Arif et al. [26] proposed an extension of the NFR framework model with fuzzy concepts to address the vagueness and imprecision of the contribution links, hence facilitating the analysis of the impact of an operation on other NFRs. Similarly, Zhang and Wang [48] uses fuzzy set theory to perform a trade-off analysis of conflicting NFRs. Paucar and Bencomo [49] proposed a mathematical probabilistic framework, based on the partially observable Markov decision processes (POMDPs) models, for decision-making during runtime to satisfy NFRs. To facilitate the identification of conflicting NFRs and their prioritization, Shah et al. [50] also utilized an ontology model for NFRs. In the same vein, Liu [51] proposed a conflict detection tool based on ontology for NFRs evolution. Last but not least, Luangwiriya and Kongkachandra [52] applied existential logic and the rules of inferential logic to conceptual graphs, representing NFR frames, to automate the detection of conflicting NFRs.

3.5. The Role of the Extensible Markup Language (XML) in the Development of NFRs

A total of 3 (about 9%) of the 34 selected publications used XML in their NFRs analysis. Almeida et al. [53] proposed an XML specification of the features model of a software product line to suggest a proposed decision algorithm to identify the best available application configuration (BSolution), based on the given NFRs specification, that best satisfies the NFRs, while Shah and Patel [54] used an XML specification of requirements description schemas which aimed to facilitate the requirements' transferability, automation, storage and traceability. Lastly, in [52], an XML model of NFRs extracted from a requirement specification document, and representing construction of conceptual graphs for NFRs, is used to detect conflicts in NFRs.

3.6. Algorithms, Prototypes and Simulations in the Development of NFRs

In this category, the publications in which algorithms, prototypes, simulations or existing software tools were included in their proposed NFRs analysis approach were considered. A total of 17 (50%) of the 34 publications selected for this study were identified. These publications are mainly those in which machine learning approaches or mathematically based solutions are used for the treatment of NFRs. In general, the algorithms are proposed as a means to process the suggested NFR models or to implement a given method. Prototypes and tools are used either for illustration or for testing.

3.7. The Role of Existing Datasets: Promise and Concordia RE Corpus in the Development of NFRs

Two datasets were encountered in this study: the PROMISE and Concordia datasets. The PROMISE corpus [55] is a research dataset repository that contains a total of 326 NFRs and 358 FRs. The Concordia RE corpus was created by [41] for their work. It contains 3064 requirement sentences manually sorted into four main classes: FR, external and internal quality, constraints and other NFRs. These datasets are mainly used to test ML/DL algorithms used for NFRs analysis, especially the extraction and classification of NFRs from large SRS. The tests are performed to determine the performance of the ML model in terms of the precision, recall and F1-score measures that provide quantitative assessments of the model's predictive capabilities [56].

Although most of the NFR approaches proposed in the selected publications do not target a specific known software development methodology, some do, which are discussed in the next section.

3.8. NFRs Analysis for Specific Software Development Methodologies

In 12 of the 34 publications (35.1%) selected for this work, the approaches proposed for NFR analysis are specific to a software paradigm or methodology. These were identified as agile [57–60], model-driven development (MDD) [25,34,37,61], aspect-oriented software development (AOSD) [35], software product line (SPL) [37], self-adoptive systems (SASs) [49] and service-oriented architecture (SOA) [62]. Figure 5 shows the percentages of the 12 publications that opted for each respective methodology.

As shown in Figure 5, agile and MDD methods were adopted in 4 publications, respectively, representing 34% of 12. These two methodologies do not inherently integrate mechanisms for NFRs development into their processes and, despite their popularity, more research efforts are still required for them to fully embrace NFRs. Each of the four other approaches was considered by only 1 publication out of 12 (8%).

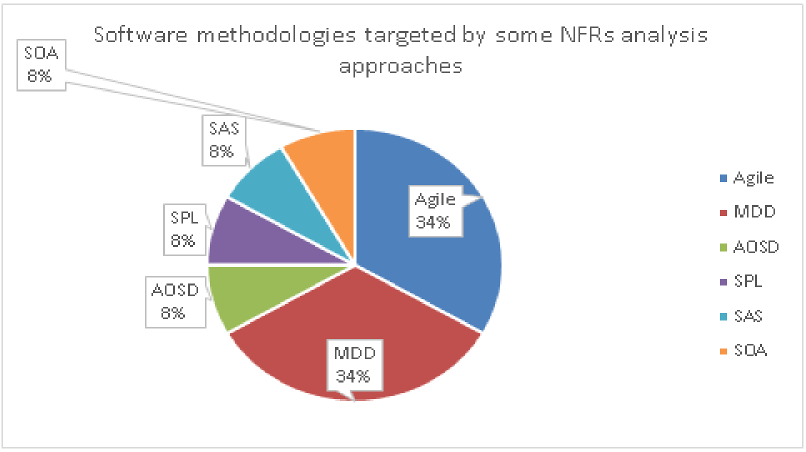

Figure 5. Statistics on methodologies targeted in NFRs analysis approaches.

4. The Analysis of NFRs in SDLC

In general, authors point out four fundamental phases for the SDLC, although they use different terminologies to describe them [1,63,64]. Since this work aims to investigate the progress made towards the development of NFRs throughout the SDLC, only the SDLC phases for which the activities have been addressed in the selected publications are here considered. For each phase, only the activities covered in the selected publications are considered. Thus, only five activities for the requirements engineering (RE) phase are identified, and only two for the software design (SD) phase. The activities for the RE phase are NFR elicitation, NFR primary analysis, NFR specification and modeling, NFR conflicts and trade-off analysis and NFR strategies selection. The two SD activities are design decisions/transformation and documentation/traceability. No activity could be linked to the software implementation phase.

Figure 6 shows the statistics of the number of publications per NFRs activity. It can be seen that, in general, each publication covers more than one activity that does not necessarily belong to the same NFRs' respective development phase, although it appears that most of the researchers' efforts have been concentrated on the requirements engineering phase. As many as 32 of the publications (94% of 34), focused on the requirements engineering phase, while only 4 (less than 12%) addressed the software design phase [29,35,37,53]. Two of these publications discussed the activities in both the RE and SD phases [37,53].

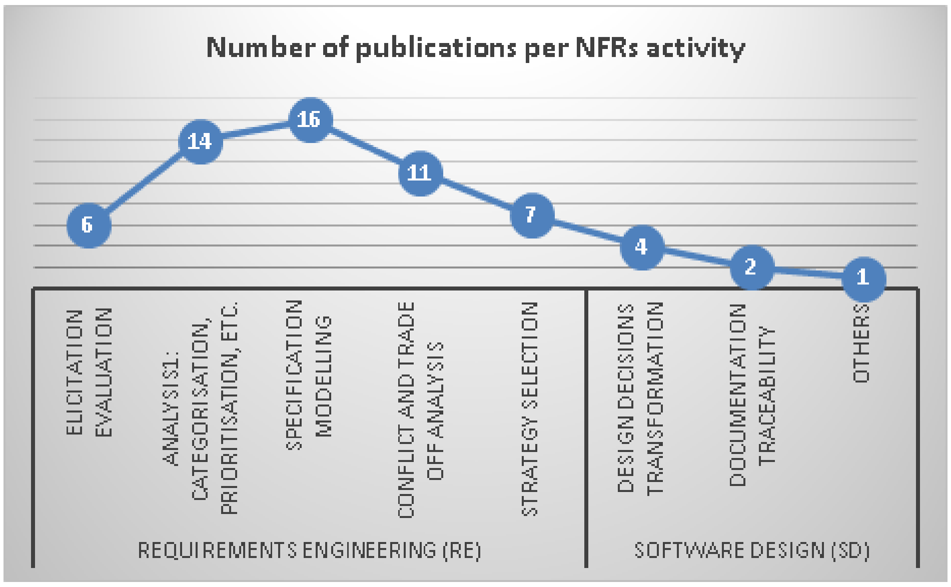

Figure 6. Statistics on publications per NFRs activities.

4.1. NFRs Requirements Engineering

In Figure 6, it can be seen that the NFR specification and modeling is the most popular activity addressed in 16 publications, followed, respectively, by the requirements categorization and prioritization (14), the NFR conflict(s) detection and trade-off analysis (11), selection of the best strategy for satisfying NFRs (7) and the requirements elicitation and evaluation (6).

4.1.1. NFRs Elicitation and Evaluation

The requirement elicitation consists, for the software engineer in collaboration with other stakeholders, of uncovering and gathering the users' true needs by identifying, searching or interrogating all the possible requirements' sources. It is the first and only of the more complex RE activities whose purpose is to produce a robust software requirement specification (SRS) [65]. At this stage of requirement gathering, it is, in general, not always possible to distinguish between the functional and non-functional requirements and, hence, the activity is common to both types of requirements. This may be one of the reasons why the present research could not find many publications on the NFRs requirements elicitation since the focus was merely on NFRs. However, in industry, the lack of NFRs elicitation was also observed by Werner et al. [66] who stated:

> While our study did not directly observe the elicitation of NFRs at the three organizations, the practices we identified were concrete actions these organizations took to support NFR realization and verification [66].

Of the six articles that addressed the requirements elicitation, two of them did not focus on requirement elicitation [26,30], but were counted among the six because they used the requirements elicitation as an intermediate step in their approach. The remaining four focused on agile: Brataas et al. [60] proposed a method for eliciting the scalability requirements in agile. While Meridji et al. [62] developed a framework for eliciting security requirements, Amorndettawin and Senivongse [59] suggested patterns/templates for eliciting and representing NFRs, and Aljallabi and Mansour [57] proposed a six-step process for eliciting and analyzing NFRs in agile based on two existing approaches [67,68]. The relative concentration of the researchers' efforts on agile may be justified by the fact that the methodology is popular, but does not naturally have mechanisms for NFR elicitation and analysis.

4.1.2. NFRs Primary Analysis: Categorization and Prioritization

Depending on the approach and techniques used to assist with the requirements gathering, some preliminary analysis and evaluation of individual requirements can be performed at that early stage. However, amongst other reasons, for the sake of automation, the bulk of the analysis is only possible when an initial list of requirements, generally expressed in natural language (e.g., English), is produced. These include, amongst others, ensuring the correctness, completeness, internal and external consistency of the requirements, classifying and prioritizing them, detecting and resolving conflicts among them, selecting the best strategies for satisfying NFRs, etc.

In this work, the analysis phase is divided into two parts to separate the activities common to both FRs and NFRs and those exclusive to NFRs. As shown in Figure 6, 14 of the selected 34 publications (41%) addressed primary analysis activities [37,41–47,50,52,57,58,69,70] with the bulk of the efforts made toward automating the process of extracting requirements from the requirements document (produced during the requirements gathering phase) and classifying and/or prioritizing them. As discussed in more detail in Section 3, the authors propose approaches that combine various techniques such as fuzzy logic, ontology, machine learning/deep learning, XML, and so forth, to automate the extraction of NFRs from requirements documents, as well as their classification and prioritization.

4.1.3. NFRs Specification and Software Modeling

Although software requirements specification can start as early as the elicitation phase, in general, and in practice, the initial requirements document, compiled at the elicitation phase, is in natural language and contains a mixture of FRs and NFRs, as well as other useful information [4,65]. Thereafter, depending on the approach adopted, the requirements are extracted from the document and specified for further analysis and development. This study observed that beyond the requirements elicitation and evaluation step, any important NFR development, proposed in a publication, precedes a requirement specification using a known or suggested method with its associated modeling language such as use cases for UML or NFR framework for the GORE method. Compared with other activities, this could explain the high number of articles that include requirements specification in their suggested NFRs analysis approaches: 16 of the selected 34 publications (47%).

It was noticed that some of the publications studied in this work utilize more than one requirement specification technique in their proposed NFRs analysis approach; the main approaches encountered and discussed in Section 3 are GORE methods (NFR framework, KAOS, iStar, GRL and TROPOS) [23–28,30,62], UML [26,28,34–37], Ontology [23,26,34,47,50,51] and XML [52,54].

4.1.4. The Second NFRs Analysis: Conflicts Detection, Trade-Off Analysis and Strategy Selection

Although separated in Figure 6, the detection and resolution of conflictual relationships between NFRs, and the selection of the best solution to satisfy NFRs are closely related activities that cannot easily be separated in practice. They count among the important NFRs analysis problems for which different approaches have been proposed, such as ontology-based methods, whereby an ontology model of the requirements is constructed and used as input to the proposed method [47,50,51]. With the goal-based approach, the goal model of the requirements is sometimes transformed or extended with new concepts, and subsequently, a method or framework is proposed to analyze the model [23–25,27]. In the same vein, approaches based on fuzzy logic have also been proposed [34,48], as well as those based on conceptual graphs [52], the partially observable Markov decision process (POMDPs) [49] and the matrix [57].

4.2. NFRs in Software Design

The influence of NFRs on software design decisions to produce software of appropriate quality has long been acknowledged by researchers [71]. Hence, the expectation is for researchers to propose software design processes that fully integrate NFRs and include the documentation and traceability of design decisions. In this regard, 3 of the 34 selected publications (less than 9%) contributed toward at least one of these expectations. In [29], a meta-model to document design decisions (hence relating design to NFRs) and facilitate traceability is proposed. Khatter et al. [61] suggested an approach for the integration of non-functional requirements in a model-driven architecture. Last but not least, Almeida et al. [53] developed an NFR-based decision-making approach to select the execution configuration that best satisfies NFRs in a dynamic environment.

4.3. NFRs in Software Implementation

The development and propagation phase of NFRs includes any methods and tools used to dispatch the influence of NFRs on the different models and the final software product. The key point here is to investigate the extent to which this phase has been of interest to researchers (and/or practitioners). The assumption here is that this phase has generally not received enough attention. No single article was found that addresses NFRs during the SDLC implementation phase. This calls for further investigation both in industry and academia.

5. Research Result

The main research objective of this work was to investigate the current state of non-functional requirements development throughout the software development life cycle. This was addressed by conducting a systematized literature review of 34 publications published between January 2013 and April 2023, all carefully selected from five well-established online databases. The main research question was divided into two complementary sub-research questions.

5.1. What Approaches or Mechanisms Are There for the Processing of NFRs? (RQ1)

Various methods, techniques and tools are used in the development of NFRs: GORE methods, UML, machine learning, deep learning and/or NLP tools, other mathematically based techniques such as fuzzy logic, ontology, graph theory and inferential logic, XML-based notation, algorithms, prototypes and existing datasets. With notation techniques such as GORE methods, UML and XML, authors sometimes find it necessary to extend the original technique with new concepts to make it more applicable to their proposed method. For instance, the transformation of an NFR framework into a directed graph to facilitate the selection of the operation that best satisfies NFRs [23] is noted. Another example is the use of a UML profile to enable the incorporation of fuzzy concepts into a UML model and capture different alternative features [34]. Some authors also combine different techniques to construct a mechanism to address NFR development activities (as depicted in Figure 6 and discussed in Section 4). For instance, Luangwiriya and Kongkachandra [52] proposed a method for detecting conflicts within NFRs that uses an extended version of XML, namely, NFR XML frames to represent NFRs extracted from software requirements documents, as well as a conceptual graph (mathematically based technique) to structure the NFRs and hence facilitate the analysis and detection of potential conflicts.

For the past ten years, researchers' efforts towards the analysis of NFRs have focused on the activities of requirements engineering and software design, with less effort devoted to software development or implementation. In RE, the activities covered include NFR elicitation and evaluation, their classification and prioritization, NFR specification/modeling, conflict detection and trade-off analysis, as well as the selection of the best operation or strategy to satisfy NFRs. Conversely, NFR elicitation has not received as much attention from the selected publications compared with other RE activities, such as NFR classification, prioritization, specification/modeling, conflict detection and trade-off analysis. In general, the methods proposed for the development of NFRs utilize GORE methods, UML, XML and ontology for the specification and modeling of activities, as well as the operation of NFRs. ML/DL techniques, together with some other natural language processing tools, fuzzy logic, graph theory and inferential logic are employed in the analysis of NFR models pertaining to their classification, prioritization and conflict detection, etc. Some of the NFR methods are proposed for a specific software development methodology or environment and may not be applicable elsewhere or need to be adopted for other environments. There are approaches for agile, model-driving development, aspect-oriented software development, software product line, self-adoptive systems and service-oriented architecture.

5.2. Can a Process Be Derived from Existing NFR Approaches That Covers the Entire, or a Part of the SDLC? (RQ2)

Software engineering is a continuous process with multiple steps that starts with the elicitation of the user's requirements and ends with the tested software product. In that regard, a task within a process should accept as an input the output from a previous activity and produce an output that can be used by other activities. Thus, there is a need to investigate the possibility of structuring existing NFR development approaches into groups for which the output from one group can be used as the input of another group. This may call for a deeper qualitative analysis of each of the existing approaches (which is not within the scope of this work and would constitute a research topic of its own). However, to illustrate the idea, we consider the RE activities in Figure 6 and the methods proposed for each activity.

The following groups can be considered: a group of approaches for NFRs elicitation and evaluation, a group for NFRs categorization and prioritization, another group of methods or techniques for the specification and modeling of NFRs, as well as a group of suggested approaches for detecting conflicts in NFRs, performing trade-off analysis and selecting, among the alternative solutions, the strategy that best satisfies NFRs. Only the elicitation activity accepts inputs from various sources: stakeholders, business documents and processes, the business environment, policies, standards, etc., to produce an initial requirements document(s). These requirements are extracted from the document and classified and prioritized, resulting in more structured and elaborated requirements documents that are to be specified, refined and operated. They can also be used to produce other models of the software. With GORE methods, FRs and NFRs from the requirements document(s) are specified, refined and operated, resulting in a SIG (soft goal inter-dependency graph) model. With UML, more models may be produced including, scenarios diagrams, use cases diagrams, class diagrams, activity diagrams, and so forth. The same input can also be specified using XML, or to produce an ontology model. This shows that various outputs with different formats can be constructed from the same input. This does not necessarily imply an impediment to the next activities. First, because the various output formats can be converted into a common format, e.g., XML. Secondly, each output may be an appropriate input for one or more methods that may be needed for the next activity.

This illustration shows that NFR development activities (from the requirements elicitation to the documentation of the architectural design decisions) can be linked (iteratively) to one another through the intermediate models of the system in construction. They are iterative, in the sense that at any stage of development, if an error is detected (e.g., from the verification and/or validation of the intermediate model), or if a new requirement is elicited, some of the previous activities will need to be repeated to correct the error or to incorporate the new requirement. Consequently, a process or a comprehensive framework can be derived to cover the first two phases of the SDLC. To have a complete NFR development process for the entire SDLC will require more work to construct new methods, techniques and tools that can be used to extend the analysis/development of NFRs from the software design phase through to the implementation and testing of the final software product. Such research could equally consist of further study to identify additional research that this article could not locate in order to combine the pieces to construct a complete process that covers all the SDLC phases.

6. Discussion, Future Work and Threats to Validity

6.1. Discussion

It appears from the discussion in Section 4.1.1 that, despite its importance and criticality, the NFR elicitation has not been the focus of attention for much of the last decade. This observation may be justified by the fact that the early activities of software requirement engineering are common to all types of requirements. At this stage, the author agrees with Eckhardt et al. [72], who argued that many NFRs actually describe behavioral properties and should be handled in a similar way to functional requirements. In practice, it is not always possible to determine the type of requirements before they have been analyzed and refined. So, it is assumed that narrowing the selection of publications to focus solely on NFRs would have excluded some of the works that treat the requirements elicitation in general. Further research may need to be carried out in that regard in order to reach a more substantive conclusion.

From the discussion in Section 4.2, less than 9% of the 34 selected publications have attempted to develop NFRs beyond the RE phase. It is also observed that the main purpose of NFRs development in the RE phase is to produce the best strategy to satisfy NFRs. A strategy is a software solution known in GORE methods as tasks or functionalities and resources needed for satisfying NFRs [22]. This perception implies the idea that the development of NFRs ends when such a strategy is obtained. Although this work seems to conform to such an idea, with only about 9% of research on the analysis of NFRs beyond the RE phase, it contrasts the pressing need for a parallel development of NFRs alongside the FRs that has long been high on the agenda [73,74].

This study has revealed, in Section 3, the use of various techniques in the development of NFRs including amongst others, machine learning/deep learning, fuzzy logic, ontology and so forth. Conversely, established formal methods (FMs) such as the Z notation [7], Object-Z, VDM, event B, etc., have not been mentioned. FMs use mathematical concepts to specify and reason about the properties of a software or a system at an early development stage [15]. They are appropriate for safety-critical and security-critical systems [7,75]. Although, in general, they do not integrate mechanisms for NFRs, considering their primary objective for quality and correctness by construction, they would be expected to find their way into the analysis of NFRs or to integrate NFRs into their own processes. This also calls for further study of the relationship and integration of FMs and NFRs.

6.2. Future Work

It is observed from this study, especially from the results presented in Section 5 and the discussion in Section 6, that the following future research is necessary to address some important aspects of NFR development which is not covered in this work.

The first future work is the construction of a comprehensive process for the analysis and development of NFRs throughout the entire software development life cycle. A process that federates the strengths of the existing methods and tools.

Having a complete process and/or methodology for NFRs throughout the SDLC may enable the development of new techniques for reverse-engineering or re-engineering NFRs. For the past few decades, traditional software reverse-engineering has developed various means to recover and document the functionalities of the software with no serious consideration of NFRs. This could very likely be due to the absence of well-established processes for NFRs in forward engineering.

An interesting and surely beneficial research area in the field of NFR analysis for producing quality software products would be to combine NFRs and established FMs such as Z/Object-Z. Two complementary alternative views could be considered: first, the integration of FMs into the development of NFRs, which could lead, for instance, to the use of theorem provers to validate NFRs at an early stage of development. Another view is to integrate NFRs into the FMs processes, for example, to ensure that the quality requirements expected from a final software system, are introduced into the process at an early stage and used to guide the development throughout the process.

This research work has noted the absence of NFR testing and validation from the selected publications. This calls for further investigations that could involve empirical research in industry and/or more refined theoretical research that focuses on the testing and validation of NFRs.

It would be interesting to extend this work by creating, for instance, a taxonomy of NFR approaches based on the subject of the software system, or the type of the system to be produced, such as information systems, intelligent systems, recommended systems, and/or to investigate the systems produced so far in the industry, in order to substantiate the applicability/efficacy of the existing approaches.

6.3. Threats to Validity

As a single-authored systematized literature review, the present author is aware of the risk of bias, especially with the selection of the publications, the extraction of data from the selected publications, as well as the analysis of the extracted data. To mitigate such risks, the paper has endeavored to follow the SLR protocols as much as possible and used Microsoft Excel (Microsoft® Excel® for Microsoft 365 MSO (Version 2409 Build 16.0.18025.20030) 32-bit) spreadsheets to guide the analysis and facilitate record keeping. For the repeatability of this work, the query string, the date and time each selected online database was queried, as well as the number of records generated per database are included in Table 2.

7. Conclusions

This article has conducted a systematized literature review to investigate the current-state-of-the-art in NFRs development. A total of 34 publications including 13 journal articles and 21 conference papers published during the past ten years were carefully selected from five reputable online databases: ACM Digital Library, ScienceDirect, Scopus, IEEE Xplore and Web of Science. Google Scholar and Springer Link databases were used to ensure the comprehensiveness of the first search.

The study of the selected publications identified the various software engineering techniques/methods and their roles in the proposed approaches to NFRs analysis and development, including GORE methods, UML, XML and Ontology mainly for the specification and modeling of NFRs, as well as their refinement and operation for some of these techniques. Machine learning, deep learning algorithms and some other natural language processing tools, fuzzy logic, graph theory, inferential logic and so forth for the extraction of NFRs from requirements document(s), NFRs classification, prioritization, conflicts and trade-off analysis and the selection of the best strategies for satisfying NFRs. It is noted that 50% of the existing approaches include algorithms to implement their methods and/or prototypes to illustrate their working or for testing. Approaches based on ML and/or DL mainly use, for their testing, the following available datasets: the PROMISE corpus [55] and the Concordia RE corpus [41].

This work has also investigated the development of NFRs throughout the software development life cycle. As illustrated in Figure 6 and discussed in Section 4, it appears that most of the existing NFR approaches focus on the activities of the requirements engineering phase, with very little efforts devoted to software design and implementation. This article suggests that the elicitation of NFRs has not received much attention in the literature. This could be explained by the fact that at an early stage of software development, the same activities are performed for both FRs and NFRs and thus the selection of publications focussing solely on NFRs might, in fact, not be entirely appropriate.

Funding: 54% of the APC was funded by Unisa's page fees fund for accredited journal articles (funding from the institutional research fund), and the remaining 46% of the APC paid from the research output incentive fund for the department of Computer Science, school of Computing, CSET, UNISA.

Data Availability Statement: All data are contained in the article.

Acknowledgments: I express my gratitude to my colleague Levine Niall for proofreading and language editing of this article.

Conflicts of Interest: The author declares no conflicts of interest.

References

1. Sommerville, I. *Software Engineering*, 8th ed.; Addison-Wesley, Person Education: Harlow, UK, 2007.
2. Dimeska, K.; Savoska, S. Model of software development using RAD methods and standard ISO/IEC 12207. In Proceedings of the 8th International Conference on Applied Internet and Information Technologies, Santa Barbara, CA, USA, 15–18 October 2018; Volume 8, pp. 48–51.
3. Chung, L.; Nixon, B.A.; Yu, E.; Mylopoulos, J. *Non-Functional Requirements in Software Engineering*; Springer Science & Business Media: Berlin/Heidelberg, Germany 2012; Volume 5,
4. *ISO/IEC/IEEE 29148:2018(E)*; ISO/IEC/IEEE International Standard–Systems and Software Engineering–Life Cycle Processes—Requirements Engineering. IEEE: Piscataway, NJ, USA, 2018; pp. 1–104. https://doi.org/10.1109/IEEESTD.2018.8559686.
5. Abrahamsson, P.; Salo, O.; Ronkainen, J.; Warsta, J. Agile software development methods: Review and analysis. *arXiv* **2017**, arXiv:1709.08439.
6. Völter, M.; Stahl, T.; Bettin, J.; Haase, A.; Helsen, S. *Model-Driven Software Development: Technology, Engineering, Management*; John Wiley & Sons: Hoboken, NJ, USA, 2013.
7. Spivey, J.M. *The Z Notation: A Reference Manual*; Prentice Hall International (UK) Ltd.: Hertfordshire, UK, 1992.
8. Horkoff, J.; Aydemir, F.B.; Cardoso, E.; Li, T.; Maté, A.; Paja, E.; Salnitri, M.; Piras, L.; Mylopoulos, J.; Giorgini, P. Goal-oriented requirements engineering: An extended systematic mapping study. *Requir. Eng.* **2019**, *24*, 133–160.
9. Dalpiaz, F.; Franch, X.; Horkoff, J. istar 2.0 language guide. *arXiv* **2016**, arXiv:1605.07767.
10. Van Lamsweerde, A. Goal-Oriented Requirements Engineering: A Guided Tour. In Proceedings of the RE'01: Fifth IEEE International Symposium on Requirements Engineering, Washington, DC, USA, 27–31 August 2001; p. 249.
11. Chung, L.; Nixon, B.A.; Yu, E.; Mylopoulos, J.; Chung, L.; Nixon, B.A.; Yu, E.; Mylopoulos, J. The NFR framework in action. *Non-Functional Requirements in Software Engineering*; International Series in Software Engineering; Springer: Boston, MA, USA, 2000; Volume 5, pp. 15–45.
12. Gomaa, H. Overview of UML, SysML and MARTE. In *Real-Time Software Design for Embedded Systems*; Cambridge University Press: Cambridge, UK, 2016; pp. 12–31.
13. Wolny, S.; Mazak, A.; Carpella, C.; Geist, V.; Wimmer, M. Thirteen years of SysML: A systematic mapping study. *Softw. Syst. Model.* **2020**, *19*, 111–169. https://doi.org/10.1007/s10270-019-00735-y.
14. Saunders, M.; Bristow, A. *2023 Research Methods for Business Students Preface and Chapter 4*, 9th ed.; Pearson: New York, NY, USA; Harlow, UK, 2023; pp. i–xxvii, 128.
15. O'Regan, G. *Mathematical Approaches to Software Quality*; Springer: London, UK, 2006; pp. I–XV, 1–231.
16. Bray, T.; Paoli, J.; Sperberg-McQueen, C.M.; Maler, E.; Yergeau, F. *Extensible Markup Language (XML) 1.0*; Wiley Publishing: Indianapolis, Indiana, 2008.
17. Dongmo, C.; Van der Poll, J.A. An improved user requirements notation (URN) models' construction approach. *Systems* **2023**, *11*, 301.
18. Dardenne, A.; van Lamsweerde, A.; Fickas, S. Goal-directed requirements acquisition. *Sci. Comput. Program.* **1993**, *20*, 3–50. https://doi.org/10.1016/0167-6423(93)90021-G.
19. Franch, X.; Lopez, L.; Cares, C.; Colomer, D. *The i* Framework for Goal-Oriented Modeling*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 485–506. https://doi.org/10.1007/978-3-319-39417-6_22.
20. Bresciani, P.; Perini, A.; Giorgini, P.; Giunchiglia, F.; Mylopoulos, J. Tropos: An agent-oriented software development methodology. *Auton. Agents Multi-Agent Syst.* **2004**, *8*, 203–236.
21. Mouratidis, H.; Giorgini, P. Secure Tropos: A Security-Oriented Extension of the Tropos Methodology. *Int. J. Softw. Eng. Knowl. Eng.* **2007**, *17*, 285–309. https://doi.org/10.1142/S0218194007003240.
22. ITU-T, Recommendation Z.151 (10/12), User Requirements Notation (URN)-Language Definition. Geneva, Switzerland, October 2012. Available online: http://www.itu.int/rec/T-REC-Z.151/en (accessed on 8 October 2024).
23. Affleck, A.; Krishna, A.; Achuthan, N.R. Optimal Selection of Operationalizations for Non-Functional Requirements. In Proceedings of the Ninth Asia-Pacific Conference on Conceptual Modelling—Volume 143, APCCM'13, Adelaide, Australia, 29 January–1 Feburary 2013; pp. 69–78.
24. DeVries, B.; Cheng, B. Goal-Based Modeling and Analysis of Non-Functional Requirements. In Proceedings of the 2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems, MODELS 2019, Munich, Germany, 15–20 September 2019; pp. 261–271. https://doi.org/10.1109/MODELS.2019.00010.
25. Goncalves, J.; Krishna, A. Dynamic non-functional requirements based model-driven agent development. In Proceedings of the 2015 24th Australasian Software Engineering Conference, ASWEC 2015, Adelaide, SA, Australia, 28 September–1 October 2015; pp. 128–137. https://doi.org/10.1109/ASWEC.2015.24.
26. Arif, M.; Mohammad, C.; Sadiq, M. UML and NFR-framework based method for the analysis of the requirements of an information system. *Int. J. Inf. Technol.* **2023**, *15*, 411–422. https://doi.org/10.1007/s41870-022-01112-7.

27. Mehta, R.; Ruiz-López, T.; Chung, L.; Noguera, M. Selecting among Alternatives Using Dependencies: An NFR Approach. In Proceedings of the 28th Annual ACM Symposium on Applied Computing, SAC '13, New York, NY, USA, 18–22 March 2013; pp. 1292–1297. https://doi.org/10.1145/2480362.2480603.
28. Peixoto, M.M.; Silva, C. Specifying Privacy Requirements with Goal-Oriented Modeling Languages. In Proceedings of the XXXII Brazilian Symposium on Software Engineering, SBES '18, New York, NY, USA, 17–21 September 2018; pp. 112–121. https://doi.org/10.1145/3266237.3266270.
29. Dermeval, D.; Castro, J.; Silva, C.; Pimentel, J.; Bittencourt, I.I.; Brito, P.; Elias, E.; Tenório, T.; Pedro, A. On the Use of Metamodeling for Relating Requirements and Architectural Design Decisions. In Proceedings of the 28th Annual ACM Symposium on Applied Computing, SAC '13, New York, NY, USA, 18–22 March 2013; pp. 1278–1283. https://doi.org/10.1145/2480362.2480601.
30. Vilela, J.; Castro, J.; Martins, L.E.G.; Gorschek, T.; Silva, C. Specifying Safety Requirements with GORE Languages. In Proceedings of the XXXI Brazilian Symposium on Software Engineering, SBES '17, New York, NY, USA, 20–22 September 2017; pp. 154–163. https://doi.org/10.1145/3131151.3131175.
31. Jacobson, L.; Booch, J.R.G. *The Unified Modeling Language Reference Manual*; Addison Wesley Longman, Inc.: Boston, MA, USA, 2021.
32. Booch, G.; Rumbaugh, J.; Jacobson, I. *The Unified Modeling Language User Guide*, 2nd ed.; Addison-Weseley: Boston, MA, USA; Pearson Education Inc.: Westford, MA, USA, 2005.
33. Fuentes-Fernández, L.; Vallecillo-Moreno, A. An introduction to UML profiles. *UML Model Eng.* **2004**, *2*, 72.
34. Saadatmand, M.; Tahvili, S. A Fuzzy Decision Support Approach for Model-Based Tradeoff Analysis of Non-functional Requirements. In Proceedings of the 2015 12th International Conference on Information Technology—New Generations, Las Vegas, NV, USA, 13–15 April 2015; pp. 112–121. https://doi.org/10.1109/ITNG.2015.24.
35. Kaur, H.; Sharma, A. A measure for modelling non-functional requirements using extended use case. In Proceedings of the 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 16–18 March 2016; pp. 1101–1105.
36. Martinez, G.G.; Carpio, A.F.D.; Gomez, L.N. A model for detecting conflicts and dependencies in non-functional requirements using scenarios and use cases. In Proceedings of the 2019 45th Latin American Computing Conference, CLEI 2019, Panama, 30 September–4 October 2019. https://doi.org/10.1109/CLEI47609.2019.235051.
37. Khalique, F.; Butt, W.H.; Khan, S.A. Creating Domain Non-functional Requirements Software Product Line Engineering Using Model Transformations. In Proceedings of the 2017 International Conference on Frontiers of Information Technology, FIT 2017, Islamabad, Pakistan, 18–20 December 2017; Volume 2017, pp. 41–45. https://doi.org/10.1109/FIT.2017.00015.
38. Mahesh, B. Machine learning algorithms—A review. *Int. J. Sci. Res. (IJSR)* **2020**, *9*, 381–386.
39. Pouyanfar, S.; Sadiq, S.; Yan, Y.; Tian, H.; Tao, Y.; Reyes, M.P.; Shyu, M.L.; Chen, S.C.; Iyengar, S.S. A survey on deep learning: Algorithms, techniques, and applications. *ACM Comput. Surv. (CSUR)* **2018**, *51*, 1–36.
40. Magnini, B.; Lavelli, A.; Magnolini, S. Comparing Machine Learning and Deep Learning Approaches on NLP Tasks for the Italian Language. In Proceedings of the Twelfth Language Resources and Evaluation Conference, Marseille, France, 11–16 May 2020; Calzolari, N., Béchet, F., Blache, P., Choukri, K., Cieri, C., Declerck, T., Goggi, S., Isahara, H., Maegaard, B., Mariani, J., et al., Eds.; pp. 2110–2119.
41. Rashwan, A.; Ormandjieva, O.; Witte, R. Ontology-based classification of non-functional requirements in software specifications: A new corpus and SVM-based classifier. In Proceedings of the International Computer Software and Applications Conference, Los Angeles, NV, USA, 14 March 2013; pp. 381–386. https://doi.org/10.1109/COMPSAC.2013.64.
42. Kaur, K.; Kaur, P. BERT-CNN: Improving BERT for Requirements Classification using CNN. *Procedia Comput. Sci.* **2023**, *218*, 2604–2611. https://doi.org/https://doi.org/10.1016/j.procs.2023.01.234.
43. Rahman, M.A.; Haque, M.A.; Tawhid, M.N.A.; Siddik, M.S. Classifying Non-Functional Requirements Using RNN Variants for Quality Software Development. In Proceedings of the 3rd ACM SIGSOFT International Workshop on Machine Learning Techniques for Software Quality Evaluation, MaLTeSQuE 2019, New York, NY, USA, 27 August 2019; pp. 25–30. https://doi.org/10.1145/3340482.3342745.
44. Yahya, A.E.; Gharbi, A.; Yafooz, W.M.; Al-Dhaqm, A. A Novel Hybrid Deep Learning Model for Detecting and Classifying Non-Functional Requirements of Mobile Apps Issues. *Electronics* **2023**, *12*, 1258. https://doi.org/10.3390/electronics12051258.
45. Alashqar, A.M. Studying the commonalities, mappings and relationships between non-functional requirements using machine learning. *Sci. Comput. Program.* **2022**, *218*, 102806. https://doi.org/https://doi.org/10.1016/j.scico.2022.102806.
46. Singh, P.; Singh, D.; Sharma, A. Rule-based system for automated classification of non-functional requirements from requirement specifications. In Proceedings of the 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Jaipur, India, 21–24 September 2016; pp. 620–626. https://doi.org/10.1109/ICACCI.2016.7732115.
47. Mairiza, D.; Zowghi, D.; Gervasi, V. Conflict characterization and Analysis of Non Functional Requirements: An experimental approach. In Proceedings of the SoMeT 2013—12th IEEE International Conference on Intelligent Software Methodologies, Tools and Techniques, Proceedings, Budapest, Hungary, 22–24 September 2013; pp. 83–91. https://doi.org/10.1109/SoMeT.2013.6645645.
48. Zhang, X.; Wang, X. Tradeoff Analysis for Conflicting Software Non-Functional Requirements. *IEEE Access* **2019**, *7*, 156463–156475. https://doi.org/10.1109/ACCESS.2019.2949218.

49. Paucar, L.H.G.; Bencomo, N. RE-STORM: Mapping the Decision-Making Problem and Non-Functional ReQuirements Trade-off to ParTIally OBseRVable MArkov Decision Processes. In Proceedings of the 13th International Conference on Software Engineering for Adaptive and Self-Managing Systems, SEAMS '18, New York, NY, USA, 28–29 May 2018; pp. 19–25. https://doi.org/10.1145/3194133.3195537.
50. Shah, U.; Patel, S.; Jinwala, D. An Ontological Approach to Specify Conflicts among Non-Functional Requirements. In Proceedings of the 2019 2nd International Conference on Geoinformatics and Data Analysis, ICGDA 2019, New York, NY, USA, 15–17 March 2019; pp. 145–149. https://doi.org/10.1145/3318236.3318257.
51. Liu, C.L. CDNFRE: Conflict detector in non-functional requirement evolution based on ontologies. *Comput. Stand. Interfaces* **2016**, *47*, 62–76. https://doi.org/https://doi.org/10.1016/j.csi.2016.03.002.
52. Luangwiriya, T.; Kongkachandra, R. Automatic Conflict Detection in Non-functional requirement Analysis using a Conceptual Graph. *ECTI Trans. Comput. Inf. Technol.* **2023**, *17*, 82–94.
53. Almeida, A.; Bencomo, N.; Batista, T.; Cavalcante, E.; Dantas, F. Dynamic Decision-Making Based on NFR for Managing Software Variability and Configuration Selection. In Proceedings of the 30th Annual ACM Symposium on Applied Computing, SAC '15, New York, NY, USA, 13–17 April 2015; pp. 1376–1382. https://doi.org/10.1145/2695664.2695875.
54. Shah, T.; Patel, S.V. A Novel Approach for Specifying Functional and Non-functional Requirements Using RDS (Requirement Description Schema). *Procedia Comput. Sci.* **2016**, *79*, 852–860. https://doi.org/https://doi.org/10.1016/j.procs.2016.03.083.
55. Menzies, T., Krishna, R., Pryor, D.; *The Promise Repository of Empirical Software Engineering Data 2015*; North Carolina State University, Department of Computer Science: Raleigh, NC, USA, 2015. Available online: http://openscience.us/repo (accessed on 8 October 2024).
56. Wang, R.; Li, J. Bayes test of precision, recall, and F1 measure for comparison of two natural language processing models. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy, 28 July–2 August 2019; pp. 4135–4145.
57. Aljallabi, B.M.; Mansour, A. Enhancement approach for non-functional requirements analysis in Agile environment. In Proceedings of the 2015 International Conference on Computing, Control, Networking, Electronics and Embedded Systems Engineering (ICCNEEE), Khartoum, Sudan, 7–9 September 2015; pp. 428–433. https://doi.org/10.1109/ICCNEEE.2015.7381407.
58. Muhammad, A.; Siddique, A.; Mubasher, M.; Aldweesh, A.; Naveed, Q.N. Prioritizing Non-Functional Requirements in Agile Process Using Multi Criteria Decision Making Analysis. *IEEE Access* **2023**, *11*, 24631–24654. https://doi.org/10.1109/ACCESS.2023.3253771.
59. Amorndettawin, M.; Senivongse, T. Non-Functional Requirement Patterns for Agile Software Development. In Proceedings of the 2019 3rd International Conference on Software and E-Business, ICSEB '19, Tokyo, Japan, 9–11 December 2019; pp. 66–74. https://doi.org/10.1145/3374549.3374561.
60. Brataas, G.; Martini, A.; Hanssen, G.K.; Ræder, G. Agile elicitation of scalability requirements for open systems: A case study. *J. Syst. Softw.* **2021**, *182*, 111064. https://doi.org/https://doi.org/10.1016/j.jss.2021.111064.
61. Khatter, K.; Kalia, A. Integration of non-functional requirements in model-driven architecture. In Proceedings of the Fifth International Conference on Advances in Recent Technologies in Communication and Computing (ARTCom 2013), Bangalore, India, 20–21 September 2013; pp. 359–364. https://doi.org/10.1049/cp.2013.2213.
62. Meridji, K.; Al-Sarayreh, K.T.; Abran, A.; Trudel, S. System security requirements: A framework for early identification, specification and measurement of related software requirements. *Comput. Stand. Interfaces* **2019**, *66*, 103346. https://doi.org/https://doi.org/10.1016/j.csi.2019.04.005.
63. Burback, R. Software Engineering Methodology: The Watersluice. Ph.D. Thesis, Stanford University, Department of Computer Science: Stanford, CA, USA, 1999.
64. Otero, C. *Software Engineering Design: Theory and Practice*; CRC Press: Boca Raton, FL, USA, 2012.
65. *IEEE Std 830-1998*; IEEE Recommended Practice for Software Requirements Specifications; IEEE: Piscataway, NJ, USA, 1998; pp. 1–40. https://doi.org/10.1109/IEEESTD.1998.88286.
66. Werner, C.; Li, Z.S.; Lowlind, D.; Elazhary, O.; Ernst, N.; Damian, D. Continuously Managing NFRs: Opportunities and Challenges in Practice. *IEEE Trans. Softw. Eng.* **2022**, *48*, 2629–2642. https://doi.org/10.1109/TSE.2021.3066330.
67. Gopichand, M.; Rao, A.A. Four layered approach to non-functional requirements analysis. *arXiv* **2012**, arXiv:1201.6141.
68. Kassab, M.; Daneva, M.; Ormandjieva, O. *Early Quantitative Assessment of Non-Functional Requirements*; Centre for Telematics and Information Technology (CTIT), University of Twente: Enschede, The Netherlands, 2007. Available online: https://research.utwente.nl/en/publications/early-quantitative-assessment-of-non-functional-requirements (accessed on 8 October 2020).
69. Sharma, V.S.; Ramnani, R.R.; Sengupta, S. A Framework for Identifying and Analyzing Non-Functional Requirements from Text. In Proceedings of the 4th International Workshop on Twin Peaks of Requirements and Architecture, TwinPeaks 2014, New York, NY, USA, 1 June 2014; pp. 1–8. https://doi.org/10.1145/2593861.2593862.
70. Matsumoto, Y.; Shirai, S.; Ohnishi, A. A Method for Verifying Non-Functional Requirements. *Procedia Comput. Sci.* **2017**, *112*, 157–166. https://doi.org/https://doi.org/10.1016/j.procs.2017.08.006.
71. Matoussi, A.; Laleau, R. A Survey of Non-Functional Requirements in Software Development Process. Research Report TR-LACL-2008-7, LACL. 2008. Available online: https://hal.science/hal-01224656v1 (accessed on 8 October 2024).

72. Eckhardt, J.; Vogelsang, A.; Fernández, D.M. Are "Non-functional" Requirements Really Non-Functional? An Investigation of Non-functional Requirements in Practice. In Proceedings of the 38th International Conference on Software Engineering, ICSE '16, New York, NY, USA, 14–22 May 2016; pp. 832–842. https://doi.org/10.1145/2884781.2884788.
73. Cai, K.Y. Non-Functional Computing: Towards a More Scientific Treatment to Non-Functional Requirements. In Proceedings of the Computer Software and Applications Conference, 2007, COMPSAC 2007, 31st Annual International, Beijing, China, 23–27 July 2007; Volume 2, pp. 493–494. https://doi.org/10.1109/COMPSAC.2007.156.
74. Rosa, N.S.; Cunha, P.R.F. An Approach for Reasoning and Refining Non-Functional Requirements. *J. Braz. Comp. Soc.* **2004**, *10*, 59–81.
75. Abdallah, A.E.; Bowen, J.P.; Nissanke, N. Formal methods for safety critical systems. In: *Dependable Computing Systems: Paradigms, Performance Issues, and Applications, Part I: Models and Paradigms*; Wiley Series on Parallel and Distributed Computing; Diab, H.B., Zomaya, A.Y., Eds.; John Wiley & Sons: Chichester, UK, 2005; Volume 9.