Towards partial fulfillment for Undergraduate Degree Level Programme
Bachelor of Technology in Computer Engineering

A Project Report on:

IDENTIFYING AND RESOLVING CONFLICTS IN THE NON FUNCTIONAL REQUIREMENTS SPECIFIED IN NATURAL LANGUAGE

Prepared by :

Admission No.	Student Name
U17CO059	JAIMIN PATEL
U18CO112	SHTAKSHI UPADHYAY
U18CO113	SHREYASH KADAM
U18CO114	ADITYA KUMAR

Class	:	B.TECH. IV (Computer Engineering) 8th Semester
Year	:	2021-2022
Guided by	:	DEVESH C. JINWALA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SARDAR VALLABHBHAI NATIONAL INSTITUTE OF TECHNOLOGY,
SURAT - 395 007 (GUJARAT, INDIA)

Student Declaration

This is to certify that the work described in this project report has been actually carried out and implemented by our project team consisting of

Sr.	Admission No.	Student Name
1	U17CO059	JAIMIN PATEL
2	U18CO112	SHTAKSHI UPADHYAY
3	U18CO113	SHREYASH KADAM
4	U18CO114	ADITYA KUMAR

Neither the source code there in, nor the content of the project report have been copied or downloaded from any other source. We understand that our result grades would be revoked if later it is found to be so.

Signature of the Students:

Sr.	Student Name	Signature of the Student
1	JAIMIN PATEL	[illegible]
2	SHTAKSHI UPADHYAY	[illegible]
3	SHREYASH KADAM	[illegible]
4	ADITYA KUMAR	Aditya

Certificate

This is to certify that the project report entitled **Identifying and resolving conflicts in the Non-functional Requirements specified in Natural Language** *is prepared and presented by*

Sr.	Admission No.	Student Name
1	U17CO059	JAIMIN PATEL
2	U18CO112	SHTAKSHI UPADHYAY
3	U18CO113	SHREYASH KADAM
4	U18CO114	ADITYA KUMAR

Final Year of Computer Engineering and their work is satisfactory.

SIGNATURE :

GUIDE JURY HEAD OF DEPT.

Abstract

Non-functional requirements (NFRs) play an important role in success of any software application. NFRs tend to interfere with one another which is inevitable. Sometimes the interference of NFRs lead to conflicts. Thus, Conflict identification and resolution is an important aspect of requirement engineering, which decides the overall quality and performance of an application. If the conflicts are not identified at the earlier stage of the software development life cycle, then it leads to customer dissatisfaction, increase the cost and the time required to build the software. Many techniques to deal with these conflicts have been developed by researchers in this field. In our report, we have discussed requirements, conflict in requirements and identification of conflicts. Conflict resolution is mostly an intuitive approach, so we focused mainly on conflict identification. We have proposed a semi-automated framework to identify conflicts among NFRs using deep learning, long-short term networks and GloVe embedding. We proposed to represent natural language NFRs via word embedding for natural language processing of text. The word embedding of sentences is then combined with our integrated features such as is-Antonym, Intersection over union and Negation for further improving the accuracy of our deep learning model. A running example is provided to illustrate the same.

Contents

List of Figures

List of Acronyms

FR	Functional Requirement
NFR	Non Functional Requirement
SDLC	Software Development Life Cycle
NFD	Non Functional Decomposition
OWL	Ontology Web Language
SWRL	Semantic Web Rule Language
SVM	Support Vector Machine
ANN	Artificial Neural Network
POS	Parts of Speech
LSTM	Long Short Term Memory

1 Introduction

All requirements for any software programme are described in an SRS document, which is divided into two categories: functional requirements and nonfunctional requirements. Stakeholders normally place a high priority on functional requirements, while non-functional requirements receive little attention. However, having clear and unambiguous Non-Functional Requirements that do not clash with each other is critical for the success of any software product.
Non-functional requirements tend to interfere and conflict with each other. The objective of this project is identifying and resolving conflicts among the Non-functional requirements (NFRs). We will be discussing about NFRs, their types, conflicts in NFRs and approaches to identify conflict and resolve them.

1.1 Application

One of the most common causes of software project failure is incomplete or defective requirements. Projects are more likely to fail if software and system requirements are not clearly defined (unclear, unprioritized, incomplete, and unreflective of business goals and objectives). As a result, it is critical to appropriately identify requirements in the SRS document. Non-functional needs are frequently neglected, resulting in conflicts, due to a variety of faults such as misunderstanding and confusing wording used to convey requirements. There have been numerous instances in the past where software projects have failed due to requirements specifications disagreements. Our initiative finds a way to solve the problem of defective requirements early in the software development cycle, saving a lot of money, time, and effort.

1.2 Motivation

Requirements won't seem to be of much significance, but it can have a huge impact on success or failure of a system/application. In section 1.2.1 there is a brief case of the famous London Ambulance services crisis which occurred in 1992 because the requirements were overlooked.

In terms of the extra effort or mistakes attributed to them, conflicting requirements are one of the three fundamental challenges in software development (Curtis, Krasner, and Iscoe 1988)[12]. According to a two-year study undertaken by Egyed Boehm, between 40 percent and 60 percent of the requirements involved are in conflict, with NFRs accounting for approximately half of the requirements conflict [6].

It is typical to change overall design guidelines rather than just one module when differences emerge [26]. As a result, it's vital to spot conflicts early in the software development process. A blueprint of a system, outlining the essential components and their relationships from many perspectives, is the basic outcome of any architectural design process.

Conflicts are more likely to be concealed if a good approach for recognising and resolving them is not outlined. If they are kept hidden, they will cause dissatisfaction with the requirements process, as well as infeasible design and execution [2]. Conflicts that are not discovered early in the software development life cycle can raise the cost of software development as well as the time it takes to complete the process. As a result, ignoring NFR conflicts will result in a succession of software development failures or challenges.

1.2.1 Case Study: London Ambulance Service

With roughly 320 ambulances serving 6.8 million people, London Ambulance Service was (is) the world's largest ambulance service (1992). The mechanism for booking and contacting an ambulance was carried out manually. The individual in need had to call the service, which was answered by a call centre employee, who took down the patient's information and address before mobilising resources. This process was quite expensive, so they wanted to automate it to save money. They planned to build a computer-based call centre and dispatch system, then dispatch/deploy everything generated at the same time (Big Bang release).

However, the entire application was a mess. The dispatch system was broken, calls were being dropped owing to exception messages, and no one knew because everything was automated. There were 46 deaths that could have been avoided. The manual system was restored in one day, but the computer system was shut down fully in eight days.

Memory leaks, a terrible user interface, inadequate error handling, and other factors contributed to the disaster. However, ignorance of inadequate SRS was the primary culprit. Before they began developing the system, the software requirements were incomplete. It was made worse by the big bang release. They should have employed an incremental release, in which the system is distributed module by module after receiving user feedback. They should have spent more time on requirements analysis and documentation if they intended everything to be delivered at the same time.

1.3 Objective

It is difficult to identify conflicts among NFRs available in natural language. Manual method for conflict identification is tedious, time-consuming and error-prone. For this research, our objective is to create a semi-automated approach for identifying conflicts in NFRs and resolving them. Conflict resolution is mostly an intuitive approach, so our focus is primarily on identifying conflicts in NFRs that arise in Natural Language. Among various methods to detect conflict, we propose to use ontological method for conflict identification. Using ontology, the knowledge can be represented in conceptual or relational form. Apart from creating ontologies, we propose to represent NFRs in a form of a meta-model. Using the information of created ontologies and NFR meta-model, rules will be queried to identify conflicts.

1.4 Contributions

In our report, we have studied about non-functional requirements and their importance in the field of requirement engineering. We studied what are conflicts and the methods to identify, analyze and resolve conflicts among NFRs. Later, we've come up with an approach for conflict identification. NFRs that are mentioned in natural language are processed and represented in the form of NFR meta-model. Ontologies are created and certain predefined rules are applied on the collective information of ontologies and the meta-model. Conflicts(if any) among NFRs are detected when the rules are applied.

1.5 Organisation of project report

This report is divided into 5 chapters. Contents of the first chapter include introduction, objective and motivation behind the project. In the next chapter, we discuss the theoretical background and literature surveys of various research papers that we read. In chapter three, proposed work for conflict identification in Non-Functional Requirements is described in detail with flow chart and a running example. Chapter four contains simulation and results. The final chapter concludes the report along with a brief discussion on scope of future work.

2 Theoretical Background

2.1 Requirements

A requirement, according to IEEE standard 729, is a condition or capacity that a system or system component must meet or possess in order to satisfy a standard, specification, contract, or other formally enforced documentation.

Requirements can further be classified as functional and non-functional.

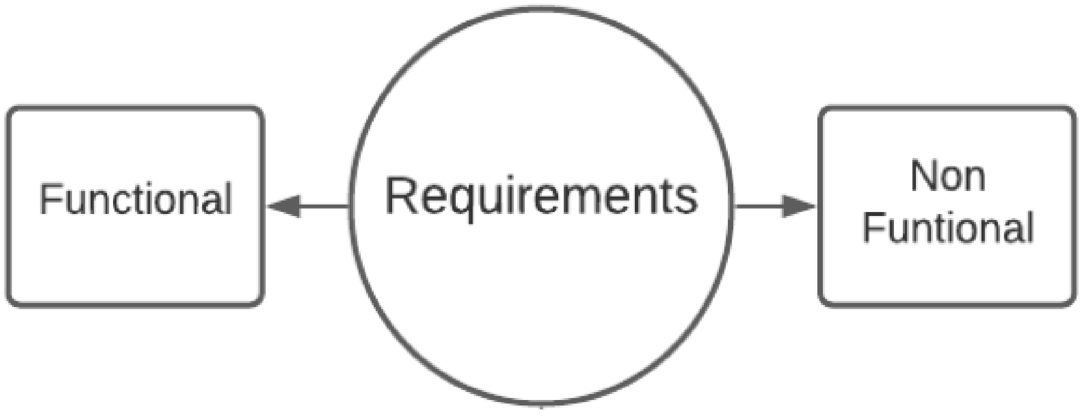

Figure 1: Classification of Requirements

2.2 Functional Requirements

Functional requirements are characteristics that enable the system to perform as planned. These functions are addressed with the client during sessions. The desired output is attained once the input is operated according to the designed system, which is how functional requirements are described.

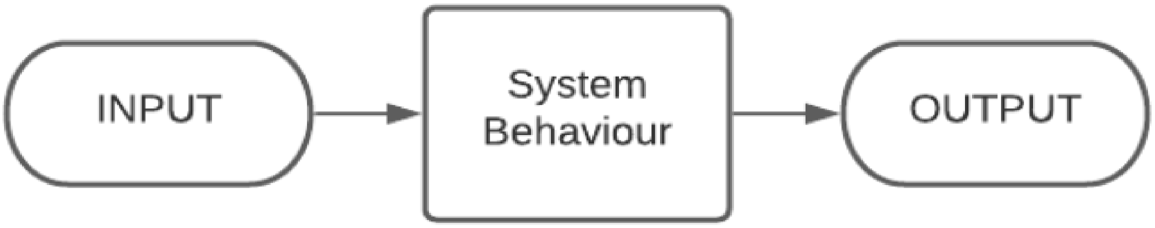

Figure 2: Functional Requirement

FRs are product attributes that are primarily concerned with the end user. For example, when a user inputs his or her OTP during a transaction, money should be debited from his or her bank account.

2.3 Non-functional requirements

Non-functional requirements indicate how the system should do something, while functional requirements define what it should do. Non-functional requirements have no bearing on an application's behaviour. They are quality attributes that are concerned with consumer expectations. The following are some of the most common NFRs encountered by software developers:

Performance	Reliability	Capacity
Recoverability	Scalability	Data Integrity
Efficiency	Interoperability	Availability
Serviceability	Maintainability	Security
Regulatory	Usability	Quality

The following illustrates the distinction between Functional and Non-Functional Requirements.

Functional Requirements	**Non-functional Requirements**
Functional requirements define the way a system works.	Non-functional requirements define the quality attributes of a system.
They are captured in use cases.	They are captured as quality attributes.
They are specified in the software requirement specification.	They are specified in the software architecture document.
They are mandatory for fulfillment.	They are not mandatory for fulfillment.
Functional requirements can exist without Non-functional requirements.	Non-functional requirements cannot exist without Functional requirements.
They are defined at a component level.	They are applied to the whole system.
They are usually easy to define.	They are usually more difficult to define.

Here is an example to understand the Non-Functional Requirements better:

Consider two kinds of bottles of ketchup. One facing upward and other which has lid at the bottom, opening back downwards. To, use the first bottle, we need to hold it upside down and shake it multiple times to bring out the ketchup. While with the second type of bottle, we just have to open the lid and the ketchup comes out without any delay.

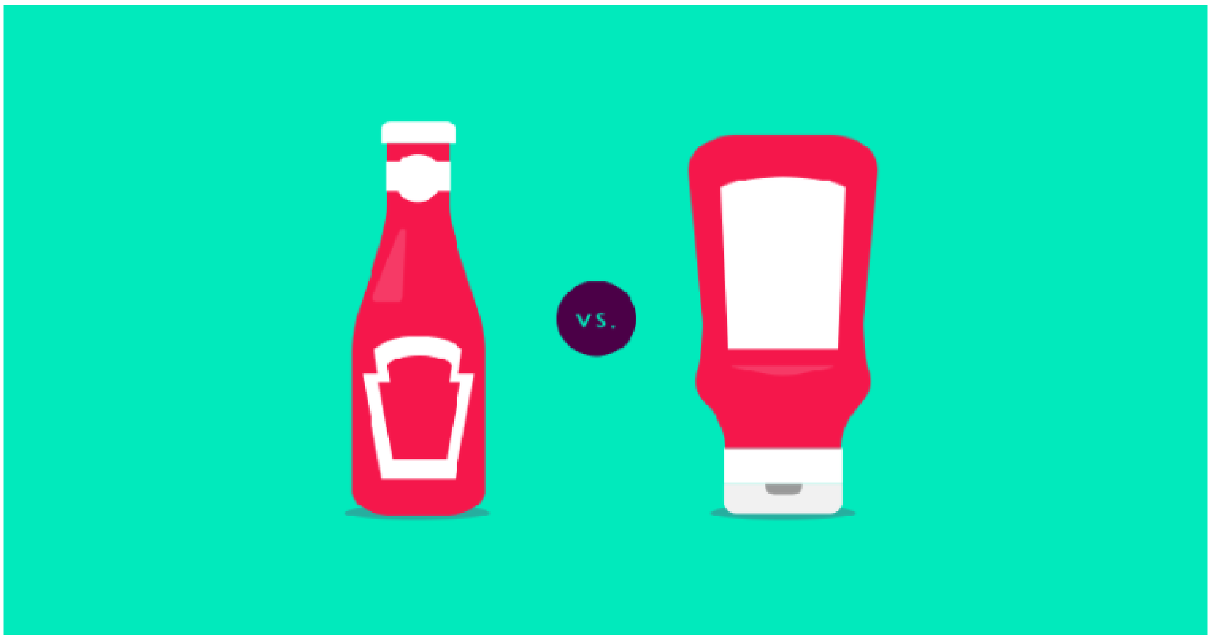

Figure 3: Two kind of ketctup bottles showing the importance of NFRs

It is obvious that the second type of bottle is more preferrable, because it minimizes the effort and time required to get the desired output. Both bottles have same functional requirements (which can be hold ketchup and have an opening to pour ketchup), but the second bottle has clearer non-fucntional requirement (which can be to provide easy and fast delivery of ketchup without any delay or extra effort). Taking care of Non-Functional

Requirements create a more usable product, which can perform better.

2.4 Quality Models

When the word "quality" is mentioned, it conjures up a plethora of associations. Many researchers have attempted to give the term a good meaning, and some quality management philosophers have succeeded. A statistician named Shewhart provided the most precise and well-known definition of quality. According to him, there are two types of quality: objective and subjective. One exists independently of man's existence (objective truth), while the other is what we perceive as a result of objective truth (subjective).

For a long time, quality models have been discussed, and a significant variety of quality models have been offered. The software product quality control is established using software quality models. They provide a more quantifiable and fixed view of a software product. McCall's Quality Model (1977), Boehm's Quality Model (1978), and others were among the quality models presented.

Quality factors are used in a hierarchical approach in McCall's Quality model. He attempted to bridge the gap between users and developers by incorporating elements that affect both the demands of users and developers. Product revision (factors that signify ease of making changes within an operating environment), product transition (easy of changing from one environment to another), and product operations (the operation characteristics) were all part of the McCall's Triangle of Quality . The model goes into greater depth on quality aspects. It's broken down into a series of factors, quality criteria, and metrics. Factors represent the point of view of users, quality criteria represent the needs of developers, and metrics provide a way of measurement. The Factors are used to characterise many forms of system behaviour. The quality criteria are characteristics associated with one or more of the quality factors. Metrics attempt to catch some of the characteristics of a quality criterion. The aim behind his Model was that the quality factors that resulted would provide a complete picture of software quality.

Boehm's Quality Model, which is analogous to the McCall Quality Model, is another commonly used model. This model is organised into three levels: high-level features, intermediate-level characteristics, and primitive characteristics, each of which adds to the total quality level. As-is utility, portability, and maintainability are among the higher-level traits, which are linked to seven intermediary characteristics. Both models appear to be similar, however McCall's quality model concentrates on the accurate measurement of high-level attributes such as "As-is utility," and Boehm's quality model is based on a broader variety of characteristics, with a stronger emphasis on maintainability.

2.5 Natural language specifications

The most important aspect in delivering a quality product that meets stakeholders' expectations is the software requirement specification document (SRS). It contains information on both user demands and system specs that are required for product development. Throughout the software development cycle, users such as system engineers, test engineers, automation engineers, managers, and others refer to the SRS document. As a result, requirements in an SRS document must be easily understood by users and stakeholders with no technical knowledge.

System requirements can be written in a variety of ways, including natural language specification, structured natural language, graphical notations, and more. However, consumers and stakeholders prefer natural language specification (NLS) since it is intuitive and widely understood. It uses natural language sentences to list the needs, with each sentence corresponding to a single requirement. For instance,

NFR: The user when requesting to delete an item in an inventory must be authenticated.

It may also be supplemented by diagrams or tables if several alternative actions have to be defined. For example a requirement may be as follows,

NFR: The machine must monitor blood pressure and deliver a dose, if required, every 4 hrs.

A tabular specification of computation for dose administration may be as follows,

Condition	**Action**
Blood pressure stable ($A = B$)	Dose = 0
Blood pressure low ($A < B$)	Dose = 0
Blood pressure high ($A > B$)	Dose = 1

Despite the fact that NLS is universally comprehensible and expressive, it frequently confronts issues. It can be difficult to be as plain and unequivocal as possible without jeopardising the document's readability, and as a result, there is often a vagueness. It's sometimes difficult to tell the difference between functional and non-functional requirements in requirements. Furthermore, non-functional needs in natural language tend to interfere and conflict with one another, resulting in requirement misunderstandings. Furthermore, many requirements may be articulated in sentences, resulting in requirement fusion.

2.6 Conflicts in Non functional requirements

Non-functional requirements are an important aspect of any software's success. Non-functional needs must be specified correctly and without ambiguity when defining user requirements [8]. Non-functional needs are more difficult to define and represent than functional requirements. Many people are involved in the software development process, and each has their own perspective on the product [2]. Different persons and circumstances in which the system is being developed can observe, interpret, and assess NFRs in different ways. The significance and perception of NFRs varies depending on the system being created and the level of stakeholder involvement [12]. Different stakeholders may have different requirements and views on the system. Different points of view result in a variety of contradictions in software requirements [6]. According to Robinson, Pawlowski, and Volkov, inconsistencies in requirements generally reflect inconsistencies in the needs of system stakeholders.

These various perspectives that make up the software development life cycle cross and overlap, resulting in conflicts [2]. Meaning, meeting one requirement may have an impact on meeting another, and two NFRs may have a negative relation [5]. As a result, non-functional requirements tend to conflict, interfere, and contradict one another. The inevitable trade-offs may be affected by certain combinations of NFRs in the software system (Boehm & In 1996b; Ebert 1998; Wiegers 2003) [6]. As a result, conflicts in NFRs are commonly recognised as one of their numerous characteristics (Chung et al. 2000). Because of the intrinsic contradictions among the NFRs, preventing disputes is unavoidable [6].

An example of two conflicting NFRs is:
Example1:
NFR1: A user should be authenticated via OTP before performing any transaction on the database (NFR type: Security)

NFR2: Performing a transaction on the database must be easy for the user and must not involve transition to a different page (NFR Attribute: Usability)

Here, in the above example, attainment of NFR1 to improve the security of the system negatively affects the fulfilment of NFR2 which is to improve usability of the system. To enter the OTP, the user might need to transition to a different page for verification. Thus fulfilling the NFR1 hinders the fulfillment of NFR2. Thus, these two requirements are in conflict with each other.

2.7 Ontology

Ontology, according to Guarino (1998), is "an artifact composed of a specific vocabulary meant to describe a particular reality, as well as a set of explicit assumptions about the lexicon's intended meaning." To put it another way, an ontology is a data framework used to portray a domain's conceptual framework and the relationships between them.

Ontology is a paradigm that captures and expresses knowledge of an organisation. This model is used to visualise the relationships between various entities inside an organisation, which can guide us in answering challenging problems.

The data could be in a variety of formats. As a result, understanding the relationships between diverse entities becomes challenging. Organizational policies, for instance, may be linked to a variety of business operations. However, based on the textual description of policies and the modular representation of business processes, it can be hard to determine their relationship.

These types of relationships in a domain can be identified using ontology. Ontologies can be used to check a knowledge's consistency or make implicit information apparent.

Web ontology language is a W3C standard that oversees the creation of ontologies (OWL). The W3C Web Ontology Language (OWL) is a computational logic-based language that allows computer programmes to use knowledge contained in OWL. Ontologies are made up of two basic components, according to OWL. 1.) Classes and 2.) Relationships.

The figure below shows the representation of classes and relationships.

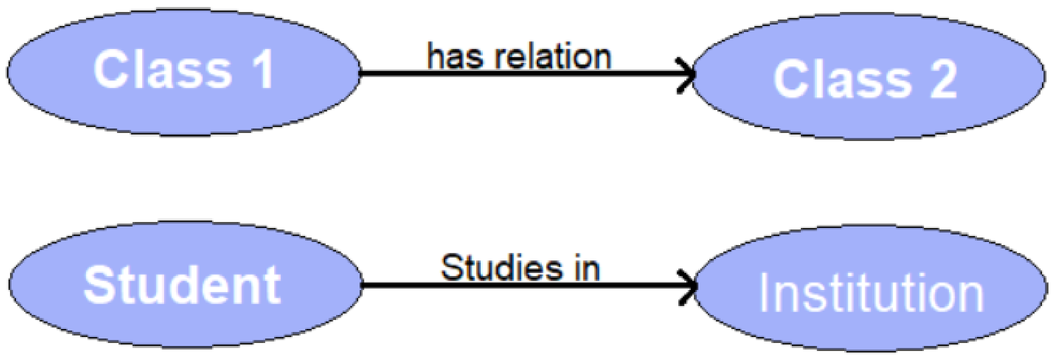

Figure 4: Ontology class and relation

When compared to plain language or source code, representing knowledge in the form of ontologies has numerous advantages.

Natural language is not understood by machines. As a result, it can't be utilised to automate or communicate with any feature.

Source code is notoriously difficult to manage and alter. It will be difficult and time consuming to add any other relationship or object using source code.

At any time, new relationships or entities can be added to ontologies. We can simply establish a new relationship between the student and the subject, as well as the institution and the subject.

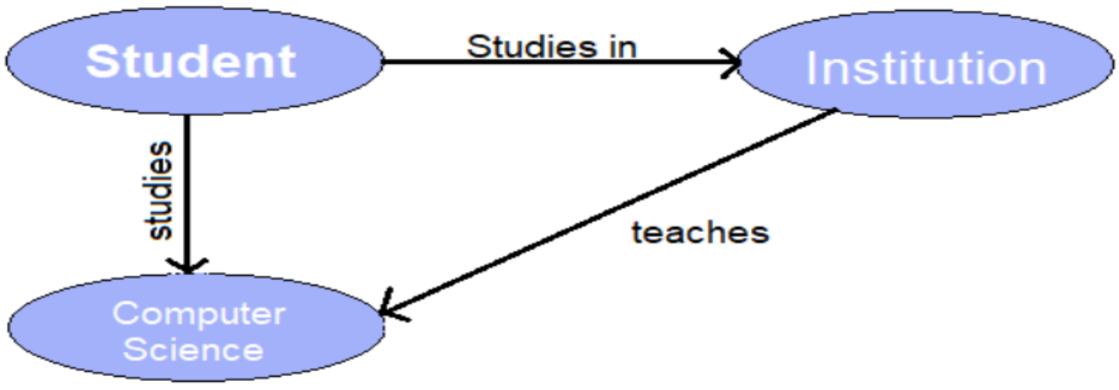

Figure 5: Adding new ontologies to existing ontology

NFRs are derived through interviews, transcripts, and discussions and are available in Natural Language. Most of the terminologies and concepts used to describe NFRs are ill-defined, and there is frequently no universally accepted term for a broad concept[1].

To facilitate successful communication and integration of activities across the RE community, a common foundation is essential. Ontologies are used to create this Common Foundation. In the domain of NFRs, ontologies will share the meaning of terms and concepts.

The application of ontology is advantageous since it aids machine thinking. Ontology also entails the modelling of domain rules, which can give an additional degree of machine comprehension [1].

2.8 Knowledge representation using Conceptual Graph

In order to enhance the understanding of given NFR, modeling of the requirements by constructing conceptual graphs was proposed in [9].

Conceptual graphs enable us to describe hybrid knowledge using semantic networks, a graphic depiction for knowledge representation made of interconnected nodes that is both human-readable and computationally calculable. It has two types of nodes: square nodes, which represent entity nodes, and oval nodes, which indicate conceptual relationships between entity nodes. The conceptual graph of the following knowledge, "Sam offers a book to Rita," is shown in Figure 6.

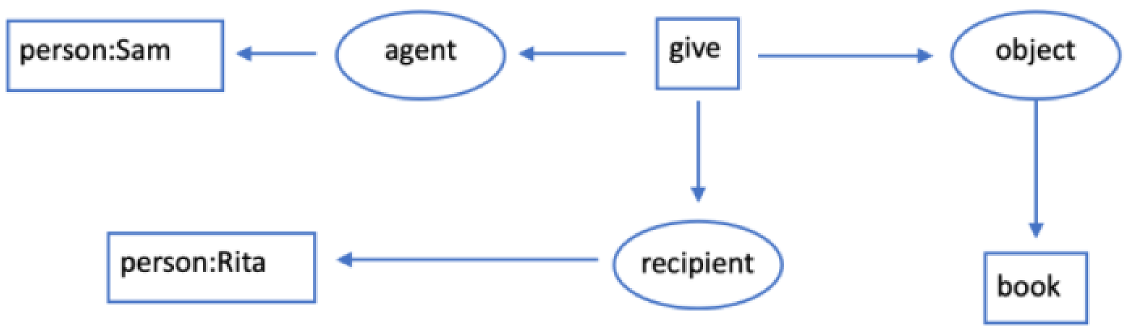

Figure 6: Conceptual graph of "Sam gives a book to Rita"

Natural language NFRs can be represented in the form of a conceptual graph to have a clear understanding of NFRs, detect ambiguity in NFRs and identify conflicts between two NFRs.

NFR can be regarded from two angles: first, NFR and its requirements, such as restrictions and business rules, and second, NFR as a requirement and its quality attributes [9]. The latter perspective of NFR is examined for the aim of discovering quality attribute discrepancies. To model the quality attributes requirement for each NFR in the SRS document, Len Bass[9] proposed a quality attribute general scenario consisting of six elements (source of stimulus, response, environment, stimulus, response, artifact and response measure).
For a security requirement such as, R1 - *" The internal user when requests to delete an item in the inventory should be authenticated."*,

the conceptual graph would be as shown below.

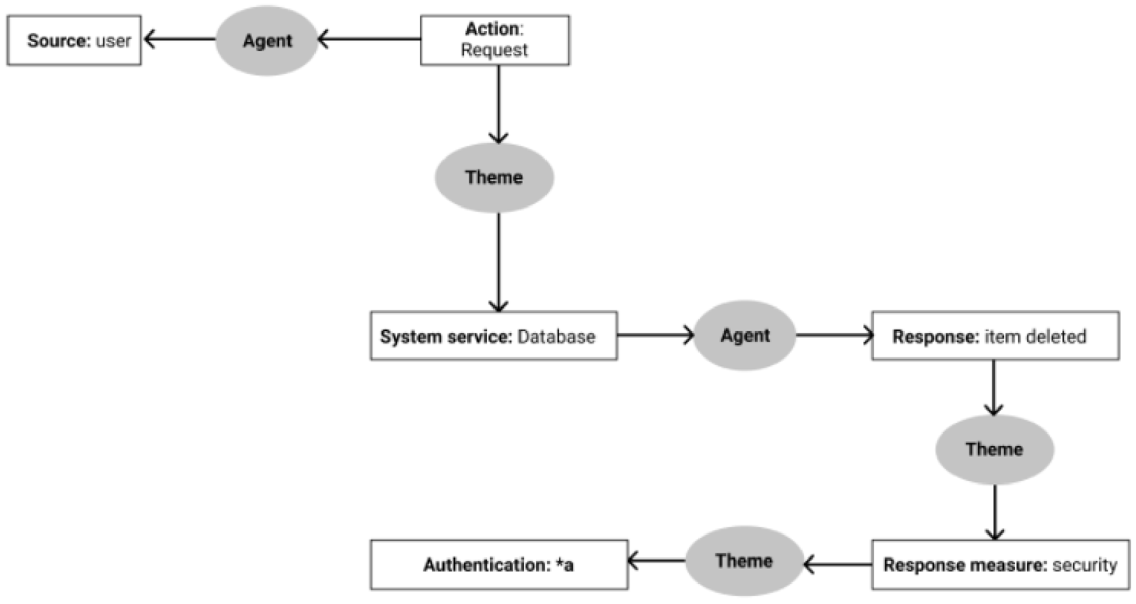

Figure 7: Conceptual graph for requirement R1

Requirements are frequently obtained from several stakeholders throughout the planning and analysis phase, and as a result, multiple requirements may refer to the same quality attribute, resulting in conflicts.
For example, the above requirement(R1) gathered from a different stakeholder may be as follows,
R2 - *" Anyone who sends a request to delete an item must be authenticated by fingerprint."*
and the conceptual graph would be as shown below.

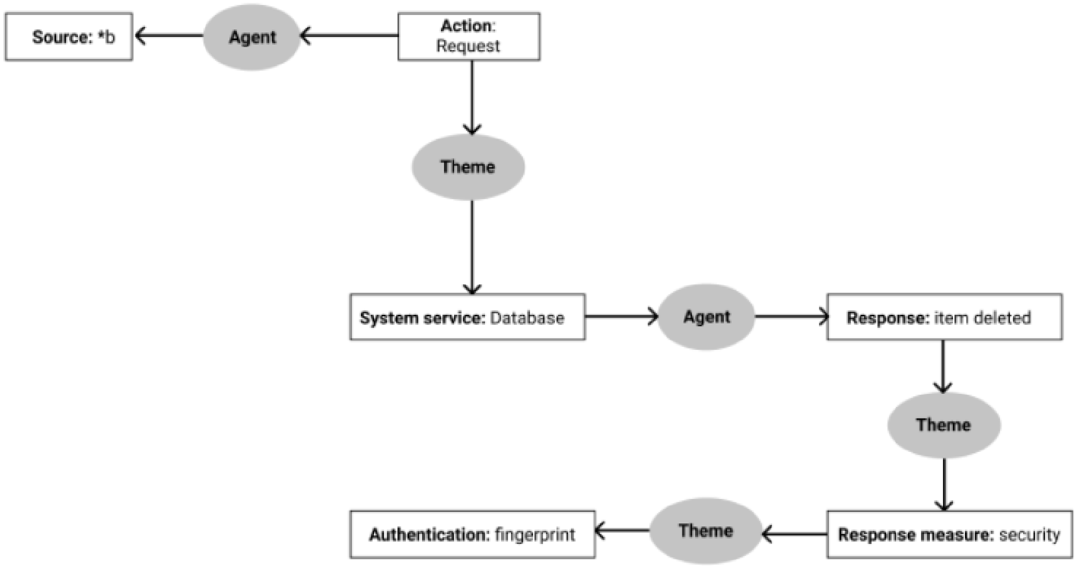

Figure 8: Conceptual graph for requirement R2

Using logical inference rules, we can examine the conceptual graphs of requirements R1 and R2 for any potential conflicts. Figure 7 shows that the stimulus comes from a user who sends a request to the database and must be authenticated to meet security requirements. Figure 8 shows a similar ph with minor differences in the source of stimulus (any general user) and the response measure of security (fingerprint authentication). We may deduce that a generic user can be a universal instantiate of a user, but fingerprint authentication cannot be a universal instantiate of general authentication because it is a specific type of authentication, as shown below. As a result, we discover that the requirements are inconsistent and in conflict with each other utilizing existential logic and inference principles.

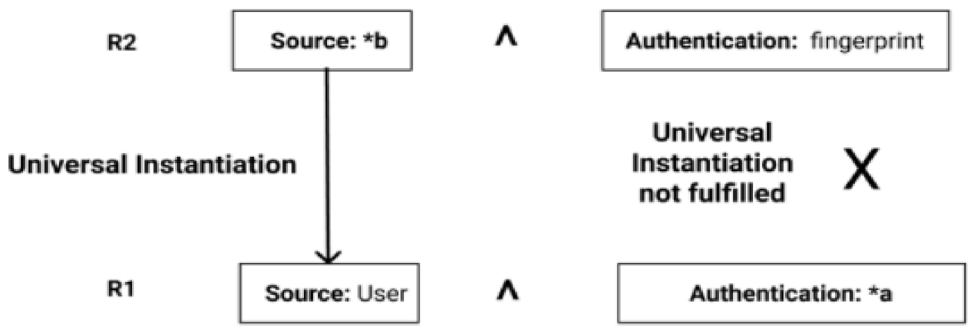

Figure 9: Conflict identification between R1 and R2 using conceptual graph

2.9 Machine Learning techniques to classify NFR

The science of machine learning occurs when a computer is programmed to learn from data. Machine learning can be defined as the creation of mathematical models to aid in the understanding of data [15]. Machine learning has lately been applied in a variety of fields, which include computer vision, natural language processing, fraud detection, chat bots, and personal support. Due to its adaptability and accuracy for automated text mining, machine learning algorithms have recently gained popularity among text categorization researchers [16]. Machine Learning (ML) has recently been investigated in the field of software engineering for both management and software development [19]. Recent technological advancements have resulted in a significant rise in the amount of data that can be used to apply machine learning techniques for a specific application. A considerable number of publications in the field of software requirements have used machine learning algorithms to detect and classify software needs [16]. In most cases, Machine Learning models are more accurate than manual labor. According to an existing NFR dataset from the 2017 Data Challenge Dataset of the International Requirements Engineering Conference, five security experts failed to unanimously classify more than half of the security requirements [18]. While applying Machine Learning approaches, we can classify NFRs considerably more accurately. This demonstrates how Machine Learning can outperform and make manual labor easier.

Any machine learning application involves these 3 stages: Feature Extraction or data-preprocessing, model training and performance measure.

Here, we study the classification of natural language NFRs using machine learning.

Feature Extraction for classifying Natural Language NFRs be done by the following:

1. **Pre-processing:** We must execute some data pre-processing before we can use the data to train any machine learning model. Noise is present in natural language requirements. Noise refers to words that are frequently used yet provide little useful information. For example, "the," "or," "is," and other words that appear frequently in sentences but contribute no useful information. Stop words are another name for these. Punctuation symbols are also considered noise because they contain no useful information.

   Other text-normalization procedures, such as case conversions and stemming, are applied once the stop words are removed. Computers, for example, distinguish between the words "Man" and "man." We'll lowercase all words so that they can all be read as one.

Also words like, “playing” and “play” have the same context as “play”. So stemming is performed for all the words. Stemming is the procedure of removing the last few characters of the word, so after performing stemming, the word “playing” will be converted to “play”.

These pre-processing steps are very important for any NLP based classifier.

2. **Bag-of-Words:** Only numbers are understood by computers. Hence, we must transform words in a phrase to numbers in order to train the model. Text tokenization or vectorization is the term used for this process. The Bag-of-Words approach is employed for this purpose. The Bag of Words model is one of the most basic still very effective text vectorization algorithms. In this approach text sentences are turned into vectors, with each text representing the frequency of all distinct words included in that sentence. [20]. We keep track of the number of times each word appears in a phrase and delete information about the structure or order of the words. This technique turns variable-length texts into a fixed-length vector since the sentences are jumbled and not of fixed length.

Following are some most common machine learning models that can be used for training:

1. **SVM:** For classification and regression, SVM is a powerful and versatile supervised learning method [21]. The hyperplane between the two classes of data points is created via SVM. The decision boundary is another name for this hyperplane. Along with this hyperplane, two more parallel planes are generated on both positive and negative sides, each passing through a point closest to the hyperplane. The marginal distance is the distance between these two parallel surfaces. The hyperplane is chosen so that the marginal distance is maximized.

2. **Multinomial Naïve Bayes:**Multinomial Naive Bayes is a version of Naive Bayes that considers the frequency of words rather than their appearance in each input [22]. The renowned Bayes theorem proposed by Thomas Bayes forms the foundation of the Naive Bayes machine learning model. The Bayes Theorem assesses the likelihood of an event A occurring given an occurrence B.

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

The independence of various features is assumed by Naive Bayes. The chance of an input being relevant to a particular predefined class is computed using Naive Bayes. The category with the highest probability is the output.

Multinomial Naïve Bayes is generally used in document and text classification [23]

3. **Logistic Regression:** Logistic Regression is a regression class in which an independent variable is utilised to predict the dependent variable, according to Abdul et al. [24]. When the dependent variable has two classes, binary logistic regression is applied. When the dependent variable includes more than two classes, multinomial logistic regression is applied. In logistic regression, a hypothesis $H = W * X + B$ is utilized. The activation function is used to predict the result of this hypothesis.

4. **Artificial Neural Networks (ANN):** An ANN is a computer programme that mimics the operation of an actual human brain. It has neurons that can discern patterns in data. There are three sections(layers) to the Artificial Neural Network. 1) The input 2) Output 3) Hidden layers. The amount of features in the input data determines the size of the input layer. The size of the hidden layer is a configurable hyperparameter, and the size of output layer is number of classes to be tested in the dataset. [25]

Performance metrics are mathematical formulas that are used to compare the model's predictions to the actual values in the dataset. Four performance measures are taken into account.

1. **Accuracy:** It is the fraction of correctly classified observations.

$$Accuracy = \frac{(TP + TN)}{(TP + FP + TN + FN)}$$

2. **Precision:** It is the ratio of proportion of true positive cases to the total positive cases.

$$Precision = \frac{TP}{(TP + FP)}$$

3. **Recall:** It is the proportion of the positive cases that were identified correctly

$$Recall = \frac{TP}{(TP + FN)}$$

4. **F1-score:** F1-score is obtained by combining both precision and recall into one. It is also called F-measure.

$$F1 - score = \frac{2TP}{(2TP + FP + FN)}$$

2.10 Word embedding

Word embedding is a term used in natural language processing to describe how words are represented in text analysis, which is usually in the form of a real-valued vector that encodes the semantic meaning of any word. The most important feature of word embeddings is that semantically related words in the vector space have a smaller distance between them than words that are not semantically related. Words like "security" and "privacy," for example, should be placed closer together than "security" and "design." Glove embedding refers to word representation using global vectors. It's a Stanford-developed unsupervised learning system for generating word embeddings from a corpus's global word-word co-occurrence matrix in order to learn the linear relationship between words.

2.11 LSTM

LSTM networks are a type of recurrent neural network that can learn order dependence in sequence prediction tasks. This is required in a variety of fields with complex issues, such as machine translation and speech recognition. When there are more than 5–10 discrete time steps between relevant input events and target signals, RNNs fail to train. The vanishing error problem raises doubts about whether standard RNNs can actually demonstrate time window-based feedforward networks in terms of practical benefits. The fact that LSTMs were one of the first methods for overcoming technical obstacles and delivering on the promise of recurrent neural networks is a key factor in their success. An LSTM layer is made up of recurrently connected blocks called memory blocks. These blocks are analogous to digital computers' differentiable memory chips. Each one contains one or more recurrently connected memory cells and also three multiplicative units (input, output, and forget gates) that offer continuous copies of memory cells' write, read, and reset operations.

2.12 Dense NN

Dense neural networks are the most basic network architecture for fitting classification models to text data, and they're a good place to start learning more advanced model designs that are more extensively used in practice for text modeling. These models contain a large number of parameters and require different preprocessing than classification and regression models. Based on the arrangement of tokens in the original text, we may tokenize and model features. This allows a model to learn from patterns in sequences and order, something that other models are unable to achieve.

2.13 Contradiction

Automatic contradiction detection detects discrepancy, inconsistency, and defiance. Consider a political debate in which two candidates take opposing positions: one declares, ”I support the new anti-corruption law,” while the other declares, ”I oppose the new anti-corruption law.” The following is an example of a negative. ”More than 50 individuals died in the plane crash,” and ”10 people died in the plane catastrophe,” are two examples of numeric disparities between contradicting assertions.

3 Literature Surveys

The author Xuan Zhang and Xu Wang proposed an efficient trade-off framework in [11]. Tradeoffs are situations involving losing one of the either entities in consideration in return for fulfilling other aspects. The demand for specific NFR encourages team to develop strategies in a methodological manner which may hinder with other NFRs. Hence it's crucial to manage trade offs for such conflicting strategies. Firstly, the NFR's are gathered from the stakeholders and then application of set theory is used to assess the importance of each NFR. After assessment of NFR's, production theory from micro economics is adapted to provide analysis by visualization of the tradeoff for conflicting NFR and then linear programming is used to make tradeoff decisions.

In [9], the authors proposed a way for knowledge representation of NFR using conceptual graphs, a directed bipartite graph consisting of square nodes to represent concept and oval nodes to represent conceptual relation between nodes, due to their ability of semantic and existential logic representation. Moreover in [9], the NFR's are taken into account as a quality attribute and proposed a quality attribute general requirements for NFR consisting of source of stimulus, the stimulus, the artifact, environment, response, and the response measure. Based on quality attribute general requirements each of the NFRs are formally represented using the graphical model. For identification of conflict the author evaluates the requirement conflict by encompassing logic in case of universal instantiation, generalization, and existential instantiation.

The author Chi-Lun Liu, proposed an Ontology based approach for identification of conflicts among NFRs in [8]. It consists of ontology, metadata and its various rules for development of ontology's. Ontology is a controlled representation of vocabulary, which aims at representing structured knowledge. It includes concept and semantic relationships between the concepts. The metadata reveals the structure of the NFR. Metadata are like empty fields of a blank form whereas ontology is analogical to a dictionary. In [8], the author proposed to represent NFRs by filling the metadata. One example of proposed metadata by the author is "Software attribute of module should be some specific values" [8]. Here, 'software attribute', 'module' and 'some specific value' are ontologies. We fill the metadata to represent NFR. Rules were proposed in [8], which encompasses ontology and metadata to analyse NFR conflicts between two given NFRs. These rules identify conflicts systematically and effectively by identifying contradiction between 2 NFRs. The author proposed 7 such metadata and rules in the paper. The author also presented an example to validate this methodology.

Unnati Shah, Sankita Patel, and Devesh Jinwala provided an ontology-based approach for finding conflicts from Natural Language NFRs in their paper [7]. The NFR meta-model, quality ontology, and operation ontology are built using Natural Language NFRs. Then, to discover conflicts among NFRs, Semantic Web Rule Language (SWRL) rules are established. Preprocessing, building ontologies, defining semantic ontologies, and finding conflicts were the four components in [7]. The programme translates NFR available in Natural Language into NFR meta-model representation during preprocessing. The Stanford parser was used for lexical-syntactic analysis, while WordNet was used to determine context information such as noun, verb, pronoun, and so on. The meta-model is also separated into six sections. The prot'eg'e tool was used to create quality ontologies and operational ontologies. Quality Attribute Relation (QAR) libraries have been proposed to store and reuse quality attribute knowledge descriptions in order to define semantic correlation. The Semantic NFR Catalogue (SNFRC) ontology instances are created from the QAR catalogues. The authors established a link between quality and operational parameters. The relationship between quality and operational variables must be defined manually. Finally, SWRL rules were established to identify conflicts among the NFRs in order to identify conflicts. Because the authors used SWRL rules, conflict detection may be done automatically.

The author of [4] presents a model for detecting NFR (non-functional requirements) conflicts . The proposed solution entails discovering and analysing non-functional requirement conflicts. The matrix approach is used in this model to detect quality-based clashes among non-functional requirements. This model was an elaboration to Sandana and Lui's model, which was further based on NFD proposed by Poort and deWith. It was a model for converting contradictory requirements into a system. The mapping of low level conflicting quality attributes to low level functionality, in [4], highlights conflicts. This model consists of a series of phases. To begin, we must write out all of the high-level non-functional requirements in a methodical manner and define quality and functionality attributes in each NFR. Second, create a hierarchy of quality attributes based on high-level functions and a series of functionality attributes to identify low-level functionalities. Finally, create a quality to quality matrix and use a 'x' to indicate any attributes that contradict. As a result, the low-level quality criteria that contradict are found. Finally, look for potential conflicts by mapping low-level conflicting quality attributes to low-level functionality. Finally, the methodology suggested in [4] identifies conflicting NFRs by mapping low level conflicting quality attributes over low level functional attributes. It not only identifies but analyzes conflicts in NFR too using matrices.

In [2, the authors Vishal Sadana and Xiaoqing Frank Liu developed a methodology that uses an integrated analysis of functional and non-functional needs to assess and detect conflicts among non-functional requirements. According to the link between quality attributes, constraints, and functionality, this model in [2] discovers and analyses conflicts. It accepts a list of high-level non-functional requirements, quality attribute hierarchies, constraint hierarchies, and functionality hierarchies as inputs. It also takes into account correlations between quality attributes, limitations, and functionality. Non-functional requirements are first expressed in a systematic manner. The relationships between quality traits, functions, and restrictions are used to identify conflicts in high-level non-functional requirements. In the following processes, a conflict is identified, examined, and filtered based on an integrated examination of functional and nonfunctional needs. This process continues until all high-level conflicts have been studied and narrowed down to specific low-level conflicts. This model is unique and superior because it allows us to capture the semantics of conflicts based on relationships between quality traits, functionality, and limitations, which other models appear to lack.

In [6], the authors have discussed classification of NFRs and why they are so essential in Requirement Engineering. 106 known NFRs are enlisted, which belong to various application types. Then essential concepts about NFR conflicts, property, definition and potential causes of conflicts are discussed, followed by the importance of dealing with conflicts among NFRs during software development. Next section states standard methods to deal with conflicts. From various definitions about conflicts among NFRs, conflict among NFRs is interference – negative contribution of one NFR on another NFR. From inference, a pair or a set of NFRs cannot be satisfied at the same time. In the research about conflicts among NFRs, researchers tend to focus on two different aspects of conflict: the relationships and the tradeoffs. Relationships focus on how NFRs contribute negatively to the other NFRs while the tradeoffs focus on the situation where we can not achieve two or more NFRs at the same time.The authors in [6] also discussed various methods to identify, analyse and resolve conflict. In the last section, a preliminary framework is proposed, which not only identifies the conflicts, but type and significance of conflict, as well as the appropriate potential strategy to resolve the conflict.

The authors Eltjo R. Poort and Peter H.N. de in [10], presented us with a solution that focuses on the mapping of non-functional requirements onto the functional requirements for architecture design. The proposed model in [10] converts conflicting requirements into a system decomposition. This NFD model reinvents requirement classification and is detailed in non-functional aspects. Functional requirements are first separated into primary and secondary FRs, with the latter being coupled with NFRs to form supplementary requirements. Supplementary requirements are further divided into Secondary Functional Requirements (SFRs), Quality Attribute Requirements (QARs) and Implementation Requirements. The three types of solution strategies are functional, structural and process solutions. The NFD model creates a 3 x 3 matrix with the supplementary requirements and architecture. The diagonal cells of this matrix contain direct strategies and all the others provide solutions that can help with achieving requirements. The NFD procedure aids in the optimization of the system's structure for all supplemental requirements. This is accomplished by adapting the system structure to the system's requirement conflicts and splitting the conflicting requirements into subsystems that can be optimised separately using functional, structural, or process strategies.

3.1 Summary of Literature Surveys

After reviewing many scholarly sources, we summarized our research and provided an overview to our present insight on the topic at discussion. Our research was divided into two areas - one was Identification of conflicts in non-functional requirements and the other was to resolve the said conflicts. From what we grasped there were many of methods to identify and analyze conflicts in NFRs, some of the said methods were using Ontologies, Conceptual graphs and Matrix maps. To resolve a conflict in NFRs, the NFD approach is given by Eltjo R. Poort and Peter in [10].

4 Proposed Work

Our proposed approach is divided into 3 models. 1) Grouping Similar NFRs 2) Tuning word embeddings 3) Contradiction Detection

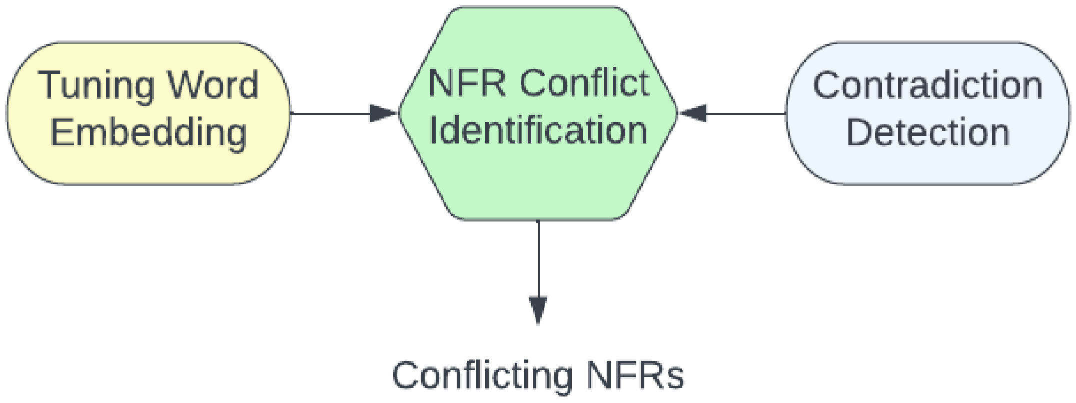

Figure 10: "High level diagram of proposed approach"

4.1 Grouping Similar NFRs

We proposed to group two similar NFRs. The group is done by taking a vector subtration of sentences, after converting the sentence into Embedding representation. We have fine tuned Glove Embedding to provide context of the domain, requirements and quality attributes.

Embedding representation provides context similarity in two sentences. If two words are similar, their embedding vector values also tend to be similar. Also to perform a vector subtraction, we require both the sentences in a single vector form. Thus, we take average of all word vectors in a single sentence, and then perform vector subtraction.

4.2 Tuning word Embeddings

We propose to use a Dense Neural Network to refine Glove embeddings. Input layer has total number of neurons equal to the total number of words present in Glove Embedding Vector. And output layer will have total number of neurons present in the input dataset.

We also plan to manually create dataset. The dataset can be created by extracting the content of different SRS from different domains. SRS documents will provide domain knowledge and also the context about requirements.

4.3 Contradiction Detection

Once similar NFRs are paired together, we can identify if the two similar sentences are contradicting each other or not.

We studied few approaches in Natural Language Processing to identify contradicting statements. Contradicting statements usually have either of 1) Negation words 2) Antonyms 3) Numeric Mismatch [30]

We also studied few Deep Learning approaches to identify contradiction. Some of the useful works are [30], [31].

From the above references papers, [30] gives highest accuracy on SEMEVAL data. Thus we plan on implementing that approach for our contradiction detecton model.

5 Simulation and Results

Our application simulation/implementation consists of 1) Tuning word embedding model 2) Grouping Similar NFRs 3) Contradiction Detection Methodology.

5.1 Tuning word embedding

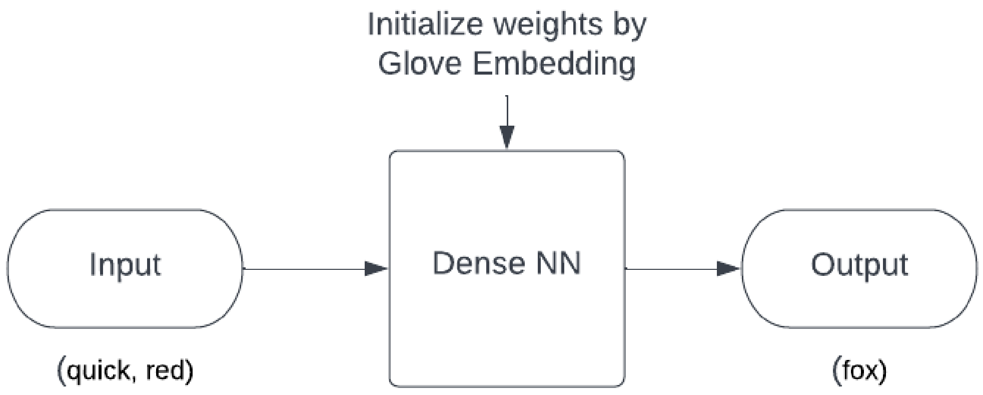

Figure 11: Tuning word embedding

Here, weights(features) will be initialized by glove embedding and use a dataset(SRS documents) to form a Dense Neural Network. This is a dummy model we use to fine-tune the actual glove embedding model to provide it the context of requirements present in SRS documents. We fetch weights after training the model, which acts as tuned word embedding.

Dataset: Our dataset comprises of content of 18 SRSs and some web sources. We manually created this dataset by extracting content from the SRS document.

Input: first two words of the tri-gram

Tri-gram: Tri-gram is a sequence of 3 consecutive letters extracted from a sentence.

Example: Sentence - ”Apple a day keeps doctor away”
Trigrams - “Apple a day”, “a day keeps”, “day keeps doctor”, ”keeps doctor away”

Dense NN: Dense layer is a network of neurons which are interconnected to each other. Previous layer neurons are connected to next layer and so on. Input layer has total number of neurons equal to the total number of words present in Glove Embedding Vector.

Output: third word of the trigram

Total number of sentences: 626

Total number of trigrams formed are: 22220

5.2 Grouping Similar NFRs

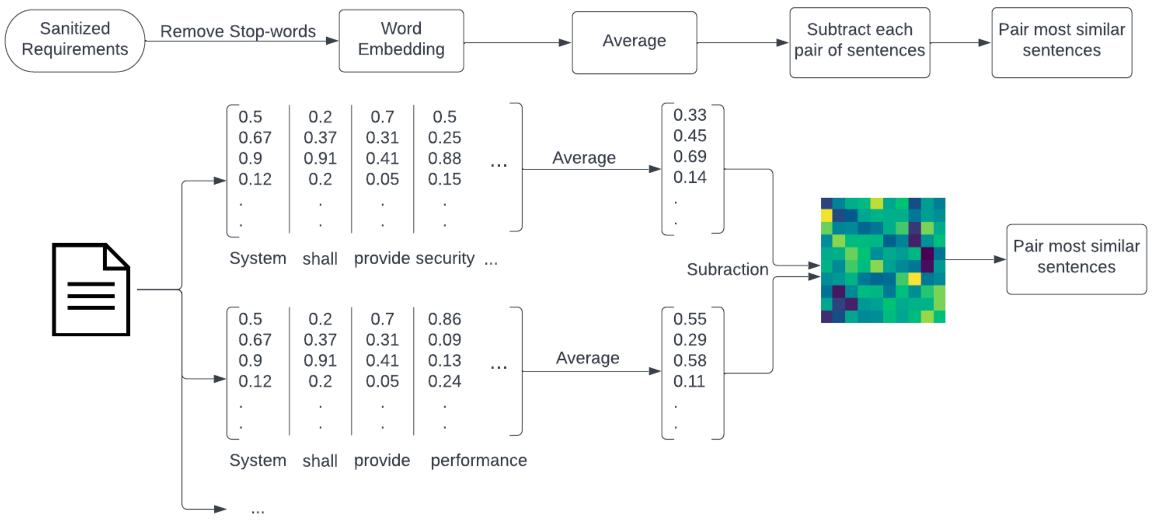

Figure 12: Grouping Similar NFRs

Natural language requirements are first sanitized. Sanitized requirements are further preprocessed using stop words removal. These sanitized requirements are now fed as input to the tuned word embedding model, where it converts each word to a vector with multiple dimensions. Each sentence or requirement can differ in length, therefore the average of all the words in a sentence is calculated to bring it down to a single vector with multiple dimensions. Average is then calculated for all the sentences or requirements by computing the mathematical mean of all the corresponding vector dimensions to result in a single vector. After calculating the average, subtraction is performed between every pair of calculated average vectors. After subtraction, the sentences with the most common context can be identified easily as they will have the least difference. Sentences with the least difference are clubbed together, as they may have contradiction.

Example:

Requirement 1: System shall provide security

Requirement 2: System shall provide performance

These two requirements have the same context and hence can have contradiction. The difference between these two sentences will come out to be very less hence they will be clubbed together to further check whether there is contradiction or not.

Now, every pair of similar requirements are passed to a contradiction detection model, which will identify whether the conflict is present or not.

5.3 Contradiction Detection Methodology

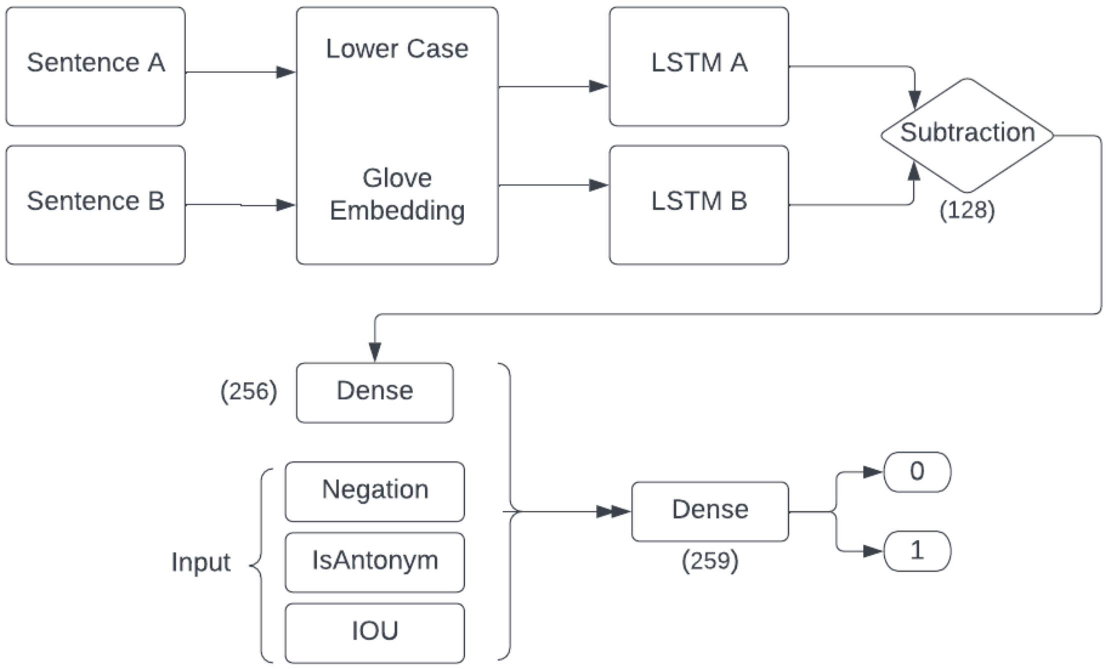

Figure 13: Contradiction Detection Methodology

Our contradiction detection model is based on the work proposed in [30]. The methodology has used deep neural network, LSTM, and Glove word embeddings. We conducted our experimental evaluation consisting of training and testing on the SEMEVAL dataset.

Total number of samples in training dataset: 3822
Total number of samples in test dataset: 3513

As shown in Fig.[13], our high level solution approach has many number of steps. The 1st step consist of taking a pair of sentences as input and convert it all into lower case. Stop words and term stemming isn't performed as terms like "and" and "not" are useful features for contradiction detection . Similarly, numeric characters like "60" or "4" are useful features for numeric mismatch contradiction detection. We implemented our model using python programming language and conducted experiments on deep learning and neural network by using the Tensor-flow python library. We convert natural language words into GloVe embedding vectors, which act as features for contradiction detection. Using GloVe embeddings, words are created into numeric vectors which signifies contextual meaning of the words. It helps in performing mathematical operations on words as shown in the Fig.[13].

Then we apply LSTM to our vector representation of sentences which extracts the contextual meaning from the sentences.

The output of the LSTM provides useful features from the sentences. Two feature vector from every pair of sentences will be produced after LSTM layer which then will be subtracted to produce a single output feature.

To further improve the accuracy of our model we further integrated three new features with the features generated by the dense neural network. As shown in Fig.[13], we integrated the following features,

1. **Negation:** A binary feature which takes value as true, when any of the sentence in a pair consists of negation words such as no, never, nothing and so on from our predefined list of negation words and the other sentence does not contain any word from our list of negation words. Otherwise, it takes value as false. It aims at identifying contradiction by capturing negative sentiment in one sentence as compared to other in a pair.

2. **IsAntonym:** A binary feature which takes value as true, when there exist no two such pairs in two sentences which are antonyms to each other. Otherwise, it take value as false. We used WordNet and NOC dataset antonyms [30] to find antonyms in given pair of sentences.

3. **Intersection over union (IOU):** Also known as Jaccard Coefficient, is a widely sued metric for measuring the similarity between two text inputs. In our pair of sentences, it is obtained by dividing the number of common words in both sentences to the total number of unique words in both sentences. It aims at capturing the likeliness between two sentences.

After combining the features from LSTM and additional 3 features, we pass the combined features to a Dense Neural network. The output of the Dense Neural Network is a single number between 0 and 1, telling whether the contradiction is present or not in a given pair of input sentences.

5.4 Experiment and Results

We have fine-tuned Glove Embedding by training it on data from 18 SRSs and few web sources that provide valuable context.

Total number of sentences: 626

Total number of trigrams formed are: 22220

The accuracy of the contradiction detection model came out be 92.26% on SEMEVAL test dataset.

Experimental testing was performed on requirements of ”Crime Criminal Tracking Network and Systems” SRS document and the results obtained are as

	Total Requirements	NFR Conflicts detected from			
Requirements Type		Contradiction detection model	Checking Antonyms	Checking Negation	Checking Numeric Mismatch
Access Module	10	2	0	2	0
Availability	5	3	1	3	0
Ease of Use	10	0	0	0	0
Performance	6	2	0	3	2
Usability	50	4	3	8	1

Experimental testing was performed on requirements of ”Tactical Control System” SRS document and the results obtained are as

	Total Requirements	NFR Conflicts detected from			
Requirements Type		Contradiction detection model	Checking Antonyms	Checking Negation	Checking Numeric Mismatch
Safety	11	2	1	2	0
Human Factor	42	5	0	7	0
Security	10	1	1	0	0
Other	18	5	2	2	1

5.5 Conclusion

We studied why Non Functional Requirements (NFRs) can play a vital role in accomplishing any successful application. We studied why and how NFRs tend to interfere and conflict with each other and the outcome if the conflicts are not resolved at the earlier stage of the Software Development Life Cycle (SDLC). We studied various techniques for conflict identification such as conceptual graph, ontological approach, etc and based on the study we proposed a approach to identify conflicts from the NFRs available in natural language. Our approach involving word embedding. We created three models, one of which is training word embedding model. Then we used it for our next step of Grouping Similar NFRs. Furthermore, we designed a contradiction detection model to detect conflicts among the grouped NFRs. We then performed experimental analysis to see the validity of our proposed approach.

5.6 Future Work

Precision of this approach can be increased by tuning the word embedding with a larger dataset, by taking content of further more number of SRS documents. Increased data can provide more contextual information for tuning word embedding which can result in better grouping of similar pair of sentences. A larger and rich set of antonyms can help to automatically identify conflicts among the similar pair of sentences (once grouped together). A huge number of conflicts are missed due to the lack of antonyms data available. Without the availability of rich antonyms, conflicts need to be identified manually. Also, validity of our proposed approach cannot be accurately checked without the availability of known conflicting requirements. Manual analysis of conflicts is needed to be done by an expert so as to compare our result with the actual conflicts.

References

[1] M. Kassab, O. Ormandjieva, M. Daneva, “An Ontology Based approach to Non-Functional Requirements Conceptualization.”, 2009 Fourth International Conference on Software Engineering Advances.

[2] V. Sadana and X. F. Liu, “Analysis of Conflicts among Non-Functional Requirements Using Integrated Analysis of Functional and Non-Functional Requirements,” in 31st Annual International Computer Software and Applications Conference (COMPSAC 2007), Jul. 2007, vol. 1, pp. 215–218. doi: 10.1109/COMPSAC.2007.73.

[3] J. Araújo, A. Moreira, I. Brito, and A. Rashid, “Aspect-Oriented Requirements with UML,” Jan. 2002.

[4] A. H, J. A, and I. U, “Conflicts Identification among Non-functional Requirements using Matrix Maps,” International Journal of Computer and Information Engineering, vol. 4, no. 8, pp. 1322–1327, Aug. 2010.

[5] R. Maia, “Dealing with Conflicts between Non-Functional Requirements of UbiComp and IoT Applications,” Sep. 2017. doi: 10.1109/RE.2017.51.

[6] D. Mairiza, D. Zowghi, and N. Nurmuliani, “Managing conflicts among non-functional requirements,” Jan. 2009.

[7] U. Shah, S. Patel, and D. Jinwala, “An Ontological Approach to Specify Conflicts among Non-Functional Requirements,” in Proceedings of the 2019 2nd International Conference on Geoinformatics and Data Analysis, New York, NY, USA, Mar. 2019, pp. 145–149. doi: 10.1145/3318236.3318257.

[8] C.-L. Liu, “Ontology-Based Conflict Analysis Method in Non-functional Requirements,” in 2010 IEEE/ACIS 9th International Conference on Computer and Information Science, Aug. 2010, pp. 491–496. doi: 10.1109/ICIS.2010.26.

[9] T. Luangwiriya, R. Kongkachandra, and P. Songmuang, “Representation of Conflicts in Non-Functional Requirement using Conceptual Graphs,” in 2021 International Symposium on Electrical, Electronics and Information Engineering, New York, NY, USA, Feb. 2021, pp. 560–567. doi: 10.1145/3459104.3459196.

[10] E. R. Poort and P. H. N. de With, “Resolving requirement conflicts through non-functional decomposition,” in Proceedings. Fourth Working IEEE/IFIP Conference on Software Architecture (WICSA 2004), Jun. 2004, pp. 145–154. doi: 10.1109/WICSA.2004.1310698.

[11] X. Zhang and X. Wang, “Tradeoff Analysis for Conflicting Software Non-Functional Requirements,” IEEE Access, vol. 7, pp. 1–1, Oct. 2019, doi: 10.1109/ACCESS.2019.2949218.

[12] D. Mairiza, D. Zowghi, and N. Nurmuliani, “Towards a Catalogue of Conflicts Among Non-functional Requirements,” 2010. doi: 10.5220/0002927900200029.

[13] V. Bajpai and R. Gorthi, “On non-functional requirements: A survey,” Mar. 2012, pp. 1–4. doi: 10.1109/SCEECS.2012.6184810.

[14] https://www.w3.org/Submission/SWRL/

[15] Canedo, E.D.; Calazans, A.T.S.; Masson, E.T.S.; Costa, P.H.T.; Lima, F. Perceptions of ICT Practitioners Regarding Software Privacy. Entropy 2020, 22, 429

[16] “Identifying Non-functional Requirements from Unconstrained Documents using Natural Language Processing and Machine Learning Approaches — IEEE Journals & Magazine — IEEE Xplore.” https://ieeexplore.ieee.org/document/9328431

[17] Iqbal, T.; Elahidoost, P.; Lúcio, L. A bird’s eye view on requirements engineering and machine learning. In Proceedings of the 2018 25th Asia-Pacific Software Engineering Conference (APSEC), Nara, Japan, 4–7 December 2018; pp. 11–20.

[18] 25th IEEE International Requirements Engineering Conference, “RE’17 Data Challenge”

[19] Lima, M.; Valle, V.; Costa, E.; Lira, F.; Gadelha, B. Software Engineering Repositories: Expanding the PROMISE Database; XXXIII Brazilian Symposium on Software Engineering; SBC: Porto Alegre, Brasil, 2020; pp. 427–436

[20] Sarkar, D. Text Analytics with Python; Apress: New York, NY, USA, 2016

[21] VanderPlas, J. Python Data Science Handbook; O’Reilly Media, Inc.: Sevastopol, CA, USA, 2016.

[22] D. Ott, “Automatic Requirement Categorization of Large Natural Language Specifications at Mercedes-Benz for Review Improvements,” in Requirements Engineering: Foundation for Software Quality, Berlin, Heidelberg, 2013, pp. 50–64. doi: 10.1007/978-3-642-37422-$7_4.Vergara, D.; Hernández, S.; Jorquera, F. MultinomialNaiveBayesforreal - timegenderrecognition.InProceedingsofthe2016XXISymposiumonSignalProcessing, Imagesan$

[23] Abdul Salam, M. Sentiment Analysis of Product Reviews Using Bag of Words and Bag of Concepts. Int. J. Electron. 2019, 11, 49–60

[24] C. Baker, L. Deng, S. Chakraborty, and J. Dehlinger, “Automatic Multi-class Non-Functional Software Requirements Classification Using Neural Networks,” in 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC), Jul. 2019, vol. 2, pp. 610–615. doi: 10.1109/COMPSAC.2019.10275.

[25] C. Ebert, ”Putting requirement management into praxis: dealing with nonfunctional requirements,” Information and Software Technology, vol. 40, pp. 175-185, 1998.

[26] E. Dias Canedo and B. Cordeiro Mendes, “Software Requirements Classification Using Machine Learning Algorithms,” Entropy, vol. 22, no. 9, Art. no. 9, Sep. 2020, doi: 10.3390/e22091057.

[27] “Detecting Intra-Conflicts in Non-Functional Requirements — International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems.” https://www.worldscientific.com/doi/10.1142/S0218488521500197 (accessed Oct. 04, 2021).

[28] “A framework for identifying and analyzing non-functional requirements from text — Proceedings of the 4th International Workshop on Twin Peaks of Requirements and Architecture.” https://dl.acm.org/doi/10.1145/2593861.2593862 (accessed Oct. 04, 2021).

[29] https://www.nltk.org/

[30] V. Lingam, S. Bhuria, M. Nair, D. Gurpreetsingh, A. Goyal, and A. Sureka, “Deep learning for conflicting statements detection in text,” PeerJ Preprints, preprint, Mar. 2018. doi: 10.7287/peerj.preprints.26589v1.

[31] L. Li, B. Qin, and T. Liu, “Contradiction Detection with Contradiction-Specific Word Embedding,” Algorithms, vol. 10, p. 59, May 2017, doi: 10.3390/a10020059.

Acknowledgement

Presentation motivation and inspiration have consistently assumed a significant part in the achievement of any endeavor. We are willing to offer our profound thanks to our project guide, Dr Devesh C. Jinwala, Professor at Computer Engineering Department, SVNIT Surat, to urge us to the most noteworthy pinnacle and to give us the occasion to prepare this project. His raising motivations, empowering direction, and kind support helped us in reaching the culmination of the project.

We also would like to express our gratitude towards Dr. Rupa G. Mehta, Head of Computer Engineering Department, SVNIT Surat, for allowing us the chance to learn new things by undertaking the project. We are additionally grateful to SVNIT Surat and its staff for giving this open door which has helped us to increase adequate information to make our work fruitful and efficacious .