Pontifícia Universidade Católica
do Rio de Janeiro

Roxana Lisette Quintanilla Portugal

Speeding-Up Non-Functional Requirements Elicitation

Tese de Doutorado

Thesis presented to the Programa de Pós-Graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Doutor em Informática.

Advisor: Prof. Julio Cesar Sampaio do Prado Leite

Rio de Janeiro

March 2020

PONTIFÍCIA UNIVERSIDADE CATÓLICA
DO RIO DE JANEIRO

Roxana Lisette Quintanilla Portugal

Speeding-Up Non-Functional Requirements Elicitation

Thesis presented to the Programa de Pós-Graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Doutor em Informática. Approved by the Examination Committee.

Prof. Julio Cesar Sampaio do Prado Leite
Advisor
Departamento de Informática - PUC-Rio

Prof. Marco Antonio Casanova
Departamento de Informática - PUC-Rio

Prof. Marcos Kalinowski
Departamento de Informática - PUC-Rio

Soelí Teresinha Fiorini
Laboratorio de Engenharia de Software - PUC-Rio

Prof. Eduardo Kinder Almentero
Departamento de Matemática - UFRRJ

Prof. Henrique Prado de Sá de Souza
Departamento de Matemática - UFRRJ

Rio de Janeiro, march 27th, 2020.

Roxana Lisette Quintanilla Portugal

Roxana has been working in the Software Engineering market since 2003. She is a Systems Technician from ISC-UNSAAC and graduated in Systems Engineering at UPN in Perú in 2008. Roxana worked as senior developer, quality assurance tester, and lead of application development until 2013. She holds a Master's degree from PUC-Rio since April 2016.

Bibliographic data

Quintanilla Portugal, Roxana Lisette

Speeding-Up Non-Functional Requirements Elicitation / Roxana Lisette Quintanilla Portugal; advisor: Julio Cesar Sampaio do Prado Leite. – 2020.
110 f.; 30 cm

Tese (Doutorado) – Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2020.

Inclui bibliográfia

1. Informática – Teses. 2. Engenharia de Requisitos 3. Elicitação de Requisitos 4. Requisitos não funcionais 5. Repositórios abertos. I. Leite, Julio Cesar Sampaio do Prado. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

To God and his Mother Maria. My faith has grown since I had the feeling that the only protection I had was their mercy. I accepted the challenges of this journey, recognizing that something in them had to be discovered by me.

Acknowledgments

To my family for unconditional support.

To my advisor, for helping me develop my research interests and for being a friend. Many times, I have thought for a quality that defines him, I found it when someone once told me, he is a gentleman.

To my co-authors, you taught me to do better research, but principally that collaboration is a must.

To my research group, I have learned and revised my ideas in every seminar we had. Thanks for being controversial.

To Prof. Sam Supakkul, for his kindness in teaching me the details of the NFR framework.

To the DI of PUC-Rio, to all professors, classmates, secretaries, and collaborators that make one proud to being part of this team.

To my relatives in the USA, for helping me with the English writing.

To Fernanda and Manuela from the Rede de Apoio ao Estudante RAE-PUC-Rio, you helped me to organize my mind better when life seemed very hard.

To Romeu, for being my friend in good times and bad.

To Eric Grandi, for giving me his apartment at the right moment.

To Rio de Janeiro, for giving me good friends, good memories, and a new breeze in my life with Lukas.

To Brazil, through CNPq, for allowing me to study at a beautiful university.

This study was financed in part by the Coordenação de Aperfeiçoamento Pessoal de Nível Superior - Brasil (CAPES) - Finance Code001.

Abstract

Quintanilla Portugal, Roxana Lisette; Sampaio do Prado Leite, Julio Cesar (advisor). **Speeding-Up Non-Funcional Requirements Elicitation.** Rio de Janeiro, 2020. 108p. Tese de Doutorado - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Considering the availability of Big Data for software engineering, as the case of GitHub, the semi-automation of non-functional requirements (NFRs) elicitation is a key strategy towards requirements definition. As such, NFRs elicitation, within the automation of document reading, can manage the mass of valuable information existing in available data. This thesis explores this context in three parts, the choice of proper sources of information, a fact-finding elicitation, and NFRs identification. The assessments performed showed that the automation faces a trade-off between efficiency and efficacy. This trade-off is detailed with different novel strategies. The acquired knowledge is organized as a SIG (Softgoal Interdependence Graph) catalog.

Keywords

Non-functional Requirements; Requirements Engineering; Requirements Elicitation; Knowledge Reuse; Text-Mining; Fact-finding; Sources of Information.

Resumo

Quintanilla Portugal, Roxana Lisette; Sampaio do Prado Leite, Julio Cesar (advisor). **Acelerando a Elicitação de Requisitos Não Funcionais.** Rio de Janeiro, 2020. 108p. Tese de Doutorado - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Considerando a disponibilidade do Big Data para engenharia de software, como no caso do GitHub, a semi-automação da elicitação de requisitos não funcionais (NFRs) é uma estratégia fundamental para a definição de requisitos. Como tal, a elicitação de NFRs, dentro da automação da leitura de documentos, pode gerenciar a massa de informações valiosas existentes nos dados disponíveis. Esta tese explora esse contexto em três partes, a escolha de fontes apropriadas de informação, uma elicitação de descoberta de fatos e a identificação de NFRs. As avaliações realizadas mostraram que a automação enfrenta um balance entre eficiência e eficácia. Esse equilíbrio é detalhado com diferentes estratégias inovadoras. O conhecimento adquirido é organizado como um catálogo SIG (Softgoal Interdependence Graph).

Palavras-Chave

Requisitos Não Funcionais; Engenharia de Requisitos; Elicitação de Requisitos; Reutilização de Conhecimento; Mineração de Texto; Pesquisa de Fatos; Fontes de Informação.

Summary

List of Figures

List of Tables

Abbreviations and Acronyms

SE	*Software Engineering*
RE	*Requirements Engineering*
API	*Application Programming Interface*
SIG	*Softgoal Interdependency Graph*
UofD	*Universe of Discourse*
SoI	*Source of Information*
IRS	*Information Retrieval Systems*
MLS	*Machine Learning Systems*
NLP	*Natural Language Processing*
POS-tagger	*Part of Speech Tagger*
BoW	*Bag of Words*
SADT	*Structured analysis and design technique*
TF	*Term Frequent*
TF-IDF	*Term Frequency–Inverse Document Frequency*
TM	*Text Mining*

“The stars are very distinct and very clear, and very high. The style can be very clear and very loud; so clear that those who don't know understand it and so loudly that they have a lot to understand those who know it. The rustic finds documents in the stars for his crops and the sailor for his navigation and the mathematician for his observations and his judgments. So that the rustic and the sailor, who don't know how to read or write, understand the stars; and the mathematician, who has read how many have written, cannot understand how much is in them. Such can be the sermon: - stars that everyone sees, and very few measure them.”

From the Sixtieth Sermon § 5. year 1655

Padre Antônio Vieira.

1 Introduction

In this section, we summarize the thesis with the context, contributions, objectives, and motivations. The background and related work enclose this chapter.

1.1. Context

According to Leite[1], the requirements elicitation process is composed of three tasks: fact-finding, communication, and fact-validation. The former can be performed using different techniques such as questionnaires, interviews, and document reading, among others. As such, an elicitor must manage the transformation of raw information into facts, restricted by the point of view of stakeholders, and the tools and methods for handling the information. In this context, existing semi-automated approaches deal with natural language (NL) documents for fact-finding, most of which are unstructured data[2].

A report from Meth et al.[2] lists facts of interest for requirements engineers, mainly related to a) abstractions of main concepts, b) domain ontology, c) classification of requirements, and d) requirements statements, which are being semi-automated[2]. However, when dealing with a semi-automation, some authors report concerns about the facts, such as the lack of completeness of domain knowledge[3,4] and the inconsistencies and ambiguities in the writing[4].

Two types of facts are related to non-functional requirements (NFR), classification of requirements, and statements of requirements[2]. In particular, authors[5-9] focuses on the classification of non-functional requirements (NFRs) over existing requirements sentences. In turn, to obtain requirement statements, the authors[2] are concerned with the discovery of functional requirements (FRs) and NFRs than may exist on available data.

Both types of requirements, NFRs and FRs, are related; however, exists the lousy practice of focusing on FRs[10] first. That is, while Chung & Leite[11] stress that "real-world problems are more non-functionally oriented than they are functionally oriented, e.g., poor productivity, slow processing, high cost, low quality, and unhappy customer," the practice is as observed by Cysneiros & Yu[10], that NFRs are seen as properties or attributes of a finished software product.

Communication[1] is the second elicitation task performed by the stakeholders who controlled by viewpoints and methods can process the facts to transform them into information. However, one of the problems found in communication[1], which is also a fact as stated by Meth et al.[2], is the vocabulary of the application, i.e., the stakeholders must reach a consensus on the keywords and the vocabulary used.

The third task of elicitation is the validation of facts[1]. This task that is similar to the practice of the courts where testimonies (points of view) are

analyzed to identify whether they complement or conflict[1], it is useful to discover missing information or false assumptions.

1.2. Motivation

Over the past 30 years, various approaches have pointed out the need for NFRs identification early on. Until now, the literature mentions that to deal with NFRs there is a missing methodology[12] and, although the fact that requirements elicitation already has a series of techniques that help those interested in defining desired software[13,14], the methods to address NFRs are more focused on modeling[15]. That is the case of the *NFR framework* conceived in the 1990s and with a focus on systems design based on NFRs. Chung et al.[16,17] created this framework with the premise that the quality of a product largely depends on the quality of the process. However, the framework assumes that stakeholders already know about the NFRs required to design their implementation. In 2000, a compendium of works[1613,17,18] on the framework is published in the book *Non-Functional Requirements in Software Engineering*[15].

It is important to emphasize the notion of *Satisficing* introduced by the framework[15] which is important to understand how to address NFRs. This term was coined by Herbert Simon[1], a combination of two words: "satisfy" and "suffice", thus, when making decisions, a person chooses the best enough option *satisficing* an NFR. Therefore, when dealing with NFRs, we can *satisfice* them only to some degree, unlike the FRs that can be fully achieved.

Regarding the elicitation of NFRs, one technique that can facilitate this is the document reading; however, due to human factors such as tiredness[2,19,20] or market time constraints[2] in elicitation[21-24], the quality of this task may be diminished. Automating document reading comes as a help, but other challenges arise, such as processing large amounts of data[25] *efficiently*, as well as performing *effective* identification of NFRs. For the latter, there is the need to address the representation problem of NFRs[26], which Glinz[26] exemplifies below:

> "Consider the following example: A particular security requirement could be expressed as *The system shall prevent any unauthorized access to the customer data*, which, according to all definitions […] is a non-functional requirement. If we represent this requirement in a more concrete form, for example as *The probability for successful access to the customer data by an unauthorized person shall be smaller than 10-5*, this is still a non-functional requirement. However, if we refine the original requirement to *The database shall grant access to the customer data only to those users that have been authorized by their user name and password*, we have a functional requirement, albeit it is still a security requirement. In a nutshell, the kind of a requirement depends on the way we represent it."

Glinz's example shows that the more concrete/refined an NFR is, the more it tends to be an RF. This is corroborated by the *NFR Framework*[15], where it is

[1] Herbert Simon: was an American economist, Nobel Prize in Economics, ACM Turing Award, pioneer of several modern-day scientific domains such as artificial intelligence, information processing, best known for the theories of "bounded rationality" and "satisficing". Source: Wikipedia.

[2] Time to market (TTM) is the length of time it takes for a product to be designed until be available for sale. Source: Wikipedia.

emphasized that NFRs must be rationalized by decomposing the problem until obtaining operationalizations (FRs) that *suffice* the NFRs.

In 2009, Chung & Leite[11] stated that in the real world the desired functionalities are accompanied by qualities. However, some authors[13,27] point out that qualities may not be simple to articulate during requirements elicitation. As such, a client may not know how to bring out sustainability requirements, but there are other qualities, such as *security*, that is more evident.

On the other hand, literature[10,11,13] stresses the need for early identification of NFRs because the later it is identified, the greater the impact on software production. But, when designing a strategy for NFRs elicitation, e.g. the disciplines/phases vision of Rational Unified Process (RUP)[28] in Fig. 1, the greatest workload gave for the requirements is at the beginning of production. This practice leads to a reduction in the identification of NFRs since they are not simple to articulate in elicitation[13,27].

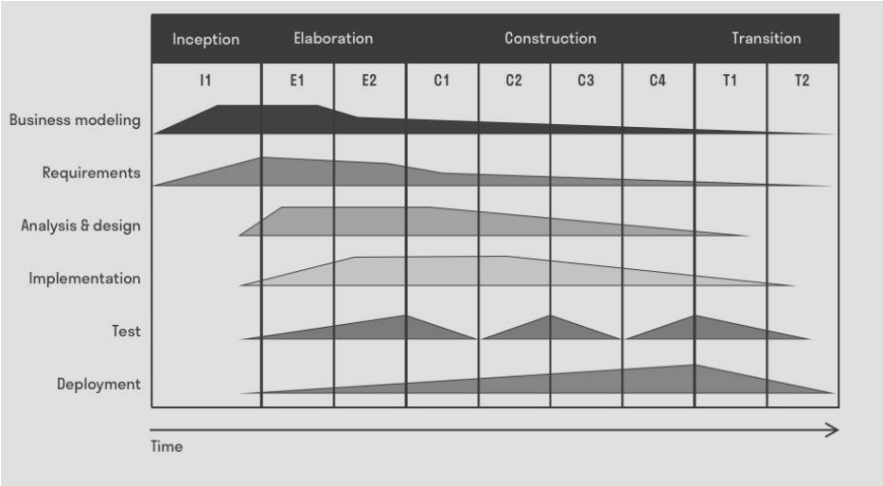


Figure 1. Rational Unified Process (RUP)[28]

Other initiatives such as agile approaches, e.g. Extreme Programming (XP)[29], aims to improve software quality by allowing more customer feedback for better requirements. In this regard, Leite[30] stress some requirements-related facts from XP:

> "The customers can't tell us what they want. When we give them what they say they want, they don`t like it". This is an absolute truth of software development. **The requirements are never clear at first**. Customers can never tell you exactly what they want."
>
> "Every project I've worked on that had fixed price and scope ended with both parties saying, "**The requirements weren't clear.**"
>
> "It seems to be in the nature of the less important requirements that they entail the greatest risk. They are typically the poorest understood, so there is great risk that the **requirements will change all during development.**"

The phrases highlighted corroborates that Requirements Engineering (RE) may not have quality when performed at the early stages of software production. As such, there is a need for domain knowledge that may improve the tasks of

elicitation[1]. And, as stressed by Leite[30], we believe that requirements are constant during software production:

> "As such, the requirements definition is performed continuously by means of stories, tests, new stories tests and so on. By doing this XP is a **learning environment** where the requirements are developed as the project goes along. The discipline of writing stories, writing tests and continuously scoping makes it possible that the on-site customer steer the development and as such achieve the desired result, which can be a **moving target**."

However, although agile approaches aim to give more time for quality requirements[29], few are concerned with the NFRs elicitation[31-34], which leads to several proposals to identify NFRs later in the requirements specifications. Thus, other problems arise, such as that the NFRs are cross-cutting[35,36] and therefore can be distributed in various requirements, just as a requirement may express the demand for various qualities to be met. This problem is evidenced by Cleland-Huang et al.[5] "*The railway gate must close at least 30 seconds before a train enters the crossing* represents both a safety and a performance requirement". To identify the NFRs, most of the approaches use quality attributes existing in fixed taxonomies[37,38], however, this practice may be inefficient earlier in elicitation[39] as facts are raw data in comparison to requirements specifications which uses quality words ending with the suffix "ity" and "ness"[15], then, exist the possibility of leaving many NFRs out as taxonomies do not consider the various ways of expressing qualities in free texts using natural language.

Therefore, the need for early identification of NFRs leads us to propose the speed-up[3] of elicitation tasks (fact-finding, communication, fact-validation) to produce NFRs facts so that stakeholders can create NFR requirements early in software production. Thus, assuming that by semi-automating the speed-up of NFRs elicitation it is possible to be more *effective*, said *softgoal* should be *satisficed* while also *satisficing* its *efficacy*. Then, we find ourselves in the same subjectivity situation of the NFR stated in the NFR framework[15], which is *satisficed* according to the points of view and contexts.

> "Now suppose you want to build the system so that it is fast and accurate. You might be able to build a system, but will it be fast and accurate? Perhaps you consider it to be so, but your manager does not. Or perhaps you consider accuracy to be most important, but your manager really values speed. Now, without even getting your manager involved, would the system be fast on a day when there are a lot of credit card transactions? And would it be accurate even if merchants are being targeted by fraudsters? If not, perhaps you can make it more accurate, by doing more validation. But this extra processing may slow down the system."

Consequently, the research question for this thesis is grounded on the NFRs we are addressing:

How can the semi-automation of NFRs elicitation be speeded up by achieving a trade-off between efficiency and effectiveness?

[3] Speed-up: an increase in speed, especially in a person's or machine's rate of working. Source: Oxford Dictionary.

1.3. Research Method

Our research is qualitative as we deal with NFRs, such as efficacy and efficiency that cannot be fully achieved in a semi-automated approach; however, these can be balanced to some degree.

Fig. 2 uses the tasks for elicitation proposed by Leite[1] and we develop it with definitions proposed in the literature.

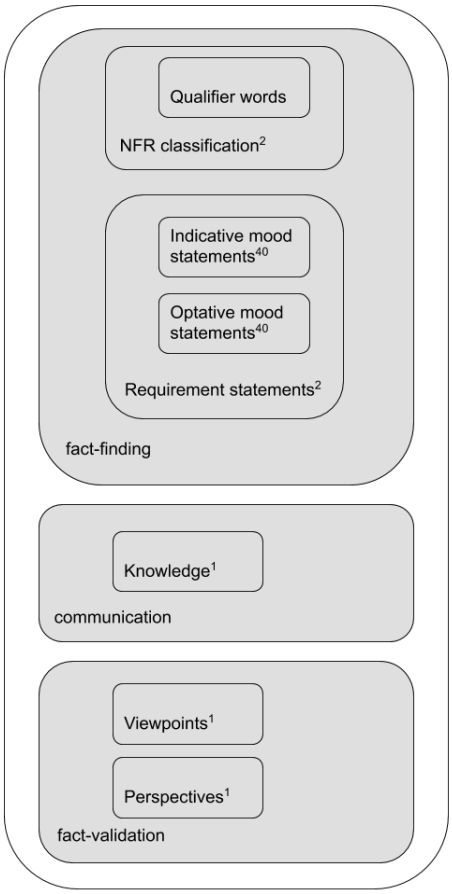


Figure 2. NFRs Elicitation Approach

Firstly, the fact-finding is composed of facts[2] that may help identify NFRs, the requirement statements, and NFRs classification. To the former, a differentiation among statements should be done according to Zave and Jackson's work[40], which state that exists statements in *optative mood* which express the users' desires, and statements in *indicative mood* which are related to domain knowledge. To the latter, we found that NFRs classification should not be addressed using taxonomies[37,38] as it would be necessary as many quality-terms to match NFRs in unknown texts. We propose the use of linguistic syntax to identify qualifiers words, such as adjectives or modifiers of verbs/nouns that, when modified, become adjectives.

Secondly, in our approach communication is related to the acquisition of domain knowledge[1,41] that is key to leverage better communication, e.g. get to know the vocabulary application.

Thirdly, the fact-validation is detailed using the viewpoints and perspectives notion used by Leite[1] which corroborates with the open-source mantra "with enough eyes, all bugs are trivial."

To develop the approach in Fig. 2, this thesis uses a bottom-up strategy that follows the spirit of the NFR framework[15], e.g. "instead of evaluating the final product, the emphasis is on trying to rationalize the development process itself in terms of non-functional requirements." Therefore, the Chapters 2-4 details the development performed to satisfice the trade-off among *efficacy* and *effectiveness*, to speed-up NFRs elicitation.

1.3.1. GitHub Big Data

GitHub Big data has attracted research interest for many purposes in Software Engineering[24,42-45]. A reason for its popularity may be that this is an open-source

platform for software projects that have grown the most in the last decade, that is, it *satisfies* the *actuality* NFR. It also provides an API from where to retrieve data, that is, it *satisfies* the *accessibility* NFR. And, by having structured metadata, it *satisfies* the *composability* NFR as well as other NFRs related to *transparency*[46].

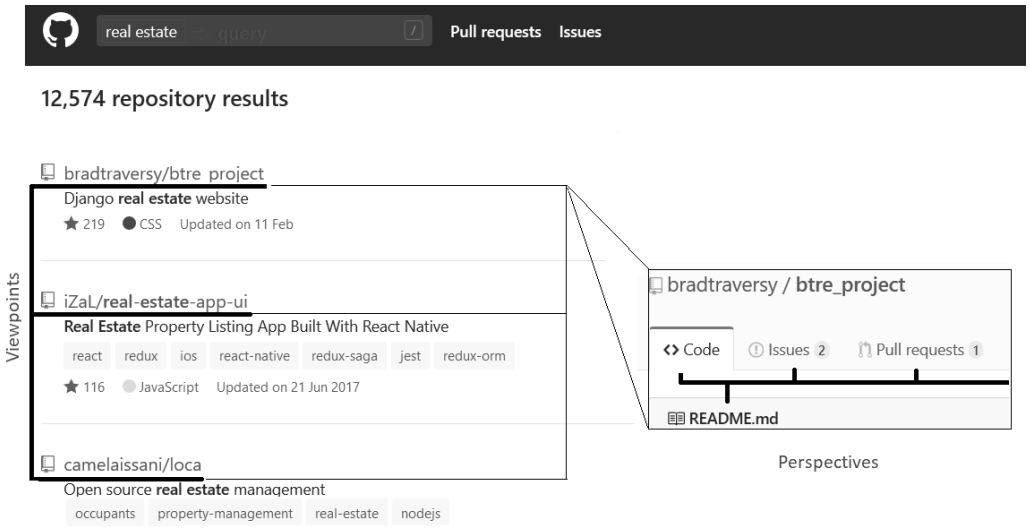


Figure 3. GitHub "Real estate" Viewpoints x Perspectives at 4/8/2020

Fig. 3 shows the GitHub metaphor that allows exploring data by using the notion of viewpoints and perspectives[1]. That is, given a domain problem e.g. "real estate" in Fig. 3., viewpoints (projects/repositories) have different perspectives (ways to solve it) such as issues, commits, or readme artifacts.

By introducing GitHub Big Data to RE, we have the opportunity to take advantage of the *wisdom of crowds*[47] that is a collective intelligence that can help in *speeding up* the rate of work in elicitation. That is, when given a domain problem, we can gain with the elicitation work – time spent – already performed in each of the millions of software projects that exist in GitHub. On the other hand, GitHub as a source of raw data, i.e. facts, provides NFRs early in elicitation which can be organized by ranking or clustering NFRs according the domain target.

1.3.2. Research Goals

Work about NFRs elicitation usually begins by eliciting facts such as requirements statements to classify them later. In this regard, related work can be organized using different facets (Fig. 4) which differ in the type of data used.

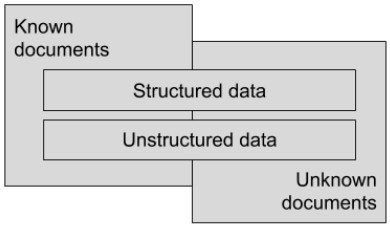


Figure 4. Facets of data used in RE elicitation work

A facet can be sought as a cut containing many parts of something. Thus, data used for fact-finding also. We have that 1) some works are dealing with known documents such as requirements specifications, which also are structured data by using a template, e.g., Volere template[48]. However, 2) other works that use requirements specifications can be written informally, i.e., free texts (unstructured data) in

existing documentation, e.g., laws[49]. And 3) works dealing with unknown documents, such as those in open repositories[24,25],[24] can have a heterogeneous structure of data.

In the previous paragraph, we have that requirement statements can be mistaken with requirements specifications. That is, the former are sentences that have information that may express a user's need[41]. The latter is the actual requirements, which through the tasks of elicitation[1], become specified. Literature that semi-automates the finding of requirement statements also calls them as requirements-related[9,34] information.

In the elicitation process[1]: fact-finding, communication, and fact-validation; we have that the former is related to data retrieval, and the others related to the viewpoints of stakeholders, which are key for requirements completeness[3,4]. Using Big data, the expectation, intentions, or needs of stakeholders, can be pushed early for a better understanding of the UofD (Universe of Discourse) which, according to Leite[50], is the general context in which the software should be developed and operated. UofD includes all sources of information and everyone related to the software. These people are also known as the actors in this universe. The UofD is the reality set forth by the set of objectives defined by those who demand the software.

To semi-automate the fact-finding, a stakeholder may need to express the target goal, which is better if they are knowledgeable about the target domain. And, to validate the facts found, a stakeholder may also need the knowledge to judge its usefulness. As such, communication encapsulates the knowledge needed to give quality to the referred elicitation process.

Having that knowledge influences the elicitation process stated by Leite[1], i.e., to retrieve facts and to judge the them, we set to this work the following sub research questions:

How can elicitors be helped to improve the quality of their inputs on a semi-automation of elicitation?

What strategies of fact-finding can speed up NFRs identification?

How can NFRs be elicited so that they can facilitate validation?

1.4. Contributions

This thesis contributes with a strategy as follows: a) The treatment of querying Big Data as a means to retrieve a quality search space (universe of discourse) for elicitation. b) A strategy to elicit facts by using the notion of viewpoints and perspectives[1]. c) The automation of NFRs identification guided by the NFR framework[15] d) An NFR catalog (SIG), which shows how a trade-off between *efficiency* and *effectiveness* of mechanisms is *satisficed* to speed-up NFRs.

1.5. Background

In this section, the concepts related to Big Data, Requirements Engineering, and Automation techniques used are set. We detail them by using examples from literature.

1.5.1. Big Data

A. Sources of Information (SoI)

In the age of Big Data, sources from where to take information are diverse, from unstructured sources such as most sites of the World Wide Web, or semi-structured sources such as GitHub or Wikipedia, to more structured ones such as Wikidata or WordNet. On the other hand, SoI can also be classified as knowledge bases, e.g., Wikipedia, or as Open Repositories, e.g., GitHub.

Big Data sources are becoming of interest in Software Engineering[43,44] as such, the techniques for automation to deal with massive data needs of diverse resources to get information completeness[3,4]. In RE, sources of information are part of to the UofD.

B. Types of Texts

Texts existing in repositories such as GitHub, are heterogeneous and can be classified in A**) Structured texts.** These are the ones that own a structure such as the name of the project, name of user, title, description, body, or metadata, among others. This structure allows data retrieval through its API. **B) Semi-structured texts.** These are the ones that can be retrieved through some structure; however, the attributes permit the insertion of free texts. e.g., the attributes body and description on GitHub. C) **Unstructured texts.** As there is not a complete unstructured of texts over the Web because of the Html markup, some texts, that are not able to be retrieved through some API, we called as unstructured. To retrieve them, some practices such as Scrawling or Scrapping are used.

To this thesis, we are dealing more with texts of type A or B. However, to test our heuristics, we also experienced C texts[51].

C. Corpus of texts

The set of documents related to a target is called *Corpus.* A corpus is highly associated with the UofD as it encloses a selection of documents with a theme or topic in common. The use of corpus comes from linguistics; as such, exists a guide for the proper building of it. In this regard, Sinclair[52] states this principle when building a Corpus: “The contents of a corpus should be selected regardless of their language, but according to their communicative function in the community in which they occur.” That is, besides domain, the context plays an important role when creating a corpus.

D. Fact-finding

After identifying a UofD, the selection of useful information (facts) becomes necessary. The facts can be described in several ways, such as lists, phrases, requirement-related phrases ("the system must ..."), tables, conceptual graphs, definitions of terms, small paragraphs, explanatory drawing, or any description that a Requirement Engineer uses after or while using elicitation techniques.

In this thesis, we are interested in facts represented as statements in Big Data texts. In this regard, a classification given by Zave & Jackson[40] is relevant.

> "The primary distinction necessary for requirements engineering is captured by two grammatical moods. **Statements in the "indicative" mood** describe the environment as it is in the absence of the machine or regardless of the actions of the machine; these statements are often called "assumptions" or "domain knowledge." **Statements in the "optative" mood** describe the environment as we would like it to be and as we hope it will be when the machine is connected to the environment. Optative statements are commonly called "requirements." The ability to describe the environment in the optative mood makes it unnecessary to describe the machine."

1.5.1. Requirements Engineering

A. Requirements-related texts

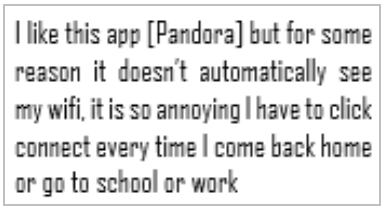


Figure 5. A requirement-related text from application review[9]

These are the information (facts) with existing knowledge (K) that can be taken as a requirement (R). Fig. 5 shows a requirement-related text[9]. K or R is different from a requirement specification (S), which is the artifact produced by stakeholders after the RE tasks are performed using R and K (See Fig. 6).

The works on semi-automation of requirements[9,53] using text-mining[4] use the notion of requirement-related texts given that the focus is on eliciting information, not requirements, nor specifications.

According to Zave & Jackson Formula[40] S, K ⊢ R, specifications (S), and knowledge (K) must be sufficient to guarantee that requirements (R) are met. That is, requirements (R) become specifications (S) when the gap of knowledge (K) is bridged. Therefore, if the text-mining of facts is capable of bringing K, an elicitor is better prepared for elaborating S from R.

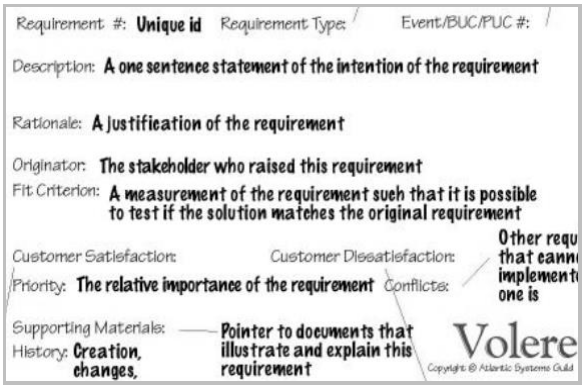


Figure 6. Volere template for requirements specification[48]

[4] "Text-mining aims at disclosing the concealed information by means of methods which on the one hand are able to cope withthe large number of words and structures in natural language and on the other handallow to handle vagueness, uncertainty and fuzziness" Hotho et al[54].

This formula is generic and can lead to misunderstanding the role of elicitation and the requirements engineering (RE) as a whole. Whereas elicitation in the three tasks defined by Leite[1] starts with R to produce validated facts, the objective of RE is to produce S that is obtained after analyzing the facts and the R of the users.

Fig. 5 and 6 clarifies the differences among R and S. This thesis is focused on eliciting relevant K, which are requirements-related statements that besides providing facts, can empower stakeholders to articulate R with NFRs. Then, the production of S is out of the scope of this thesis.

B. NFR framework

In the *NFR Framework*[15] book, a notation for a design reasoning is proposed. That is, with the requirements at hand, stakeholders can model the goals by considering NFRs (qualities) and their interdependencies.

The notation defines NFR goals as Softgoals; that is, it is soft because it is flexible in the sense that it deals with quality achievement. E.g., a goal may be to provide biometric security, which can be achieved/denied; however, one cannot expect to achieve 100% of Softgoal security. Thus, as it is not likely a binary result (achieved/denied) as when dealing with goals, the NFR framework provides elements to deal with qualities, as the proposed notion of ***Satisficing***.

C. SIG (Softgoal Interdependency Graph)

SIGs are proposed to organize the design reasoning of NFRs. To draw a graph, SIG's follow the spirit of AND/OR trees for decomposing a problem; however, the NFR Framework adds more expression by using relationships named contributions; i.e., a Softgoal can *help* or *hurt* to other Softgoal. Nonetheless, as stressed by Chung et al.[15], SIGs should be understood as graphs rather than trees. SIGs are also called Catalogs.

The *NFR framework*[15] proposes an organization of NFRs through the interdependencies of nodes (Softgoals), that is, there is the need to express what Mylopoulos et al. call of controversial relationships[55], as in Fig. 7. A secure account needs of confidentiality which can be restricted by the degree of availability desired.

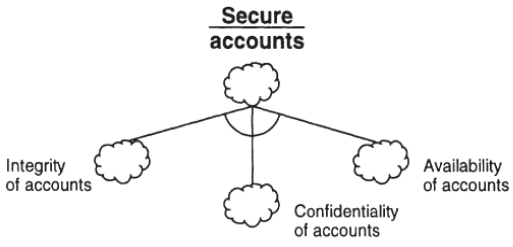


Figure 7. NFRs decomposition in SIG[15]

SIGs are used to represent the knowledge about a quality (parent node), e.g., in Fig. 7, Secure is a parent node that is decomposed in (offspring nodes) using the interdependence relation AND. To deal with problem decomposition, Softgoals are specialized in NFR Softgoal, NFR Operationalizations, and NFR Claims; this thesis uses the first two, which can be seen in Fig. 8.

An NFR Softgoal is represented with a cloud with a thin border, it can be described using two attributes, NFR type, and NFR topic. E.g., Fig. 7 shows the NFR Softgoal with the type (secure) and topic (accounts).

The NFR operationalizations are represented with the clouds with a thick border (Fig. 8). These express the mechanisms that address an NFR Softgoal.

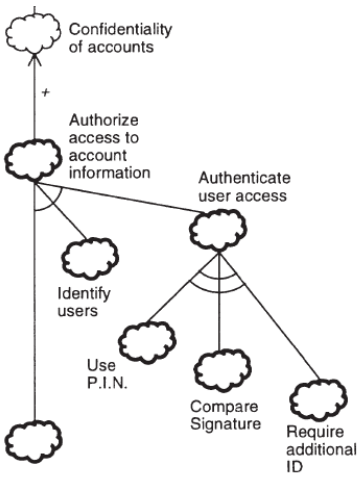


Figure 8. NFRs operationalizations in SIG[15]

D. NFRs classification

One of the most reported problems in requirement engineering is the difficulty of identifying NFRs[56-63]. Generally, stakeholders want these requirements to be present, but such requirements are not always explicit; in other words, they can be part of the so-called tacit knowledge[64].

Mairiza et al.[37] studied the literature on the notion of NFRs and list 114 types of NFRs; among them, 20% (Fig. 9) are qualities that have been defined and have attributes, that is, for security NFR, they found as NFRs attributes integrity, availability, and confidentiality. As such, Fig. 9 does not present a taxonomy of qualities/attributes since attributes can describe various qualities.

accessibility; adaptability; availability; efficiency; fault tolerance; functionality; integratability; integrity; maintainability; modifiability; performance, portability; privacy; readability; reliability; reusability; robustness; safety; scalability; security; testability; understandability; and usability

Figure 9. NFRs with existing definition and attributes[37]

The ISO/IEC 25030 for *quality requirements*[38] organizes NFRs in two groups (Fig. 10), which relates to the ISO 9126 quality model. Besides the classification presented by authors[38], which is said that can vary, they stress the importance of acceptance and a shared terminology about qualities.

Another approach to organizing qualities by using catalogs (SIGs) is in Fig. 11, it shows a transparency catalog[39] with interdependencies, which, by using the HELP contribution, add more semantics to the SIG.

Figure 10. A new standard on software quality requirements[38]

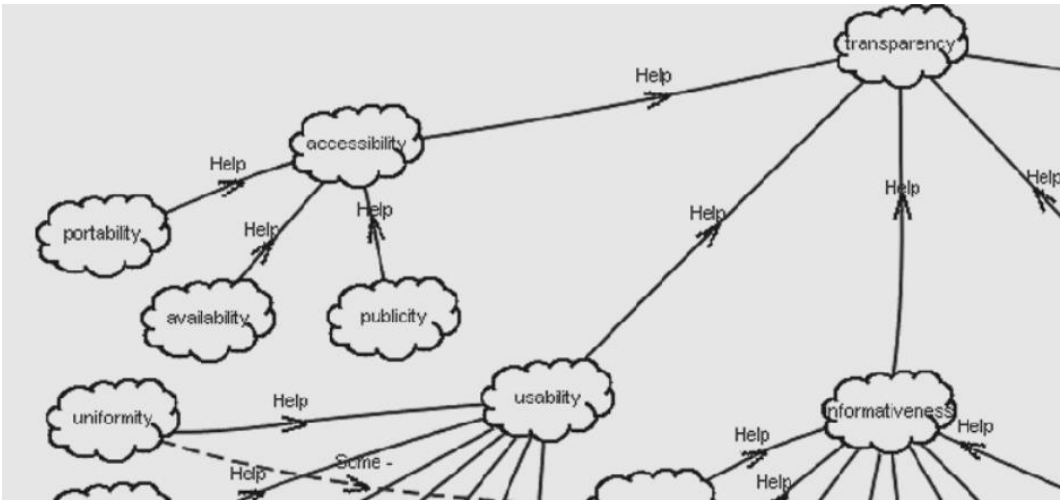


Figure 11. Part of the transparency SIG[39]

The approaches presented to organize NFRs show that there is no consensus in organizing NFRs, nor have a way to cover all possible types of qualifiers allowed by natural language.

1.5.2. Automation Techniques

Existing techniques can be divided into two paths, Information Retrieval Systems (IRS) and Machine Learning Systems (MLS). Although IRS was formerly used in library science[65], the search engines have changed the way Information Retrieval (IR) is perceived, which now is better known as *search*[65]. Therefore, the IRS brings together the tasks of querying, indexing, filtering, and browsing mostly over a collection of text documents.

In the querying is where elicitation takes place by expressing a need for information; however, a necessary clarification is stated by Manning et al.[65].

> "An information need is the topic about which the user desires to know more, and is differentiated from a query, which is what the user conveys to the computer in an attempt to communicate the information need. A document is relevant if it is one that the user perceives as containing information of value with respect to their personal information need."

Nowadays, with the exponential growth of data, MLS brings an opportunity to simulate human reasoning by using statistics, i.e., a machine is capable of identifying patterns or learning from feedback. Unlike IRS that does not infer the terms that may be more appropriate for representing a query.

MLS are divided into supervised techniques, unsupervised, and semi-supervised techniques. The former needs of metadata so the ML models learn to make predictions. The second does not need information for learning; that is, the discovering of patterns or structures are let to techniques such as clustering, trend analysis, correlation, among others. The third combines both methods to balance efficiency and efficacy.

A. Natural Language Processing (NLP)

NLP is a technique that aims to automate what a human reader does. Tasks like summarizing, highlighting, among others[66] are the interest of requirements engineers. NLP is used for both, MLS and IRS.

To process texts, the usual NLP pre-processing steps[67] used in this thesis are a) tokenizing: the work of separating words when a space between them appears; b) tagging: the work of assigning a grammatical tag, e.g. verb; and c) name entity recognition (NER): which aims to recognize entities, people, location, or codes among others.

B. Part of Speech Tagging (POS-tagging)

POS-tagging[67] is one of the techniques of NLP which identify the grammatical function of a word in a given language. To this thesis, most of our operationalizations relied on this technique.

An example in Fig. 12 uses the requirement-related text in Fig. 5 where words tagged as nouns are: app, Pandora, reason, wifi, time, home, school, work; these nouns may indicate the entities involved in the desired software. Also a differentiation with proper nouns may help to identify entities such as Pandora.

```
> req.rel.text<-"I like this app Pandora but for some reason it doesn't automatically see my$
> nouns<-lapply(document, extractPOS, toMatch)
> nouns
[[1]]
[1] "app/NN Pandora/NNP reason/NN wifi/NN time/NN home/NN school/NN work/NN"

> toMatch <- c("NP")
> proper.nouns<-lapply(document, extractPOS, toMatch)
> proper.nouns
[[1]]
[1] "Pandora/NNP"
```

Figure 12. POS-tagging using the openNLP library[5]

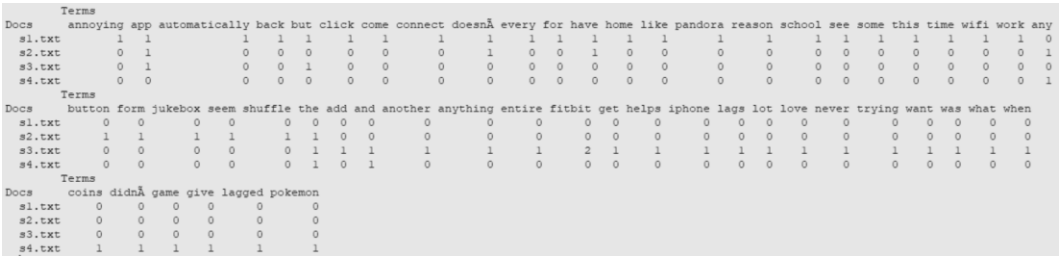
C. Bag of Words (BoW)

A set of words tokenized is called of bag of words. When applying automation techniques for NLP, words are usually organized in a matrix using the document/term or term/document format. The number of rows and columns in a BoW indicates the dimension of a matrix.

A BoW is used for IRS and MLS for the numeric representation of words in a text. Fig. 13 represents the matrix for two related-requirement texts from work[9] where the numbers are the frequency of the terms (TF) in the BoW.

> "I like this app Pandora but for some reason it doesn't automatically see my wifi, it is so annoying I have to click connect every time I come back home or go to school or work"
>
> "The app Jukebox doesn't seem to have any form of shuffle button"

[5] https://opennlp.apache.org/



Terms																										
Docs	annoying	app	automatically	back	but	click	come	connect	doesnÃ	every	for	have	home	like	pandora	reason	school	see	some	this	time	wifi	work	any
s1.txt	1	1		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
s2.txt	0	1		0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	1
s3.txt	0	1		0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
s4.txt	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Terms																										
Docs	button	form	jukebox	seem	shuffle	the	add	and	another	anything	entire	fitbit	get	helps	iphone	lags	lot	love	never	trying	want	was	what	when
s1.txt	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
s2.txt	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
s3.txt	0	0	0	0	0	1	1	1	1	1	1	2	1	1	1	1	1	1	1	1	1	1	1	1
s4.txt	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Terms						
Docs	coins	didnÃ	game	give	lagged	pokemon
s1.txt	0	0	0	0	0	0
s2.txt	0	0	0	0	0	0
s3.txt	0	0	0	0	0	0
s4.txt	1	1	1	1	1	1

Figure 13. A BoW for requirement-related texts in app reviews[9]

D. Frequent Terms

Automation techniques from IRS and MLS rely on statistics, as such, representations such as BoW record the count of words using diverse schemes. This thesis uses the Term Frequency–Inverse Document Frequency (TF-IDF) scheme[68] which calculates how important is a word in a corpus. Fig. 14 shows the counts using a TF scheme where *app* is the frequent word. In contrast, Fig. 15, shows that *app* is not than relevant in the corpus, whereas *fitbit* appears to be important as it is mentioned in two documents.

app	the	but	doesnÃ	have	any	and	fitbit	png	annoying
3	3	2	2	2	2	2	2	1	1
automatically	back	click	come	connect	every	for	home	like	pandora
1	1	1	1	1	1	1	1	1	1
reason	school	see	some	this	time	wifi	work	button	form
1	1	1	1	1	1	1	1	1	1
jukebox	seem	shuffle	add	another	anything	entire	get	helps	iphone
1	1	1	1	1	1	1	1	1	1
lags	lot	love	never	trying	want	was	what	when	coins
1	1	1	1	1	1	1	1	1	1
didnÃ	game	give	lagged	pokemon					
1	1	1	1	1					

Figure 14. Ranking of words using TF scheme on texts from app reviews

fitbit	annoying	automatically	back	but	click	come	connect	doesnÃ
4.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000
for	have	home	like	pandora	reason	school	see	some
2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000
time	wifi	work	any	button	form	jukebox	seem	shuffle
2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000
and	another	anything	entire	get	helps	iphone	lags	lot
2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000
never	trying	want	was	what	when	coins	didnÃ	game
2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000
lagged	pokemon	app	the					
2.000000	2.000000	1.245112	1.245112					

Figure 15. Ranking of words using TFIDF scheme on texts from app reviews

1.6. Related work

Few works are similar to our approach to speed up NFRs elicitation which considers two NFRs towards its automation: efficiency and efficacy. The closer works are the ones focused on classifying requirement statements by NFR; their semi-automation uses IR[56,58] techniques and also ML techniques[56-63]. However, most of them classify NFRs by considering a taxonomy of quality attributes[37,38] as well as using a corpus of known requirements specifications.

Our approach differs in that we are dealing with unknown data from Big Data, where statements are semi-structured texts that, after some pruning, they can become in a set of requirements-related texts. In this regard, Table 1 shows the works that are comparable in different aspects of our approach. The selection of works followed this criteria exclusion: a) we avoid works using requirements specifications; b) works using interviews as NFRs are difficult to articulate early

in elicitation; c) works that elicit NFRs guided by ontology, as this practice can direct elicitation towards the entities of such ontologies, which leads to omitting NFRs; d) works that elicit NFRs guided by quality taxonomies, e.g. ISO 25010, ISO 9126, as also leads to omitting NFRs.

Table 1. Related Work according to our approach

Works	The work Speed-up elicitation? (+) increase rate/quality of work (-) diminish rate/quality of work	Query SoI?	NFR Elicitation?	NFR Classification?	Semi-automated Classification?	NFR Interdependencies?
[69]	+Proposes use of Knowledge Bases - No viewpoints	No	Yes	No	No	No
[70]	+extract keywords from SIGs	No	No	Yes	No	No
[71]	+ deal with the ambiguity problem - No viewpoints - No use of knowledge bases	No	Yes	No	No	No
[8]	+use App reviews (unstructured texts)	No	Yes	No	Yes	No
[72][73][74]	+identify antonyms/synonyms in SIGS	No	No	No	No	Yes

As our approach avoids fixed-quality attributes in taxonomies, and we propose the use of linguistic syntax to identify qualifiers words in GitHub texts, we did not find a benchmark to verify our heuristics. However, we verified this by using an existing gold standard of NFR statements[75], obtaining positive results[76].

An important question arose about gold standards for NFRs, about its subjectivity when a human classifies it. As such, we developed our gold standard based on the existing one[75], and we found that the classification of NFRs vary[76].

Our work is novel in exposing NFRs early in elicitation. Given that this is a task with time constraints, there is a need for *efficiency* which is provided using existing knowledge on GitHub. On the other hand, *efficacy* is related to how accurate the automation is in identifying NFRs; here, similar approaches[5,6] work in a search space (corpus) where the NFRs are known; as such, it is possible to verify how accurate automation is. In our case, our approach deal with unknown search spaces[77]; therefore, the recall of our heuristics is not guaranteed, however, the precision can *suffice* as compared to the manual work that would be done[77].

By using the NFR framework[15] as a reference, one is knowledgeable about interdependencies among NFRs. As such, the design of our automation considers that NFRs need to be identified in a context by showing the NFRs that are correlated, to later recognize their contributions (Fig. 8, Fig. 11). To the best of our knowledge, there is no work performing this type of text-mining for NFRs elicitation.

Structure of the thesis

The remainder of this thesis is structured as follows. Chapter 2 explores the importance of corpus creation (search space) as the first step towards automated elicitation. Chapter 3 details the facts elicitation approach to get NFRs-related texts. Chapter 4 focuses on strategies used to identify interdependencies in NFRs. Chapter 5 present a SIG for NFRs elicitation, which organizes the knowledge of mechanisms for automated NFRs elicitation. Finally, Chapter 6 concludes the thesis and presents future work.

2 Corpus Creation

In this chapter, we present a strategy to explore search spaces in the GitHub Big Data. It begins with the creation of corpora using GitHub data, which is explained by using the viewpoints and perspectives notion. Later, two approaches, one semantic and another syntactic, explains the importance of a quality query to retrieve a relevant corpus. The assessments of strategies are presented while explaining our findings.

2.1. Corpus Creation from GitHub

Our work to create corpora for text-mining of RE assets, relies on our previous work[78] that needed efficiency in creating a corpus for any query on the GitHub environment. As such, a tool[79] for corpus retrieval was developed.

The tool was designed to explore the notion of viewpoints and perspectives[1] on the GitHub environment, thus, the retrieval of corpus can be implemented to retrieve artifacts (perspectives) such as Issues, Readme, or Commits, among others. The Readme artifact serves as a summary of projects indicating the goals as well as instructions for use. The Issues artifact is used as a support ticket system; as such, it allows the labels (bug, enhancement, help-wanted, among others). The Commits artifact enables developing tracking through comments on project code or documentation increments.

So far, the tool[6] has been improved to support two artifacts, Readmes and Issues. A test version[7] explores the description attribute of projects as a way to reduce the processing of Readmes.

To maintain the recall of results in a GitHub search, the tool keeps the order of results[79] (GitHub recall). Despite not existing official information on how this order is computed, there are attributes such as *stars* given by users or *forks* (reusing a project) that may be used for such a measure.

2.1.1. A GitHub Recommendation System

Strategies as the tool[79] help in speed up the retrieval of a corpus; however, we found constraints, such as limiting the number of projects to be retrieved, e.g. GitHub only give access to 1000 projects when the results can usually be more than 2000. This restriction creates uncertainty about the existence of relevant information that could be in the projects that are not accessible, as well as in the last locations of the GitHub results ordering.

In this regard, our subsequent work[80] proposes a Recommendation System (RecSys) approach to filter relevant projects from GitHub by using the Readme perspective. The criteria to select this artifact is because Readme is the front end to communicate the goals and features that a project implements.

[6] Stable version: http://corpus-retrieval.herokuapp.com/
[7] Test version http://corpus-retrieval-2.herokuapp.com/

A common approach in RecSys is the content-based filtering, which is used to extract features in items such as movies, and the use of preferences existing in user profiles. Nonetheless, for our research, two issues arose. The first is that the readme is not categorized, like a book or a movie, since it contains unstructured texts with an undefined purpose, i.e., a readme may provide information about features as well as installation instructions. The second is that users are rating GitHub projects and not their perspectives separately.

RecSys in Requirements Engineering (RSREs) has been a growing interest in the field of Requirements Engineering (RE) as surveyed in work[81]. One approach[82] stresses the necessity to process a lot of information by stakeholders, which includes what are the users' needs, why they are needed, what are competitors offering, and what are technological advances and the feasible features. As such, it is important to note that much interesting information about users' needs can be found in documents from similar projects.

To the best of our knowledge, there are a few studies that focus on requirements elicitation using recommendation systems[83-88]. The difference with our approach is that they know in advance the documents they text-mined, as well as the domain. This is not the case of our work[78] where we evidenced the ambiguity nature in natural language, and unstructured texts, e.g. *real estate* was identified by two domains, one domain is related to *house and lands business*, and the other as an analogy in HCI Usability lingo ("the amount of space available on display for an application to provide output"). A similar approach to our is proposed by Guendouz et al.[89]; this work predicts useful repositories according to developer needs. Such predictions are made by exploring the fork perspective based on users' activity history.

Figure 16. Syntactic filtering of frequent-nouns from Real Estate domain[80]

2.1.2. GH4RE Strategy

GitHub for RE (GH4RE)[80] uses content-based filtering, for which it is needed, project features, and user preferences. To supply the lack of user preferences, we propose to: (1) obtain project features by revealing frequent terms that may cause an interaction with the user; and (2) obtain the user preferences through the visualization of relevant terms and, based on user interest in one of them, we can cluster and rank the term- related projects for a recommendation.

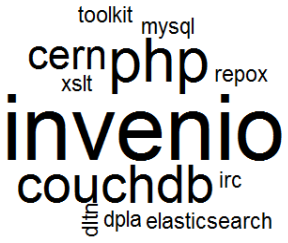


Figure 17. DL Technologies from Conventional GitHub Results

Fig. 16 shows the relevant words in a *real estate* corpus. We filtered proper-nouns to discard nouns that can be unrelated to the target domain. However, still, some of them are not related to the domain. Using another corpus as the case of the Digital Library (DL) domain, we also filtered the proper-nouns. Fig 17 shows the relevant technologies for DL. Later, by ordering the projects

where the phrase “digital library” is frequent, we extracted other latent words for DL (Fig. 18).

We also envisioned in a situation when a stakeholder is unknowledgeable about a domain. That is, we need to have more semantic in visualization, such as the proximity relation of the words. As such, with a correlation of words measure[65], we got other DL words that are relevant (Fig. 19). However, the correlation value of words is not significant to have a differentiated size.

Figure 18. DL technologies in Relevant Projects of GitHub

Figure 19. Technologies related to Bibtex

Figs.17-19 shows diverse ways to inform about the features that a project has. Then, according to user selections, the task of clustering and ranking of projects of GH4RE[80] allows a new organization of projects, which may be a recommendation.

2.1.3. GH4RE Assessment

Six people with experience in Software Engineering were selected assess the quality of 20 readme texts from a corpus of the Real Estate domain. Among them, 10 readmes are from our recommendation approach. We randomly combine both groups of readmes, and then we asked six users to assess the usefulness of the information in them. We measured using the Likert scale technique[90]. The following text was presented to users:

> *Imagine a scenario where a client desires an application, for instance, an application for the Real Estate domain. One of the tasks you may need to perform as a (requirements engineer, developer, project-manager or designer) is the learning about the Real Estate domain. By observing the Wordcloud (Fig. 16), you may perceive that Zillow is an important word in this domain. The excel file[91] presents 20 links to Readme texts containing in-formation about Zillow. We want to measure the usefulness of information in texts for the concept Zillow and for the scenario described. Each readme text is named following the pattern below to keep the traceability to its sources:*
>
> *Number of original relevance in GitHub.-.userName.-.projectName*

Our results in Table 2 show that projects in GH4RE recommendation have no relevance in GitHub results. That is, the better position is to the project 0774.

From the GH4RE recommendation, the first two readme texts were rated as extremely useful by five people (83% of users). By contrast, the first two readme texts of GitHub’s recommendation were evaluated with the lowest scores (useless or not very useful). We note that the assessment of 80% of readmes from GH4RE recommendation range from somewhat useful to extremely useful, which we consider as a positive result, given the various profile of participants.

Related to GitHub’s recommendations, 90% of the readme texts were qualified with the lowest scores. This situation supports our belief that GitHub is envisioned for development purposes, and as such, it is needed strategies to organize a corpus retrieved from GitHub according to the RE needs.

Table 2. Top10 Projects Recommended by GitHub vs. Top10 projects of GH4RE Recommendation

GitHub Recommendation	GH4RE Recommendation	Frequency of *Zillow* in GH4RE
0001.-.jdemaris.-.real	1380.-.CurleySamuel.-.Thesis	26
0002.-.litianbo.-.AndroidZillowFetch	1357.-.MichaelAHood.-.real_estate_recommender	21
0003.-.annaplusdavid.-.real-estate-comps	0774.-.hanneshapke.-.pyzillow	16
0004.-.hi08060204.-.Real-Estate-Search	1541.-.shawncxc.-.zillow-analysis	8
0005.-.matlai17.-.Zillow-Classification-599	1336.-.verdi327.-.zillow_api	6
0006.-.eternalmothra.-.real_estate_values	1586.-.aminge37.-.prime-group-project	6
0007.-.samidakhani.-.zillow_web_search	2094.-.imFORZA.-.re-pro	6
0008.-.Brian-Koscielniak.-.realtorApp	1073.-.fascinatingfingers.-.ZillowR	5
0009.-.wilk916.-.ZPropertyEvaluator	1349.-.Tim-K-DFW.-.zillow_scraper	5
0010.-.jamesxuhaozhe.-.Real-Estate-Information-Search-Engine-using-Zillow-API-web-based	1534.-.cran.-.ZillowR	5

2.2. Query Definition

Literature in IR[92] research on mechanisms that brings documents that are relevant for a need, such as partial or exact match techniques. In requirements elicitation, a need can be expressed in concepts or sentences which a stakeholder can be able to articulate if he/she is knowledgeable about the domain. However, in a reduced situation, such as a query summarizing a need, the influence of stakeholders in the choice of keywords will influence the obtaining of quality information. As such, stakeholders influence the quality of the query negatively when they are unknowledgeable about a domain.

We departed by exploring a syntactic approach using an exact matching technique to find requirements-related to transparency NFR (2.2.1). Later, to discover relevant keywords in unknown corpora, we develop a semantic strategy to assist stakeholders in creating queries (2.2.3).

2.2.1. Querying transparency related-information

Transparency is an NFR that have been elicited and modeled by the RE group at PUC-Rio by using the NFR framework. Fig. 20 shows the SIG produced[93]. As other approaches that use quality-indicator words[5] for text-mining sentences, the NFR Softgoal of SIGs can be used for querying related-information.

As such, we[51] performed the following task using three elements, a target problem (e.g., Bills related to transparency NFR), a corpus of texts (e.g., Votenaweb Bills[8]), and a SIG of transparency[93]. With them, we manually created the keywords for each NFR Softgoal on SIG (Fig. 20), and then we form queries to search Bills related to Transparency (Table 3, second column).

Our findings for the task are the following: 1) we perceived on stakeholders the lack to articulate keywords, which were limited to few central words such as *transparência*, *informação*, *dados*, as well as their previous knowledge about the transparency SIG. 2) After text-mining, and by reading of findings, we perceived other keywords in the corpus that could have been useful

[8] http://votenaweb.com.br/

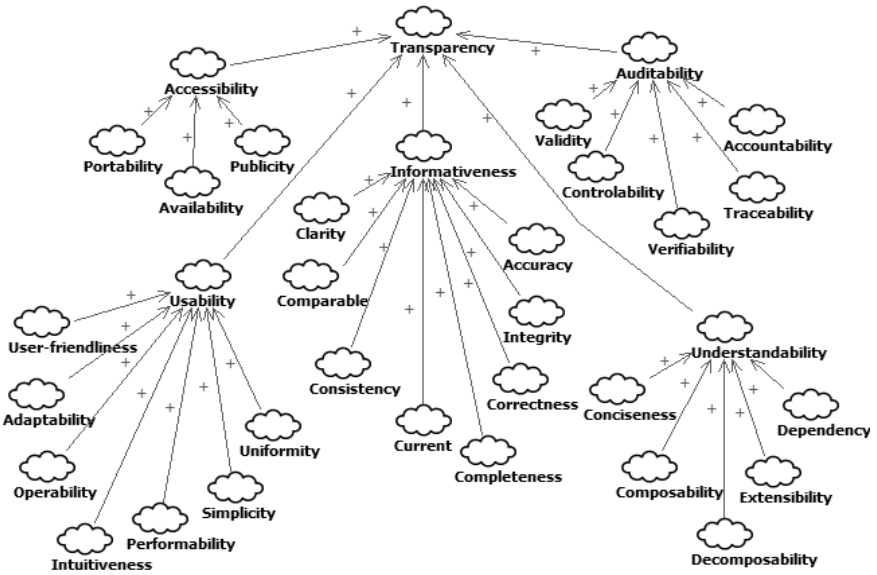


Figure 20. Transparency Softgoal Interdependency Graph (SIG)[51]

to retrieve more related sentences (third column in Table 3). 3) The use of the exact-matching technique limits the match of similar words in its root form, e.g., *divulgarem, divulgar, informem, informando*. 4) SIGs can be leveraged to perform the IR task of ranking, i.e., the SIG interdependencies in Fig. 20, availability *helping* accessibility *helping* transparency, can be used to compute that a sentence is more transparent (parent node) if a sentence has matches with offspring nodes[51].

Table 3. Stakeholders Keywords using NFR transparency Catalog

NFR Softgoal	Queries based on NFR Softgoal	Keywords found after text-mining the Corpus of Votenaweb Bills
transparência	transparência da informação, transparência das informações, transparência dos dados, transparência de software, transparência do processo, transparência + informação, transparencia + informações, transparência + dados, transparência + software, transparência + processo	
acessibilidade	acesso à informação, acesso + informação, acesso + dados	visiveis, divulgado, divulgação, informem, informações, divulgar, internet, informando, divulgarem, facil, visualização, mostrar, informar, preciso, claro, divulgarem, local visivel, acompanhar, sigilo, divulgar, publicação
portabilidade	diferentes plataformas, diferentes + plataformas, portabilidade da informação, portabilidade + informação	
disponibilidade	disponibilizar informação, disponibilizar + informação, disponibilizar + dados	
publicidade	divulgação da informação, divulgação + informação, divulgação dos dados, divulgação + dados	

entendimento	fácil entendimento, fácil + entendimento	ensino, estudo
dependência	informação relacionada, informação + relacionada	
compositividade	fácil organização, fácil + organização	
detalhamento	dado detalhado, dados + detalhado, dados + detalhamento	
divisibilidade	dado dividido, dados + dividido	
concisão	dado resumido, dado + resumido, dado conciso, dado + conciso, dado sucinto, dado + sucinto	
informativo	informação ao cidadão, informação + cidadão	ensino, estudo, divulgado, divulgação, informem, informações, conhecimento, divulgar, internet, informando, divulgarem, facil, visualização, mostrar, informar, preciso, claro, divulgarem, local visivel, prestar contas, acompanhar, enviar
clareza	informação clara, informação + clara	
consistência	dados consistentes, dados + consistente	
integridade	informação íntegra, informação + integridade, dados + integridade	
corretude	sem erro, erro + informação	
acurácia	informação precisa, informação + precisa, informação + precisão	
atualidade	informação desatualizada, informaçãoes desatualizadas, conteúdo desatualizado, informação + desatualizada, informaçãoes + desatualizadas, conteúdo + desatualizado	
completeza	toda informação, toda + informação	
usabilidade	facilidade de uso, facilidade + uso, fácil + usar	Visiveis, internet, facil, visualização, mostrar, local visivel
uniformidade	forma única, forma + única	
amigabilidade	leitura fácil, leitura + fácil, menor esforço, menor + esforço	
simplicidade	usos simples, usos + simples, uso +simples	
operabilidade	fácil de operar, fácil + operar, pronto para uso, pronto + uso	
intuitividade	uso intuitivo, uso evidente, uso + intuitivo, uso + evidente	
adaptabilidade	capacidade de ser alterado, capacidade + alterado, capacidade de ser adaptável, capacidade + adaptável, capacidade de mudar, capacidade + mudar	
desempenho	agilidade de uso, agilidade + uso	
auditabilidade	prestação de contas, prestação + contas, auditoria da informação, auditoria + informação, fiscalização dos dados, fiscalização + dados	divulgarem, internet, prestar contas, acompanhar, enviar
validade	qualidade da informação, qualidade + informação	
controlabilidade	controle dos dados, controle + dados	
verificabilidade	verificação dos dados, verificação + dados, verificação da informação, verificação + informação	
rastreabilidade	rastreabilidade + dados, rastreabilidade + informação, rastro + dados, rastro + informação	
responsabilidade	responsabilidade + informação, responsabilidade + dados	

2.2.2. Assessing the Retrieval of Transparency Bills

The legislative process begins with the proposal of Bills by members of Congress, which, after discussions in the legislative chamber, can become laws. In our

work[51], we use data from the Votenaweb portal, as it provides a summary of Brazilian Bill's. The portal allows citizens to evaluate whether they agree or not with a Bill, and they can comment on these.

To the best of our knowledge, few works[5,70,94] use the qualities of NFR catalogs as a drive for text mining. Other works[95,96] also rely on keywords to identifying/reuse components of software libraries[95], as well as to treat privacy and security requirements[96]. Previous work that explores Votenaweb data can be summarized as follows: a) Engiel et al.[97] performed a manual identification of Bills related to transparency in the year 2013. b) Engiel et al.[98] presents an initial approach for text-mining transparency Bills. c) Portugal et al[99] is an initial approach of text-mining citizen comments about the transparency Bills.

The assessment of heuristics created for text-mining based on SIGs related-keywords, and ranking findings using SIGs structure, confirms a previous manual search[97]. From 27 Bills identified in 2013, we found 13. Despite that our process got 48% percent of Bills in 2013, it is worth noting that the process mined another 15 that were not identified by previous work. Finally, we checked the Bills that our approach did not find, and we annotated the keywords that could have brought them (Third column in Table 3).

This approach has shown us the need for mechanisms to identify corpus-related keywords.

2.2.3. A knowledge-based approach to assist Querying

Despite the limitations of our first approach, which used exact matching, we perceived that it was also limited due to two factors: 1) we were unknowledgeable about the corpus we text-mined, and 2) queries may not have been created effectively. To tackle both factors, we developed a strategy that relies on Wikipedia to obtain concepts given a domain, and by using a correlation of words measure, we can use those concepts to find others related that will become the keywords of a corpus.

Fig. 21 uses the NFR framework[15] to model our problem. That is, given that our main target is to perform an automated elicitation, we designed that it can be performed by 1) satisficing on time elicitation (efficiently), and 2) by obtaining relevant projects from software repositories (efficacy). GitHub is proposed as an NFR operationalization (a mechanism) to address two softgoals, the timely elicitation and the capacity of being knowledgeable about a domain. Then, if satisfied that an elicitor is knowledgeable then he/she can help in retrieving relevant projects. However, a conflict occurs since the timely NFR impacts negatively on being

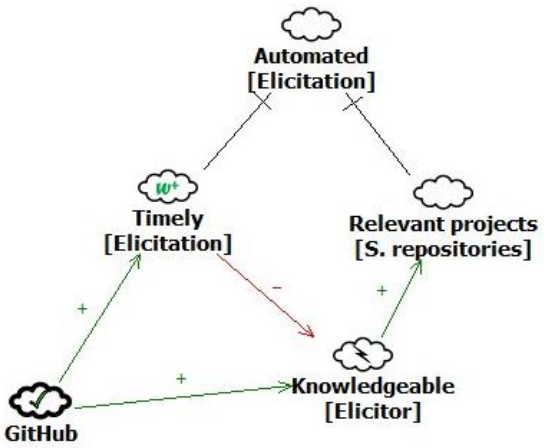


Figure 21. The Problem modeled as an evaluated SIG

knowledgeable. The NFR label propagation procedure[15] detected the conflict. The result obtained is the NFR Softgoals, such as Relevant and Automated without a label (Fig. 22), which means that the GitHub operationalization does not satisfice relevant projects, and consequently, the automation of elicitation being not *satisficed*.

Figure 22. Offspring Softgoals labels

Fig. 23 shows our proposal to improve queries based on three types of words when composing a query: 1) **a keyword** is a relevant word from a corpus but without the certainty that it belongs to the target domain, 2) **a concept** is a relevant word that belongs to the target domain, and 3) **a seed-concept** is the target-related problem, e.g., *digital library.* The query is used to retrieve a corpus from the source selected. In our case, we use GitHub.

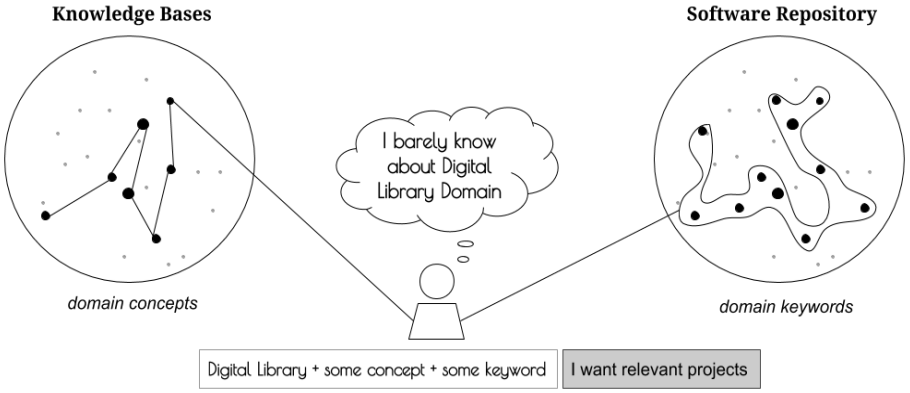


Figure 23. Semi-Automated Strategy for Finding Relevant Projects in Software Repositories

Assuming that a knowledgeable elicitor is capable of performing a query, there are still limitations when performing this task. That is, as, in any search engine, the more words are used, the fewer results will be produced. For such, an elicitor must have assistance in searching, so that, when the search results are inefficient, the elicitor is informed.

Three tasks are key for automation: 1) finding domain concepts, 2) finding keywords in a preliminary corpus, and 3) assistance to combine concepts to produce a better query.

The task of finding concepts has always been a need in RE[,3,19,100-103]. Nowadays, with the progress in Artificial Intelligence (AI) techniques, there exists the opportunity to apply automated mechanisms in texts of Big Data. There are some approaches[104,105] that automate access to knowledge and assist in finding relevant assets, such as key concepts[104] or features[105]; however, its searching is over a corpus of documents whose nature is known, e.g., Software Requirement Specifications (SRS's). Hence, applying these approaches in a corpus with

descriptions of different GitHub projects, with heterogeneous data, may bring irrelevant data, as seen in the work of Morales-Ramirez et al.[106].

The strategy we propose for better querying GitHub is modeled by using SADT[107] (Fig. 25). It begins when the elicitor inputs a *seed concept* (Fig. 25 A-0), which triggers the finding of *concepts* and *keywords*, and ends with a corpus of *relevant projects*. Following, we detail the process accompanied by a virtual environment where an elicitor is assisted.

A. RETRIEVE Concepts from Knowledge Bases

Our strategy relies on knowledge bases, such as Wikidata[108], to tackle this task (SADT-A0). Wikidata has structured information from Wikipedia, where articles are represented as a set of entities with properties.

The task begins by introducing a *seed concept* queried on Wikidata for disambiguation (Fig. 24). To exemplify, we use the target problem *digital library* related to libraries such as the ACM library or the IEEEXplore library.

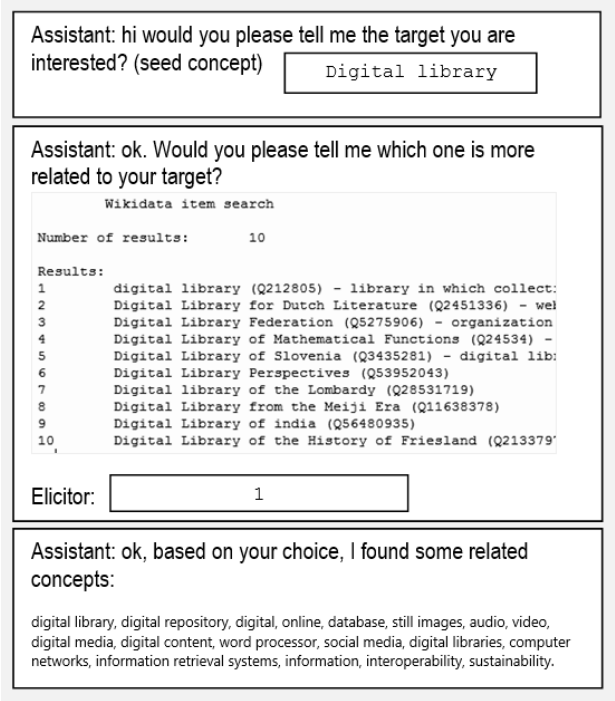


Figure 24. Disambiguation of target problem

Disambiguation step is needed as related work reported on the polysemy of words[24,102,109,110]. As such, our previous work[24] faced the case of a query "real estate" for *land and buildings* domain, but we found that it was also related to the real state property of *screens on electronic devices*. As seen in Fig. 26, Wikidata facilitates disambiguation, since any variation of a concept is a unique entity. Towards its automation, a wrapper WikidataR for R language[111] provides this function.

The second function after disambiguation is the finding of relevant concepts; for that matter, we use the Wikifier[112,113] service to process a text, e.g., an article from Wikipedia. Using the properties of an entity in Wikidata, e.g., entity Q212805 (Fig. 24), we have access to items in different languages linked to an entity. As such, we selected the main English article[9]. Wikifier[112,113] results are the concepts in Fig. 26, which are in the context of a given text, e.g., the first paragraph of an article.

[9] https://en.wikipedia.org/wiki/Digital_library

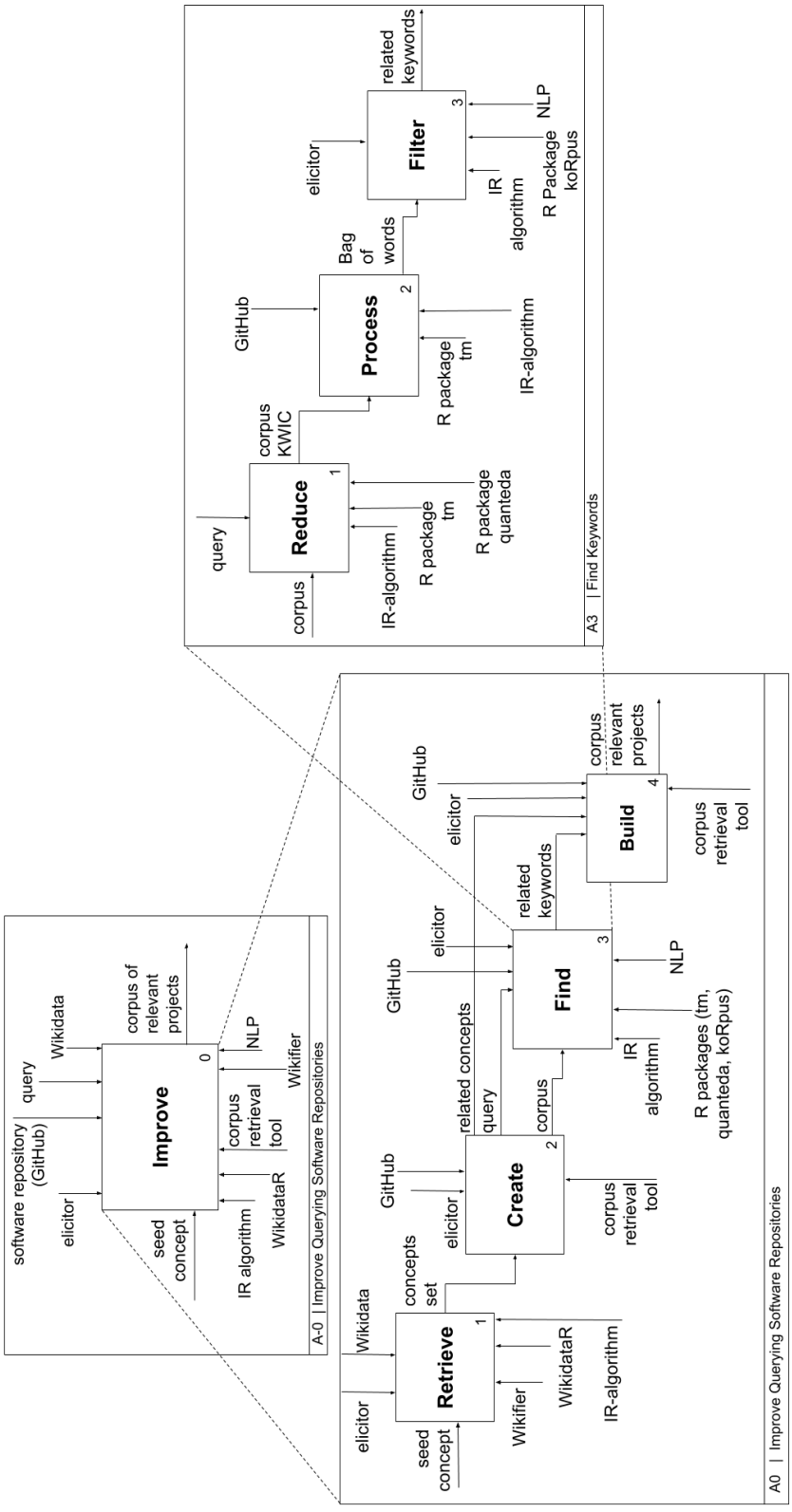


Figure 25. SADT model for knowledge-based querying

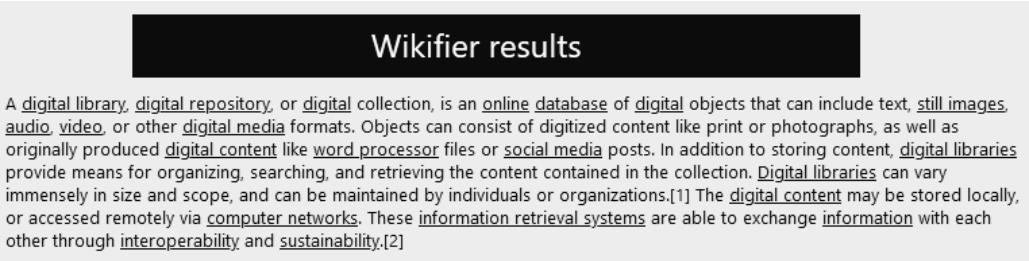


Figure 26. Wikifier annotation using a digital library article from Wikipedia

B. CREATE Corpus of Readme-Projects based on Concepts

In this stage of interaction, an assistant (Fig. 27) may help to create a GitHub corpus using the concepts retrieved.

The activity is as follows: i) it is asked to select 5 related concepts ii) the elicitor is asked to create a query with the *seed-concept* and *concepts* selected from Fig. 26, iii) it is reminded about the length of the query strings, and advise if the number of projects returned is below a certain threshold (Fig. 27).

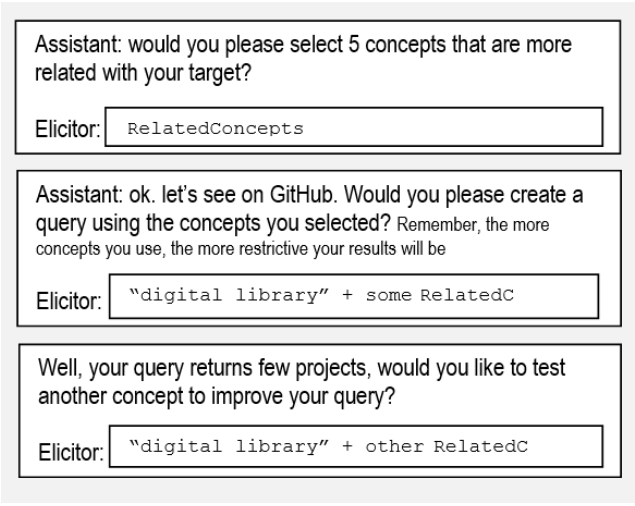


Figure 27. Interaction for Corpus Creation

C. FIND Keywords related to a domain

The FIND activity is decomposed into three activities to process a corpus from the CREATE.

The first is to REDUCE the time in processing a large amount of data. We reduce by transforming the corpus in a corpus of short descriptions (160 words). We use the Keywords in Context (KWIC) technique for this purpose. Similar work[102] and our work[114] use this mechanism to reduce the scope of searching.

Fig. 28 depicts this reduction, which consists in first, filtering the texts that match the *seed concept*, to then locate the ones that also match with *concepts* in the query. The implementation of this activity makes use of the R packages, tm[115] and quanteda[116].

The second activity is to PROCESS the corpus of KWICs by using a bag of words representation. On this subject, there is a discussion about pre-processing when dealing with noisy texts[102,117]; therefore, we manually read some samples to have an idea of what to prune on GitHub texts.

The third activity, FILTER, uses two techniques. The first is the TF-IDF weighting schema[118] to extract the more relevant words according to its inverse frequency. The second is the part-of-speech tagger (POS-tagger) technique[119], used to filter the nouns words. We select noun words since concepts are usually expressed as nouns. Here, we use the koRpus package[120] for the R language.

```
retrieval system, or [ digital library ] .[ Usage]
  / invenio- Invenio [ digital library ] software http:/
 BibTeX entry from a [ digital library ] , edit it carefully
     of tracks in my [ digital library ] . I often come
 Google Scholar, ACM [ Digital Library ] , IEEE Xplore and
     , is building a [ digital library ] of Internet sites and
       project,[ the [ Digital Library ] of Mathematical Functions]
Miller from the Persian [ Digital Library ] and their TEI files
```

Figure 28.Corpus Reduction using KWIC

The result of this activity is a set of corpus-relevant keywords selected by a stakeholder, with the help of an assistant (Fig. 29).

E. BUILD a relevant corpus.

The set of *concepts* and the set of *keywords* selected by elicitors are necessary to assists in the creation of a final query that would bring a corpus of relevant projects.

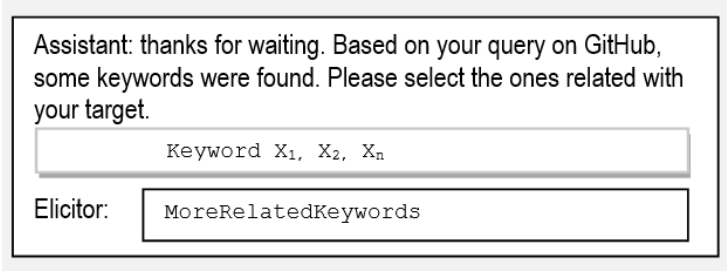


Figure 29. Interaction to show domain-related keywords

By using both sets of terms, it is assumed that the elicitor has more knowledge to perform a better query, as shown in Fig. 23. Nevertheless, an automated assistant should be capable of warming at any moment about the number of projects that bring a query (efficiency), and with that, the elicitor can improve it.

2.2.4. Related Work on the knowledge-based strategy for querying

Several works, from different areas, are related to the tasks we have automated. However, we did not find a direct benchmark with which we could compare our strategy.

1) Exploring Software Repositories

There is a well-known conference, *Mining Software Repositories* (MSR), where we can find diversity in approaches to explore massive data. To the best of our knowledge, few works are focused on mining requirements-related information; however, several mechanisms can be reused to automate RE tasks. Three works were selected, as they stress the importance of finding relevant projects, the use of knowledge bases, and the mining of concept keywords.

Grechanik et al.[121] propose a strategy to find relevant projects in software repositories. Unlike us, they are not worried about finding domain words on natural language as they want to filter projects based on source code. For such, they, as developers, know what keywords to use. Another work[122] that aims software reuse uses a knowledge base such as WordNet to find similarities in names that programming methods/classes in code could have. On this subject, the path of using knowledge bases is a mechanism to reduce the syntactic bias of the approach.

Relate to IR-mechanisms, Ohba & Gondow[123] proposes the ckTF/IDF technique to find concept keywords from code. They found that their strategy performs better than the TF/IDF weighting method[118]. However, they were dealing with source code and not with natural language texts.

2) Knowledge Discovery

Another area that tackles the task of the finding of relevant information is knowledge discovery, where at least there are three venues dedicated to this topic. From them, we selected three works related to our strategy.

Timonen et al.[124] improve the TF/IDF weighting technique[118] to perform keyword extraction in short documents where important words would occur rarely. This work is related to our case, as, after the reduction of corpus achieved with the KWIC technique, the Readme descriptions become short documents. For such, we place this technique as a future work to be applied.

Arcelli-Fontana et al.[125] identified that the task of finding Open Source Projects that satisfy a requirement has limitations when using a vague query. They found and stated "the low precision of RepoFinder is influenced mainly by the generality of the keywords," as such they propose as future work online query support.

Aljamel et al.[126] performed a supervised learning approach to obtain relations among entities in different sources of unstructured data. This work stresses that natural language data requires further processing to get meaningful information. In this regard, they state "knowledge services that require Information Extraction techniques to be able to search and extract specific knowledge directly from the unstructured text should be guided by the domain knowledge that details what type of knowledge is to be obtained and for which exploration scenario." This work is similar to us in the task performed; however, they rely on a supervised technique. We highlight this work as it shows that extracting information in massive data is complex, and it is necessary to be guided by knowledge bases.

3) Requirements engineering approaches

The task of finding concepts has always been a need in RE[3,19,100-104]. In this regard, our strategy is very similar to authors[3,19,103,127] for the use of an IR strategy; however, strategy[19] does not rely on a part-of-speech analysis to rank concepts.

Regarding the KWIC mechanism and the use of part-of-speech analysis, our strategy is very similar to approach[102]. In works[3,76] knowledge bases are used as a

way to reduce ambiguity; however, our work does not explore concepts relationships, as performed in work[3,]

Lian et. al[127] report the demands from industry to perform a quick identification of relevant information given a domain. Towards its automation, the step of identifying keywords that may anchor documents was performed manually. The authors reported that such a task took 35 hours. Although the author's goal is not to find domain concepts, it is worth stressing the time it took to get them manually.

4) NLP for Requirements Engineering

NLP (Natural Language Processing) is a core field to automate RE tasks[128]. On this topic, two works stress qualities in the tasks automated: easiness for requirements comprehension[129], and identification of similar requirements for reuse[130]. NLP is also used for recommendation systems in RE[131,80,82-88], that are looking for a system that recommends from documents to development artifacts. Our knowledge-based querying approach is similar to our previous work[80] that recommends GitHub projects to elicitors.

2.2.5. Assessing the Querying Assistant

By using the Vignette[10] technique[132,133] (Fig.30), we asked three students to create a GitHub corpus for the Digital Library domain we want to assess the quality of the corpus they got by using the queries assistant. The students are from a non-local university who volunteered to participate (from hereafter, elicitors A, B, C). They are in the last year of their undergraduate studies and have an intermediate knowledge of English.

> You are an undergraduate student responsible for building a repository for papers to be published in a local conference. The conference will be held within a month. You probably barely know about the Digital Library domain. In that context, a faculty member advises you to explore GitHub as a way to save time learning about the domain problem.

Figure 30. Digital Library Vignette

It was asked their level of knowledge about Digital Library; they reported that all they know is that ACM and IEEE Xplore are websites where you can find scientific papers. However, they seldom use these platforms.

A) Pilot

Two subjects were asked to test the strategy before assessment with elicitors A, B, and C. It served to improve the assistant interaction. However, a concern related to the *seed concept* appeared, which is the possibility of creating many *seed concepts* after reading the Vignette. Another concern was the selection of more entities in disambiguation. Finally, a subject performed this type of query: (document* OR metadata OR proceedings) AND (storing OR repositor*). In this regard, the assistant could guide the elicitor in another sense, that it is possible to use more concepts in a query, as long as it is used with the logic operators.

To simplify the assessment, we set the seed-concept to the main topic: Digital Library, and limited the choice of entities to 1. Also, it was added a

[10] The Vignette technique uses small impressionist narratives that are easier to understand.

reminder: “Remember, the more concepts you use, the more restrictive your results will be” (as in Fig. 27).

B) Tasks

The assistant asks elicitor A to create the basic query using the fixed-seed concept “digital library” in the corpus retrieval tool[79] and to send us the corpus retrieved.

The assistant guides Elicitor B and C in the strategy. The values expected are the concepts, the <concepts + seed-concept> query, the keywords selected, and the better query: <seed-concept + concepts + keywords> (Table 4). Finally, using the better query, the assistant asks for elicitors B and C to create a corpus of projects using the tool[79].

Table 4. Results from students using the strategy

Elicitor B	Elicitor C
Concepts Selected	
Information, digital content, database, online, digital libraries	digital repository, systems, interoperability, social media, digital
Seed Concept+ Concepts Query	
"digital library" + database	"digital library" + systems
Keywords Selected	
framework, system, access, repository, information	jcdl, zetoc, collection, project, kitodo.
Seed Concept+ Concepts+Keywords Query	
“digital library” + repository	“digital library” + jcdl

C) Results

From corpora of elicitors A, B, and C, we filtered the top 10 projects. Table 5 shows a list of the ten projects with the links (each number enacts a link to GitHub Readme per subject).

Table 5. Students top 10 projects

C_A	1, 2, 3, 4, 5, 6, 7, 8, 9, 10
C_B	1, 2, 3, 4, 5, 6, 7, 8, 9, 10
C_C	1, 2, 3, 4, 5, 6, 7, 8, 9, 10

The analysis of the results (Fig.31 and Fig.32) was performed by the author and advisor of this work (appraisers) which consider themselves knowledgeable in the domain. Using the Likert scale[90] from 1 to 5, 1:Poor, 2:Fair, 3:Good, 4:Very Good, 5:Excellent; each appraiser should answer the following question: *How useful is the project's readme in empowering the subject with domain knowledge, for the given Vignette in Fig. 30?*

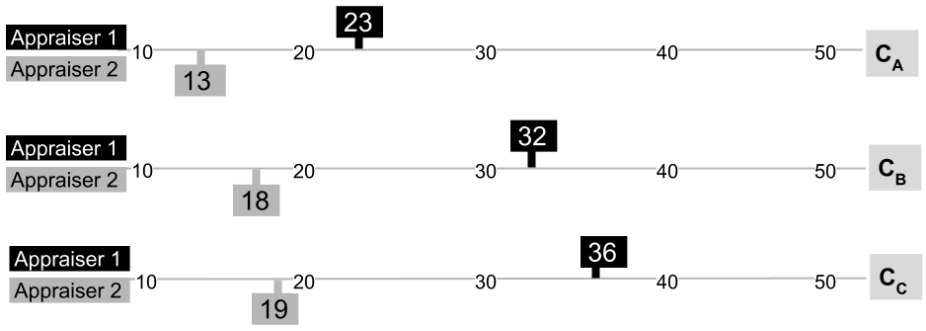


Figure 31. Overall Assessment of Corpus created with Querying Assistant Strategy

Fig. 31 shows the final grade given to each corpus (C_A, C_B, C_C). We sum the grades (from 1-5) given to each Readme. Then, the maximum grade for a corpus is 50, and the minimum 10. Then using the Liker scale, the grade values are 10:Poor, 20:Fair, 30:Good, 40:Very Good, 50:Excellent.

One appraiser considered good the corpora created using the strategy (C_B, C_C); however, the other appraiser thought the three corpora are close to fair.

We found that Fig. 31 is not enough to conclude, then we performed a more detailed analysis of results. In this regard, we wanted to know how many projects in each corpus got a high value (4 and 5) given for each appraiser (Fig. 32). With this, it was calculated the proportion of most valued projects within each corpus:

C_A: 5% C_B: 25% C_C: 35%

From this analysis viewpoint, we found that corpora using the strategy, C_B and C_C, are 5 and 6 times better than C_A, respectively.

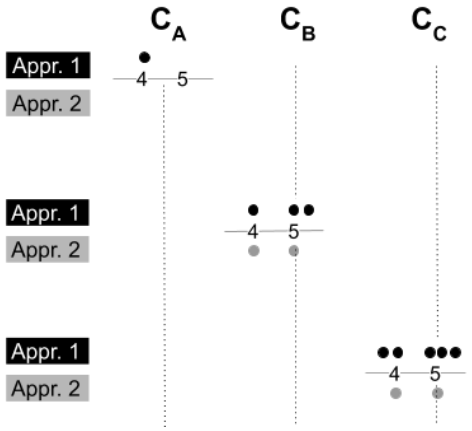


Figure 32. Projects with higher value in assessment

D) NFR evaluation.

The problem modeled in Fig. 21, is a conflict in the Softgoal knowledgeable elicitor. It was produced by the contribution + received from the operationalization GitHub. We propose a strategy that improves the results an elicitor will have if the querying assistant is used. We posit that our results are sufficient to change the NFR operationalization GitHub to Improving Querying GitHub, as well as change the contribution interdependence to ++. (see Fig. 33).

Figure 33 is the result of applying the propagation procedure[15]. The label of Softgoal Knowledgeable [Elicitor] with this new configuration shows that Relevant projects are weekly satisficed, which means that we get to relevant projects. Our problem is, of course, a wicked problem[134], so there is no solution from the mathematical perspective. However, it is possible to argue about satisficing wicked problems, since there are qualitative by nature. We stress that our strategy contributes towards a better situation (++) than the everyday use of GitHub queries.

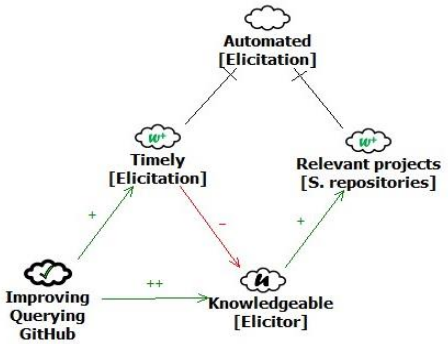


Figure 33. NFR Model Using the Strategy

Note: we made available the IR-based algorithm and evaluation of elicitors A, B, C, in a persistent repository[135].

2.3. Chapter summary

We explored the operationalizations created for corpus creation. It is explained that GitHub helps in producing a corpus with different viewpoints (projects) and perspectives[1] (project's artifacts). Then, the creation of a relevant corpus for RE purposes is addressed with a RecSys approach called GH4RE. Later, two strategies to perform better queries which may impact in better corpora are presented. The first strategy explores the exact-matching technique by mining Bills related to the transparency NFR. Due to issues encountered, the second strategy uses a knowledge base such as Wikidata to improve the quality of queries before the retrieval of a corpus from GitHub.

3 Facts Elicitation

This chapter details two type of facts[2] explored in this thesis. These are, requirement sentences, and requirements classification. To the former, we made a distinction of statements by using the Zave & Jackson[40] classification, one is related to domain information, and the other related to the user's desires. To the latter, we develop a Gold Standard for Non-functional Requirements classification.

3.1. Requirement Statements

One of the tasks while performing Requirements Elicitation is the gathering of facts related to the target in question. In the work of Shepperd & Schofiled[136] about case-based reasoning (CBR)[137], we found that it can serve to speed up elicitation, as it points out that analogies can avoid problems related to knowledge elicitation such dealing with a poorly understood domain. The activities proposed for evaluating an analogy are stated such: "the identification of a problem as a new case, the retrieval of similar cases from a repository, the reuse of knowledge derived from previous cases and the suggestion of a solution".

Our work proposes to operationalize the finding of similar cases by exploring the Issues perspectives from GitHub. From our experience, this artifact is the most used while producing software; as such, there may be more domain information than in Readme perspective.

According to Zave & Jackson Formula[40] S, K ⊢ R, specifications (S), and knowledge (K) must be sufficient to guarantee that requirements are met. That is, requirements (R) become specifications (S) when the gap of knowledge (K) is bridged. Therefore, if the text-mining of facts is capable of bringing K, an elicitor is better prepared for elaborating S from R.

Two types of sentences are identified[40], indicative mood, and optative mood sentences; the former are the ones describing domain knowledge as it would be in the absence of a machine. The latter is the sentences describing desires from users. Thus, we propose that in a semi-automation of elicitation, the patterns for mining should seek the identification of both types of sentences.

3.1.1. Optative-mood Sentences

In the GitHub environment, as projects are described as free texts (free natural language descriptions), software desires can be expressed in different ways. For that matter, the finding of patterns that provides efficacy in a corpus is a challenge. An approach can be found in our previous work[24,138] by using morphology patterns such as modal verbs, e.g., must, wants, allows, found in phrases with requirements-related information.

According to Zave & Jackson formula[40], optative-mood sentences are the ones that express software desires, that can be present in sentences [S] and [R].

However, as stated by them[40], there is a difference among requirement specifications [S], and requirements sentences [R]. The former is better in the sense that these are requirements already elicited and specified. The latter is information related to requirements, where desires are expressed in different ways, and expressing functional and non-functional needs.

Regarding sentences S, previous work found patterns[78,138] in GitHub texts that may facilitate the finding of specifications, e.g. the pattern "as a user want to be able…" used in agile approaches such as Scrum[78]. Thus, the finding of many of these type of sentences (S) can speed up the retrieval of relevant facts from where to elicit NFRs.

We develop an operationalization to get optative-mood sentences using the Issues perspective on GitHub. In particular, we are interested in Issues with the label Enhancement, as they usually describe demands of features to be implemented in a project. However, as issue labeling is not mandatory on GitHub, there exists a large number of unlabeled Issues that can be of Enhancement type; for that, we used Supervised Learning Techniques[139] to speed up its retrieval by learning from existing data and to classify the non-labeled ones.

3.1.2. Eliciting Optative-mood sentences on GitHub Issues

We propose to speed up elicitation by using case-based reasoning (CBR), where analogies can avoid problems related to a poorly understood domain. Thus the finding of GitHub Issues given a target domain can be *satisficed* by using a supervised learning techniques[139].

1) Dataset of Issues from GitHub.

We retrieved 50506 issues by querying trending topics that can be opportunities for startups investors[141]. The dataset is composed of some attributes existing in raw JSON files retrieved through REST Web services, such as: id, title, body (the Issue description), state (if an Issue is open or closed), user, project_url, up-date_at, html_url, comments (indicates the number of comments), score (according to GitHub relevance), and labels (an array of labels). To keep the trace of data, we added other attributes such as query (the query that brought that Issue). We also added seven attributes for each possible label defined by GitHub: bug, duplicate, enhancement, help-wanted, invalid, wontfix, and question.

a) Dataset Sample

An SVM (Support Vector Machine)[139] model requires classified items for learning; thus, our dataset was split into two groups: the ones with at least one label and the ones with no labels at all. It resulted in a sample dataset composed of 10469 issues, and a non-labeled dataset with the 36629 Issues.

b) Features in Dataset

For training purposes of the dataset sample, it was considered only three attributes of the dataset: title, body, and enhancement; this last informs rather an Issue is an enhancement or not. Then, the Bag of Words technique and the TF-IDF scheme[118] are used to create a matrix of features (words), which represent the importance (frequency) of each word (column) in each Issue (row).

Some common preprocessing steps are required, such as taking out the English stopwords. As our texts are very similar to the Enron-email dataset[142,143] in size and type of texts (unstructured), we decided to consider the English stopwords.

2) The classifier of Enhancement Issues

The dataset sample was trained following an n-fold Cross-validation strategy[144], to train and test all Issues in dataset sample. Another reason to use this strategy is to verify the behavior (recall-precision) in each portion of training/test data (See Table. 6 and Table. 7). Table 6 shows a dataset sample divided into two parts, 90% for cross-validation (9423 Issues), and 10% (1046 Issues) reserved for validation. At the end of the 10-fold strategy (Table 7), a single model was trained by using the 9423 Issues; then, the model was tested in the validation set. The performance achieved by the model is **Precision = 87% and Recall = 84%.**

Table 6. 10-fold Strategy over Dataset Sample

Validation set (unseen issues) 10% of Dataset sample	10-fold set (seen issues) 90% of Dataset sample	
1046	9423	
	Each fold Training	Each fold Testing
	8480(90%)	941 (10%)

3) SVM Model Verification

One of the results provided by an SVM model is a vector W (of feature weights) that contains all the features been trained. In SVM, the features with the highest and the lowest W are relevant as these influence the classification. Table 8 shows the top 10 of each portion of features, the highest and lowest weight. We found that the ones that influence the classification of an Issue as Enhancement, are words that may describe an Issue Bug; contradictorily, the ones that not classify an Issue as Enhancement, are the ones that are commonly used to describe a functionality/feature.

Table 7. 10-fold Strategy for Training Dataset Sample

	10-fold set		
	Precision	Recall	F-measure
fold-1	0.86	0.79	0.823515152
fold-2	0.84	0.84	0.84
fold-3	0.82	0.71	0.761045752
fold-4	0.77	0.58	0.66162963
fold-5	0.92	0.78	0.844235294
fold-6	0.74	0.61	0.668740741
fold-7	0.8	0.62	0.698591549
fold-8	0.74	0.45	0.559663866
fold-9	0.83	0.61	0.703194444
fold-10	0.8	0.44	0.567741935
		Average F-measure	**0.712835836**

4) Feature Engineering to Improve SVM Model

Features existing in data need attention as their influence in classification. However, this task is costly, as it will require manual work or automated strategies to add features that are relevant to the domain knowledge. Our target is to classify Issues related to functionalities, but our first automated model brought unexpected results that, at first sight, may suggest that GitHub Issues are misclassified. The following are the steps towards an improved model using a semi-automated feature engineering[145].

Table 8. Features that Influence Enhancement Classification

Higher Weight	Lower Weight
Fix	Feedback
Error	Request
Bug	Relevance
Incorrect	Use
Crash	Friendlier
Broken	Ideal
Wrong	Improve
Can’t	Should
Affect	Enhance
Detect	Feature

a) A Bug is an Enhancement

In the work of Antoniol et al.[117], it is mentioned that words such as *crash, critic, broken* lead to classify bugs, and we obtained crash and broken to classify an Enhancement. This assumption made us hypothesize about an existing scenario in GitHub: a user reports a Bug, then, after revision/discussion with the development team, they agree that indeed, this cannot be a Bug but an Enhancement to be done. In this regard, an author[146] helps to clarify the nature of bugs by defining three kinds of them: "Bug is an error: The software is not doing what we wanted it to do. A bug is an enhancement: The software doesn´t do quite what we wanted it to do. A bug is a feature: Something that the software should do that it does not do". After the manual reading of several issues with the "fix" word, we assume that we are facing the second kind of issue. That is, that a bug is a future enhancement.

b) Selecting Features from SVM

In our SVM model, we have a vector W of features with the highest weight (in positive) and the lowest (in the negative). The ones from the extremes of vector W are the most important as these classify with more precision, whether an Issue is an Enhancement or not. For such, it was removed the negative sign in the vector and then ordered it. From this new vector W, ordered, we took almost 10% of all the features existing in the dataset. That is, from 65732 features (words), we took the top 5000. The criteria considered to select only 10% of features, is the threshold set: features should appear at least in 100 issues.

c) Boilerplates as Features

Manual feature engineering[145] is a task that is used to select attributes useful for training a model. A feature may help to a better understanding of the context of a problem. For such purpose, we used a previous work[138] in the context of the Readmes perspective of GitHub, where a list of boilerplates (patterns) was found to describe software functionalities. Table. 9 lists 30 boilerplates selected from work[138] by considering only the ones with a frequency higher than ten.

Table 9. Requirements boilerplates in GitHub readmes[138]

Boilerplates	Freq. in 291 Readmes
you can	926
can be	655
enable	312
to create	279
should be	222
used to	164
that can	120
you should	97
to provide	77
in order to	76
designed to	66
to manage	61
can do	60
which can	57
to allow	54
let's you	44
can be used to	43
should be able	38
I want to	36
designed for	34
project to	32
that allows you to	31
users can	29
allows users to	28
one can	27
it should be	23
intended for	23
that shows	22
allows for	20
I can	20

Boilerplates are features that are composed of more than one word. This kind of feature is also called n-grams[147]. To add the boilerplates to the new dataset, we needed to calculate their TF-IDF frequency in the sample dataset. With n-grams frequency, a new dataset of 5030 features was created; i.e., the top 5000 features from a dataset sample plus 30 n-grams features.

d) Model SVM with Features Engineering

With a new dataset, the 10-fold Cross-Validation strategy was applied to verify its behavior (precision-recall) in every part of the dataset trained. Table 10 shows a better performance than the previous model (Table 7). Therefore, all Issues in folds are trained to create a single model. The classification results for this model are: **Precision = 89% and Recall = 84%.**

In comparison with the previous model, the new improves **precision, from 87 to 89%.**

Table 10. 10-Fold Strategy in New Dataset

Dataset with Features Engineering			
	Precision	***Recall***	***F-measure***
fold-1	0.87	0.82	0.844260355
fold-2	0.87	0.84	0.854736842
fold-3	0.81	0.72	0.762352941
fold-4	0.79	0.64	0.707132867
fold-5	0.9	0.7	0.7875
fold-6	0.71	0.66	0.684087591
fold-7	0.83	0.62	0.709793103
fold-8	0.87	0.69	0.769615385
fold-9	0.85	0.73	0.785443038
fold-10	0.82	0.57	0.672517986
		Average F-measure	**0.757744011**

5) SVM Model Over-Fitting

We bring again the main goal for which this model is being built. We want to create a model to classify unlabeled Issues that can be retrieved for any domain. Thus, we want to test if the SVM+features model is not over-fitted to the data trained. In this regard, we used the Issues not used for any SVM model trained, i.e., the unlabeled issues group, from where 50 Issues were randomly selected as a sample. Two researchers were asked to manually classify the 50 Issues for later comparison with the classifier results.

Table 11. Performance of svm+features classifier against manual classification

Hits researcher 1		**Hits researcher 2**	
Enhancement	20	Enhancement	20
True positives	15	True positives	16
True negatives	8	True negatives	9
False positives	22	False positives	21
False negatives	5	False negatives	4
Performance			
Precision	**0.40**	***Precision***	**0.43**
Recall	**0.75**	***Recall***	**0.80**

a) SVM+features Classification

In the unlabeled group of Issues: 36629 Issues we found the existence of 2145 features of the 5030 from our SVM+features model. We filter those 2145 features to train a model using the group of unlabeled-Issues dataset minus the 50 issues from the sample. Then, the model is used to classify the 50 Issues of the sample.

b) Assessment of SVM approach

Two researchers were asked to classify issues according to the seven labels of GitHub. Table 11 shows the results of manual classification (Appendix A) against the SVM model classifier. Both researchers agree in 20 Issues being an Enhancement; however, they only coincide in 15 Issues. Further evaluation is needed for other labels such as Question or Help-wanted because these labels are close to the meaning of Enhancement.

c) Results Discussion

The results are promising if we consider the recall as an important measure when dealing with hairy tasks[77], as such Berry states:

> "The two components of "correctness" are recall, that all the desired information is found, and precision, that only the desired information is found. Of these two components, for a hairy task, recall is more in need of tool assistance. For any task for which tool assistance is truly needed, finding a unit of desired information among the many documents available for the Computer-based system development is generally significantly harder than dismissing a found unit of information that is not desired. It is like finding needles in a haystack when one does not know how many needles the haystack has. If recall were not the harder component of correctness".

To the set of 50 samples, we can value *recall* as an important measure as we know which the Enhancement issues are; thus, we can measure if the desired information is found. Regarding the performance of both models built, precision is more important, given that we are in this context stated by Berry[77]:

> "Not every instance of a hairy task needs close to 100% recall, and it may be enough for a tool for it to achieve only some recall. For example, if the task is to determine from NL app reviews whether a particular app has been actively used, finding any review that describes the app as being used suffices61 […]. In such a case, it would be more important for the tool to achieve high precision."

As such, the models in their 10-fold analysis (Table 7 and 10) performs better in precision, which, according to the criteria for hairy tasks defined by Berry, is a positive result.

Kolodner[137] states about CBR: "Case-based reasoning can mean adapting old solutions to meet new demands, using old cases to explain new situations, using old cases to critique new solutions, or reasoning from precedents to interpret a new situation (much as lawyers do) or create an equitable solution to a new problem (mush as labor mediators do)." By managing to classify unlabeled Issues, we improve the recall of Enhancement Issues by using a Machine Learning approach, so effectively saving time, and improving the efficiency of Requirements Engineers as they look for sources of information (cases) from which reuse of information can be applied as stated in the Kolodner citation.

Regarding Feature Engineering, we perceive this is a crucial technique to be applied to RE approaches with Machine Learning, as this improves the classification model. Tables (7 and 10) shows how an SVM model with features becomes more stable in every fold tested. That is, through this strategy, we can verify if the data trained requires some tuning.

Through this SVM case, we learned that SVM should be applied carefully by taking into consideration the data and the context of data. The strategies used, such n-fold Cross-validation and Feature Engineering, helped to improve our classifier by having a better understanding of the nature of Issues in GitHub, instead of assuming a misclassification.

Towards our main goal, to find analogies in similar cases to solve a new case, we believe this approach contributes to facilitating the reuse of such content. It is needed for Requirements Engineers to speed up the understanding of a domain before eliciting, modeling, and analyzing software requirements. Also, GitHub Issues are valuable assets as these are free texts in optative mode[40], that is: they describe what is required.

3.1.3. Indicative-mood sentences

According to Zave & Jackson[40], Indicative-mood sentences are the phrases that indicate domain information in the absence of a machine. That is, sentences with only domain knowledge. From our exploration in GitHub data, we found a difference between domain knowledge and domain-technology knowledge, although the latter can be referred to as some machine (software). E.g., for the *Real Estate* domain, the word Zillow[11] appeared as a frequent acronym in GitHub texts[80,] which may suggest its relevance in a corpus of projects related to Real estate. Another relevant acronym found in the previous work[78] is RETS[12]. Both terms, which are not specific to the real estate domain, are a technology used in the domain, which are important to speed up elicitation; i.e., by knowing which technology is used, it is possible to know the functions (goals) implemented by them.

Then, we extend the Zave & Jackson Formula[40] to S, Kd, Kc ├ R, where Kd is related to domain information, and Kc to context information. Context covers the domain-technology information as well as other information not directly related; e.g., a query <real estate + pets> opens the search to other domains. The formula proposed S, Kd, Kc ├ R can be read as follows: a requirement R is met in every interpretation when the specification S, domain knowledge Kd, and context domain knowledge Kc, are sufficient.

Figure 34. Syntactic filtering of frequent-nouns from Real Estate domain[80]

3.1.4. Semantic filter of indicative-mood sentences

Our work proposes the use of Knowledge bases to speed up the finding of domain-related sentences. That is, we want to separate the domain-related nouns from nouns of other domains (Fig. 34), even though these are related in a GitHub project.

Our approach[80] applies the keywords strategy[148] where nouns are filtered to provide important information. Nonetheless, it is necessary pre-processing steps to deal with the phenomena of polysemic words that may cause the finding of ambiguous sentences[24], for which we used two techniques, clustering[80] and NLP[24,80,78] (natural language processing). Both perform syntactic analysis of texts. The former groups by the similarity of texts, and the latter performs POS-tagging[119] to filter the nouns.

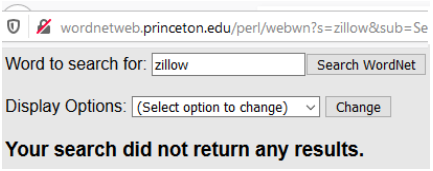


Figure 35. Filtering proper-nouns using WorldNet

As a noun is a broad category that includes proper-nouns, abstract-nouns,

[11] Zillow.com: The leading real estate marketplace. Search millions of for-sale and rental listings, compare Zestimate® home values and connect with local professionals.
[12] RETS is an acronym which stands for Real Estate Transaction Standard.

among others. In particular, we found that to have domain words we need to filter and remove the proper-nouns so that the domain nouns can appear. As such, we propose a semantic filtering by using a knowledge base such as WordNet, which is a database formed of open-class words[149] (nouns, verbs, adjectives, and adverbs), that can allow to identify proper nouns as these are not in its database. Therefore, by using a WordNet wrapper[150], we can differentiate nouns from proper-nouns (Fig. 35).

3.2. Requirements Classification

Two types of sentences used in RE are Functional requirements (FRs) and Non-functional requirements (NFRs). Existing approaches on classification of NFRs[56,57,58] consider FRs as one of the types for classification, and, although automating this task with IA (Artificial Intelligence) is having increasing attention of researchers[151,152,153], still the approaches are automating based on fixed structures/taxonomies (section 1.5.2) of NFRs attributes which limits the task as NFRs are cross-cutting[39].

We propose a hybrid approach that is syntactic in the use of NLP to find qualifiers on sentences. e.g. adjective words, and it is semantic by using existing SIGs to rank findings as performed in section 2.2.1. Thus, we developed heuristics[76] that gave positive results when compared it with a Gold Standard[75].

3.2.1. NFRFinder

A) Finding Similar Operationalizations on Catalogs

Catalogs are a rich source of information as the knowledge related to an NFRs is organized in clusters and relations. Two of them are the SIGs[15] and i* models[154]. An initial idea to take advantage of this organization is to consider that *operationalizations Softgoals* are clustered by *NFR Softgoals* (Fig. 8, Fig. 11). Then, to classify a potential text (unstructured) or a requirement sentence (structured), we will need to search similar operationalization Softgoals, or similar NFR Softgoals, so that we can find the closest context from which we can extract keywords. This approach is based on strategies of search engines where the finding of similar texts with more precision needs to enclose its context, which is very expensive in time processing[155].

Fig. 36 is an SADT[107] model that resumes the strategy for finding similar texts in SIGs, where a series of filtering activities is performed in a given order, as explained as follows.

The first filter is to identify if a requirement text (input) matches with adjectives

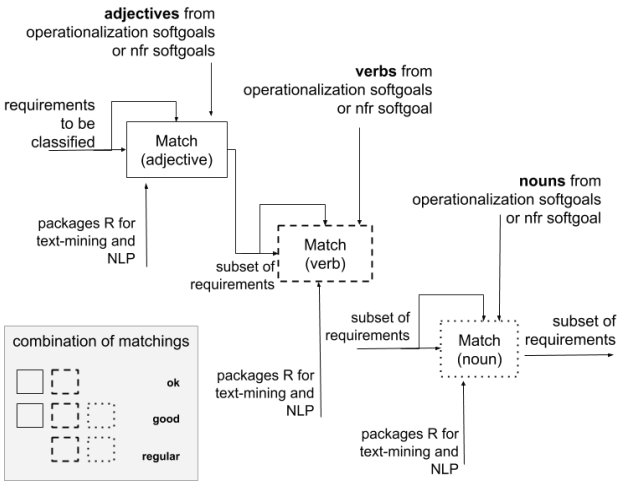


Figure 36. Filtering Process over SIGs

existing in NFR Softgoals/NFR operationalizations in a target SIG. This heuristic is based on finding qualities first. Since qualities are usually adjectives, the match with adjectives identifies the requirement as an NFR.

The subset of requirements is the input of the second filter to verify if some of these texts also match with some verb existing in the target catalog. This heuristic is based on the reasoning of which is most valuable, among verbs or nouns, for our search of similar texts. For such, we selected verbs, as they are more domain-independent than nouns.

The last filter is applied to a subset of requirements with an adjective and verb found in the catalog, to verify if some of them also match with a noun.

The legend in Fig. 36 shows the combination of matches; that is, if a sentence is filtered in the order proposed, then the sentence in evaluation is very similar (good); thus, it can be classified as NFR. It is ok when a sentence match with some operationalization Softgoal or NFR Softgoal in adjectives and verbs; texts are similar, except that they are referring to different subjects (nouns). It is regular when the match is in verbs and nouns. In this last case, texts can be similar but with few chances to be an NFR as they do not match in qualities (adjective).

This approach (Fig. 36) returned few matches due to the lack of more SIGs related to the qualities existing on the texts we used[75]. Nevertheless, from this approach, we note that for our task of finding similar texts, some nouns made no difference at all as hints: user, system, product, customer, website, among others. We placed them in a list of stop words. Furthermore, some verbs were not relevant to identify an NFR as they are more useful when identifying functional requirements, as seen in our previous work[138]. Some of these verbs are: provide, set, be, make, shall, have, allow, are, take. They were also placed as stop words.

B) A Knowledge-based Approach to find Qualifiers

We designed a strategy to classify NFRs by using the knowledge in SIGs to weight qualifiers. Tables 12, 13, and 14 define the multi-criteria that we exemplify by using a requirement sentence from a gold standard[75]: "*The product shall be able to support multiple remote users*". First, Table 12 indicates the conditions to identify adjectives, verbs and nouns using a POS-tagger, thereby the words in bold are adjectives. To the case of verbs (in italics) *shall be* is the type of word in our verb-stopwords list. Finally, nouns as product and user are also a noun stop word. The sentence ends with the following qualifiers:

Table 12. Criteria to identify qualifiers

Type	Condition
Adjectives	All adjectives
Verbs	All except stopwords
Nouns	All except stop words and proper nouns

"The product shall be **able** to *support* **multiple remote** users."

Table 13 defines the criteria to give more relevance to qualifiers identified. A qualifier is relevant (key) if this, being adjective, can be found in some SIG used. If a qualifier is a verb in past participle, it is

Table 13. Criteria to find key qualifiers

Type	**Adjective**
Condition	Only when match an adjective of a catalog
Type	**Verb**
Condition	Only the verbs in past participle
Type	**Nouns**
Condition	Only when lemmatized becomes a verb

also relevant, as it acts as an adjective[156]. And the nouns, when lemmatized in its singular form, becomes a verb[157]. This last heuristic can be used to avoid the nouns that are entities and are not as relevant as the ones cited in stopwords.

Finally, table 14 defines the heuristics to classify an NFR based on qualifiers on sentences. It is important to note that we are not performing classification by NFR Type, but identifying whether a sentence is or not NFR.

Our approach does not use fixed attributes for NFR classification but relies on knowledge existing in SIGs. Then, to classify by NFR, it could be performed by tracing the NFR Softgoals that have the qualifiers in a sentence. Therefore, a SIG allows the inference of the parent NFR Softgoal related to some offspring NFR Softgoal.

Table 14. Criteria for NFRfinder classification

Criteria	Value
It is a **NFR** if **a word** in sentence match with **1 NFR softgoal** in any catalog	NFR
It is a **NFR** if **many words** in sentence match **with many NFR operationalizations)**	NFR
It is **strong candidate** if a sentence has **1 match** with a **catalog** and **key qualifiers**	S
It is **strong candidate** if a sentence has **1 match** with **many catalogs.**	S
It is a **medium candidate** if the sentence has **many key qualifiers**	M
It is a **medium candidate** if the sentence has **1 match** with a catalog and **various qualifiers**	M

3.2.2. Gold Standard of NFRs

When identifying requirements, we are dealing with a *hairy task*[77]. As such, any tool for a hairy task should be evaluated by comparing the recall of humans performing the task with and without the proposed tool[77]. In this regard, we conducted an exercise to verify how four domain experts would classify the existing NFR/FR statements[75], with and without catalogs.

In the work of Leite and Freeman[158], viewpoints are built using different perspectives and are compared pairwise to produce requirements of better quality. As such, there is a two-fold verification strategy: each actor (viewpoint) performs an elicitation using different perspectives and then provides their final viewpoint. The viewpoints are then compared, and consistency and completeness problems may be detected. This strategy is used to build our gold standard, which is an identification/classification exercise over a list of requirements[75]. This strategy is also similar to the N-fold Inspection strategy[159].

A) Sample Fabrication

We began by taking a random sample of the data used in the existing gold standard[75]. Other approaches[57,58,61,152] in classifying NFR used this data. We decided to use thirty requirements sentences as our sample. Empirically, thirty observations are enough to conduct a test[160].

Fig. 37 shows the number of sentences per type of requirement in the sample. Most of them, 12, are functional requirements. We use this sample to build our own Gold Standard.

B) Knowledge Bases

Knowledge about NFRs exists in various approaches for modeling NFRs. Two of them are the SIGs[15] and i* models[154]. We have used four models[76] (catalogs) as our knowledge base. The actors performing the classification of NFRs use these catalogs as a reference in deciding if a requirement sentence is an NFR or not.

The catalogs are about the following NFRs: usability, privacy, and transparency. Three are SIGs, and the other is one i* model.

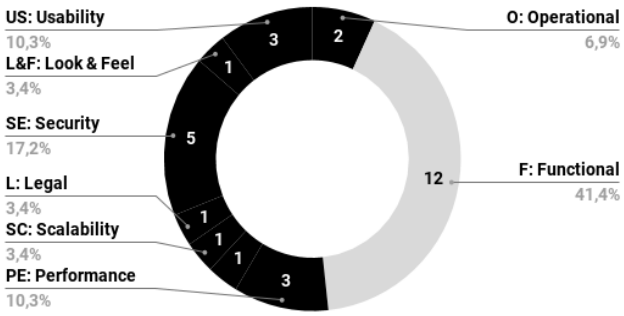


Figure 37. Distribution of Requirements Types in Sample

A. Evaluators of Classification

Four actors, with several years of NFR experience, evaluated the requirements sentences of sample (Appendix B)[161]. These evaluators did not have previous knowledge of the gold standard used[75] to create the sample. Each actor received two spreadsheets with the 30 requirements sentences[161]. Each file (spreadsheet) had different instructions; these are:

a. Using your knowledge, identify each requirement, whether it is an NFR or not. You do not need to classify them according to its NFR type, e.g., a security NFR.

b. Highlight the words that influenced the classification.

c. Provide argumentation to your decision using two criteria: common sense (cs) or implicit (imp). Use implicit (imp), only when no words influenced the classification.

d. Once the tasks (1)(2)(3) are finished, use the other spreadsheet, and repeat (1)(2) with the use of catalogs provided.

e. Comment the identification using the criteria: (cs), (imp), or (cat) if catalogs influenced the decision.

Each actor performed the task using two different perspectives[1], one with catalogs and another without them, to form their viewpoint. Our work[76] details in the process to summarize data collected from actors and to create our gold standard.

Table 15. Disagreements in Gold Standards

Req. sentence ID	153	462
NFR type of sentence	Usability	Functional
Gold Standard[75]	1	0
GS-4-View	0	1

Table 15 shows the sentences where both gold standards, the reference Gold Standard[75], and the (GS-4-View), disagree. In both sentences, we found that NFRs has a representation problem as stated by Glinz[26].

153: *100% of cardmember services representatives shall be able to successfully create a dispute case on the first encounter after completing the training course.*

462: *Website must be able to support free trial periods with various parameters set by the Izogn Manager.*

To allow transparency of this study, we made available all artifacts and results in a public repository at GitHub and persisted in the repository Zenodo[161].

3.2.3. NFRFinder Assessment

Our work[76] used both gold standards, the one used in literature[75], and the one we created (3.2.2). NFRFinder approach is a binary classification. It indicates, whether a sentence is an NFR or not. We aim to apply this strategy in unstructured texts, to suggest or making notice to a requirement engineer that in some texts, there is a high possibility of an NFR is valuable. ‘

Table 16. NFRfinder vs. Gold Standards[76]

Data	P	R	F1	Fβ=0,51	Fβ =1,08
GS-4-View	0.73	0.61	0.66	0.66	0.68
Gold Standard[75]	0.80	0.66	0.72	0.72	0.74

Table 16 presents the accuracy of NFRFinder with three measure scores: recall, precision, and f-measure. In Berry’s work[77], it is recommended for hairy tasks the measure β, where β is the time for a human to find, manually, a true positive in the original documents. In the 4-viewpoints gold standard, the four authors together made 120 classifications for each perspective, with or w/o catalogs. The total time spent in 120 classifications per perspective is 62 minutes’ w/o catalogs, and 130 minutes with catalogs. Thus, per perspective, the average classification required 62/120 = 0,51 person minutes per classification (w/o catalogs), and 130/120 = 1,08 person minutes per classification (with catalogs).

Table 16 shows that NFRFinder performed better in the reference gold standard[75,] with an F1-measure of 72%. Moreover, if considering the β value, mainly when a person uses a catalog for classifying (1.08 minutes), the Fβ performance of NFRFinder is higher.

3.3. Chapter Summary

It was described the facts elicitation by using two type of facts stated in the work[2], requirements statements, and requirements classification. For the first, two types of sentences are elicited, optative-mood sentences (user desires), and indicative-mood sentences (domain information). For both type of sentences, we presented operationalizations, such as an ML approach to elicit similar GitHub Issues, and a filter to identity proper-nouns as a means to identify domain-information. Finally, for requirements classification, we present an approach to classify NFRs called NFRFinder. Besides, a four-viewpoint Gold standard is presented.

4 Finding Interdependencies with Sentiment Analysis

This chapter presents the mechanisms used to find interdependencies among NFRs. As NFRs are controversial, that is, the choosing of one NFR impacts in another, we identified that chunks of texts, which we named as locations, can be used to find the closest NFR impacted. We propose the use of Sentiment Analysis to identify interdependencies (sentiment) in the middle of both NFRs. By using the NFR framework, the notion of interdependencies and contributions help in the operationalizations used.

4.1. NFR Interdependencies

Must have NFRs in a software, usually, are hard to elicit during requirements construction. Therefore, NFRs have less attention during requirements tasks, which results in the lack of proper analysis concerning their possible conflicts. Chung et al[15] states that "Design decisions may positively or negatively affect particular non-functional requirements. These positive and negative interdependencies can serve as a basis for arguing that a software system indeed meets a certain non-functional requirement or explain why it does not". This citation is relevant given that by eliciting a target NFR, there is the need also to elicit its context where other NFRs are involved, i.e., by using the NFR framework notation[15], NFRs have **interdependencies** that should be elicited. Besides, NFRs may lead to different design decisions, e.g., *the login procedure be quick and easy, and be secure*, such as having more security or less usability, or more efficiency and less usability. Those decisions, expressing a degree, are contributions in the NFR framework notation[15], and also should be elicited.

To the best of our knowledge, few works exist on the elicitation of interdependencies and its contribution (positive or negative) among qualities[72-74]. For automation of such elicitation, we used as operationalization three techniques: KWIC, word correlation, and sentiment analysis (SA)[114].

Fig. 38 depicts the approach. It uses SIGs as a reliable source from where to take quality keywords. That is, if our target is to find NFRs related to usability NFR, a reliable source is the SIGs existing about usability.

This approach is looking at two questions. First, *can the connection be detected by the SA technique (ranging from positive to negative)?* Second, *does detection provide useful information to determine interdependency among these qualities?* We decided to use usability NFR as the seed quality in exploring those questions.

4.2. Sentiment Analysis in GitHub

There is a variety of work focused on Sentiment Analysis. Following, we cite those related to artifacts similar to projects' Issues in GitHub.

Guzman et al.[162] reported about emotions found in GitHub commits. They used the tool SentiStrength[163] to identify emotions in comments and correlating them with spatial data existing in commits metadata. Blaz & Becker work[164] argues that feedback about the quality of services can be found in user requests "tickets" to expose sentiments that may bring contextual information, e.g., "I have no access to email again. It is the third time this week!" This work[164] reported its method that outperformed SentiStrength[163]. Our work differs from work[164] in the sense that tickets tend to be more polite[165] than issues because customers write tickets as part of a service contract. A similar goal is addressed by Ortu et al.[166] using Sentiment Analysis to find factors that make a multicultural development team healthy and productive. They used SentiStrength[163] and found that diversity in GitHub teams is related to "good manners" in communication. Imtiaz et al.[167] focused on job satisfaction according to the developer's mood. They[167] used GitHub Pull-request comments and GitHub Issues. It was found that tools, as well as humans, are unreliable. To a similar conclusion came Jongeling et al.[168] that made a study about this threat to validity: that tools such as SentiStrength[163] have been trained on non-software engineering texts like the ones in the movie/product reviews. Yang et al.[169], highlighted the influence of emotional factors in software development, specifically Bugs fixing. They mention that watching movies or listening to music while programming can impact in developers' emotions. The processing of GitHub texts was performed with their tool[169]. Finally, Jurado & Rodriguez work[170], is concerned with estimating the degree of satisfaction of the development team on a specific feature, as this can affect the process and the product quality. By using an NLP approach to find sentiments, they also found as

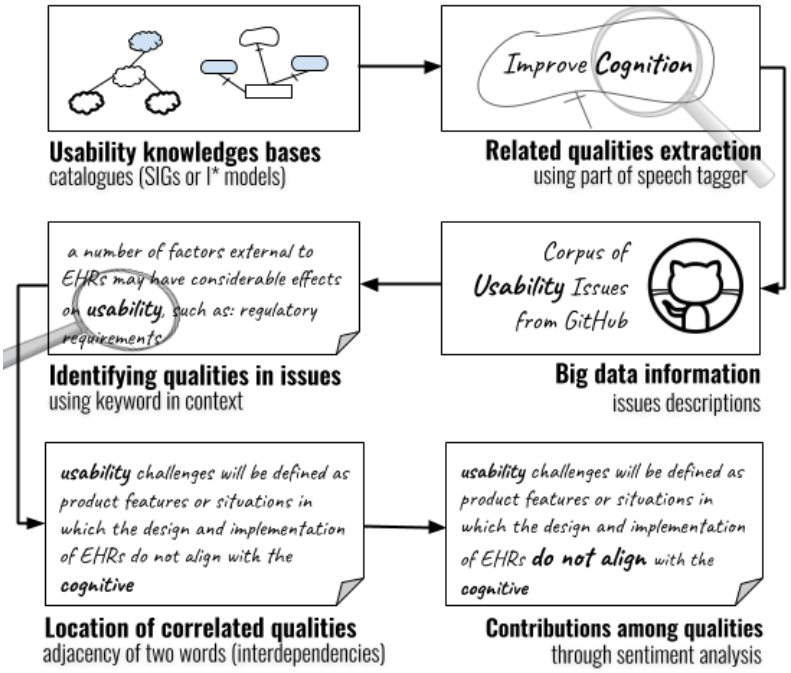


Figure 38. Strategy for Identifying Interdependences among Qualities

important the use of named-entity recognition (NER) for retrieval of information needed, which is important from our perspective on the use of emotions as a surrogate to interdependences.

4.3. Knowledge Bases for Keywords Extraction

Our work[114] uses catalogs, where usability NFR is present. Fig. 39 is a usability i* model[171]; it shows the relation of quality usability (cloud shape) with usefulness, performance, security, and availability, also expressed with the same shape. The catalog has goals (oval shape) and tasks (hexagonal shape), which represent the operation to attain goals and qualities.

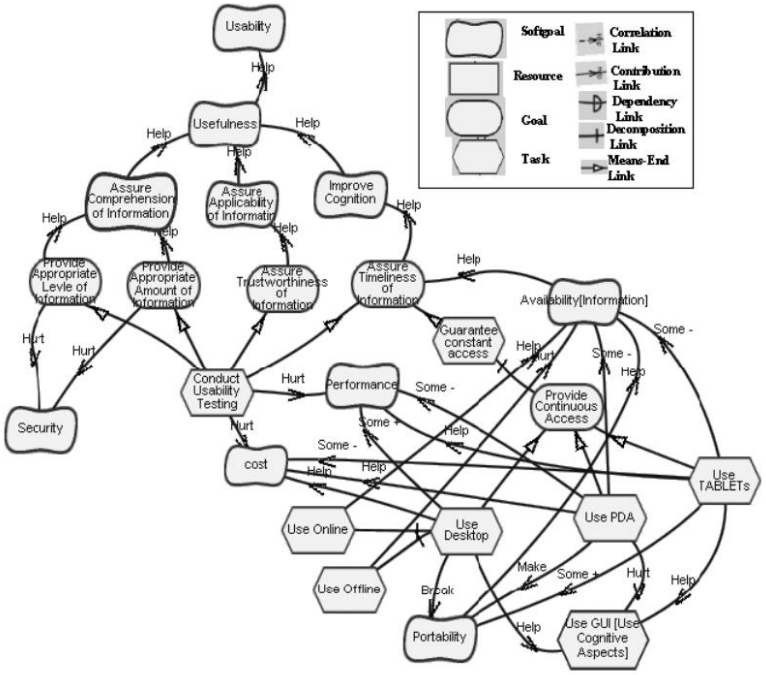


Figure 39. Part of a Usability Catalog[171]

4.4. Strategy for Eliciting Interdependencies

Following, we describe the strategy depicted in Fig. 38:

1) Filtering Qualities from Adjectives.

The most straightforward approach to extract quality keywords is through a syntactic analysis, i.e., tagging adjectives. From the texts in Fig. 41, the keywords identified are *appropriate, continuous, constant, online,* and *cognitive*. As such, a heuristic is to assume that adjectives are indicative of qualities (Table 17).

2) Filtering Qualities from Nouns

When POS-tagging texts, words tagged as nouns are the next candidates to be quality words. From this word set, we have a group of nouns derived from adjectives (Table 17) ending with the suffix “ity” and “ness” that are usually taken

as qualities[15]. These nouns can be filtered by using a regular expression (regex) function.

3) Corpus of GitHub Issues

In our work work[114], we use an example of usability in WhatsApp application. As such, with the query usability+chat+messaging, we created a corpus of GitHub Issues. The selection of this GitHub perspective is that this is more used while developing software; as such, stakeholders have a better knowledge about the target domain.

Table 17. Quality keywords filtered from a usability catalog[171]

Token	Tag
appropriate	adjective
continuous	
constant	
online	
cognitive	
usability	nouns
applicability	
availability	
security	
portability	
usefulness	
trustworthiness	
timeliness	

4) Location

The KWIC (Keyword in Context) technique is used to determine the location of the target NFR, e.g., usability. As such, it is first necessary to use the partial matching technique to filter the texts with the target NFR. Then, the KWIC technique selects the contexts (pre and pos) of a keyword (See Figure 43).

5) Correlations among Qualities from the Catalog

Table 18. Usability correlated qualities in issues

Quality keywords	Correlation value
cognitive	0.95
timeliness	0.95
appropriate	0.92
accessibility	0.43
usefulness	0.28
custom	0.34
security	0.13

Having the usability-related qualities elicited from a catalog (Table 17), and the locations, we performed a rank of qualities as to how related they are to the target NFR. We used a statistical measure (correlation) that indicates the adjacency among two words. The keywords with a correlation above 0.1 were selected. Table 18 shows the qualities that are closer to the usability NFR in the corpus used. Fig 41 shows the NFR security in its context.

```
 and comply with QWASP-[ x] No new | security | vulnerabilities-[ x] Internal team code revi
 and comply with QWASP-[ x] No new | security | vulnerabilities-[ x] Internal team code revi
pp, F_book and other less privacy/ | security | focused apps and plattforms, it has to be as
ut we also want to provide maximum | security | for people who depend on it with their lives
pp, F_book and other less privacy/ | security | focused apps and plattforms leak data, which
 case is:- casual user- moderately | security | concious user- tinfoilhat™/ protester in$ no
ove mentioned features, as well as | security | features like#175,#226 and#328 and future co
   UI( e.g. new user advice, tips, | security | warnings). Things like" Save/ update passwor
```

Figure 41. KWIC of security NFR[114]

6) Sentiment Analysis through Natural Language Processing (NLP)

Our strategy uses sentiments as a surrogate of interdependencies. As such, we selected the SentiStrength tool[163] used in related work. By using the locations, we used the NLP technique of POS-tagging as a means to identify the grammar type of word that we judge express a sentiment. Then, we compare with the findings of the tool.

To the best of our knowledge, SentiStrength is the only one that has been validated in various bases, that is, comments from Myspace, Last FM, YouTube,

and Flickr among others, to extract a lexicon robust enough to be applied in other social contexts. It has, an emoticon dictionary, the treatment of words that give a greater emphasis (strength), and a fixed vocabulary of negative and positive words in its *EmotionLookupTable*[172].

4.5. Assessment of NLP Approach to Identify Interdependencies

Figure 42 shows a location[173] where usability is the key NFR, and cognitive NFR is related. By POS-tagging the location we interpret the sentence. We argue that the word *challenges* coded as NNS (plural noun) and the word *not* coded as RB (adverb), are a strong indication of a negative sentiment between the two marked NFRs. If we can identify this situation by NLP based-heuristics, we may present the case (context) to the requirements engineers and suggest that usability may hurt cognitive requirements, bringing upfront a possible conflict and improving a future design decision.

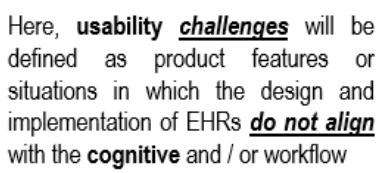


Figure 42. Usability location

Fig. 43 shows the POS-tagging performed on location. From this, we noticed that the grammatical type adverb might be the type of word that helps identifying sentiments related to qualities. As such, Table 19 summarizes the adverbs found in all usability-locations[173]. Besides, other types of words, such as modal verbs, verbs, and nouns, may be candidates to express emotions.

Here, **usability** challenges[NNS] will[MD] be[VB] defined[VVN] as[IN] product[NN] features[NNS] or[CC] situations[NNS] in[IN] which[WDT] the[DT] design[NN] and[CC] implementation[NN] of[IN] EHRs[NNS] do[VVP] not[RB] align[VV] with[IN] the[DT] **cognitive** and/or workflow requirements and preferences of users within and across professional and patient roles and settings.

Figure 43. POS-tagging of Usability location

A) Comparison with a Sentiment Analysis Tool

By using the SentiStrength tool[163], we processed all sentences from GitHub corpus to verify which words are determining whether a sentence is positive or negative according to its lexicon. Table 20 lists sentiment with their grammar type.

From Tables 19 and 20, there are words where both coincide with defining a sentiment, such as the verbs: improve, interrupt, and the noun: challenges. From our list of adverbs, none of them were considered for SentiStrength. From our list of verbs, there are also many not considered

Table 19. Type of words that may identify sentiment on GitHub issues[114]

Adverbs	Modal Verbs	Verbs	Plural Nouns
Not	should	reclaim	challenges
urgently	will	affect	effects
Very	can	deal	
however	cannot	face	
potentially	would	agree	
maybe	could	interrupt	
manually	may	depend	
		increase	
		increased	
		hide	
		improve	
		re-configure	

by SentiStrength.

From Table 20, words that SentiStrength qualified are mostly nouns, verbs, and adjectives. Our approach identified the adjectives (as a surrogate of NFRs), and we could have filtered them from the ones ending with suffixes “ity” and “ness” to have qualifier words that may express a sentiment.

Table 20. Words Qualified by SentiStrength

Negative (-1) and Positive (1)	POS-tag
WHACK [-1]	Noun
improved [1]	Verb, past participle
improve [1]	Verb, base form
challenges [-1]	Plural noun
problem [-1]	Noun
Care [1]	Proper Noun
cumbersome [-1]	Adjective
poor [-1]	Adjective
interrupt [-1]	Verb, non-3rd person singular present
lack [-1]	Noun
worthwhile [1]	Adjective
security [1]	Adjective
unpopular [-1]	Adjective
protesting [-1]	Verb, gerund or present participle
gay [-1]	Adjective
jail [-1]	noun
executed [-1]	Verb, past participle

Our NLP approach is an initial result that needs to be further explored towards the use of Sentiment Analysis to find interdependencies among NFRs. We envisage the possibility of speeding up the elicitation of possible conflicts among NFRs early on, as requirements engineers are acquainted with the domain. Finding chunks of text, where qualities are mentioned together, presents the opportunity of looking for “sentiment” in this context. This strategy has yielded early positive signs.

4.6. Chapter Summary

This chapter explained how the Sentiment Analysis could be useful as a means to identify interdependencies in unstructured texts, e.g., GitHub texts. The strategy proposed is based on SIGs as a source of quality-related keywords to be located in corpora texts. Then, by applying NLP, we identified types of part-of-speech (POS) that can express a contribution (emotion) among two NFRs. Finally, we compared our findings using POStagging with the SentiStrength tool[163].

5 Using a SIG to Map Semi-Automated NFRs elicitation

This chapter presents a SIG that organizes the knowledge gained in operationalizations used in previous chapters. We detail NFR framework notions that guided the SIG creation. Then, we map the NFR Softgoal from NFR Operationalizations of this thesis on SIG. Finally, by using a syntaxes called *pretty-print*, we describe the SIG.

The NFR framework[15] boosts the creation of catalogs to record experiences, standard techniques, development techniques, and knowledge about NFRs. Thus, creativity, which is important in developing software, may have a reliable source of information.

This work, which followed a bottom-up strategy, explored mechanisms (NFR operationalizations) to semi-automate NFRs elicitation based on the NFR framework vision. As such, from the RE point of view, we developed strategies according to the goal identified in Chapter 1, which is to balance efficacy (accurate information) and efficiency (timely information). Therefore, through this research, we perceived an NFR catalog (SIG) would help reutilization for future NFRs identification strategies based on text-mining.

The organization (chapters) of this work may lead to perceive it as a process, however, each part is an independent NFR operationalization that helps in Leite's[1] proposed elicitation tasks, with a focus on NFRs elicitation (Table 21).

Table 21. Organizing NFRs Operationalizations in Elicitation

Fact Finding	Communication	Fact Validation
Corpus creation	Query formulation	Finding Interdependencies with SA
Facts Elicitation	UofD definition	
NFRs identification		

5.1. NFR Foundations

Following, we define the NFR framework notions used to understand our modeling reasoning. In Chapter 1, some notions are defined, such as SIGs, interdependencies, *satisficing*. Following. We detail relevant notions that guided the creation of the SIG proposed.

5.1.1. NFR framework Notions

Operands and Operators

With the notation proposed, we classify nodes of a SIG graph as Operands (Table 22) and arrows as Operators (Table 23). It is important to note that operands are types of Softgoal. As such, while explaining interdependencies (operators) among nodes, we can use only the abstraction Softgoal.

Table 22. SIGs operands in the NFR Framework

Operands	Description
NFR Argumentation (Claim Softgoals)	Claims help to justify design decisions such as adding or refining a Softgoal.
NFR Softgoal	It is a quality goal to be satisficed
NFR Operationalization	It is a mechanism to satisfice an NFR Softgoal

Table 23. SIGs operators (interdependencies) in the NFR Framework

Operators	Description	
Refinements	Softgoals can be refined in three different ways:	
	Refinement Types	**Description**
	Decompositions	Any Softgoal can be decomposed in the same type of Softgoals through OR/AND operators
	NFR Operationalizations	NFR Softgoals and NFR operationalizations can be refined in NFR operationalizations
	NFR Argumentations	Softgoals can be refined in NFR argumentations
Contributions	Add expression to refinements by adding a contribution value (Fig. 46) Its reading is from the offspring to the parent node.	

Refinements Types

Show refinements of "parent" Softgoals downwards into other, more specific, "offspring" Softgoals. Refinements can be decompositions among Softgoals following the AND/OR strategy. Refinements can be operationalizations of Softgoals. Refinements can be argumentations to support/deny Softgoals or interdependencies.

Interdependencies Types

Interdependencies show the contribution (impact) of offspring Softgoals upwards upon the meeting of other (parent) Softgoal. Fig. 44 depicts the types of contributions proposed by NFR Framework[15].

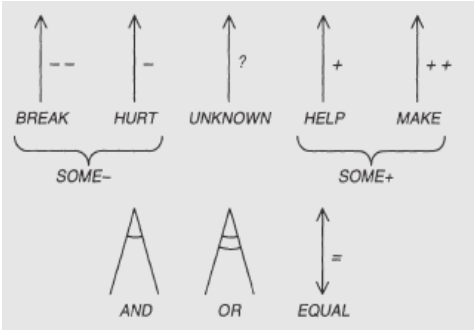


Figure 44. Contributions elements of NFR framework[15]

When all of several sub-Softgoals are needed together to meet a higher Softgoal, we say it is an AND type of contributions; then, as a group, their contribution will be to achieve a Softgoal. On the other hand, when one

or more sub-Softgoals are needed to meet a higher Softgoal together, we can use the OR type of contribution. However, when only a sub-Softgoal meets a Softgoal, its contribution can be set as equal (See Fig. 44).

5.1.2. The Dynamics of the NFR Framework

Chung et al.[15] stress that the steps to design a system by using the NFR framework are not sequential and depend on the strategy of modelers. The following steps summarize the steps for modeling NFRs.

a. Identify the top/main NFR Softgoals to be achieved. These are the goals that need to be elaborated upon[15]

    i. Softgoals have an NFR type, such as security or performance

    ii. Softgoals have a subject matter called NFR topic, e.g., accounts

b. Decomposing the NFR Softgoals into more specific sub-Softgoals

    i. Decomposing by NFR type or NFR topic

    ii. Specific sub-Softgoals which together, through decompositions or contributions, satisfice

c. Deal with ambiguities, domain information, and priorities[15]

d. Consider interdependences among Softgoals

e. Synthesize solutions (operationalizations) to build quality.

5.1.3. Organizing Softgoals

Chung et al.[15] state, "The transition from NFR Softgoals to operationalizing Softgoals is a crucial step in the process because NFRs need to be converted into something that can be implemented." The process to concretize operationalizations may not be performed in one step, it is needed for further refinements by decomposing Softgoals and measuring relationships contributions.

To this matter, Suppakul[174] defines NFRs modeling patterns to deal with the production of an NFR model properly. Among these patterns, we adopt the refinement pattern to our NFR foundations. A refinement (Table 23) can be of three types: decompositions, NFR Operationalizations, and NFR argumentations.

Fig. 45 shows the refinement of a Softgoal by using the NFR type method. That is, the Softgoal with NFR type security is decomposed in confidentiality, integrity, and availability NFR types, while the NFR Topic *System* remains. For Softgoals, such as NFR operationalizations, solutions are decomposed in other NFR Operationalizations; although there is no restriction, NFR Operationalizations does not specify a topic.

This way, using the refinement pattern, modelers can choose one solution (if OR operator) or model all solutions needed (if AND operator) satisficing a general solution. Another refinement is by NFR topic, that is, while topics are

refined, the NFR Type is maintaining. When using an NFR Topic refinement, it is important to keep the grammar used. That is, if an NFR topic is written in the infinitive form, then the topic of NFR offspring's also.

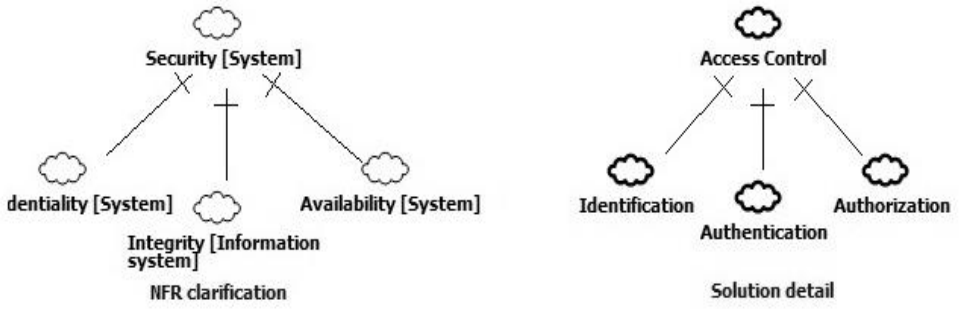


Figure 45. Softgoals Refinement Pattern[173]

5.1.4. Type of interdependencies among Softgoals.

Chung et al.[15] states, "Developers can explicitly state interdependencies among Softgoals, by using refinements." We call these **explicit interdependencies**." However, as Softgoals are antagonists, e.g., having more transparency may impact security, the NFR framework defines that **implicit interdependencies** occur when achieving one Softgoal impact in another Softgoal achievement. Then, while explicit interdependencies are shown with a solid line, implicit ones use dashed lines.

Therefore, the reuse of catalogs is encouraged to find correlations (implicit interdependencies). Chung et al states[15], "implicit interdependencies can be detected as the graph is being developed […] by consulting catalogs (correlation catalogs) of positive and negative interdependencies among Softgoals".

5.2. SIG for a Semi-Automated Approach to Elicit NFRs

When developing software, we do not start from scratch; we are reusing different artifacts throughout the construction of the software, from code to models, among other artifacts. As such, as stated by Leite et al.[39] when we search for what to reuse, we are used to searching for functionality. Although searching by quality may be more complicated given its cross-cutting nature, the catalogs can facilitate this searching, as their interdependencies provide the links to related NFRs.

In this regard, as we perceived the growth of AI strategies for RE, we consider to creating a catalog that provides knowledge about qualities that are needed to satisfice a Softgoal such as "Speed-up."

Two NFRs, effectivity and efficiency, constrain our approach for NFRs elicitation. As such, In the SIG (Fig. 46), we organize NFR Softgoals elicited from the NFR operationalizations presented in this thesis (Table 24). The SIG only shows NFR Softgoals as a means to be reusable. Stakeholders are free to choose different NFR operationalizations to meet NFR Softgoals.

We produced the SIG during a year while understanding better the NFR framework, and validating this artifact with different viewpoints such as our research group and a researcher with experience in the NFR framework.

5.2.1. Balanced NFRs Elicitation SIG

From left to right, the SIG in Fig. 46 intends to express an order needed to satisfice a balanced NFR elicitation. To the left, are the qualities that satisfice efficacy. To the right, the ones that satisfice efficiency. The NFR Softgoals were decomposed following the refinement patterns.

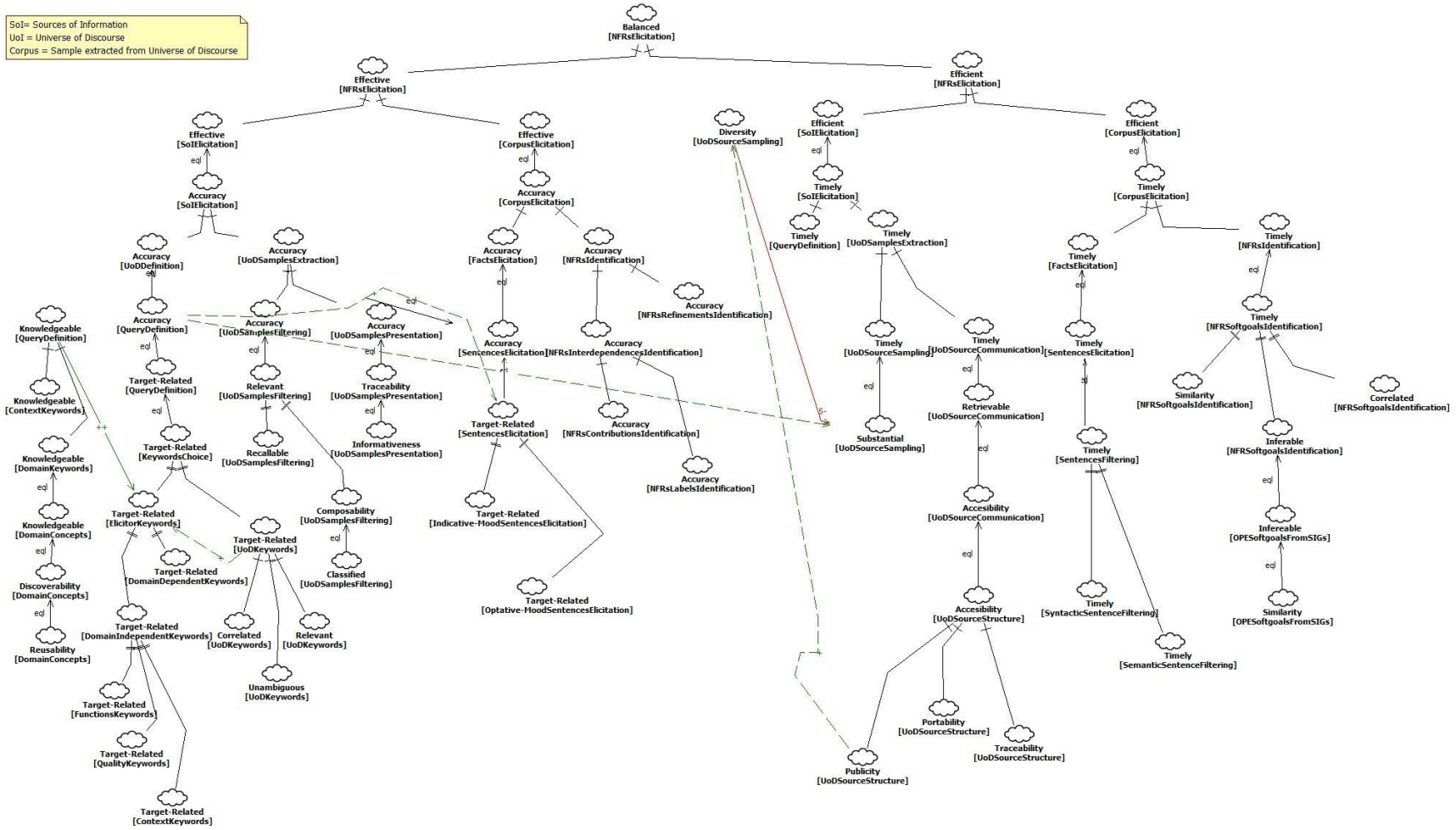


Figure 46. NFR Softgoals related to a balanced elicitation of NFRs

Table 24. NFRs identified in Operationalizations Used

		Operationalizations / NFRSoftgoal	**Querying** - Exact/partial Matching - Wikidata - SIGs	**Corpus Creation** - GitHub API - Corpus Builder	**Corpus Creation** - GH4RE - NLP	**Optative-Mood Sentences** - SVM - Boilerplates - N-Fold	**Indicative-Mood Sentences** - WordNet - NLP	**NFRs Classification** - NFRFinder - SIGs - NLP	**Interdependencies** - SA - SIGs
Balanced [NFRs Elicitation]	Effective[NFRsElicitation]	Effective [SoIElicitation]	x	x					
		Accuracy [SoIElicitation]	x	x					
		Accuracy [UofDDefinition]	x	x					
		Accuracy[QueryDefinition]	x	x	x			x	
		Target-Related[QueryDefinition]	x		x				
		Target-Related[KeywordsChoice]	x		x			x	
		Target-Related[ElicitorKeywords]	x						
		Target-Related[DomainDependetKeywords]	x		x				
		Target-Related[DomainIndependetKeywords]	x		x				
		Target-Related[FunctionKeywords]	x		x				
		Target-Related[QualityKeywords]	x		x				
		Target-Related[ContextKeywords]	x		x		x		
		Target-Related[UofDKeywords]	x		x				
		Correlated[UofDKeywords]							x
		Unambiguos[UofDKeywords]			x				
		Relevant[UofDKeywords]	x						
		Accuracy[UofDSamplesExtraction]		x	x				
		Accuracy[UofDSampleSiltering]		x	x				
		Relevant[UofDSamplesFiltering]		x					
		Recallable[UofDSamplesFiltering]		x					
		Composabilty[UofDSamplesFiltering]		x					
		Classified[UofDSamplesFiltering]		x		x			
		Accuracy[UofDSamplesPresentation]		x		x			
		Traceability[UofDSamplesPresentation]		x		x			
		Informativeness[UofDSamplesPresentation]		x		x			
		Effective [CorpusElicitation]	x	x					
		Accuracy [CorpusElicitation]			x				
		Accuracy [FactsElicitation]				x			
		Accuracy [SentencesElicitation]			x	x			
		Target-Related[SentencesElicitation]			x	x			
		Target-Related[Indicative-MoodSentencesElicitation]			x	x	x		
		Target-Related[Optative-MoodSentencesElicitation				x			

		Operationalizations / NFRSoftgoal	**Querying** - Exact/partial Matching - Wikidata - SIGs	**Corpus Creation** - GitHub API - Corpus Builder	**Corpus Creation** - GH4RE - NLP	**Optative-Mood Sentences** - SVM - Boilerplates - N-Fold	**Indicative-Mood Sentences** - WordNet - NLP	**NFRs Classification** - NFRFinder - SIGs - NLP	**Interdependencies** - SA - SIGs
Balanced [NFRs Elicitation]		Accuracy [NFRsIdentification]						X	x
		Accuracy [NFRsInterdependenciesIdentification]							x
		Accuracy [NFRsIContributionsIdentification]							x
		Accuracy [NFRsLabelsIdentification]							x
		Accuracy [NFRsRefinementsIdentification]						X	x
	Efficient [NFRsElicitation]	Efficient [SoIElicitation]		x					
		Timely[SoIElicitation]		x					
		Timely[QueryDefinition]	x						
		Timely[UofDSamplesExtraction]	x	x	x				
		Timely[UofDSourceSampling]		x	x				
		Substantial[UofDSourceSampling]		x					
		Timely[UofDSourceCommunication]		x					
		Retrievable[UofDSourceCommunication]		x					
		Accesibility[UofDSourceCommunication]		x					
		Accessibility[UofDSourcesStructure]		x		x			
		Publicity[UofDSourceStructure]				x			
		Portability[UofDSourceStructure]				x			
		Traceability[UofDSourceStructure]		x		x			
		Efficient[CorpusElicitation]	x	x	x				
		Timely[CorpusElicitation]	x	x	x				
		Timely[FactsElicitation]				x	x		
		Timely[SentencesElicitation]				x	x		
		Timely[SentencesFiltering]				x	x	x	
		Timely[SyntacticSentencesFiltering]	x			x		x	
		Timely[SemanticSentencesFiltering]	x			x	x	x	
		Timely[NFRsIdentification]				x		x	x
		Timely[NFRsSoftgoalsIdentification]						x	
		Similarity[NFRsSoftgoalsIdentification]						x	
		Infereable[NFRsSoftgoalsIdentification]						x	
		Correlated[OPESoftgoalsIdentification]							x
		Similarity[OPESoftgoalsFromSIGs]						x	
		Correlated[NFRSoftgoalsIdentification]							x
		Knowledgeable[QueryDefinition]	x						
		Knowledgeable[ContextKeywords]	x						
		Knowledgeable[DomainKeywords]	x						
		Knowledgeable[DomainConcepts]	x						

		Operationalizations / NFRSoftgoal	**Querying** -Exact/partial Matching -Wikidata -SIGs	**Corpus Creation** - GitHub API - Corpus Builder	**Corpus Creation** - GH4RE - NLP	**Optative-M Sentences** - SVM - Boilerplates - N-Fold	**Indicative-M Sentences** - WordNet - NLP	**NFRs Classification** - NFRFinder - SIGs - NLP	**Interdependencies** - SA - SIGs
		Discoverability[DomainConcepts]	x				x		
		Reusability[DomainConcepts]	x					x	x
		Diversity[UofDSourceSampling]			x	x			

5.2.2. SIG Pretty-prints

We use a pretty-print[13] format to explain the SIG. As such, we defined the following syntax:

i) The "achievement" of a softgoal is expressed as satisficing or satisficing. The morphology of satisficing varies according to what allows a better reading of a phrase.

ii) Some phrases use <are + Softgoal> when satisficing diminish the meaning of phrase.

iii) NFR Type of Softgoals are in bold. Its morphology also can vary.

iv) NFR Topic of Softgoals are in brackets

v) Refinement-decompositions are in capital letters.

vi) Explicit interdependencies vary according its contribution. If (+), then positively impacts; if (-), then negatively impacts.

vii) Implicit interdependencies vary according its contribution. If (+), then positively correlates; if (-), then negatively correlates.

[13] Prettyprint (or pretty-print) is the application of any of various stylistic formatting conventions with indent style and syntax highlighting. Source: Wikipedia

A **Better** [NFRs Elicitation] is satisficed if **Effective** [NFRsElicitation] AND **Efficient** [NFRsElicitation] is satisficed

Effective [NFRsElicitation] is satisficed if the [Sources of Information Elicitation] AND the [Corpus Elicitaion] are **Effective**

Effective [Sources of Information Elicitation] is satisficed if it has **Accuracy**

Accuracy in [Sources of Information Elicitation] is satisficed if the [Universe of Discourse Definition] AND the [Universe of Discourse Samples Extraction] have **Accuracy**.

The [Universe of Discourse Definition] has **Accuracy** if the [Query Definition] has **Accuracy**

Accuracy in [Query Definition] is satisficed if it is **Target-Related**

The [Query Definition] is **Target-related** if [Keywords choice] is **Target-related**

Target-related [Keywords choice] is satisficed if [Elicitor Keywords] OR [Universe of Discourse Keywords] are **Target-related**

Target-related [Elicitor Keywords] is satisfice if [Domain-Dependent Keywords] OR [Domain-Independent Keyword] are **Target-related**

Target-related [Domain-Independent Keyword] is satisficed if [Functions-Related Keywords] OR [Quality-Related Keywords] OR [Context-related Keywords] are **Target-related**

Target-related [Universe of Discourse Keywords] is satisficed if they are **Correlated** AND **Unambiguous** AND **Relevant**

Accuracy in [Universe of Discourse Samples Extraction] is satisficed if [Universe of Discourse Samples Filtering] AND [Universe of Discourse Presentation] satisfice **Accuracy**

Accuracy in **[**Universe of Discourse Samples Filtering**]** is satisficed if it satisfices **Relevance**

Relevance of **[**Universe of Discourse Samples Filtering] is satisficed if **[**Universe of Discourse Samples Filtering] is **Recallable** OR has **Composability**

Composability of [Universe of Discourse Samples Filtering] is satisficed if [Universe of Discourse Samples Filtering] is **Classified**

Accuracy in [Universe of Discourse Presentation] is satisficed if **Traceability** of [Universe of Discourse Presentation] is satisficed.

Traceability of [Universe of Discourse Presentation] is satisficed if **Informativeness** of [Universe of Discourse Presentation] is satisficed.

Effective [Corpus Elicitation] is satisficed if it has **Accuracy**

Accuracy in [Corpus Elicitation] is satisficed if [Facts Elicitation] AND [NFRs Identification] has **Accuracy**

[Facts Elicitation] has **Accuracy** if [Sentences Elicitation] satisfices **Accuracy**

Accuracy in [Sentences Elicitation] is satisficed if [Sentences Elicitation] is **Target-Related**

Target-Related [Sentences Elicitation] is satisficed if [Indicative-Mood Sentences Elicitation] AND [Optative-Mood Sentences Elicitation] are Target-Related.

Accuracy in [NFRs Identification] is satisficed if [NFRs Interdependencies Identification] AND [NFRs Refinement Identification] satisfices **Accuracy**

Accuracy of [NFRs Interdependencies Identification] is satisficed if [NFRs Contributions Identification] AND [NFRs labels Identification] satisfices **Accuracy**

Efficient [NFRsElicitation] is satisficed if the [Sources of Information Elicitation] AND the [Corpus Elicitation] are **Efficient**

An **Efficient** [Sources of Information Elicitation] is satisficed if it is **Timely**

A **Timely** [Sources of Information Elicitation] is satisficed if [Query Definition] AND [UofD Samples Extraction] is **Timely**

A **Timely** [Universe of Discourse Samples Extraction] is satisficed when [Universe of Discourse of Source Sampling] AND [Universe of Discourse Source Communication] are **Timely**

A **Timely** [Universe of Discourse of Source Sampling] is satisficed when the [Universe of Discourse of Source Sampling] is **Substantial**

A **Timely** [Universe of Discourse Source Communication] is satisficed when it is **Retrievable**

A [Universe of Discourse Source Communication] is Retrievable if it has **Accessibility**

The **Accessibility of** [Universe of Discourse Source Communication] is satisficed when the [Universe of Discourse Source Structure] has **Accessibility**

The **Accessibility of** [Universe of Discourse Source Structures] is satisficed if it satisfices **Publicity, Portability, Traceability**

An **Efficient** [Corpus Elicitation] is satisficed if it is **Timely**

A **Timely [**Corpus Elicitation] is satisficed if [Facts Elicitation] AND [NFRs Identification] are **Timely**

A **Timely** [Facts Elicitation] is satisficed if the [Sentences Elicitation] is **Timely**

A **Timely** [Sentences Elicitation] is satisficed If [Sentences Filtering] satisfices **Timely**

Timely [Sentences Filtering] is satisficed if a [Syntactic Sentences Filtering] and a [Semantic Sentences Filtering] satisfices **Timely**

A **Timely** [NFRs Identification] is satisficed if the [NFRs Softgoal Identification] is **Timely**

A **Timely** [NFRs Softgoal Identification] is satisficed when it satisfices **Similarity**, are **Infereable** and are **Correlated**

[NFRs Softgoal Identification] are **Infereable** if [Operationalization Softgoal from SIGs] are **Infereable**

[Operationalization Softgoal from SIGs] are **Infereable** if they satisfices **Similarity**

Following, two NFR Softgoals, Knowledgeable and Diversity, are not the decomposition of any NFR Softgoal. However, they impact with implicit (dashed lines) and explicit interdependencies (solid lines).

A **Knowledgeable** [Query Definition] is satisficed when [Context Keywords] AND [Domain Keywords] are Knowledgeable

Knowledgeable [Domain keywords] are satisficed when [Domain Concepts] are **Knowledgeable**

Knowledgeable [Domain Concepts] are satisficed when [Domain Concepts] satisfice **Discoverability**

Discoverability [Domain Concepts] are satisficed when [Domain Concepts] satisfice **Reusability**

The implicit and explicit interdependencies in the model are the following:

A satisficed **Knowledgeable** [Query Definition] positively impacts **Target-Related** [Keywords from Elicitor] being satisficed.

A satisficed **Target-Related** [Universe of Discourse Keywords] positively correlates that **Target-Related** [Keywords from Elicitor] being satisficed.

A satisficed **Accuracy** in [Query Definition] positively correlates in **Substantial** [Universe of Discourse Sampling] being satisficed.

A satisficed **Accuracy** in [Query Definition] positively correlates in **Target-Related** [Sentences Elicitation] being satisficed.

A satisficed **Diversity** in [Universe of Discourse Sampling] negatively impacts in **Substantial** [Universe of Discourse Sampling] being satisficed

A satisficed **Publicity** of [Universe of Discourse Structure] positively correlates in **Diversity** of [Universe of Discourse Sampling]

5.2.3. Reusing NFRs in SIG catalog

Our thesis has performed NFR Operationalizations towards the semi-automated elicitation of NFRs (Chapter from 2-4). From them we have elicited NFRs Softgoal needed to get a balanced NFRs elicitation (SIG in Fig. 46). Both, NFR operationalizations and NFR Softgoal are mapped in Matrix of Table 24.

As SIG contribution is its reusability, we show in Fig. 47 and Fig. 48, how this can be performed with two operationalizations used in this thesis, Wikidata and Wikifier. We took one NFRTopic: Sources of Information (SoI) Elicitation to verify if this can be satisficed balancing efficacy and efficiency.

One of the paths to get The Effective [SoIElicitation] and Efficient [SoIElicitation] in Fig. 47, is by satisficing a Query Definition. In Section 2.2.3 we have shown a knowledge-based approach helps in improving Querying. As such, from our developing experience, we found that by using only the Wikidata Operationalization we cannot satisfice the Timely required in a [QueryDefinition]. But when we found an alternative operationalization such as Wikifier, we retrieved concepts faster. With the latter operationalization, Fig. 48 shows a Balanced SoI Elicitation.

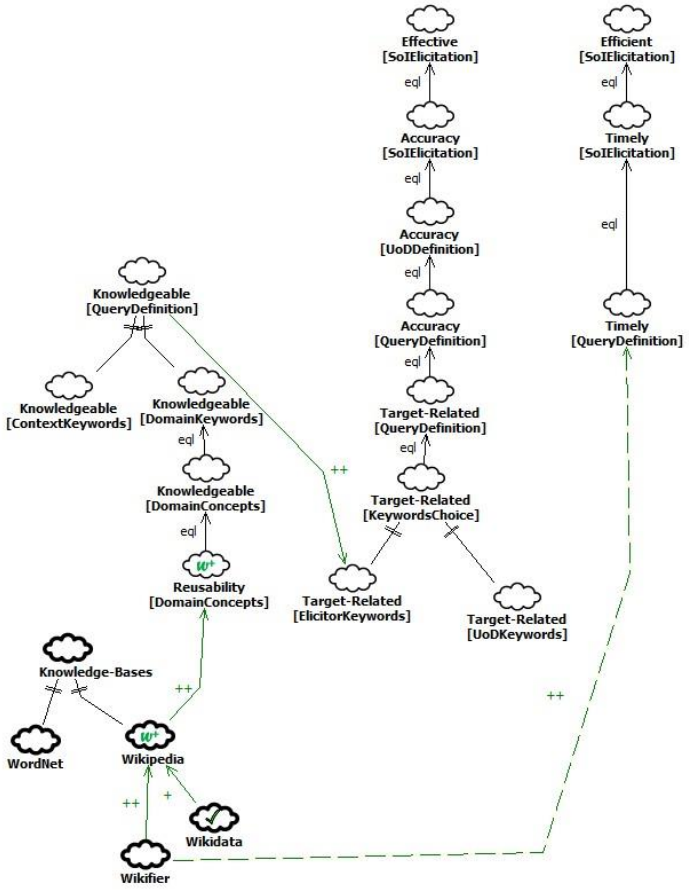


Figure 47. Unbalanced SoI Elicitation

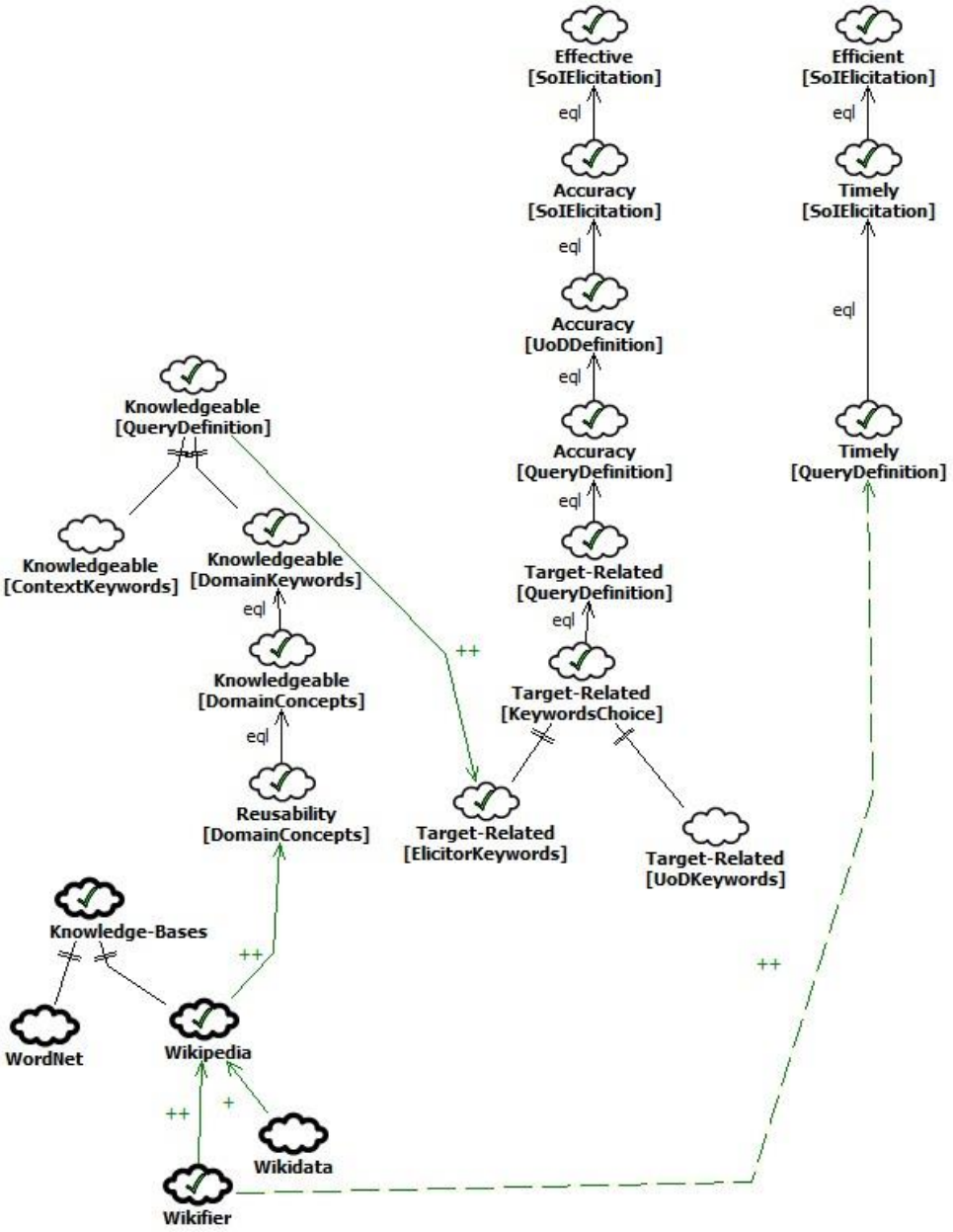


Figure 48. Balanced SoI Elicitation

5.3. Chapter Summary

We defined the notions of the NFR Framework needed to understand the SIG we produced for reuse of strategies to semi-automate NFRs elicitation. Using an operationalizations we have described in this thesis, Querying Assistant, we showed how the reuse of SIG can be performed. Also, a matrix explains how the NFRs are related to NFR Softgoals. Finally, a description of the SIG is performed by using a pretty-print syntax.

6 Conclusion

This chapter summarizes the thesis, stressing the contributions and pointing out future work as well as limitations of the work.

6.1. Contributions

This work has made an effort to speed-up NFRs elicitation by performing a qualitative bottom-up research, which, according to the NFR Framework, to meet a Softgoal one can start by exploring NFRs Operationalizations as means of finding design alternatives. After producing different ways to achieve Speed-Up in text-mining RE information, we wrapped up the acquired knowledge using the NFR framework. By doing this, we modeled this knowledge as a catalogue (SIG). The resulting SIG should serve as a basis in the pursuit of quality reuse.

The main research question of this thesis is: *How can the semi-automation of NFRs elicitation be speeded up by achieving a trade-off between efficiency and effectiveness?* This work contributes with a SIG that organizes knowledge about qualities that are key for semi-automated NFRs elicitation. This SIG shows interdependencies among qualities, which, if satisficed, can satisfice the main Softgoal.

Three Sub-goals were set up. The first one, *how elicitors can be helped to improve the quality of their inputs on a semi-automation of elicitation?* Is answered with the strategies proposed for enhancing the querying of sources of information such as GitHub. This research identified how critical, to efficiency, is an elicitor that lacks knowledge of a domain. We have shown that keywords (inputs) depend on the discourse of a corpus; as such, we developed semantic and syntactic approaches to identify corpus-relevant keywords.

The second sub-question, *what strategies of fact-finding can speed up NFRs identification?* Is answered by decomposing the problem in types of facts, such as requirement statements and requirements classifications. To the first type, we found in Zave and Jackson's work that sentences can have an optative-mood (desires information) and indicative-mood (domain information). To the latter, we built a strategy to classify sentences, either are NFRs or Not. Another contribution is the Gold Standard developed, which showed that Gold standards for NFRs are not accurate, as this is a qualitative task.

The third sub-question. *How NFRs can be elicited so that they can facilitate validation?* is partially answered by the Sentiment Analysis strategy. That is, by using locations of two related NFRs, it can be possible to find their independencies, with one or more contributions (emotions as a surrogate of contributions in Fig. 44). The locations of NFRs facilitate their validation by providing a context. The use of knowledge-bases (SIGs) also facilitates the validation as this knowledge has an NFR type and an NFR Topic, which can give

more precision eliciting locations. However, the heuristics were not assessed by elicitors.

6.2. Future Work

We suggest improvements to the following Operationalizations performed:

Querying Sources of Information. There is a need to explore more reliable data existing in Knowledge bases such as Wikipedia. We grasped this type of data by using the service Wikifier.org; however, we have not used it with a mass of data, such as many Wikipedia articles, or other domain-related texts of the same domain, to filter relevant concepts. Also, it is needed strategies to compose queries that allows retrieving recallable data from sources of information.

Corpus Creation. The first step for giving efficiency to this task was done by developing a tool[79] that retrieves data from GitHub. However, still, the corpus is presenting imbalanced data[175]; that is, the samples of a class are not equal; then, the ML approaches may diminish its accuracy.

The Finding of Optative-mood Sentences. The strategy presented, still needs to be compared with other models, and strategies, e.g., word embedding's. On the other hand, we need to verify a ML approach[176] that surpassed the performance on NFRFinder. The mentioned work is interesting as it uses unstructured scenarios, similar to GitHub issues, to classify NFRs as well as identify NFR-related-keywords.

The Finding of Indicative-mood Sentences. This task is related to querying strategies; then, both may be beneficial in eliciting more domain information.

NFRs classification. We developed NFRFinder, which its heuristics were tested in structured texts of a Gold Standard. Future work should check unstructured data, i.e., requirement-related texts.

Sentiment Analysis (SA). Our approach was based on non-software engineering texts. After the publication, this technique has been improved with and for SE texts. Future work should verify our approach against the new tools for SA, as well to assess our strategy with stakeholders.

SIG for NFRs elicitation. Still, the SIG can explore the propagation procedure[15,] as seen in Fig. 22. Fig. 47, and Fig. 48. As such, we need to use more operationalizations to verify the degree of achievement on NFR Softgoals.

Out of the operationalizations used, we found the main future work is finding strategies to cope with Search Space. That is, elicitors know little about what they are looking for, so the search expression in an automated context is in itself an obstacle to what they will have as a return. This problem of tuning the search space is more difficult when the interest is to search for NFRs, considering two factors: subjectivity and transversality. NFRs must be identified in a broad spectrum because NFRs are transversal.

Thus, the delimitation of search spaces becomes spaces of uncertainty, because what is recovered depends on the search expression (query in an automated context). We need strategies that allow reducing such uncertainty helping in the creation of queries that allow the coverage of relevant information in a search space.

6.3. Limitations

It worth pointing out limitations of the work. We present them per each part of the speeding up strategy.

Corpus Creation is unstable due to GitHub policies of its API that lately deprecated a component (Typhoeus[78]) we used in the corpus builder architecture. Also, querying GitHub is limited by the lack of phrase queries in its search engine.

Querying Sources of Information has limitations in the knowledge bases used which are efficient when queries are about stable domains. Also, retrieving concepts can be ineffective if the **concept seed** is not well formulated.

The Finding of Optative-mood Sentences is limited to the patterns used which are based in functional requirements sentences e.g. “should be able”, “to allow”. Using qualifiers directly in raw data may have brought other facts, probably not requirement-related sentences but with chances of obtaining NFRs.

The Finding of Indicative-mood Sentences. Our heuristic, based in POS-tagging, has used proper nouns to anchor domain information. The application of ontologies may have better results in stable domains.

NFRs classification relies on SIGs to identify qualifiers in texts. However, as mentioned with taxonomies, it can restrain the relevance of other texts such as numeric characters (not identified by a POS-tagger) that express some degree of NFRs.

Sentiment Analysis (SA) is limited to interdependencies that can be expressed with contribution value (Fig. 44). Exist other types of interdependencies such as the decomposition OR/AND in NFR graphs that do not have a description to discover its relation.

The SIG for NFRs elicitation is limited to the NFRs discovered while implementing the NFR operationalizations.

One of the limitations for the replicability of the operationalizations discovered is the instability of GitHub data. Notwithstanding we provided the corpora used in each operationalization[91,135,140,173].

References

1 LEITE, J.C.S.P. **Viewpoint Resolution in Requirements Elicitation**. University of California, Irvine, 1988. Available at http://www-di.inf.puc-rio.br/~julio/Viewpoint.pdf.

2 METH, H., BRHEL, M., AND MAEDCHE, A., 2013. **The state of the art in automated requirements elicitation**. Information and Software Technology, 55(10), pp.1695-1709.

3 KAIYA, H. AND SAEKI, M., 2006, September. **Using domain ontology as domain knowledge for requirements elicitation.** In 14th IEEE International Requirements Engineering Conference (RE'06) (pp. 189-198). IEEE.

4 LEE, Y. AND ZHAO, W., 2006, May. **Domain requirements elicitation and analysis-an ontology-based approach**. In International Conference on Computational Science (pp. 805-813). Springer, Berlin, Heidelberg.

5 CLELAND-HUANG, J., SETTIMI, R., ZOU, X. AND SOLC, P., 2007. **Automated classification of non-functional requirements**. Requirements Engineering, 12(2), pp.103-120.

6 CASAMAYOR, A., GODOY, D. AND CAMPO, M., 2010. **Identification of non-functional requirements in textual specifications**: A semi-supervised learning approach. Information and Software Technology, 52(4), pp.436-445.

7 SLANKAS, J. AND WILLIAMS, L., 2013, May. **Automated extraction of non-functional requirements in available documentation.** In 2013 1st International Workshop on Natural Language Analysis in Software Engineering (NaturaLiSE) (pp. 9-16). IEEE.

8 LU, M. AND LIANG, P., 2017, June. **Automatic classification of non-functional requirements from augmented app user reviews**. In Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering (pp. 344-353).

9 JHA, N. & MAHMOUD, A., 2019. **Mining non-functional requirements from App store reviews.** Empirical Software Engineering, 24(6), pp.3659-3695.

10 CYSNEIROS, L.M. AND YU, E., 2004. **Non-functional requirements elicitation**. In Perspectives on software requirements (pp. 115-138). Springer, Boston, MA.

11 CHUNG, L. & DO LEITE, J.C.S.P, 2009. **On non-functional requirements in software engineering. In Conceptual Modeling: Foundations and applications** (pp. 363-379). Springer, Berlin, Heidelberg

12 YEH, R.T., ZAVE, P., CONN, A.P. AND COLE JR, G.E., 1980. **Software Requirements: A Report on the State of the Art**. Maryland Univ College Park Dept Of Computer Science.

13 NUSEIBEH, B. AND EASTERBROOK, S., 2000, May. **Requirements engineering a roadmap.** In Proceedings of the Conference on the Future of Software Engineering (pp. 35-46). ACM.

14 LEITE, J.C.S.P. **Livro Vivo: Engenharia de Requisitos**. 1994. Disponível em:http://livrodeengenhariaderequisitos.googlepages.com/ERNOTASDEAULA.pdf. Último acesso: 04-04-2016.

15 CHUNG, L., NIXON, B.A., YU, E., and Mylopoulos, J., 2000. **Non-functional requirements in software engineering.** Springer Science & Business Media New York

16 MYLOPOULOS, J., CHUNG, L., AND NIXON, B., 1992. **Representing and using nonfunctional requirements: A process-oriented approach**. IEEE Transactions on software engineering, 18(6), pp.483-497

17 K. L. CHUNG, **Representing and Using Non-Functional Requirements: A Process-Oriented Approach**. Ph.D. Thesis, Dept. of Comp. Sci., Univ. of Toronto, June 1993. Also Tech. Rep. DKBS-TR-93-1

18 CHUNG, L., AND NIXON, B.A., 1995, April. **Dealing with non-functional requirements: three experimental studies of a process-oriented approach**. In Proceedings of the 17th international conference on Software engineering (pp. 25-37). ACM.

19 GOLDIN, L. AND BERRY, D.M., 1997. **AbstFinder, a prototype natural language text abstraction finder for use in requirements elicitation**. Automated Software Engineering, 4(4), pp.375-412

20 GEORGE, B., BOHNER, S. A., & PRIETO-DIAZ, R. (2004, April). **Software information leaks: a complexity perspective.** In Proceedings. Ninth IEEE International Conference on Engineering of Complex Computer Systems (pp. 239-248). IEEE. IEEE

21 ANDREOU, A.S., 2003. **Promoting software quality through a human, social and organizational requirements elicitation process.** Requirements Engineering, 8(2), pp.85-101

22 LARBURU, N., BULTS, R.G., HERMENS, H.J. & NAPOLITANO, C., 2013, July. **Early phase telemedicine requirements elicitation in collaboration with medical practitioners.** In Requirements Engineering Conference (RE), 2013 21st IEEE International (pp. 273-278). IEEE.

23 EBERLEIN, A. & LEITE, J.C.S.P., 2002, September. **Agile requirements definition: A view from requirements engineering.** In Proceedings of the International Workshop on Time-Constrained Requirements Engineering (TCRE'02) (pp. 4-8).

24 PORTUGAL, R.L.Q., DO PRADO LEITE, J.C.S. & ALMENTERO, E., 2015, August. **Time-constrained requirements elicitation: reusing GitHub content.** In Just-In-Time Requirements Engineering (JITRE), 2015 IEEE Workshop on (pp. 5-8). IEEE

25 BAVOTA, G., 2016, March. **Mining unstructured data in software repositories: Current and future trends**. In 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER) (Vol. 5, pp. 1-12). IEEE.

26 GLINZ, M., 2007, October. **On non-functional requirements.** In 15th IEEE International Requirements Engineering Conference (RE 2007) (pp. 21-26). IEEE.

27 RATURI, A., PENZENSTADLER, B., TOMLINSON, B. AND RICHARDSON, D., 2014, June. **Developing a sustainability non-functional requirements framework.** In Proceedings of the 3rd International Workshop on Green and Sustainable Software (pp. 1-8). ACM.

28 KRUCHTEN, P., 2004. **The rational unified process: an introduction**. Addison-Wesley Professional.

29 BECK, K., 2000. **Extreme programming explained: embrace change**. Addison-wesley professional.

30 LEITE, J.C.S.P., 2001. **Extreme requirements.** Keynote at the Jornadas de Ingeniería de Requisitos Aplicadas, Sevilha, June, 2001

31 AZIZ, Y., AZIZ, T., MALIK, M.I., BAIG, M.K., ALI, M.Z. AND BAQER, M., 2017. **Non Functional Requirement in Agile Software Development. University of Engineering and Technology Taxila**. Technical Journal, 22(1), p.107.

32 RAMOS, F., PEDRO, A., CESAR, M., COSTA, A., PERKUSICH, M., ALMEIDA, H. AND PERKUSICH, A., **Evaluating Software Developers' Acceptance of a Tool for Supporting Agile Non-Functional Requirement Elicitation.** SEKEDOI reference number: 10.18293/SEKE2019-1

33 AMORNDETTAWIN, M. AND SENIVONGSE, T., 2019, December. **Non-functional Requirement Patterns for Agile Software Development.** In Proceedings of the 2019 3rd International Conference on Software and e-Business (pp. 66-74).

34 LIU, Y., LIU, L., LIU, H. AND LI, S., 2019. **Information Recommendation Based on Domain Knowledge in App Descriptions for Improving the Quality of Requirements**. IEEE Access, 7, pp.9501-9514.

35 KULESZA, U., SOARES, S., CHAVEZ, C., CASTOR, F., BORBA, P., LUCENA, C., MASIERO, P., SANT'ANNA, C., FERRARI, F., ALVES, V. AND COELHO, R., 2013. **The crosscutting impact of the AOSD Brazilian research community.** Journal of Systems and Software, 86(4), pp.905-933.

36 DA SILVA, L.F., 2006. **Uma Estratégia Orientada a Aspectos para a Modelagem de Requisitos** (Doctoral dissertation, Tese de Doutorado, Computer Science Department, PUC-Rio).

37 MAIRIZA, D., ZOWGHI, D. AND NURMULIANI, N., 2010, March. **An investigation into the notion of non-functional requirements.** In Proceedings of the 2010 ACM Symposium on Applied Computing (pp. 311-317).

38 BØEGH, J., 2008. **A new standard for quality requirements**. IEEE software, (2), pp.57-63.

39 LEITE, J.C.S.P, YU, Y., LIU, L., ERIC, S.K. AND MYLOPOULOS, J., 2005, June. **Quality-based software reuse**. In International Conference on Advanced Information Systems Engineering (pp. 535-550). Springer, Berlin, Heidelberg.

40 ZAVE, P. AND JACKSON, M., 1997. **Four dark corners of requirements engineering.** ACM transactions on Software Engineering and Methodology (TOSEM), 6(1), pp.1-30.

41 LIAN, X., RAHIMI, M., CLELAND-HUANG, J., ZHANG, L., FERRAI, R., & SMITH, M. (2016, September). **Mining requirements knowledge from collections of domain documents.** In 2016 IEEE 24th International Requirements Engineering Conference (RE) (pp. 156-165). IEEE..

42 SALO, R. **A guideline for requirements management in GitHub with lean approach**. Master's Dissertation. University of Tampere. School of Information Sciences (Computer Science). Tampere, Finland, 2014.

43 HUQ, S.F., SADIQ, A.Z. AND SAKIB, K., 2019, December. **Understanding the Effect of Developer Sentiment on Fix-Inducing Changes: An Exploratory Study on GitHub Pull Requests**. In 2019 26th Asia-Pacific Software Engineering Conference (APSEC) (pp. 514-521). IEEE.

44 DABBISH, L., STUART, C., TSAY, J. AND HERBSLEB, J., 2012, February. **Social coding in GitHub: transparency and collaboration in an open software repository**. In Proceedings of the ACM 2012 conference on computer supported cooperative work (pp. 1277-1286).

45 KALLIAMVAKOU, E., GOUSIOS, G., BLINCOE, K., SINGER, L., GERMAN, D.M. AND DAMIAN, D., 2014, May. **The promises and perils of mining GitHub.** In Proceedings of the 11th working conference on mining software repositories (pp. 92-101).

46 PORTUGAL, R.L.Q. AND DO PRADO LEITE, J.C.S., 2015. **Mineração de informações no Ecossistema Github para apoiar à Elicitação de Requisitos**. WTDSoft 2015, p.26

47 SUROWIECKI J. **The wisdom of crowds**. Anchor; 2005.

48 ROBERTSON S. and ROBERTSON J. (1999) **Volere Requirements Specification Template**. In Mastering the Requirements Process, p 353–391, ACM Press/Addison-Wesley Publishing Co, New York, NY, USA

49 ENGIEL, P., LEITE, J.C.S.P, AND MYLOPOULOS, J., 2017, May. **A tool-supported compliance process for software systems**. In 2017 11th International Conference on Research Challenges in Information Science (RCIS) (pp. 66-76). IEEE.

50 LEITE, J. C. S. P. **Livro Vivo: Engenharia de Requisitos**. 1994. Available at http://livrodeengenhariaderequisitos.googlepages.com/ERNOTASDEAULA.pdf. Último acesso: 16-02-2020.

51 PORTUGAL, R.L.Q., ENGIEL, P., ROQUE, H. AND LEITE, J.C.S.P, 2017, September. **Is There a Demand of Software Transparency?** In Proceedings of the 31st Brazilian Symposium on Software Engineering (pp. 204-213).

52 SINCLAIR, J. 2005. **Corpus and Text - Basic Principles in Developing Linguistic Corpora: a Guide to Good Practice**. Appendix: How to build a Corpus. Oxford-Oxbow Books.

53 HELMING, J., NARAYAN, N., ARNDT, H., KOEGEL, M. AND MAALEJ, W. **From informal project management artifacts to formal system models**. System 2010

54 HOTHO, A., NÜRNBERGER, A. AND PAAß, G., 2005, May. **A brief survey of text mining**. In *Ldv Forum* (Vol. 20, No. 1, pp. 19-62).

55 MYLOPOULOS, J., CHUNG, L. AND YU, E., 1999. **From object-oriented to goal-oriented requirements analysis.** Communications of the ACM, 42(1), pp.31-37

56 CLELAND-HUANG, J., SETTIMI, R., ZOU, X. AND SOLC, P., 2007. **Automated classification of non-functional requirements**. Requirements Engineering, 12(2), pp.103-120.

57 ZHANG, W., YANG, Y., WANG, Q. AND SHU, F., 2011, December. **An empirical study on classification of non-functional requirements.** In The twenty-third international conference on software engineering and knowledge engineering (SEKE 2011) (pp. 190-195).

58 RAHIMI, M., MIRAKHORLI, M. AND CLELAND-HUANG, J., 2014, August. **Automated extraction and visualization of quality concerns from requirements specifications**. In 2014 IEEE 22nd international requirements engineering conference (RE) (pp. 253-262). IEEE.

59 MUNAIAH, N., MENEELY, A. AND MURUKANNAIAH, P.K., 2017, September. **A domain-independent model for identifying security requirements.** In 2017 IEEE 25th International Requirements Engineering Conference (RE) (pp. 506-511). IEEE.

60 LI, T., 2017, December. **Identifying security requirements based on linguistic analysis and machine learning.** In 2017 24th Asia-Pacific Software Engineering Conference (APSEC) (pp. 388-397). IEEE.

61 KURTANOVIĆ, Z. AND MAALEJ, W., 2017, September. **Automatically classifying functional and non-functional requirements using supervised machine learning.** In 2017 IEEE 25th International Requirements Engineering Conference (RE) (pp. 490-495). IEEE.

62 TÓTH, L. AND VIDÁCS, L., 2018, May. **Study of various classifiers for identification and classification of non-functional requirements.** In International Conference on Computational Science and Its Applications (pp. 492-503). Springer, Cham.

63 BINKHONAIN, M. AND ZHAO, L., 2019. **A review of machine learning algorithms for identification and classification of non-functional requirements**. Expert Systems with Applications: X, p.100001.

64 GOGUEN, J.A. AND LINDE, C., 1993, January. **Techniques for requirements elicitation**. In [1993] Proceedings of the IEEE International Symposium on Requirements Engineering (pp. 152-164). IEEE.

65 MANNING, C.D., RAGHAVAN, P. AND SCHÜTZE, H., 2008. **Introduction to information retrieval**. Cambridge university press.

66 KULKARNI, N., PARACHURI, D., DASA, M. AND KUMAR, A., 2012, December. Automated Analysis of Textual Use-Cases: **Does NLP**

Components and Pipelines Matter? In 2012 19th Asia-Pacific Software Engineering Conference (Vol. 1, pp. 326-329). IEEE.

67 TUFIS, D., 2009. **Algorithms and Data Design Issues for Basic NLP Tools.** Language Engineering for Lesser-studied Languages, 21, p.3.

68 SALTON, G., AND BUCKLEY, C. **Term-weighting approaches in automatic text retrieval**. Information processing & management 24, no. 5 (1988): 513-523.

69 KNAUSS, E., LIEBEL, G., SCHNEIDER, K., HORKOFF, J. AND KASAULI, R., 2017, September. **Quality requirements in agile as a knowledge management problem: More than just-in-time**. In 2017 IEEE 25th International Requirements Engineering Conference Workshops (REW) (pp. 427-430). IEEE.

70 MARINHO, M., ARRUDA, D., WANDERLEY, F. AND LINS, A., 2018, September. **A Systematic Approach of Dataset Definition for a Supervised Machine Learning Using NFR Framework**. In 2018 11th International Conference on the Quality of Information and Communications Technology (QUATIC) (pp. 110-118). IEEE.

71 MACASAET, R., CHUNG, L., GARRIDO, J.L., NOGUERA, M. AND RODRÍGUEZ, M.L., 2011, June. **An agile requirements elicitation approach based on NFRs and business process models for micro-businesses**. In Proceedings of the 12th International Conference on Product Focused Software Development and Process Improvement (pp. 50-56).

72 CAPPELLI, C., CUNHA, H., GONZALEZ-BAIXAULI, B. AND DO PRADO LEITE, J.C.S., 2010, March. **Transparency versus security: early analysis of antagonistic requirements.** In Proceedings of the 2010 ACM symposium on applied computing (pp. 298-305).

73 GONZALEZ-BAIXAULI, B., LAGUNA, M. AND DO PRADO LEITE, J.C.S., 2005. **Applying personal construct theory to requirements elicitation**. IEEE Latin America Transactions, 3(1), pp.82-89.

74 NIU, N. AND EASTERBROOK, S., 2006, May. **Discovering aspects in requirements with repertory grid.** In Proceedings of the 2006 international workshop on Early aspects at ICSE (pp. 35-42).

75 RE17 DATA TRACK. DATASET: **Quality attributes (NFR).** 2017. Retrieved April 22, 2018 from: https://web.archive.org/web/20171216053325/http://re2017.org/pages/submission/data_papers/. Available at:

76 PORTUGAL, R.L.Q., LI, T., SILVA, L., ALMENTERO, E. AND DO PRADO LEITE, J.C.S., 2018, September. **NFRfinder: a knowledge based strategy for mining non-functional requirements.** In Proceedings of the XXXII Brazilian Symposium on Software Engineering (pp. 102-111).

77 BERRY, D.M., 2017, September. **Evaluation of tools for hairy requirements and software engineering tasks**. In 2017 IEEE 25th International Requirements Engineering Conference Workshops (REW) (pp. 284-291). IEEE.

78 PORTUGAL, R.L.Q., 2016. **Mineração de Informação em Linguagem Natural para Apoiar a Elicitação de Requisitos** (MSc. Dissertation. PUC-Rio University, Rio de Janeiro, Brasil).

79 PORTUGAL, R.L.Q., ROQUE H., AND LEITE. J.C.S.P. **A Corpus Builder: Retrieving Raw Data from GitHub for Knowledge Reuse in Requirements Elicitation**. In 2016 3rd International Conference on Information Management and Big Data (SIMBig). pp. 48-54. 2016

80 PORTUGAL R.L.Q, CASANOVA M.A., LI. T., AND LEITE J.C.S.P. **GH4RE: Repository Recommendation on GitHub for Requirements Elicitation Reuse.** In CAiSE-Forum-DC, pp. 113-120. 2017.

81 MOHEBZADA JG, RUHE G, EBERLEIN A. **Systematic mapping of recommendation systems for requirements engineering.** Proc. Int'l Conf. on Software and System Process. pp. 200-209. IEEE Press. (2012)

82 MAALEJ W, THURIMELLA AK. **Towards a research agenda for recommendation systems in requirements engineering**. Proc. 2nd Int'l. Workshop on Managing Requirements Knowledge. pp. 32-39. IEEE Computer Society. (2009)

83 CASTRO-HERRERA C, DUAN C, CLELAND-HUANG J, MOBASHER B. **Using data mining and recommender systems to facilitate large-scale, open, and inclusive requirements elicitation processes**. Proc.16th IEEE Int'l. Requirements Engineering Conf. pp. 165-168. IEEE. (2008)

84 CASTRO-HERRERA C, DUAN C, CLELAND-HUANG J, MOBASHER B. **A recommender system for requirements elicitation in large-scale software projects**. Proc. Symposium on Applied Computing. pp. 1419-1426. ACM. (2009)

85 CASTRO-HERRERA C, CLELAND-HUANG J, MOBASHER B. **Enhancing stakeholder profiles to improve recommendations in online requirements elicitation**. In 17th IEEE International Requirements Engineering Conference. pp. 37-46. IEEE. (2009)

86 CASTRO-HERRERA C, CLELAND-HUANG J. **Utilizing recommender systems to support software requirements elicitation**. In Proceedings of the 2nd International Workshop on Recommendation Systems for Software Engineering. pp. 6-10. ACM. (2010)

87 LIM SL, FINKELSTEIN A. **StakeRare: using social networks and collaborative filtering for large-scale requirements elicitation**. IEEE Trans. on Software Eng. pp. 707-35. (2012)

88 HARIRI N, CASTRO-HERRERA C, CLELAND-HUANG J, MOBASHER B. **Recommendation systems in requirements discovery.** Recommendation Systems in Software Eng. pp. 455-476 (2014)

89 GUENDOUZ, M., AMINE, A., & HAMOU, R. M. **Recommending relevant GitHub repositories: a collaborative-filtering approach.** on Networking and Advanced Systems, 34. (2015)

90 LIKERT R, 1932. **A technique for the measurement of attitudes.** Archives of psychology.

91 PORTUGAL, R.L.Q. **Artifacts used in GH4RE. CAiSE-Forum 17 version**. Zenodo Repository. DOI: 10.5281/zenodo.3921767 |

92 INGWERSEN, P., 1992. **Information retrieval interaction** (Vol. 246). London: Taylor Graham.

93 RE - PUC Rio. **Transparency Catalog.** Online. Available at: http://transparencia.inf.puc-rio.br/wiki/index.php/Catálogo_Transparência. Last Access: 20-03-2017

94 CLELAND-HUANG, J., SETTIMI, R., BENKHADRA, O., BEREZHANSKAYA, E. AND CHRISTINA, S., 2005, May. **Goal-centric traceability for managing non-functional requirements.** In Proceedings of the 27th international conference on Software engineering (pp. 362-371). ACM

95 FRANCH, X., PINYOL, J. AND VANCELLS, J., 1999, June. **Browsing a component library using non-functional information.** In International Conference on Reliable Software Technologies (pp. 332-343). Springer, Berlin, Heidelberg.

96 ANTON, A.I., BOLCHINI, D. AND HE, Q., 2003. **The use of goals to extract privacy and security requirements from policy statements**. North Carolina State University. Dept. of Computer Science.

97 ENGIEL, P., LEITE, J.C.S.P., CAPPELLI, C. 2014. **Confirmando a Demanda por Transparência: Um Estudo Inicial sobre um Sistema de Avaliação de Projetos de Lei.** Anais do II Workshop de Transparência em Sistemas. Londrina, Brasil.

98 ENGIEL, P., PORTUGAL, R.L.Q., LEITE, J.C.S.P. 2016. **Descobrindo Projetos de Lei relacionados a Transparência**. IV Workshop de Transparência em Sistemas, Rio de Janeiro, Brasil.

99 PORTUGAL, R.L.Q., ENGIEL, P., LEITE, J.C.S.P. 2017. **Existe uma Demanda de Transparência? Análise de comentários à Projetos de Lei.** V Workshop de Transparência em Sistemas. São Paulo, Brasil

100 BORGIDA A, GREENSPAN S, AND MYLOPOULOS J. **Knowledge representation as the basis for requirements specifications**. In Wissensbasierte Systeme, pp. 152-169. Springer, Berlin, Heidelberg, 1985.

101 LEITE J.C.S.P, AND FRANCO A.P.M. **A strategy for conceptual model acquisition**. In [1993] Proceedings of the IEEE International Symposium on Requirements Engineering, pp. 243-246. IEEE, 1993.

102 SAWYER P, RAYSON P, AND COSH K. **Shallow knowledge as an aid to deep understanding in early phase requirements engineering.** IEEE Transactions on Software Engineering 31, no. 11 (2005): 969-981

103 GACITUA R, SAWYER P, AND GERVASI V. **On the effectiveness of abstraction identification in requirements engineering.** In 2010 18th IEEE International Requirements Engineering Conference, pp. 5-14. IEEE, 2010.

104 SMITH A.E, AND HUMPHREYS M. S. **Evaluation of unsupervised semantic mapping of natural language with Leximancer concept mapping**. Behavior research methods 38, no. 2 (2006): 262-279

105 HAMZA M, AND WALKER R. J. **Recommending features and feature relationships from requirements documents for software product lines**. In 2015 IEEE/ACM 4th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering, pp. 25-31. IEEE, 2015.

106 MORALES-RAMIREZ I, KIFETEW F.M, AND PERINI A. **Speech-acts based analysis for requirements discovery from online discussions**. Information Systems (2018).

107 ROSS, D. T. **Structured analysis (SA): A language for communicating ideas**. Software Engineering, IEEE Transactions, 1977. (1) pp. 16-34.

108 VRANDEČIĆ D, AND KRÖTZSCH M. **WIKIDATA**: **a free collaborative knowledge base.** Communications of the ACM. Volume 57 Issue 10, October. pp. 78-85 (2014).

109 TOEWS D, HOLLAND L.V. **Determining Domain-specific Differences of Polysemous Words Using Context Information**. In 2nd Workshop on Natural Language Processing for Requirements Engineering (NLP4RE), 2019

110 NIGAM A, ARYA N, NIGAM B, AND JAIN D. **Tool for automatic discovery of ambiguity in requirements.** International Journal of Computer Science Issues (IJCSI) 9, no. 5 (2012): 350

111 KEYES O. AND GRAUL C. (2016). **WikidataR: API Client Library for 'Wikidata'.** R package version 1.1.0. https://CRAN.R-project.org/package=WikidataR

112 BRANK J, LEBAN G, AND GROBELNIK M. **Semantic Annotation of Documents Based on Wikipedia Concepts**. Informatica 42, no. 1 (2018).

113 BRANK J, LEBAN G, AND GROBELNIK M. **Annotating documents with relevant Wikipedia concepts**. Proceedings of SiKDD (2017).

114 PORTUGAL R.L.Q, LEITE J.C.S.P. **Usability Related Qualities Through Sentiment Analysis**. In 2018 1st International Workshop on Affective Computing for Requirements Engineering (AffectRE), pp. 20-26. IEEE, 2018

115 FEINERER I AND HORNIK K. (2015). **tm: Text Mining Package.** R package version 0.6-2. https://CRAN.R-project.org/package=tm

116 BENOIT K AND NULTY P. (2016). **quanteda: Quantitative Analysis of Textual Data**. R package version 0.9.8.3. https://CRAN.R-project.org/package=quanteda

117 ANTONIOL G, AYARI K, DI PENTA M, KHOMH F, AND GUÉHÉNEUC Y. G. **Is it a bug or an enhancement? a text-based approach to classify change requests**. In CASCON, vol. 8, pp. 304-318. 2008.

118 SALTON, G., AND BUCKLEY C. **Term-weighting approaches in automatic text retrieval**. Information processing & management 24, no. 5 (1988): 513-523.

119 CHARNIAK E. **Statistical techniques for natural language parsing.** AI magazine 18, no. 4 (1997): 33-33

120 MICHALKE M. (2016). **koRpus: An R Package for Text Analysis**. Version 0.06-4. http://reaktanz.de/?c=hacking&s=koRpus

121 GRECHANIK M, CONROY K.M, AND PROBST K.A. **Finding relevant applications for prototyping**. In Fourth International Workshop on Mining Software Repositories (MSR'07: ICSE Workshops 2007), pp. 12-12. IEEE, 2007.

122 AMIN R, CINNÉIDE M. O, AND VEALE T. **Laser: a lexical approach to analogy in software reuse**. In Proceedings of the Workshop on Mining Software Repositories, pp. 112-116. 2004

123 OHBA M, AND GONDOW K. **Toward mining concept keywords from identifiers in large software projects**. In ACM SIGSOFT Software Engineering Notes, vol. 30, no. 4, pp. 1-5. ACM, 2005.

124 TIMONEN M, TOIVANEN T, KASARI M, TENG Y, CHENG C, AND HE L. **Keyword extraction from short documents using three levels of word evaluation.** In International Joint Conference on Knowledge Discovery, Knowledge Engineering, and Knowledge Management, pp. 130-146. Springer, Berlin, Heidelberg, 2012.

125 ARCELLI FONTANA F, ROVEDA R, AND ZANONI M. **Discover knowledge on FLOSS projects through RepoFinder.** In Proceedings of the International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management-Volume 1, pp. 485-491. SCITEPRESS-Science and Technology Publications, Lda, 2014.

126 ALJAMEL A, OSMAN T, AND ACAMPORA G. **Domain-specific relation extraction: Using distant supervision machine learning.** In 2015 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K), vol. 1, pp. 92-103. IEEE, 2015.

127 LIAN X, RAHIMI M, CLELAND-HUANG J, ZHANG L, FERRAI R, AND SMITH M. **Mining requirements knowledge from collections of domain documents**. In 2016 IEEE 24th International Requirements Engineering Conference (RE), pp. 156-165. IEEE, 2016.

128 GROEN E.C, SCHOWALTER J, KOPCZYNSKA S, POLST S, AND ALVANI S. **Is there Really a Need for Using NLP to Elicit Requirements? A Benchmarking Study to Assess Scalability of Manual Analysis.** In REFSQ Workshops. 2018.

129 SCHLUTTER A, AARON, AND VOGELSANG A. **Knowledge representation of requirements documents using natural language processing**. In REFSQ Workshops. 2018

130 BORRUL R, COSTAL D, FRANCH X, QUER C. **Research on NLP for RE at UPC: a Report** In REFSQ Workshops. 2018.

131 FUCCI D, STANIK C, MONTGOMERY L, KURTANOVIC Z, JOHANN T, MALEEJ W. **Research on NLP for RE at the University of Hamburg: a Report** In REFSQ Workshops. 2018.

132 HIBSHI H, BREAUX T.D, AND BROOMELL S.B.. **Assessment of risk perception in security requirements composition.** In 2015 IEEE 23rd

International Requirements Engineering Conference (RE), pp. 146-155. IEEE, 2015.

133 RIEDL, M. O., AND LEÓN C. **Toward vignette-based story generation for drama management systems.** In Workshop on Integrating Technologies for Interactive Stories-2nd International Conference on INtelligent TEchnologies for interactive enterTAINment, pp. 8-10. 2008.

134 BUBENKO J.A. **Challenges in requirements engineering**. In Proceedings of 1995 IEEE International Symposium on Requirements Engineering (RE'95), pp. 160-162. IEEE, 1995.

135 PORTUGAL, R.L.Q. **Artifacts of Querying Assistant using Knowledge Bases**. **Version RE19**. Zenodo Repository. DOI: 10.5281/zenodo.3923120

136 SHEPPERD, M AND SCHOFIELD C. **Estimating software project effort using analogies.** IEEE Transactions on software engineering 23.11 (1997): 736-743.

137 KOLODNER, J. 2014. **Case-based reasoning**. Morgan Kaufmann

138 PORTUGAL, R. L. Q AND LEITE, J.C.S.P. **Extracting Requirements Patterns from Software Repositories**. Em Requirements Patterns (RePa), IEEE Workshop in Requirements Engineering Conference, 2016. 138

139 HAYES, J. H., LI, W., & RAHIMI, M. **Weka meets TraceLab: Toward convenient classification: Machine learning for requirements engineering problems:** A position paper. In 2014 IEEE 1st International Workshop on Artificial Intelligence for Requirements Engineering (AIRE) (pp. 9-12). IEEE.

140 PORTUGAL, R.L.Q. **Artifacts used in SVM approach, a Bug is an Enhancement. Version SVM-Issues**. Zenodo Repository. DOI: 10.5281/zenodo.3924460

141 MAGAZINE CIO FROM IDG. **The 10 biggest startup opportunities in 2016**. Available at: http://www.cio.com/article/3019718/startups/the-10-biggest-startup-opportunities-in-2016.html. Last Access: 03/08/20

142 KLIMT, B, & YANG, Y. 2004. **The enron corpus: A new dataset for email classification research.** In European Conference on Machine Learning (pp. 217-226). Springer Berlin Heidelberg.

143 COHEN, W. W., CARVALHO, V. R., & MITCHELL, T. M. 2004. **Learning to Classify Email into``Speech Acts''**. In EMNLP (pp. 309-316).

144 KOHAVI, R., 1995, August. **A study of cross-validation and bootstrap for accuracy estimation and model selection**. In Ijcai (Vol. 14, No. 2, pp. 1137-1145).

145 LAM, H. T, THIEBAUT, J. M, SINN, M, CHEN, B, MAI, T, & ALKAN, O. (2017). **One button machine for automating feature engineering in relational databases.** arXiv preprint arXiv:1706.00327.

146 MCKENNA J., 2014. **Conscious Software Development**.

147 BEKKERMAN, R., & ALLAN, J. (2004). **Using bigrams in text categorization**. Technical Report IR-408, Center of Intelligent Information Retrieval, UMass Amherst.

148 PAY, T., 2016, December. **Totally automated keyword extraction**. In 2016 IEEE international conference on big data (Big Data) (pp. 3859-3863). IEEE

149 BOND, F. & FOSTER, R., 2013, August. **Linking and extending an open multilingual wordnet.** In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (pp. 1352-1362).

150 FEINERER I & HORNIK K (2017). **WordNet: WordNet Interface**. R package version 0.1-14, https://CRAN.R-project.org/package=wordnet.

151 JU, R. & LICEA, G., 2017, October. **Towards supporting software engineering using deep learning: A case of software requirements classification.** In 2017 5th International Conference in Software Engineering Research and Innovation (CONISOFT) (pp. 116-120). IEEE.

152 FONG, V.L., 2018. **Software Requirements Classification Using Word Embeddings and Convolutional Neural Networks**. (MSc. Dissertation. California Polytechnic State University, San Luis Obispo).

153 BAKER, C., DENG, L., CHAKRABORTY, S. & DEHLINGER, J., 2019, July. **Automatic Multi-class Non-Functional Software Requirements Classification Using Neural Networks**. In 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC) (Vol. 2, pp. 610-615). IEEE.

154 YU, E.S 2009. **Social Modeling and i*.** In Conceptual Modeling: Foundations and Applications, pp. 99-121. Springer, Berlin, Heidelberg.

155 HAAHR, P. & BAKER, S., **Google LLC, 2011. System and method for providing search query refinements.** U.S. Patent 8,086,619

156 YU S. **Part-of-Speech Tutorial.** Available at: https://sites.google.com/site/partofspeechhelp/home/jj_vbn. Last Access: 3/07/2020

157 VIENNA OXFORD INTERNATIONAL CORPUS OF ENGLISH. 2014. **Part-of-Speech Tagging and Lemmatization Manual**. Available at: https://www.univie.ac.at/voice/page/tagging_manual_information. Last Access: 3/07/2020

158 LEITE, J.C.S.P. & FREEMAN, P.A., 1991. Requirements validation through viewpoint resolution. IEEE transactions on Software Engineering, (12), pp.1253-1269.

159 MARTIN, J. & WEI-TEK T. 1990. **N-fold inspection: A requirements analysis technique.** Communications of the ACM 33, no. 2: 225-232.

160 CONOVER WJF & IMAN RONALD L. 1976. **On some alternative procedures using ranks for the analysis of experimental designs**. Communications in Statistics-Theory and Methods 5, no. 14: 1349-1368.

161 PORTUGAL, R.L.Q. **NFRFinder artifacts until SBES18**. Zenodo Repository. DOI: 10.5281/zenodo.3906608

162 GUZMAN E, AZÓCAR D, LI Y. **Sentiment analysis of commit comments in GitHub: an empirical study.** In Proceedings of the 11th Working Conference on Mining Software Repositories 2014 May 31 (pp. 352-355). ACM.

163 THELWALL M. **The Heart and soul of the web? Sentiment strength detection in the social web with SentiStrength**. In Cyberemotions 2017 (pp. 119-134). Springer, Cham

164 BLAZ CC & BECKER K. **Sentiment analysis in tickets for it support**. In Mining Software Repositories (MSR), 2016 IEEE/ACM 13th Working Conference on 2016 May 14 (pp. 235-246). IEEE

165 IMTIAZ N, MIDDLETON J, GIROUARD P, MURPHY-HILL E. **Sentiment and Politeness Analysis Tools on Developer Discussions Are Unreliable**, **but so Are People**, SEmotion'18, June 2, 2018, Gothenburg, Sweden

166 ORTU, M., DESTEFANIS, G., COUNSELL, S., SWIFT, S., TONELLI, R. AND MARCHESI, M., 2017. **How diverse is your team? Investigating gender and nationality diversity in GitHub teams**. Journal of Software Engineering Research and Development, 5(1), p.9.

167 IMTIAZ N, MIDDLETON J, GIROUARD P, MURPHY-HILL E. **Sentiment and Politeness Analysis Tools on Developer Discussions Are Unreliable, but so Are People**, SEmotion'18, June 2, 2018, Gothenburg, Sweden

168 JONGELING R, DATTA S, & SEREBRENIK A, **Choosing your weapons: On sentiment analysis tools for software engineering research**. In Software maintenance and evolution (ICSME), 2015 IEEE international conference on 2015 Sep 29 (pp. 531-535). IEEE.

169 YANG B., WEI X., & LIU C.. **Sentiments Analysis in GitHub Repositories: An Empirical Study**. In Software Engineering Conference Workshops (APSECW), 2017 24th Asia-Pacific 017 Dec 4 (pp. 84-89). IEEE

170 JURADO F & RODRIGUEZ P.**, Sentiment Analysis in monitoring software development processes: An exploratory case study on GitHub's project issues.** Journal of Systems and Software. 2015 Jun1;104:82-9

171 CYSNEIROS L.M., **Evaluating the Effectiveness of Using Catalogs to Elicit Non-Functional Requirements**. 2007. In Workshop on Requirements Engineering WER. pp. 107-115.

172 PORTUGAL, R.L.Q. & LEITE, J.C.S.P, 2018, July. **Análise de Sentimento no Contexto de Comentários a Projetos de Lei Relativos à Transparência.** In Anais do VI Workshop de Transparência em Sistemas. SBC.

173 PORTUGAL, R.L.Q. **Usability Related Qualities through Sentiment Analysis: Artifacts until AffectRE18**. Zenodo Repository. DOI: 10.5281/zenodo.3908477

174 SUPAKKUL, S. **NFRs Modeling Patterns and Anti-Patterns**. Available at https://personal.utdallas.edu/~supakkul/NFR-modeling/index.html. 2011. Last access: 03/11/2020

175 HAKIM, L. & ROCHIMAH, S., 2018, October. **Oversampling Imbalance Data: Case Study on Functional and Non Functional Requirement**. In 2018 Electrical Power, Electronics, Communications, Controls and Informatics Seminar (EECCIS) (pp. 315-319). IEEE.

176 ABDELQADER A.A. 2019. **A novel intelligent model for classify and evaluating non-functional security requirements form scenarios**. Indonesian Journal of Electrical Engineering and Computer Science.

Appendix A

A.1. Manual Classification of Issues

Researcher 1

url	query	enhancement	bug	duplicate	helpwanted	invalid	wontfix	question
https://github.com/roee88/meta4kodi/issues/48	movies	1	0	0	0	0	0	0
https://github.com/liberapay/liberapay.com/issues/59	money-transfer	1	0	0	0	0	0	0
https://github.com/Bernie-2016/Connect-SharkWeek/issues/3	share-economy	0	0	0	1	0	0	0
https://github.com/faizaanshamsi/canary_v2/issues/27	inf-availability	1	0	0	0	0	0	0
https://github.com/w3c/personalization-semantics/issues/1	clothing	0	1	0	0	0	0	0
https://github.com/poetic/fancy-maps/issues/11	pollution	0	1	0	0	0	0	0
https://github.com/tomokas/pathfinding-genetic/issues/1	autonomous	0	1	0	0	0	0	0
https://github.com/CleverRaven/Cataclysm-DDA/issues/14112	clothing	1	0	0	0	0	0	0
https://github.com/odoo/odoo/issues/7237	manufacturing	0	1	0	0	0	0	0
https://github.com/saurabhdogspot/zirco-browser/issues/40	market	0	1	0	0	0	0	0
https://github.com/code-google-com/arora/issues/483	recommendation-system	0	1	0	0	0	0	0
https://github.com/StarQuestMinecraft/StarQuestPublic/issues/1080	travel	0	1	0	0	0	0	0
https://github.com/18F/tts-public-comments/issues/18	digital-health	1	0	0	0	0	0	0
https://github.com/Sylean92/MAGD-150-Assignments/issues/4	music	0	0	0	0	1	0	0
https://github.com/inspirehep/inspire-next/issues/1461	conferences	1	0	0	0	0	0	0
https://github.com/reissjohnson/MakersBnbRails/issues/13	booking	1	0	0	0	0	0	0
https://github.com/anselmorenato/lightlang/issues/114	developing-world	1	0	0	0	0	0	0
https://github.com/Mimi2k15/google-opt-out-plugin/issues/39	florist	0	1	0	0	0	0	0

https://github.com/rainbow702/flexigrid/issues/148	wedding	0	0	0	0	1	0	0
https://github.com/obbiwan/Testimonium/issues/5	recommendation-system	1	0	0	0	0	0	0
https://github.com/kurokama/nulldc/issues/354	manufacturing	0	1	0	0	0	0	0
https://github.com/keralapsctips/keralapsctips/issues/590	sports	0	0	0	0	1	0	0
https://github.com/vahid8028/droidwall/issues/202	privacy	1	0	0	0	0	0	0
https://github.com/humanytek/mer/issues/8	manufacturing	0	1	0	0	0	0	0
https://github.com/matryer/bitbar-plugins/issues/544	travel	0	0	0	1	0	0	0
https://github.com/contiki-os/contiki/issues/583	conferences	0	0	0	1	0	0	0
https://github.com/mausquirk/skytraq-datalogger/issues/7	travel	0	0	0	1	0	0	0
https://github.com/google-code-export/jquery-i18n-properties/issues/38	pollution	0	1	0	0	0	0	0
https://github.com/MarkKing12/In-The-France/issues/96	taxi	0	0	0	0	1	0	0
https://github.com/bharaninb/eyes-free/issues/367	accessibility	1	1	0	0	0	0	0
https://github.com/twcctz500000/redis/issues/596	pollution	1	0	0	0	0	0	0
https://github.com/CodeForFoco/org/issues/40	real-estate	0	0	0	1	0	0	0
https://github.com/mocruz/movist/issues/69	movies	1	0	0	0	0	0	0
https://github.com/brownplt/pyret-lang/issues/186	movies	1	0	0	0	0	0	0
https://github.com/HippieStationCode/HippieStation13/issues/3017	cooking	0	1	0	0	0	0	0
https://github.com/techcompiler/gwt-oauth2/issues/82	inf-availability	0	0	1	0	0	0	0
https://github.com/dusblinov/jodconverter/issues/102	soft-simplicity	0	1	0	0	0	0	0
https://github.com/WhiteCoatAcademy/whitecoatacademy.org/issues/1	diseases	1	0	0	0	0	0	0
https://github.com/JoeProgram/monster/issues/19	music	0	0	0	0	0	0	0
0	market	0	1	0	0	0	0	0
https://github.com/hallmark/gitwebhook/issues/3300	autonomous	0	0	0	0	1	0	0
https://github.com/unisonweb/unison/issues/101	soft-portability	1	0	0	0	0	0	0
https://github.com/cam-technologies/time-booker/issues/36	booking	1	0	0	0	0	0	0

https://github.com/openreferral/api-specification/issues/20	privacy	1	0	0	0	0	0	0
https://github.com/Thorium-Sim/thorium/issues/75	medical	0	0	0	0	0	0	0
https://github.com/malathicode/foursquared/issues/132	smart-city	1	0	0	0	0	0	0
https://github.com/codeforamerica/lv-trucks-map/issues/65	food-truck	0	1	0	0	0	0	0
https://github.com/kdekorte/gecko-mediaplayer/issues/96	movies	0	1	0	0	0	0	0
https://github.com/suiyuchen/dashclock/issues/314	transparency-inf	0	1	0	0	0	0	0
https://github.com/sckott/soylocs/issues/30	vegan	1	0	0	0	0	0	0

Researcher 2

url	query	enhancement	bug	duplicate	helpwanted	invalid	wontfix	question
https://github.com/roee88/meta4kodi/issues/48	movies	0	0	0	0	0	0	1
https://github.com/liberapay/liberapay.com/issues/59	money-transfer	1	0	0	0	0	0	0
https://github.com/Bernie-2016/Connect-SharkWeek/issues/3	share-economy	1	0	0	0	0	0	0
https://github.com/faizaanshamsi/canary_v2/issues/27	inf-availability	1	0	0	0	0	0	0
https://github.com/w3c/personalization-semantics/issues/1	clothing	0	1	0	0	0	0	0
https://github.com/poetic/fancy-maps/issues/11	pollution	1	0	0	0	0	0	0
https://github.com/tomokas/pathfinding-genetic/issues/1	autonomous	0	1	0	0	0	0	1
https://github.com/CleverRaven/Cataclysm-DDA/issues/14112	clothing	0	0	0	0	0	0	1
https://github.com/odoo/odoo/issues/7237	manufacturing	0	1	0	0	0	0	0
https://github.com/saurabhdogspot/zirco-browser/issues/40	market	0	1	0	0	0	0	1
https://github.com/code-google-com/arora/issues/483	recommendation-system	1	0	0	0	0	0	1
https://github.com/StarQuestMinecraft/StarQuestPublic/issues/1080	travel	0	1	0	0	0	0	0
https://github.com/18F/tts-public-comments/issues/18	digital-health	1	0	0	0	0	0	0
https://github.com/Sylean92/MAGD-150-Assignments/issues/4	music	0	0	0	0	1	0	0
https://github.com/inspirehep/inspire-next/issues/1461	conferences	1	0	0	0	0	0	0
https://github.com/reissjohnson/MakersBnbRails/issues/13	booking	1	0	0	0	0	0	0
https://github.com/anselmorenato/lightlang/issues/114	developing-world	0	0	0	0	1	0	0

https://github.com/Mimi2k15/google-opt-out-plugin/issues/39	florist	0	1	0	0	0	0	0
https://github.com/rainbow702/flexigrid/issues/148	wedding	0	0	0	0	1	0	0
https://github.com/obbiwan/Testimonium/issues/5	recommendation-system	1	0	0	0	0	0	1
https://github.com/kurokama/nulldc/issues/354	manufacturing	0	1	0	0	0	0	0
https://github.com/keralapsctips/keralapsctips/issues/590	sports	0	0	0	0	1	0	0
https://github.com/vahid8028/droidwall/issues/202	privacy	1	1	0	1	0	0	0
https://github.com/humanytek/mer/issues/8	manufacturing	0	1	0	0	0	0	0
https://github.com/matryer/bitbar-plugins/issues/544	travel	0	0	0	0	0	0	1
https://github.com/contiki-os/contiki/issues/583	conferences	1	0	0	0	0	0	1
https://github.com/mausquirk/skytraq-datalogger/issues/7	travel	1	0	0	0	0	0	1
https://github.com/google-code-export/jquery-i18n-properties/issues/38	pollution	0	0	0	0	1	0	0
https://github.com/MarkKing12/In-The-France/issues/96	taxi	0	0	0	0	1	0	0
https://github.com/bharaninb/eyes-free/issues/367	accessibility	0	1	0	0	0	0	1
https://github.com/twcctz500000/redis/issues/596	pollution	1	0	0	0	0	0	0
https://github.com/CodeForFoco/org/issues/40	real-estate	0	0	0	0	0	0	1
https://github.com/mocruz/movist/issues/69	movies	1	0	0	0	0	0	1
https://github.com/brownplt/pyret-lang/issues/186	movies	1	0	0	0	0	0	0
https://github.com/HippieStationCode/HippieStation13/issues/3017	cooking	0	1	0	0	0	0	0
https://github.com/techcompiler/gwt-oauth2/issues/82	inf-availability	0	0	0	0	0	0	1
https://github.com/dusblinov/jodconverter/issues/102	soft-simplicity	0	0	0	1	0	0	0
https://github.com/WhiteCoatAcademy/whitecoatacademy.org/issues/1	diseases	1	0	0	0	0	0	0
https://github.com/JoeProgram/monster/issues/19	music	0	0	0	0	1	0	0
https://github.com/nevzathuruzoglu/Homework-1---Black-box-test/issues/3	market	0	1	0	0	0	0	0
https://github.com/hallmark/gitwebhook/issues/3300	autonomous	0	0	0	0	1	0	0
https://github.com/unisonweb/unison/issues/101	soft-portability	1	0	0	0	0	0	0
https://github.com/cam-technologies/time-booker/issues/36	booking	0	0	0	0	0	0	1
https://github.com/openreferral/api-specification/issues/20	privacy	1	0	0	1	0	0	0
https://github.com/Thorium-Sim/thorium/issues/75	medical	0	0	0	0	1	0	0

https://github.com/malathicode/foursquared/issues/132	smart-city	1	0	0	0	0	0	0
https://github.com/codeforamerica/lv-trucks-map/issues/65	food-truck	0	1	0	0	0	0	0
https://github.com/kdekorte/gecko-mediaplayer/issues/96	movies	0	0	0	0	0	0	0
https://github.com/suiyuchen/dashclock/issues/314	transparency-inf	0	1	0	0	0	0	1
https://github.com/sckott/soylocs/issues/30	vegan	1	0	0	0	0	0	0

Appendix B

B.1. 4-Viewpoints Classification

index	RequirementText	Square w Catalogs		Square w/o Catalogs		Diamon w Catalogs		Diamon w/o Catalogs		Oval w Catalogs		Oval w/o Catalogs		Oval w Catalogs		Oval w/o Catalogs	
		Class.	Crit.	Class.	Crit.	Class.	Crit.	Class.	Crit.	Class.	Crit.	Class.	Crit.	Class.	Crit.	Class.	Crit.
49	The product shall be able to support multiple remote users	0		0		1	cs - operationalization	1	cs - operationalization	1	cs e cat: availability (SIG Usability), multiusers	1	multiusers	1	cat	0	
295	The product shall create an exception log of problems encountered within the product for transmission to our company for analysis and resolution.	1	imp	0		1	cs - operationalization			1	cs e cat: fault tolerance, auditability (SIG Transparency)	1	fault tolerant	1	cat	0	
593	The Brio portion of the WCS system must be able to export files in spreadsheet form (Microsoft Excel and Lotus 1-2-3 formats). Brio will provide buttons in the user interface that produce/export reports in .xls or .123 file formats.	0		0		1	cs - operationalization ; cat (usa - oliv)		cs - operationalization ; cs	1	cs e cat: consistency (SIG transparency), interoperability	1	interoperability	0		0	
222	The product shall achieve a 98% uptime. The product shall not fail more than 2% of the available online time.	0		0		1	cs - operationalization		cs	1	cs: availability	1	cs:availability	1	cs	1	cs

25	The system shall display the local and exercise time in separate clocks	0		0		1	cs - operation alization			0		0		0		0	
156	The Disputes application shall support 350 concurrent users without any degradation of performance in the application.	1	cat	1	cs	1	cs - operation alization		imp	1	cs: concorrenc y, performanc e	1	cs:multiuse rs, performanc e	1	cs	1	CS
364	Product formula ingredients shall allow defining substitutionary ingredients.	0		0		0				0		0		0		0	
333	The product will update existing room equipment.	0		0		0				0		0		0		0	
576	The product shall have security.The product shall provide authentication and authorization.	1	cat	1		1	cat(cysn-priv)		imp ; cs - operati onaliza tion	1	cs: security authenticati on, authorizatio n	1	cs: security, authenticati on, authorizatio n	1	cs	1	CS
173	The Disputes application must conform to the legal requirements as specified by the Merchant Operating Regulations.	1	cat	1	cs	1	cs		cs	1	cs: legal requirement s	1	cs: legal req	1	cs	1	CS
153	100% of cardmember services representatives shall be able to successfully create a dispute case on the first encounter after completing the training course.	0		0		0				0		0		1	cs	1	CS
360	The System shall allow on demand generation of all Inventory Quantity Adjustment documents since certain point of time.	0		0		0				1	cs: persistence	1	persistence	0		0	

433	Data integrity scripts will be run on a weekly basis to verify the integrity of the database.	1	cat	0		1	cat (transp)		imp ; cs - operationalization	1	cs: integrity	1	cs: integrity	1	cs	1	cs
353	The System shall utilize currently owned computer equipment.	0		0		0			?	1	cs: portability	1	imp: portability, compatibility	1	imp	1	imp
5	If projected the data must be understandable. On a 10x10 projection screen 90% of viewers must be able to determine that Events or Activities are occuring in current time from a viewing distance of 100	0		0		1	cat(transp)		imp ; cs - operationalization	1	cs: understandability, accessibility	1	cs: understandability, accessibility	1	cs	1	cs
217	The search for recycled parts shall take no longer than 15 seconds. The search results shall be returned in under 15 seconds.	0		0		1	imp; cs		imp ; cs - operationalization	1	cs: performance	1	cs: performance	1	cs	1	cs
324	The product shall record meeting entries.	0		0		0				0		0		0		0	
462	Website must be able to support free trial periods with various parameters set by the Izogn Manager.	0		0		1	cs - operationalization			1	cs e cat: publicity (SIG transparency), availability	1	imp: availability	1	imp	1	imp
117	A Program of Study shall consist of a program name and listing of required classes (both clinical and non-clinical) that must be completed.	0		0		0				0		0		0		0	

320	The product shall have the ability to receive automatic software updates as new threats emerge. 100% of customers will be able to receive automatic software updates transmitted to the installed product.	0		0			cs - operation alization		cs - operati nalizati on	1	cs e cat: security, adaptability (sig transparenc y), evolution	1	cs: security, evolution	1	cs	1	cs
219	The recycled parts audit report shall be returned to the user within 10 seconds. The audit report shall be returned within 10 seconds.	0		0			cs- opertiona lizaton		cs - operati nalizati on	1	cs: performanc e	1	cs: performanc e	1	cs	1	cs
440	The website should cater to all tribes in Nigeria.	1	imp	1	cs					1	cs: availability completene ss	1	imp: availability, completene ss	1	imp	1	imp
434	The website shall make its user aware of its information practices before collection data from them via a Privacy Policy accessible on all pages of the website.	1	cat	1	cs		imp ; cat (cys-priv)		imp	1	cs: privacy, accessibilit y	1	cs: privacy, accessibilit y	1	imp	1	imp
14	The data displayed in both the nodes within the graph and the rows in the table are MSEL Summary data	1	cat	0						1	imp and SIG Transparen cy: consistency , comparable , redundancy	0		0		0	
362	The System shall allow entering storing and modifying product formulas.	1		0						1	imp: adaptability , configurabi lity, persistence	1	cs: configurabi lity, persistence	0		0	

67	The system shall notify the realtor when a seller or buyer responds to an appointment request	1	imp	1	imp					0		0		0		0	
169	If a user account is revoked it can only be re-instantiated by the System Administrator.	0		0			cs - operationalization		cs - operationalization	1	imp: legal requirement	1	imp: legal requirement	1	imp	1	imp
516	Once a game is initiated the product shall allow each player to position their 5 ships on their respective defensive grids.	0		0						0		0		0		0	
151	The list of dispute cases that are displayed after a search is performed must be color coded for easy identification of dispute cases based upon the dispute case status.	1	cat	1	cs		cs-opertionalizaton		imp	1	cs: usability	1	cs: understandability	1	cs	1	cs
331	The product will be able to delete conference rooms.	0		0								0		0		0	

B.2. 4-Viewpoints Keywords