Diogo Henrique de Jesus Lopes Albuquerque

BSc in Computer Engineering

A Sustainability Catalogue for Software Modelling

Dissertation submitted in partial fulfillment
of the requirements for the degree of

Master of Science in
Computer Science and Informatics Engineering

Adviser: Ana Maria Diniz Moreira, Associate Professor with Habilitation,
FCT NOVA University of Lisbon

Co-adviser: João Baptista da Silva Araújo Junior, Assistant Professor, FCT NOVA University of Lisbon

Examination Committee

Chairperson: Pedro Abílio Duarte de Medeiros
Raporteur: Isabel Sofia Sousa Brito
Members: Ana Maria Diniz Moreira
João Baptista da Silva Araújo Junior

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

July, 2020

A Sustainability Catalogue for Software Modelling

Copyright © Diogo Henrique de Jesus Lopes Albuquerque, Faculty of Sciences and Technology, NOVA University Lisbon.
The Faculty of Sciences and Technology and the NOVA University Lisbon have the right, perpetual and without geographical boundaries, to file and publish this dissertation through printed copies reproduced on paper or on digital form, or by any other means known or that may be invented, and to disseminate through scientific repositories and admit its copying and distribution for non-commercial, educational or research purposes, as long as credit is given to the author and editor.

This document was created using the (pdf)LaTeX processor, based in the "novathesis" template[1], developed at the Dep. Informática of FCT-NOVA [2].
[1] https://github.com/joaomlourenco/novathesis [2] http://www.di.fct.unl.pt

For a better and sustainable world...

Acknowledgements

To my adviser, professor Ana Maria Diniz Moreira, and my co-adviser, professor João Baptista da Silva Araújo Junior, I thank you for the opportunity to develop this work under your guidance and all the support, both moral and technical, throughout this journey. I will never forget, and I will be forever grateful for all the time, advices and guidance made available throughout the course of this MSc dissertation. It was a pleasure to be your student.

To the creator of the piStar Tool, João Pimentel, I appreciate all the availability you had to help me understand your tool and develop new extensions for it. Your help was indispensable for the success of this work.

To my friend André Malafaia, for all the strength he transmitted, and for all the help he provided during the development of this work. A special thanks also to all my heart-felt friends who were essential in this process, who provided moments of fun and relaxation.

To my girlfriend Catarina, I appreciate the patience and aid that she always tried to give me, so that I could overcome the toughest moments. Thanks also for the good mood that facilitated the passage through this phase.

To my parents and sister, a great thank you, for all the effort made since the beginning of my journey until its end, for all the patience and advices. Thank you for believing in me.

Finally, I thank all the people who I shared this phase, teachers, family, colleagues and friends who helped me in some way and with whom I shared good times.

Abstract

Sustainable development is the development that meets the needs of the present without compromising the needs of our future generations. It covers five different dimensions: environmental, economic, social, technical, and individual. Such dimensions are also of interest for software. For example, memory and power efficiency have an impact on the environmental dimension, the reduction of costs in software development and evolution relates to the economic dimension, the use of software for general improvement of people's lives affects the social dimension, the software's ability to cooperate with other systems impacts the technical dimension, and the improvement of well-being of individuals relates to the individual dimension. These various dimensions and their properties impact on each other and on the base requirements of a system. Therefore, well-informed design decisions require improved support to reason on such intra- and inter-relationships and impacts, early in development. The objective of this dissertation is to propose a catalog of sustainability requirements for later reuse during the software development process. The envisioned solution involves using requirement engineering activities to address sustainability in the early stages of the software development. The first step towards a solution was to perform a (agile) systematic mapping study in order to gain a complete and profound knowledge about the existing sustainability and requirement engineering techniques. This study was the base of our work. Our final artifact is a sustainability catalogue. This catalogue addresses four out of the five dimensions of sustainability, as well as their qualities and relationships. We did not treat the individual dimension, for sake of simplicity and time constraints, although we consider that some of its properties are included in the social dimension. The catalogue was developed using the iStar framework, and it was implemented in the piStar Tool. Such catalogue offers a generic approach that can be instantiated for particular application domains, and for any combination of dimensions. Hence, this work will contribute to the field of sustainable software development.

Keywords: Sustainable development, sustainability dimensions, requirements engineering, software development, sustainability requirements, trade-off analysis.

Resumo

O desenvolvimento sustentável é o desenvolvimento que atende às necessidades do presente sem comprometer a capacidade de futuras gerações de satisfazerem as suas próprias necessidades. Este desenvolvimento abrange cinco dimensões diferentes: ambiental, económica, social, técnica e individual. Tais dimensões são também de interesse para o software. Por exemplo, memória e eficiência energética têm um impacto na dimensão ambiental, a redução de custos no desenvolvimento e evolução do software relaciona-se com a dimensão económica, o uso para a melhoria geral da vida das pessoas afeta a dimensão social, a capapicade de cooperar com outros sistemas tem um impacto na dimensão técnica, e o melhoramento da qualidade de vida de cada indivíduo relaciona-se com a dimensão individual. Essas várias dimensões e as suas propriedades têm impacto umas nas outras e nos requisitos básicos de um sistema. Portanto, decisões de desenho informadas requerem melhor suporte para raciocinar sobre tais relações, sejam inter- ou intra-, no início do desenvolvimento. O objetivo desta dissertação é propor um catálogo, de requisitos de sustentabilidade para posterior reutilização durante o processo de desenvolvimento de software. A solução prevista envolve o uso de atividades de engenharia de requisitos para abordar a sustentabilidade nos estágios iniciais do desenvolvimento de software. Assim, o primeiro passo para uma solução foi realizar um mapeamento sistemático, a fim de obter um conhecimento completo e profundo sobre as técnicas existentes de sustentabilidade e engenharia de requisitos. Este estudo foi a base do nosso trabalho. O nosso artefato final é um catálogo de sustentabilidade. Este catálogo aborda quatro das cinco dimensões da sustentabilidade, bem como suas qualidades e relacionamentos. Não considerámos a dimensão individual, por razões de simplicidade e constragimentos de tempo, contudo considerámos que algumas das suas propriedades estão incluídas não dimensão social. O catálogo foi desenvolvido usando o iStar framework e implementado na ferramenta piStar. Este catálogo oferece uma abordagem genérica que pode ser instanciada para domínios de aplicação específicos, e para qualquer combinação de dimensões. Portanto, este trabalho contribuirá para o tópico de desenvolvimento sustentável de software.

Palavras-chave: Desenvolvimento sustentável, dimensões da sustentabilidade, engenharia

de requisitos, desenvolvimento de software, requisitos da sustentabilidade, análise de conflitos.

Contents

List of Figures

List of Tables

Glossary

non-functional requirement	A non-functional requirement (NFR) is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors [48].
requirement	A singular documented physical or functional need that a particular design, product or process aims to satisfy [48].
requirements prioritization	Which candidate requirements of a software product should be included in a certain release [32].
requirements elicitation	Is the practice of researching and discovering the requirements of a system from users, customers, and other stakeholders [48].
stakeholder	A person, group or organization that has interest or concern in something, especially a business [48].
sustainability	The process of maintaining change in a balanced environment, in which the exploitation of resources, the direction of investments, the orientation of technological development and institutional change are all in harmony and enhance both current and future potential to meet human needs and aspirations [50].
sustainability dimensions	A division made on sustainability which covers: social, economic, environmental, and technical aspects of sustainability [12].
sustainable development	Development that meets the needs of the present without compromising the ability of future generations to meet their own needs [50].
trade-off	Is a situational decision that involves diminishing or losing one quality, quantity or property of a set or design in return for gains in other aspects [45].

Acronyms

AHP	Analytic Hierarchy Process.
HTML	Hyper Text Markup Language.
IT	Information Technology.
MSc	Master in Science.
NA	Not Applicable.
NFR	Non Functional Requirement.
PICOC	Population, Intervention, Comparison, Outcome, Context.
PNG	Portable Network Graphics.
RE	Requirements Engineering.
ROI	Return Over Investment.
SIG	Softgoal Interdependency Graph.
SVG	Scalable Vector Graphics.
TXT	Text.
UI	User Interface.
UN	United Nations.
URL	Uniform Resource Link.

CHAPTER 1

Introduction

This research was triggered by the growing and urgent need to make our actions as sustainable as possible. Although the theme of sustainability is one that is increasingly discussed by the media and people in general, there is still a lack of resources and practical and effective solutions to make software development a sustainable process, or at least have their roots and ideologies based on sustainability. With this work, we hope to contribute and help in this collective and important demand. This introductory chapter presents the context and motivation of this MSc dissertation work, discusses the problems and respective challenges encountered, and ends with the objectives of this work and the expected contributions.

1.1 Context & motivation

Sustainability is "the process of maintaining change in a balanced environment, in which the exploitation of resources, the direction of investments, the orientation of technological development and institutional change are all in harmony and enhance both current and future potential to meet human needs and aspirations" [58]. Sustainability is discussed now more than ever. Measures are getting applied to wider and broader domains in order to protect our future generations and our planet. Measures such as sustainability reports [50], international treaties [59] and protocols [51] have one goal in common: to reduce the impact of our actions on our planet. Sustainability takes a major role in current times, since we cannot maintain our quality of life as humans nor of all other species on Earth unless we embrace it as part of our everyday life.

Since sustainability is such a wide and broad topic, a decomposition of it was made so that the needs and challenges became clearer. Sustainability addresses three main dimensions, also known as the sustainability pillars: social, economic, and environmental

[7]. However, some authors also consider a technical dimension [12] and an individual dimension [39] [1]. Dimensions have inter- and intra-relationships, i.e., the requirements that relate to one dimension, can relate to another, either inside the same dimension, or between dimensions. Therefore, a decision made on one dimension can impact decision on another dimension. For example, a decision to promote environmental sustainability may have a negative impact on the cost (economic dimension) of the system. A high-level view of various interactions and overlaps between the social, economic and environmental dimensions is shown in Figure 1.1.

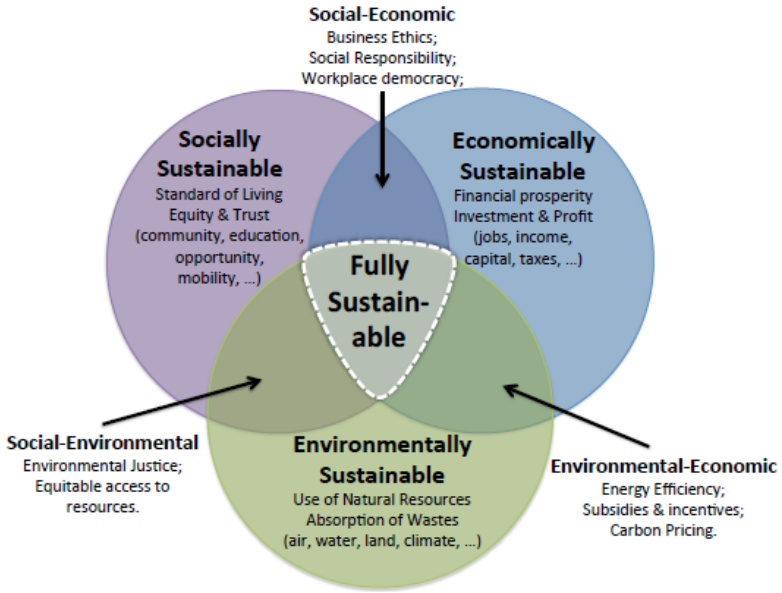


Figure 1.1: Relationship between the three main sustainability dimensions, from [7].

The motivation for this dissertation lies on this concept of sustainability, but directed to software development. Ideally, the software product and the entire development process should consider sustainability as a first-class concern. However, addressing sustainability in software, on late stages of development, is not recommended, because we run the risk of not addressing it properly. This dissertation offers a configurable sustainability catalogue to be totally (or partially) reused throughout software development process, particularly during the early development stages, such as requirements engineering. Requirements engineering is the process of discovering, defining, documenting, managing and maintaining requirements in the engineering of the software design process with the ultimate goal of producing a set of consistent, complete and relevant system requirements that reflect the needs of stakeholders [48]. This area is very important in any software or product development since, without it, the risk of project failure is potentially much higher [2], as summarized in Figure 1.2.

[1] The individual dimension is not addressed in this work, but parts of it are included in the social dimension

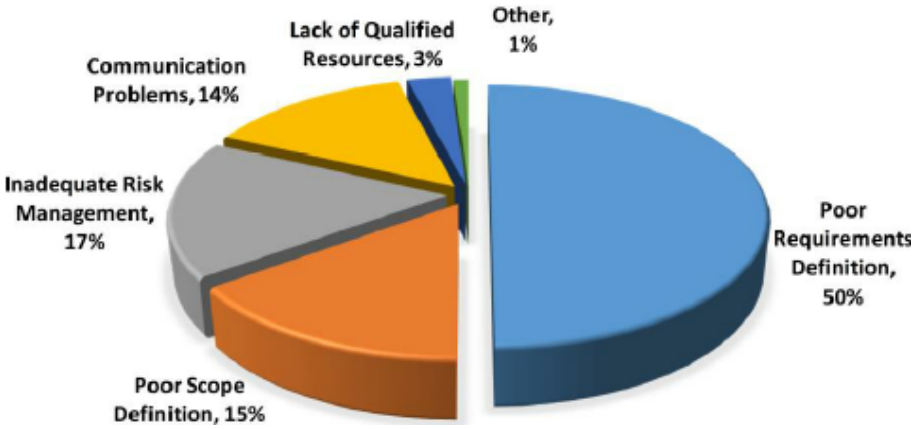


Figure 1.2: Impact of requirements in software project failures, from [2].

1.2 Problem statement

The problem that triggered the research work for this dissertation is **the lack of approaches to support the development of sustainable software**. This problem has been confirmed by the results found by a systematic mapping study we performed as part of the first step of this dissertation. The discussion that follows decomposes the identifies problem into two sub-problems, from which four major challenges have been derived. Each challenge is discussed together with the solution we envision to tackle it. A relation between the problem statement, its major problems and respective challenges is summarized and detailed in Figure 1.3.

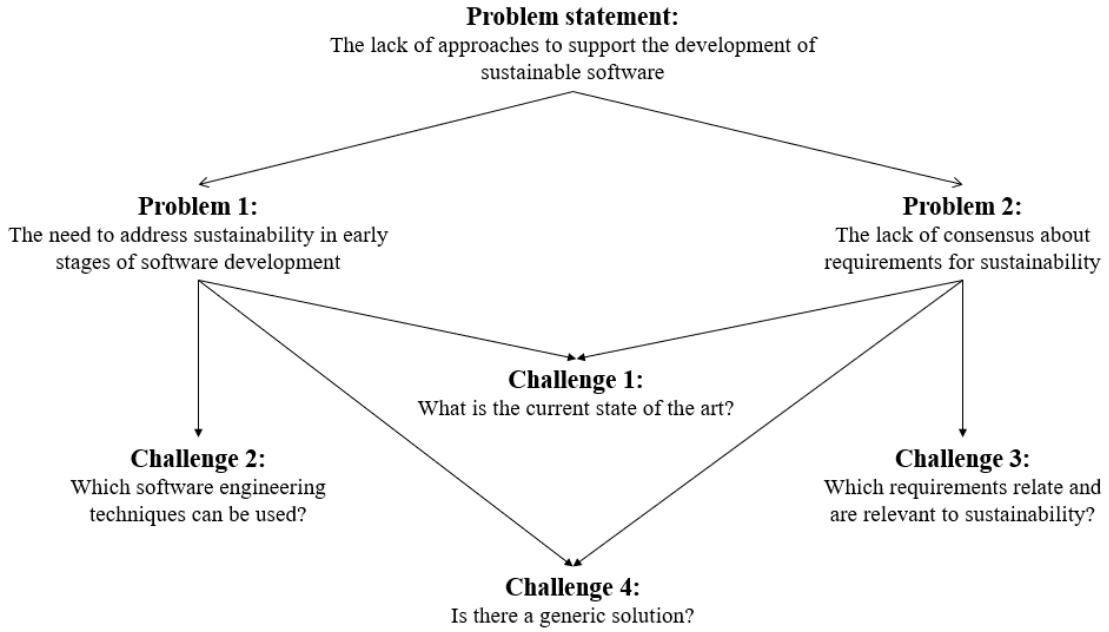


Figure 1.3: Problem statement, main problems and respective challenges.

Problem 1 → The need to address sustainability in early stages of software development.
For well-informed decisions about sustainability, we cannot simply handle the sustainability impact during the implementation of a project or after it is completed. Sustainability needs to be reasoned about as early as possible, and its impact between their constituent dimensions and on the system should be studied from the initial stages of software development, such as the requirements engineering, so that well-informed decision can be made before code starts to be written.

Problem 2 → The lack of consensus about requirements for sustainability
As an emerging topic, there is still a discussion about what sustainability really is and what it means to software development. In order to develop sustainable software, first we must understand the concept, as well as its properties or requirements.

Challenge 1 → What is the current state of the art?
A simple search is not enough since it would not produce full and viable results. A deeper and sheerer search is needed to decrease the possibility of relevant studies being left apart. To solve this challenge, we carried out a systematic mapping study on requirements for sustainability and requirements engineering, with the aid of focused research questions. This work is covered in Section 2.1.1.1. An automatic search, using the research questions, was made in Dblp, which compiles various online digital libraries, such as IEEExplore or ACM. Such search provided us with many studies, where a selection of some was done following inclusion and exclusion criteria. A later manual search was done so that the automatic search could be validated and corroborated. This manual search used two approaches: Snowballing [61], by using the reference list of each selected study performed to identify additional potentially relevant studies; a manual keyword search through the most relevant articles in Google Scholar.

Challenge 2 → Which software engineering techniques can be used?
Since sustainability should be addressed in early stages of software development, preferably on requirements stage, we need to know which good practices there are for requirements engineering. The proposed catalogue was based on the systematic study findings, particularly the existing techniques, their recommended conditions, strengths and weaknesses.

Challenge 3 → Which requirements relate and are relevant to sustainability?
Sustainability is a topic which touches many areas in our world and for that reason it is difficult to reach a consensus to what it is. We need to understand which requirements are important to sustainability, and to which dimension, since the same requirement can have different relative importance's on different dimensions. Only when we know this, can we draw conclusions and start working on the approach and applying software engineering techniques. The solution for this challenge lies on the systematic search done.

Challenge 4 → Is there a generic solution?
A solution for any problem is only useful if first, it works, and second, if it is useful given different contexts. So to speak, the solution needs to be as general as possible while providing the expected results. The way to overcome this challenge was to design an approach capable of being configurable according to the user's needs, and to cover as many sustainability requirements as possible. By doing this, the solution is the most generic possible while generating the optimal result for a particular context.

1.3 Objectives

The aim of this dissertation is to collect the body of knowledge in the area and propose a catalogue for sustainability requirements to facilitate the reuse of sustainability properties and relationships during software development. This catalogue takes into account the requirements that are important and relevant for an application domain while considering any combination of four (out of five) sustainability dimensions. The main objective lies on developing a working tool that systematizes the sustainability requirements in a configurable and understandable way. For that we made an in-depth study of the state of the art on two main topics: requirements engineering and sustainability requirements. The catalogue is one of its kind, since it combines all the information mentioned above with the goal of being as general as possible to accommodate the maximum number of domains of software development. One important thing to know is that our catalogue does not provide a solution for the problem at hand, but rather a set of requirements that the system needs, or at least, should consider, to solve such problem. Before implementing a solution, we must first know what we need for it to be viable and effective.

1.4 Contributions & results

This dissertation aims to contribute and help software developers with an approach that allow more efficient specification of sustainability requirements, with the goal of developing sustainable software. By taking into account the needs of the various sustainability dimensions and combining that with requirements engineering techniques, the result is a sustainability catalogue (with tool support) that highlights the most important requirements for sustainability given an application domain. Since this catalogue is generic, it has more field of action, which leads to a possible increase of sustainability in wider domains. The outcome of the above mentioned contributions are the following results:

- A sustainability catalogue that covers the social, economic, environmental and technical dimensions and respective properties, such as requirements, goals, and relationships. The sustainability catalogue is our main artifact and contribution of this MSc thesis.

- A catalogue for each dimension, which can be perceived as a sub-catalogue of the sustainability catalogue. Each one of these catalogues also address the respective properties of each dimension.
- Three plugins developed as an extension of the piStar tool. The main plugin is responsible for the configuration of the catalogue, where the user can filter the catalogue to only show certain type of requirements he wants. The other two plugins, one is responsible to label the elements of the catalogues, while the other one label the colors of the elements.
- An assessment of the sustainability catalogue, considering its readability, interest, utility and usefulness.

1.5 Structure of the document

This document is composed of the following six chapters:

- **1. Introduction** → This chapter introduces the theme to be addressed by this dissertation, exposing the context, motivation, problem statement, objectives and contributions of the work.
- **2. State of the Art**→ This chapter provides an overview of sustainability and requirements engineering and their processes. It also contains the research work made during the initial period of this dissertation, particularly, a systematic mapping study. It discusses the research methodology for such study, as well the results and a discussion of the search done.
- **3. Sustainability Catalogue Conceptualization** → This chapter introduces the conceptualization of the core end result of our work, the sustainability catalogue. It presents and explicitly details each one of its constituents, and further analysis the numerous relationships between them.
- **4. Sustainability Catalogue Implementation** → This chapter focuses on the implementation of the sustainability catalogue, and the development decisions we took and what were the reasoning behind them. We also present the functionalities of our catalogue and some application examples.
- **5. Evaluation** → This chapter contains the steps performed to do an assessment of our work. This assessment aims for us to evaluate our sustainability catalogue across different aspects of it. It also presents all the results and respective discussion.
- **6. Final Conclusions & Future Work** → This chapter clarifies our conclusions and final appreciations. It also provides some ideas for future work to be developed aligned with the topic of this dissertation.

CHAPTER 2

State of the Art

In order to have a thorough analysis of the state of the art, we performed a systematic mapping study. In this chapter, we first present and detail all the phases needed to carry out a systematic mapping study, and latter we introduce the reader to the themes that this dissertation is based on: requirements engineering and sustainability requirements.

2.1 A systematic mapping study

The first step in our research work was to make a thorough analysis of the state of the art on requirements engineering and sustainability, for that we performed a systematic mapping study. A systematic mapping study aims to provide an overview of a research area and synthesize and report the results available. The typical process is done by selecting one or more research questions and planning a search and study selection strategy. After collecting and extracting the data, the results are analyzed against each research question. All the phases needed to carry out a systematic mapping study are presented and detailed in this section.

2.1.1 Research methodology

A mapping study process is composed of three main phases: planning, conducting, and discussion [40]. The planning phase aims at defining the research questions, the search and study selection strategy, as well as defining the data to be extracted. The conduction phase shows the execution of the search while presenting the results for each research query. Finally, the discussion phase intents to present a detailed analysis taking into account the results given in the previous phase. The methodology, for the search in general, was to first gain an overview of techniques regarding elicitation, prioritization and trade-off analysis. Secondly, what requirements are important or relate to sustainability (see Section 3.1).

Last but not least, if there is any approach that combines these two main topics, or which ones do already exist to recognize what has been done and what needs to be done.

Each search string, presented on Section 2.1.1.1, was run, solely, in a general digital library, DBLP, as it compiles a large amount of publications from different sources, such as: IEEEXplore, Science Direct, and SpringerLink. These are the sources that cover the most important forums in Software Engineering, and Computer Science in general. It indexes the best journals, conferences and workshops in Computer Science. Also, DBLP has a great search engine with reasonable matches for each query. ACM Digital Library and Google Scholar, although very good sources, for each query we had thousands, or even millions, of matches which was not feasible, for us, to read and choose those which were suited for this study. Also the number of false positives was too high.

2.1.1.1 Planning

In the first phase of the mapping study, we start by formulating the research questions and respective search string to run in the digital library. Next we define the search and studies selection strategy, such as inclusion and exclusion criteria. An assessment of the quality of the studies was also done. Finally, we specify what data should be extracted from the selected studies, i.e., what information is relevant. All these phases are presented in the following sections.

Research questions

The definition of the goal of our study follows the **PICOC** [41] structure, as described in Table 2.1. A PICOC analysis is a method used to describe the five elements of a searchable question.

Table 2.1: PICOC Analysis.

Population	Papers addressing sustainability and its requirements as well approaches for elicitation, prioritization and trade-off analysis of those same requirements on the maximum types of industries, systems and application domains.
Intervention	By searching on electronic libraries using specific queries containing relevant keywords.
Comparison	Given a set of criteria and by understanding the main characteristics of the existing approaches for assessment, prioritization and trade-off analysis, compare them and choose the ones that are better or fit better in this context.
Outcome	Overview and analysis of the state of art.
Context	Preparation for the Master Thesis on Informatics Engineering.

This led to the definition of the following research question: "*What are the existing approaches for the elicitation, prioritization and trade-off analysis of requirements for sustainability?*". For a more complete search, we subdivided the main research question into four, where each one has a research query associated with the key terms, which makes a search string. The four sub research questions (RQ1, RQ2, RQ3, and RQ4) and the main research query (Main RQ), with the respective research query, are presented in Table 2.2.

Table 2.2: Search query building for the various research questions.

Research Question	Approach	Elicitation	Prioritization	Trade-off Analysis	Requirements	Sustainability
What are the existing **approaches** for **requirements elicitation**?	(method | process | technique | model | tool | approach | framework)	(elicit)	NA	NA	(requirement | attribute)	NA
What are the existing **approaches** for **requirements prioritization**?	(method | process | technique | model | tool | approach | framework)	NA	(important | priorit)	NA	(requirement | attribute)	NA
What are the existing **approaches** for **requirements trade-off analysis**?	(method | process | technique | model | tool | approach | framework)	NA	NA	(trad | conflict | relation)	(requirement | attribute)	NA
What are the **requirements** that contribute or relate to **sustainability**?	NA	NA	NA	NA	(requirement | attribute)	(sustain | green)
What are the existing **approaches** for **elicitation**, **prioritization** and **trade-off analysis** of **requirements** for **sustainability**?	(method | process | technique | model | tool | approach | framework)	(elicit | important | priorit | trade | conflict | relation)			(requirement | attribute)	(sustain | green)

Some important things to note about the research queries are: " | " represents an "*OR*" operator; the first column shows the research question with the key terms in bold; the following columns, one column for each key term, are connected with an "*AND*" operator; "NA" stands for non-applicable; words like "sustain" are incomplete since the search engine can search any variation of this word, for example, "sustainable" or "sustainability".

Search strategy

The search strategy used two complementary search methods: automatic and manual. The automatic search used the search strings for each research question from Section 2.1.1.1, to find primary studies on electronic data sources (see Section 2.1.1). After the automated search was made, we proceeded to make a manual search to complete the previous one. This manual search was made using two approaches: snowballing [61], by using the reference list of each selected study performed to identify additional potentially relevant studies, and keyword manual search. The keyword manual search was shaped by joining a couple of keywords from each search query, and scroll through the most relevant articles containing these keywords on Google Scholar. The search was done using the following terms:

- **Elicitation** → Requirements elicitation techniques
- **Prioritization** → Requirements prioritization techniques
- **Trade-off analysis** → Requirements trade-off analysis techniques
- **Sustainability requirements** → Sustainability requirements

Study selection strategy

To help selecting the relevant studies for analysis and data extraction, inclusion and exclusion criteria were defined. The study selection was done in two different iterations. In the first iteration, we analyzed the title and abstract of all candidate studies. Next, with the pre-selected candidate studies, we performed the second iteration through full text reading. The studies only pass to the second iteration, or are chosen as final studies, if they confirm all of the inclusion criteria, and are excluded if they confirm one or more exclusion criteria. The inclusion and exclusion criteria for the first iteration are listed in Table 2.3, and for the second is presented in Table 2.4.

Table 2.3: Study inclusion and exclusion criteria for the first iteration.

ID	Study inclusion criteria
IC1.1	The article is available in English
IC1.2	The article is from a conference, workshop or journal
IC1.3	The article is from year 1987 to the point of conducting the search (2018). 1987 was chosen as the starting year because it was on this year that the topic "Sustainable development" was approached on United Nations [35] (This criteria only applies to the main research question and to the fourth sub research question)
IC1.4	The article is available in full text
IC1.5	The title and/or abstract of the article is related to one of the research questions
	Study exclusion criteria
EC1.1	The article is a duplicate
EC1.2	The article is not in English
EC1.3	The article is informal (slides, extended abstracts, blogs)
EC1.4	The article is older than 1987 (only for the main research question and to the fourth sub research question)
EC1.5	The title and/or abstract does not relate to any of the research questions

Table 2.4: Study inclusion and exclusion criteria for the second iteration.

ID	Study inclusion criteria
IC2.1	The article content discusses and/or answers any of the research questions in a direct way, i.e., if it's a summary, a comparison, a validation or a proposal of different approaches regarding one or more research questions
	Study exclusion criteria
EC2.1	The content of the article does not answer any of the research questions
EC2.2	The article mainly discusses challenges and problems in this domain but does not provide any beneficial solution or suggestion to solve such problem

Assessing the quality of the studies

The quality of the primary studies is critical for obtaining trustworthy results in empirical studies [28]. For each study we focused on two variables. The number of citations, which are presented in the respective digital library, that the study has, and its CORE Rank [1]. The first was chosen as a quality attribute since a study with a high count of citations is, probably, more trustable than a paper with little to none citations, although the year of the paper can take a major role on it. The CORE Conference Ranking provides assessments of major conferences and journals in the computing disciplines [14]. The ranking has four different levels. Rank C, satisfactory, B, good, A, very good, and A*, excellent.

Data collection and extraction

To promote the understandability of what's relevant, and facilitate the data extraction, an excel worksheet was created, where each study has a corresponding sheet. Each sheet has the same template, containing the basic information of the paper, such as, title, citations, year, and a section for each sub research question. In this section, we present relevant information (if any) regarding the research question, accompanied by how the validation was made. An example of a sheet is presented in Appendix A.

Protocol review

This systematic mapping study protocol should have undergone a review process. This process is typically done by external entities. However, given the short time, this step was not performed, being only evaluated and commented by the advisers of this MSc dissertation.

2.1.1.2 Conduction

The conduction of the search was done by running the search strings (2.1.1.1) on DBLP. The first step was to select papers based on title and abstract while applying the first iteration criteria shown in Table 2.3. Then, the papers that successfully passed the first step, were fully read while applying the second iteration criteria listed in Table 2.4. Finally, the relevant data and validation were extracted and added to a spreadsheet workbook previously structured as a form (see Appendix A). The date of this procedure was December 13th, 2018. A synthesis of this process, with the number of articles first encountered and excluded by the various criteria, and with the final results, is shown in Table 2.5. From Table 2.5, we can see that the final total number of articles is 24. However, as previously mentioned in Section 2.1.1.1, a manual search was done to complete the automatic search. From this manual search, 7 more articles were added, where 3 were through snowballing and 4 from the keyword manual search. Therefore, a total of 31 articles were chosen as final and primary studies.

[1] http://portal.core.edu.au/conf-ranks/

Table 2.5: Synthesis of the automatic search.

	1st Sub Query	2nd Sub Query	3rd Sub Query	4th Sub Query	Main Query
1st Matches	375	113	193	169	5
EC1.1	-4	-4	-5	-4	0
EC1.2	-7	-5	-7	-10	0
EC1.3	-13	-6	-9	-15	0
EC1.4	NA	NA	NA	-6	0
EC1.5	-337	-77	-165	-117	-5
2nd Matches	14	21	7	17	0
EC2.1	-5	-6	-2	-4	0
EC2.2	-5	-5	-2	-6	0
Final Total	4	10	3	7	0

2.1.1.3 Discussion of the results

After we conducted the search and extracted the relevant information (Section 2.1.1.1), we were finally ready to answer our research questions and discuss the results of this study. This discussion will take into account the articles found in the automatic search as well as the manual ones. One thing to notice is that more than half of the articles found in the automatic search were also found in the manual search, which validates, in part, the efficiency of the search queries. This section is divided in various subsections, one for each sub-research question and another for the main one,highlighting the relevant information associated for each one.

RQ1. What are the existing approaches for requirements elicitation?

Although this sub question was the one with more first matches, it was the second last in second matches and in final studies. A reason behind this could be the fact that elicitation is a broad topic and we can make an elicitation of almost anything within a domain. Such reason leads to many articles being too specific, which is not of interest, since what we want is to have an overview of the techniques that exist for the elicitation of requirements. From the four final studies related to this question, two articles ([37], [46]) presented a detailed explanation of various elicitation techniques, while the other two ([47], [9]) offered a guide and framework to help requirements engineers to select the most adequate elicitation techniques at given time. To answer our question, there is a wide variety of elicitation approaches of requirements, however only some show a good effectiveness [37]. The main approaches are: interviews, questionnaires, brainstorming, use cases, workshops, focus groups, and prototyping.

RQ2. What are the existing approaches for requirements prioritization?

We found a relative high amount of studies that offer a list of techniques regarding prioritization of requirements. Some offer a comparison of each one based in selected criteria ([20], [55], [26], [29]) , others present strengths and weaknesses of each technique ([22]), while others only detail and explain the process behind each technique ([25], [31]. We

also selected two studies where each one proposes a new prioritization technique, weight assessment model [21], and RePizer [27]. The rest of the studies that we have chosen speak of frameworks and guidelines for the prioritization of requirements ([53], [4], [30]) and the importance that this topic has in the development of a project and how we can improve the effectiveness of prioritization ([32]). From this research, we find out which techniques are most used in the real world and which present better results. They are: Numerical Assignment, MoSCoW, Bubble Sort, Binary Search Tree, Hundred Point Method, AHP, Minimal Spanning Tree and Planning Game.

RQ3. What are the existing approaches for requirements trade-off analysis?

An interesting fact that we have noticed in trade-off analysis studies is that, contrary to what we have seen in elicitation and prioritization, there is no published work offering a list of techniques, nor a comparison of the various. It is also added that different approaches can produce very different results. The main two results that we found were: a way of showing the relation between the various requirements ([65], [11]) in terms of contribution in order to know if one requirement help or harms another one(s); a selection of the best candidate requirements, i.e., the ones that have a greater value for the application domain. Some approaches were found: EvenSwap method [18], Quantitative WinWin [45], i* framework [15], NFR framework [11].

RQ4. What are the requirements that contribute or relate to sustainability?

From this research we have selected articles that talk about software sustainability and its importance ([17], [8], [7], [10], [6]), models and frameworks ([44], [33], [39]) for a better treatment of sustainability in software development, and finally articles that specifically relate requirements and sustainability ([12]). The main thing we discovered was that there is an urgent need of addressing sustainability in software development and the treatment of requirements for sustainability, specially in trade-off analysis [12].

Main RQ: What are the existing approaches for the elicitation, prioritization and trade-off analysis of requirements for sustainability?

As we can see from Table 2.1.1.2, we only had five matches, where none passed to the second iteration, let alone to be considered a final study. Such finding, represents that there is little to nothing that relates elicitation, or prioritization, or trade-off analysis with sustainability requirements. A reason behind this could be the fact that software sustainability and sustainable development, is still an immature and new topic, where even the concept and what it does mean is not clear for the science community. Knowing this plus the fact, as previously mentioned, that there is a need to relate requirements to sustainability, as well to be a treatment, specially in trade-off analysis, of requirements so that software could be more sustainable, makes this work an important and useful

one. Therefore, we will focus our efforts in developing an approach capable of showing the various sustainability requirements and its impacts to one another.

2.1.2 Challenges revisited

With this study we were able to analyze the current state of the art, both regarding sustainability requirements and requirement engineering activities (Challenge 1 of the problem statement, 1.2). We gathered numerous techniques for requirements elicitation, prioritization and trade-off analysis, while understanding their process and constraints of use (Challenge 2). We were able to find out which requirements relate and are relevant to sustainability (Challenge 3) [12], however, a more detailed study should be made on this topic. Finally we discovered that there is yet a solution to be developed that combines these two topics, let alone a generic one, however we know where our focus needs to be, the analysis of trade-offs of sustainability requirements (Challenge 4).

2.1.3 Threats to validity

No research work can be flawless in its entirety. There are always threats to the validity of any study. Nonetheless, we are able to mitigate those threats. A validity threat of our study is the accuracy of our automated search. We tried to optimize our search strings, however, they can be incomplete or inaccurate. Studies could be left out, simply because they don't have the keywords we chose in their titles. To mitigate this issue, a manual search was done to make sure that the number of relevant articles that could be left out, was the minimal possible. Non-English studies and unavailable ones were excluded, and, therefore, we could have missed some articles that could be relevant to our study. Finally, since the study selection was only made by one person, and the number of articles to analyze was high, such selection could be biased or some studies could be left out due to fatigue. A list of inclusion and exclusion criteria was made in order to facilitate the study selection.

2.2 Background

One of the results of our systematic mapping study was the identification of several works that form the foundations of our work. Thus, this section aims to introduce the reader to the themes that this dissertation is based on: requirements and sustainability engineering. For the former, a brief description is presented containing the basics of what requirements engineering is, followed by its activities and associated techniques. In the latter, an explanation is made about what sustainability is, its challenges and possible solutions, and its importance in software development.

2.2.1 Requirements engineering

To understand what requirements engineering is, we first must comprehend the definition of *requirement* in software engineering. A requirement is a physical or functional need that a particular design, product or process aims to satisfy [48]. There are two types of requirements: functional and non-functional. The first one describes what the system should do, i.e., a function or a behaviour, while the second one describe how the system should behave; they specify the system's "quality characteristics" and restrictions. Knowing this, we can define *requirements engineering* as the process of defining, documenting and maintaining requirements in the engineering design process [48].

Requirements engineering is crucial on software development. When starting a project, we should include all the features and functions a system should have in order to accommodate all the stakeholders needs. Requirements should be comprehensible for all interested parties and be free of any ambiguities. Accurate requirements helps to estimate more precisely the time and money that will be spent on certain features while mitigating possible risks of implementation [48].

Requirements engineering have different activities, which facilitate to understand the stakeholders needs, define the system requirements and constraints, analyze them, evaluate their feasibility, validate their specification and manage them. Indeed, requirements engineering encompasses the following five major phases [46]:

- **Elicitation** → Elicitation is the process of engaging with stakeholders to elicit and understand their business needs and requirements. The identifying of system requirements is the main objective of this activity.

- **Analysis and Negotiation** → After the identification and documentation of requirements we must analyze and negotiate them. Some processes of requirements analysis are: classification, resolution of conflicts and prioritization. In negotiation, we must justify solutions and benefits for all parties involved and understand and explain what approach is better in deterioration of another. This second phase, consists of a set of processes aimed to discover problems within the system requirements and achieve agreement on changes to satisfy all stakeholders involved.

- **Documentation** → This activity aims to put the requirements in a documented format which will make clear sense to the stakeholders as well as the technical teams. In order to be perceptible, the requirements should be clear, complete and accurate.

- **Validation** → This activity is related to ensuring that models and documentation accurately express the stakeholders needs along with checking all the system specification and requirements so that there is no margin left for error or misunderstanding.

- **Management** → Since any software development is dependent of the needs and desires of stakeholders, which translates on requirements, it leads to a volatile environment.

Therefore, the software can suffer changes as we are developing it, but be that as it may, we must manage those changes in relation to our requirements. All the changes should be tracked efficiently and accurately so that there is no process derailment. Even if no changes are made during the software development, we must ensure the system maintenance.

The following sections present three distinct requirements engineering processes: elicitation, prioritization, and trade-off analysis. Each of these processes is explained and a set of respective techniques used are detailed.

2.2.1.1 Elicitation

Requirements elicitation is the practice of researching and discovering the requirements of a system from users, customers and other stakeholders [48]. There are a variety of techniques for elicitation of requirements, but for the purpose of this dissertation, we will only consider those that have demonstrated a good effectiveness as stated in [37]. These techniques are presented below, and also a summary containing all of them with the respective characteristics and recommended conditions is presented in Annex I.

- **Interviews** → Interviews are a human based social activity, where typically, an interviewer asks questions to someone related to the application domain with the intent of gather large size of information. The quickness of gathering and usefulness of the information is strongly correlated to how experienced the interviewer is [46]. Interviews can be structured or unstructured. In the first one, the goals and contents have been prepared before hand, while on the other, they were not [9]. This technique is useful since it is effective to elicit relevant aspects from the stakeholders' needs and in quickly obtaining complete information [37].

- **Surveys/Questionnaires** → This technique is based on a set of questions on paper, or online, sent to one or more informants [9]. The questionnaires can have open or closed questions, and must be clear, well defined and concise along with the domain knowledge. It is an efficient way to collect information from multiple stakeholders while being one of the methods with lower costs and time consumption associated [46].

- **Scenarios/Use cases** → It consists of a description of a set of possible actions, scenarios and events that describe part of the system behaviour [9]. They are useful to understand the general requirements of the system as well as to identify ambiguous requirements.

- **Workshops** → Workshops are a meeting with a set of clear objectives. During a workshop, a group of people engage in intensive discussion and activity in order to reach the agreed objectives. They are useful since it is easier to solve conflicts

when interested parties are present and encourage cooperation and consensus among participants [37]. Nevertheless, it is very time consuming and can be really hard to have all the participants in one room due to conflicting agendas or geographical issues.

- **Brainstorming** → Brainstorming is a group technique for generating preliminary ideas about the software to be developed in the project. Participants come up with ideas, deliberately and in no particular order, in a familiar discussion to quickly produce as many as possible without focusing on a particular one [46], [9]. One of the major advantages of using brainstorming is that it inspires creative ideas which can be used to solve existing problems in new and advanced ways [46]. Although this technique is similar to workshops, it presents a major difference which is the fact of not having well defined objectives.

- **Focus Groups** → Focus groups are a mixture of interview and workshops. It is a semi-structured group interviews where the open discussion is encouraged by an interviewer, whose job is to elicit the requirements [9]. It promotes collaboration which translates in effectiveness to solve conflicts [37].

- **Prototyping** → Prototyping is basically the development of a simplified version of the system under construction to help capture the wanted requirements [9]. They are typically developed using preliminary requirements or existing examples of similar systems. It is mainly used on complex systems and on the development of human-computer interfaces [46].

2.2.1.2 Prioritization

In software development, decisions must be made among numerous choices. Decisions such as, "which requirement should be implemented first?". As all the requirements cannot be programmed in a single increment, we do not know which requirements have higher priority. This issue is more difficult when more stakeholders are involved. Requirements prioritization appears to help developers make such decisions. This activity can be defined as the process to determine the implementation order of the requirements for implementing the system or the process to determine the order of importance of the requirements to the stakeholders [32]. In short, requirements prioritization, says what the most important requirements of the system are.

This process is very important since projects face limited resources such as short timelines, small budgets, limited human power and technology resources. Therefore, a project can contain different sets of requirements to be implemented in a release time. Requirements prioritization helps the project developers to select the most appropriate set within resource constraints. However, when many stakeholders are involved in a project, decisions are harder to make, since different stakeholders, typically, have different perspectives. For example, financial managers look for requirements with low cost, market

managers look for requirements with market value while end users look for requirements which are easy to use. It can be a challenge to reach a consensus among all stakeholders. Requirements prioritization excels at uncovering the most important requirements to maximize the stakeholders' satisfaction.

As mentioned above, any project has constraints regarding resources. Therefore, a prioritization of a certain requirement should consider different aspects. The main aspects are: importance, time, cost, penalty, and risk [31]. A brief description of each is presented below:

- **Importance** → Importance might be the most important aspect, however, since different stakeholders have different understandings of this concept, it is essential to specify the meaning of "importance" to reduce the confusion among stakeholders [31].
- **Time** → Time is the time spent on, successfully, implementing the candidate requirement. There are some factors that can influence this measure, such as degree of parallelism in development and staff training time [31].
- **Cost** → Cost is the money spent on, successfully, implementing the candidate requirement. Cost can be directly influenced by staff hours and extra resources needed.
- **Penalty** → Penalty is how much it is necessary to pay if a requirement is not fulfilled. Although some requirements may have low values of importance, but failing to fulfill them may cause a high penalty [31].
- **Risk** → Risk is defined as the degree of likelihood that a project will fail to achieve its time, cost or quality goals. Each requirement has a risk associated and consequently the conjunction of all system requirements defines the risk of the project. Project risk can be associated with unrealistic schedules and budgets, developing the wrong functions, continuing stream of requirements changes, or adding more features than necessary [31].

To help software developers prioritize requirements, a panoply of techniques are available, each one with specific strengths and weaknesses. It is quite difficult to say which technique is the best one, since they produce different results in different situations. To maximize the efficiency of requirements prioritization, we should understand the conditions of our projects and choose the technique that suits best. A combination of techniques may also be a valid and viable option [55]. Techniques can be categorized as nominal scale, ordinal scale and ration scale [31]. A description of them, grouped by their category, is presented below, while a summary containing all these techniques with a brief description, respective strengths and weaknesses, and recommended size of requirements set is presented in Annex II.

Nominal scale

On nominal scale, requirements are assigned to different priority groups, with all requirements in one priority group being of equal priority. The main techniques that use this scale are [31]:

- **Numerical assignment** → It is a simple technique based on grouping requirements into different priority groups. The number of groups can vary, but it is common to have three, "low", "medium" and "high" [26].
- **MoScoW** → This technique is very similar with numerical assignment, but it groups the requirements in the same four priority groups. "MUST have", "SHOULD have", "COULD have"importance" and "WON'T have" [31]. **Must have** means that requirements in this group must be implemented before release. **Should have** means that the implementation of these requirements is beneficial. **Could have** is similar with **Should have** but are less important. **Won't have** means that these requirements are of low priority and can be implemented in later iterations [26].

Ordinal scale

Ordinal scale methods result in an ordered list of requirements. The following techniques use this scale to prioritize requirements:

- **Bubble sort** → As the name says, it is a method for sorting elements. The idea behind it is that the users compare two requirements at a time and swap them if they are in the wrong order [31]. Basically we outline the requirements in a vertical column, then we compare the two top requirements, and if the second one is more important than the first one, we swap. Repeat this process until we reach the bottom and until no swaps are left to be made [55]. The result of this process is a ranked column of requirements where the most important requirement is at the top, and the least important one is at the bottom.
- **Binary search tree** → This is also a method for sorting elements, where each node of the tree contains, at most, two children. The idea of the binary search tree method for ranking requirements is that each node represents one requirement. All requirements placed on the left sub-tree of a node are of lower priority and the ones placed on the right, are of higher priority [31].

Ratio scale

Ratio scale methods provide the relative difference between requirements. The techniques that follows, all use some sort of process to ensure that each requirement has an "importance value" associated:

- **Hundred point method** → The idea of this technique is that each stakeholder has 100 points (hours, money, etc.) to distribute to the requirements [26]. The result

is presented on a ratio scale which can provide the information on how much one requirement is more/less important than another one [31].

- **AHP - Analytic hierarchy process** → AHP is the most widely studied requirement prioritization technique [55]. The basic idea of AHP is to calculate the priorities of requirements by comparing all unique pairs of requirements to estimate their relative importance. In other words, the one performing the comparison has to judge and decide which requirement is more important, using a scale from 1 to 9, where 1 indicates equal value and 9 indicates extreme value [20]. 2, 4, 6 and 8 are intermediate values between two judgments [25]. The scale is shown in Table 2.6.

Table 2.6: Basic scale of AHP, from [25].

How Important	Description
1	Equal Importance
3	Moderate Difference in Importance
5	Essential Difference in Importance
7	Major Difference in Importance
9	Extreme Difference in Importance

This comparison is made by putting all requirements in a matrix n x n, where n is the number of requirements. 1s are put on the diagonal of the matrix, since a requirement has equal importance with itself. Then, for each unique pair of requirements, insert the relative importance. At the same time, the reciprocal values are inserted to the transposed positions. Finally, to get the relative priority of each requirement, the eigenvalues of the resulting comparison matrix are needed to be calculated. The final result is, consequentially, the relative priorities of the requirements [55]. An example matrix using AHP is shown in Figure 2.1.

	SR-1	SR-2	SR-3	SR-4	SR-5
SR-1	1	8	1/5	3	1
SR-2	1/8	1	1/5	1/7	1/7
SR-3	5	5	1	1	2
SR-4	1/3	7	1	1	1/2
SR-5	1	7	1/2	2	1

Figure 2.1: An example matrix created with AHP, from [55].

AHP requires a total of n x $(n - 1) \div 2$ comparisons which make AHP not that scalable. This means that for higher number of requirements AHP is not that suitable. However tools are available which mitigate this problem [26]. Another way to solve the scalability problem encountered with AHP is to use Hierarchy AHP. Since requirements are often structured in a hierarchy in large projects, Hierarchy AHP only prioritizes the requirements that are at the same level of hierarchy. This method reduce the number of pair wise comparisons, since not all the requirements are compared [31].

- **Minimal Spanning Tree** → It consists of a directed graph which is minimally connected. It is a subset of the edges of a connected, edge-weighted and non-directed graph that connects all the vertices together, without any cycles and with the minimum total edge weight. While it eliminates redundancy and the number of pair wise comparison's, it is very unreliable and has poor fault tolerance [22].

Combining techniques

As the name says, combining techniques are techniques that combine two or more techniques to wield better results. An example could be by joining numerical assignment and AHP. We would first group the requirements and then apply AHP to each group. Ergo, for each group we would know which requirements are most important. This could be useful if we are dealing with a large number of requirements or if we want a narrow scope. An existing combining technique which wields good results is presented below [20]:

- **Planning game** → Planning Game combines the numerical assignment technique and ranking technique together to prioritize requirements [31]. Requirements are first prioritized into three groups: "those without the system will not function", "those that are less essential but provide significant business value" and "those that would be nice to have". At the same time, the programmer estimates how long each requirement would take to implement and then begin to sort the requirements into three different piles: "those that can be estimated precisely", "those that can be estimated reasonably well" and "those that cannot be estimated at all". The final result, is a sorted list of requirements on an ordinal scale [20].

2.2.1.3 Trade-off analysis

A trade-off is a decision that involves losses of, or decrease, one quality, aspect, property of something in return for gains in other aspects. In simple terms, a trade-off is when we give up on something to get more of what we want. This concept suggests a tactical or strategic choice made with full comprehension of the advantages and disadvantages of each possibility. Taking this topic to software engineering, it is clear that it is an important one with a broad spectrum of action. The overall effort that can be spent on a project is limited in terms of money, time, human resources, and the final product has to fulfill certain minimum quality requirements. Therefore, decisions of what should be implemented is very important. However, we can not make such decisions without context. We need to understand, in the most possible extent way, the implications that each requirement has on another ones and in the general project overview. Let us take an example: we have four requirements, A, B, C and D; Requirement A is very important but it's implementation harms requirements B and C, which are equally important; Requirement D is of medium importance but its implementation avails requirements B and C. We need to know which requirement should be implemented first taking into account all the dependencies related

among them. Trade-off analysis aims to help making such decisions while making the different objectives and constraints more transparent. More transparency regarding these relationships supports the negotiation for the most appropriate solution in the context of a concrete project [45]. Approaches have been proposed to help software developers do trade-off analysis of requirements. A description of some is presented below:

- **NFR framework** → Non-Functional Requirements framework (NFR framework) is a common solution regarding trade-off analysis. Despite being a "NFR framework", it also addresses functional requirements of the system, such as operationalizations. There are several major steps in the design process: acquiring knowledge about the system domain; identifying particular NFR's for the domain; decomposing NFR's; identifying "operationalizations"; dealing with ambiguities, trade-offs, interdependencies; selecting "operationalizations"; supporting decision with design rationale; evaluating the impact of decisions. These are not necessarily sequential steps, and one may need to iterate over them many times during the design process [11]. The final result is an interdependency graph containing the relations that each element has. An example of a NFR framework is shown in Figure 2.2.

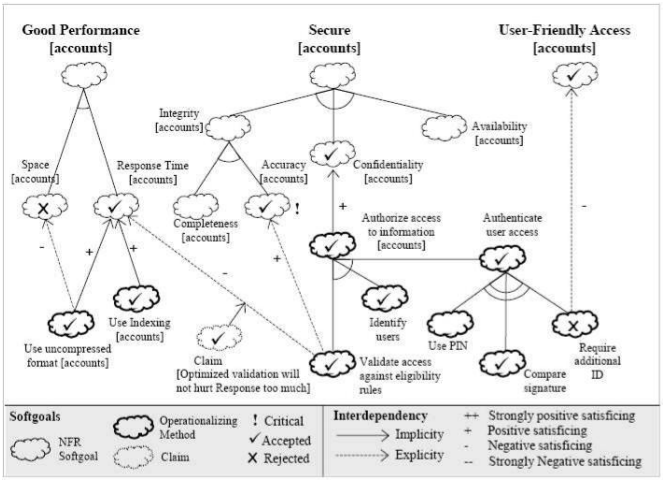


Figure 2.2: Softgoal interdependency graph, from [63].

- **FQQSIG** → The fuzzy qualitative and quantitative software interdependency graph (FQQSIG) method has on its core the NFR framework. However, it modifies the NFR framework with some very interesting and useful changes. The NFR framework does not depict the value of each relation. It only gives a general overview of the relationships between requirements. Yet, an importance degree among them would be beneficial. The fuzzy qualitative and quantitative software interdependency graph (FQQSIG) method tries to solve this issue [65]. This model integrates qualitative and quantitative methods to describe, analyze, calculate and evaluate the NFR alternatives as well as the relationships amidst them [65]. It uses fuzzy linguistic variables to indicate the experts assessment, a defuzzication process in order to quantify those

same assessments, and a relation matrix to calculate the nodes (requirements) degree of importance. The result is an interdependency graph, as shown in Figure 2.3, which helps the decision-makers select the optimal NFRs [65].

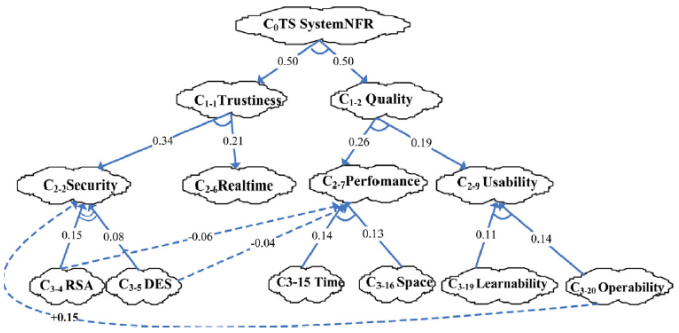


Figure 2.3: FQQSIG example, from [65].

- **Quantitative WinWin** → Quantitative WinWin supports the determination of "best" requirements by using quantitative models in conjunction with an iterative approach that adapts to changing sets of requirements and evolving problem parameters. It focus on value-based selection of requirements and resolution of conflicts in the presence of quality, time and effort constraints [45]. As a final result, this approach generates a small set of the most promising candidate solutions from which the decision-maker can select the one which is the best match with additional implicit and subjective preferences depending on the context [45]. This methods improves the transparency and efficiency of decision making.

- **Even swaps** → In Even Swaps, stakeholders qualitatively compare consequences of alternatives on decision criteria. This method extracts stakeholder's value trade-offs, i.e., how many stakeholders would give up on one goal for more satisfaction than another. Stakeholders are asked to specify the extent of the difference between consequences of alternatives on each goal. Value trade-offs are obtained by asking stakeholders to hypothetically relax their demand on one goal in order to gain more value of another goal. This method is supplemented with a semi-automated decision aid tool which takes the value trade-offs and alternatives comparisons and finds the optimum solution, through interaction with stakeholders [18].

- **iStar framework**→ The iStar (or i*) framework was presented in the mid-nineties as a goal- and actor-oriented modeling and reasoning framework. It consists of a modeling language along with reasoning techniques so that one can analyze created models [15]. In i*, the central conceptual modeling construct is the **actor**. It is an abstraction referring to an entity (human, software, hardware). Actors are autonomous entities that aim to achieve their goals [64]. Furthermore, actors can be separated into a role (abstract) or an agent (concrete). An actor has a respective boundary, inside it are the **intentional elements**. Intentional elements are the things actors want, they model

different kinds of requirements. There are four different types: goal, quality, task and resource. Both actors and the intentional elements can be linked between each other. **Social dependencies** represent social relationships between actors, it must exist an actor that depends for something (a quality, a goal or a task) to be provided by another actor. **Intentional element links** as the name says, links intentional elements. There are four types of links: contribution (help, make, hurt, break), refinement (AND, OR), needed by and qualification. However, there are rules on which link to use given the linked elements. Figure 2.4 explains these rules.

		Arrowhead pointing to			
		Goal	*Quality*	*Task*	*Resource*
Link starts from	*Goal*	Refinement	Contribution	Refinement	n/a
	Quality	Qualification	Contribution	Qualification	Qualification
	Task	Refinement	Contribution	Refinement	n/a
	Resource	n/a	Contribution	NeededBy	n/a

Figure 2.4: Links between intentional elements: overview, from [15].

Finally there are two types of iStar models, the Strategic Rationale model (SR) and the Strategic Dependency model (SD). An SD model describes a network of dependency relationships among various actors in an organisational context, while a SR model allows modeling of the reasons associated with each actor and their dependencies, and provides information about how actors achieve their goals [15]. An hybrid SD/SR model example is shown in Figure 2.5.

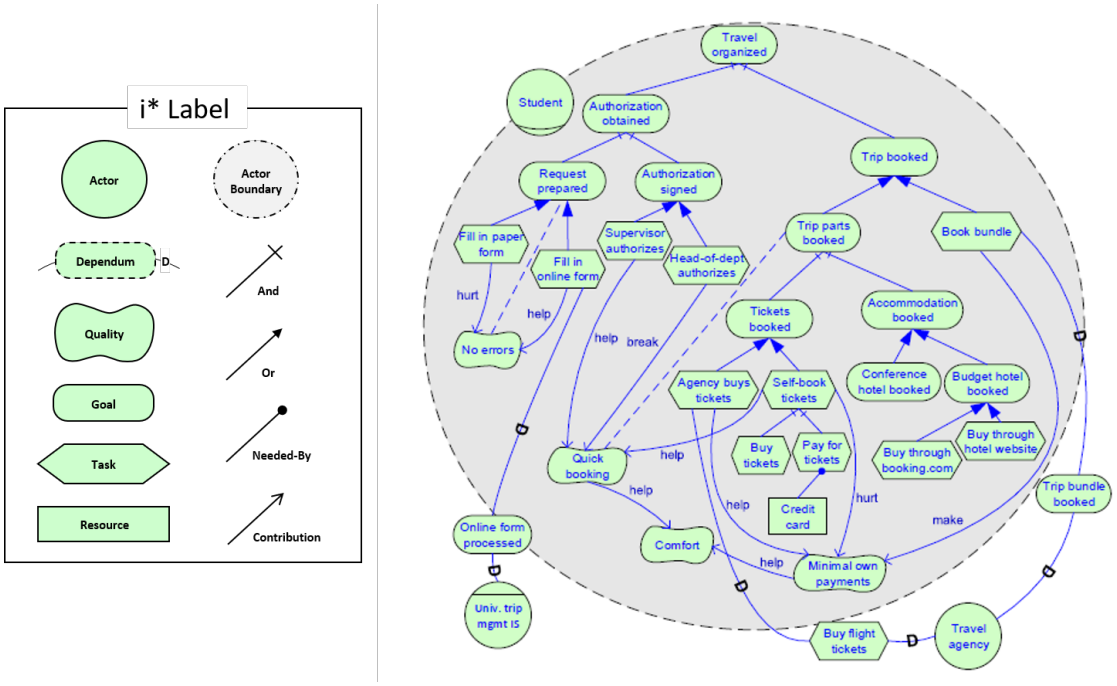


Figure 2.5: iStar hybrid model example, from [15].

2.2.2 Sustainability

Sustainability focuses on meeting the needs of the present without compromising the ability of future generations to meet their needs [58]. From this definition it is clear that sustainability is very important, since if we want to leave a better world for our children, we need to be conscious of the actions we do. But since complex problems require complex solutions, and since sustainability is a complex issue, measures must be taken, globally and individually, so that the obstacles and challenges are overcome to achieve the final objective, global sustainability.

2.2.2.1 Environmental challenges

There are three main challenges that need to be addressed, or at least, their impact in our world. They are: energy consumption and global warming; water scarcity; waste management. For each of these challenges, solutions and decisions must be implemented and carried out [49].

Energy consumption and global warming

Energy consumption alone, despite concerning, is not that big of a problem. The ways that the energy is produced to feed our needs is the real problem. If the produced energy was made from renewable and green sources we had particularly nothing to worry about. Sadly that is not the case, since only a quarter of all the energy produced is from renewable sources [62]. This leads to the most problematic part of this challenge: global warming. Global warming is the rise on the average temperature of the earth with some devastating effects such as the rising of sea levels by the molting of polar caps, ocean acidification, expansion of deserts [23]. The main cause of global warming is the expansion of the hole in the ozone layer. This expansion is directly influenced by the releasing of greenhouse gases into the atmosphere which leads to a greenhouse effect. These gases are strongly related to the energy production sources since the majority of their emission comes from them [49].

Water scarcity

Only 3% of the world's water is fresh water, and two-thirds of that is tucked away in frozen glaciers or otherwise unavailable for our use. As a result, 1.1 billion people worldwide lack access to water, and a total of 2.7 billion find water scarce for at least one month of the year. Many of the water systems that keep ecosystems thriving and feed a growing human population have become stressed. Rivers, lakes and aquifers are drying up or becoming too polluted to use. More than half the world's wetlands have disappeared. Climate change is altering patterns of weather and water around the world, causing shortages and droughts in some areas and floods in others. At the current consumption rate, this situation will

only get worse [57]. Water scarcity was listed as the as the largest global risk in terms of potential impact over the next decade by the world economic forum [56].

Waste management

Waste management is the collection, transport, processing, recycling or disposal of waste materials. Waste management is also carried out to reduce the effect of the materials on the environment and to recover resources from them. Waste management can involve solid, liquid or gaseous substances, with different methods and processes for each of them. The main disposal methods for waste management are: landfill and incineration. This challenge is important since almost every industrial process produces waste. Such waste must be managed or else it can have serious impacts in the environment, like the pollution of rivers or atmosphere, and can be prejudicial to our health [3].

2.2.3 Sustainability in software engineering

A system's sustainability describes how well it will continue to exist and function, even as circumstances change [7]. Sustainability has often been equated with environmental issues, but it is clear that it requires simultaneous considerations of environmental resources, societal and individual well-being, economic prosperity and the long-term viability of technical infrastructure [6]. We can also have sustainability by IT or sustainability in IT. This difference is also important, since the first one applies when the IT is the tool used to support sustainability goals and the second one is when the term sustainability is related to the IT, software or hardware themselves. In general, the definitions of sustainable software mix these two perspectives [8]. Therefore we can define sustainable software "*as software, whose direct and indirect negative impacts on economy, society, human beings and environment that result from the development, deployment and usage of it are minimal and/or which as a positive effect on sustainable development*" [33]. However, a green and sustainable software product can only be achieved if the process is also sustainable. Thus, the software architects, designers and developers should be conscious about the impacts, negative and positive, that their product can have on sustainability. Hereupon, we can define green and sustainable software engineering as "*the art of developing green and sustainable software with a green and sustainable software engineering process, i.e., the art of defining and developing software products in a way, so that the negative and positive impacts on sustainable development that result and/or are expected to result from it, over its whole life cycle, are continuously assessed, documented and used for a further optimization of the software product*" [33].

As stated previously, sustainable development can not only accommodate the environmental aspect, but also the social, economic, technical and individual ones. Dimensions are interdependent, i.e., they are related to one another. The same requirement can be applied to one or more dimensions, i.e., can be beneficial to more than one "aspect" of sustainability. This relations will be better explained in Section 3.2. There are five interrelated

dimensions [6]:

- **Social Dimension** → This covers relationships between individuals and groups. For example, it covers the structures of mutual trust and communication in a social system.
- **Economic Dimension** → This deals with the financial aspects and business values. It includes capital growth, investment questions and financial operations.
- **Environmental Dimension** → This considers the use and stewardship of natural resources. It includes questions ranging from immediate waste production and energy consumption to the balance of local ecosystems and climate change concerns.
- **Technical Dimension** → This covers the ability to maintain and evolve artificial systems, such as software, over time. It refers to maintenance and evolution, resilience, and the ease of system transitions.
- **Individual Dimension** → It covers individual freedom, human dignity, and fulfillment. It includes individuals ability to thrive, exercise their rights, and develop freely.

Complex software systems can affect sustainability in any of these dimensions. Nevertheless, changes in one dimension can affect other dimensions. These trade-offs must be analyzed and discussed before any change implementation. However this analysis can be hard to make, and since sustainability is such a broad subject, and in many ways subjective, a deeper understanding is needed. With this in mind, the following principles regarding sustainability for software engineering were proposed [7]:

- Sustainability is **systemic** → A system can never be treated in isolation from its environment.
- Sustainability is **multidimensional** → The four key dimensions are economic, social, environmental and technical.
- Sustainability is **interdisciplinary** → Sustainability design in software engineering requires an appreciation of concepts from other disciplines and must work across disciplines.
- Sustainability **transcends the software's purpose** → Any software can impact the sustainability of its socioeconomic, sociotechnical, cultural, and natural environments.
- Sustainability is **multilevel** → It requires us to consider at least two spheres during system design: the system under design and its sustainability, and the wider system of which it will be part.
- Sustainability is **multi-opportunity** → It requires us to seek interventions that have the most leverage on a system and to consider the opportunity costs.

- Sustainability **involves multiple timescales** → It requires long-term thinking to address the timescales on which sustainability effects occur.
- Sustainability is **not zero-sum** → Changing a system's design to consider the long-term effects does not automatically imply making sacrifices now.

Yet, there is still a lack of knowledge, support and responsibility regarding sustainability and sustainable development and software [10]. The urge of applying sustainability principles in real projects grows. Education presents a major role for improvement. We need to include sustainability principles in software engineering courses, and educate software costumers about sustainability within requirements elicitation and software users about the choices they are making. Critical reflection is essential at the individual, organizational and community level in order to achieve real sustainable software.

2.3 Conclusion

Requirements engineering is an important stage of any software development project. It includes five major activities: elicitation, analysis and negotiation, documentation, validation, and management. In the context of our work, we studied the processes of: elicitation, prioritization, and trade-off analysis. Various techniques associated to each of these processes were identified and discussed. Depending on the context, each technique can wield optimal or non-optimal results,that is, each one has strengths and weaknesses, under recommended conditions. It is up the to the responsible developer, to choose the one that best suits the application domain in question. Focusing now on sustainability, we introduced the three main environmental challenges (energy consumption and global warning, water scarcity, and waste management), and highlighted some threats that come with them. Regarding software sustainability, we explained the difference between Green in IT and Green by IT. We pointed out that sustainability can not only accommodate the environmental aspect, but also the social, economic and technical ones. These four dimensions of software sustainability are connected and interrelated. We discovered that sustainability in software development is still a very young and unclear topic. Due to these characteristics, there are, still, a lack of approaches regarding sustainable software development, despite of the growing need.

CHAPTER 3

Sustainability Catalogue Conceptualization

A catalogue is a model that offers various development decisions, as well as the re-utilization of its information. It also offers, typically, context of a certain topic or research area. In our case, it aims to show the various sustainability requirements and their contributions to the various dimensions of sustainability, which was the result of months of work and research in the sustainability requirements field. Our sustainability catalogue addresses four, out of five, sustainability dimensions: social, economic, environmental and technical. Some of the coarse-grained requirements of each dimension can be shared among dimensions but their refinements can be different depending on to which dimension they belong. Furthermore, a wide scope of relationships (inter-dimension and intra-dimension) exists. These relationships, and the coarse-grained requirements and their respective refinements are the main components of our catalogue. In this chapter we introduce the reader to the conceptualization of the sustainability catalogue, by explaining the sustainability requirements and relations alike.

3.1 Sustainability dimensions & requirements

Our sustainability catalogue is composed, in essence, by sustainability requirements and each of those requirements is part of one, or more, sustainability dimensions. These composing sustainability requirements are typically non-functional requirements that relate, in some way, to sustainability. As stated in Section 2.2.3 there are five sustainability dimensions: social, economic, environmental, technical and individual. However for the context of this work, we did not treat the individual dimension, although we consider that some of its properties are included in the social dimension, such as well-being of individuals composing the society. We know, of course, that the social dimension includes properties beyond the individual needs, such as those concerning organizations, companies

and institutions [6]. Each of these four dimensions and its respective requirements will be explained next. Furthermore, a feature model was defined for each dimension, in order to gain an high-level view of each dimension.

3.1.1 Social dimension

The social dimension relates to societal communities and the factors that erode trust in society [5]. It can also be seen as the well-being of humans living in such society [36]. This dimension relates with notions such as: honesty, transparency, communication, security and safety [5]. Given this, we can divide the social dimension in three main concerns: satisfaction of the stakeholder, security of the system and social safety. Each one of them can be refined into more specific requirements.

Satisfaction

Satisfaction is the degree to which user needs are satisfied when a product or system is used in a specified context of use [24]. In the social dimension, satisfaction is divided into usefulness, trust and fairness [12]. It can be linked with the achievement of pragmatic goals (usefulness) and confidence in the company, regarding equality and honesty, that provides such system or product (trust and fairness) [12]. A definition of such qualities as follows:

- **Usefulness** → Usefulness is the degree to which a user is satisfied with perceived achievement of pragmatic goals, including the results of use and the consequences of use. When eliciting usefulness requirements, one should consider ease of learning, likability, and possible metrics [24].
- **Trust** → Trust is the degree to which the disclosure and shadiness of information, principles and usage is the minimal possible. When eliciting trust requirements, one should consider aspects regarding honesty, transparency, communication, user support [1].
- **Fairness** → Fairness is the quality of treating people equally or in a way that is right or reasonable. This relates directly to justice and equality. Both the company and the system should provide equal user treatment [1].

Security

Security is the protection of information and data so that unauthorized people or systems cannot read or modify them and authorized people or systems are not denied access to them; it is the degree to which a product or system protects information and data [24]. Security is one of the main requirements of the social dimension [12], since we, as individuals, or as a society, don't want our data and information compromised when we use a system. The

definition and possible examples of the main qualities that contribute to the security of a system, in what regards the social dimension, are [12]:

- **Confidentiality** → Confidentiality is the degree to which the software system protects sensitive data and allows only authorized access to the data. When eliciting confidentiality requirements, one should consider aspects related to access control, privacy of communication channels, input interfaces, secure storage of sensitive data and anonymity [24].
- **Authenticity** → Authenticity is the degree to which the software system can confirm the truth of an attribute of a single piece of data claimed true by an entity. When eliciting authenticity requirements, consider needs regarding user registration, user authorization, and user authentication [24].
- **Integrity** → Integrity is the degree to which the data maintained by the software system are accurate, authentic, and without corruption. When eliciting integrity requirements, one should consider needs regarding routine backups of data to prevent loss, backing up data to multiple locations, data restore procedures, and authenticity of data with respect to the original data source [24].
- **Accountability** → Accountability is the degree to which the action of an entity can be traced uniquely to the entity. When eliciting accountability requirements, one should consider aspects related to traceability and authenticity [24].

Social safety

Safety in a social context is the degree to which a product or system mitigates the potential risk to people in the intended contexts of use [1], but also in terms of social justice in what regards legislation. Thus, social safety can be divided into two different aspects:

- **Freedom from risk** → Freedom from risk is the preservation of the human life. We, as a society, or as an individual, must be free of accidents that could cause harm to our well being [12].
- **Legislation** → The legal information (such as, terms and conditions, specific policies) should be available and without dubious information [38].

Relationships between social requirements

Regarding relationships between the social requirements, the security of the system helps the trust of stakeholder, since a secure system is one that inspires trust to the user [1]. A system's authenticity can help both its integrity and its accountability [24]. It helps the first in the sense that integrity is all about how accurate and authentic the system data is. Thus authenticity helps to prove that certain data is trustworthy. It helps accountability since

it helps to trace all system's entities and confirm their legitimacy. However confidentiality may be prejudicial for accountability [24], since it could be harder to trace the origin of the data, due to possible anonymity. A feature model of the social dimension, which depicts its decomposition into the main requirements and shows the various relationships among them, is shown in Figure 3.1.

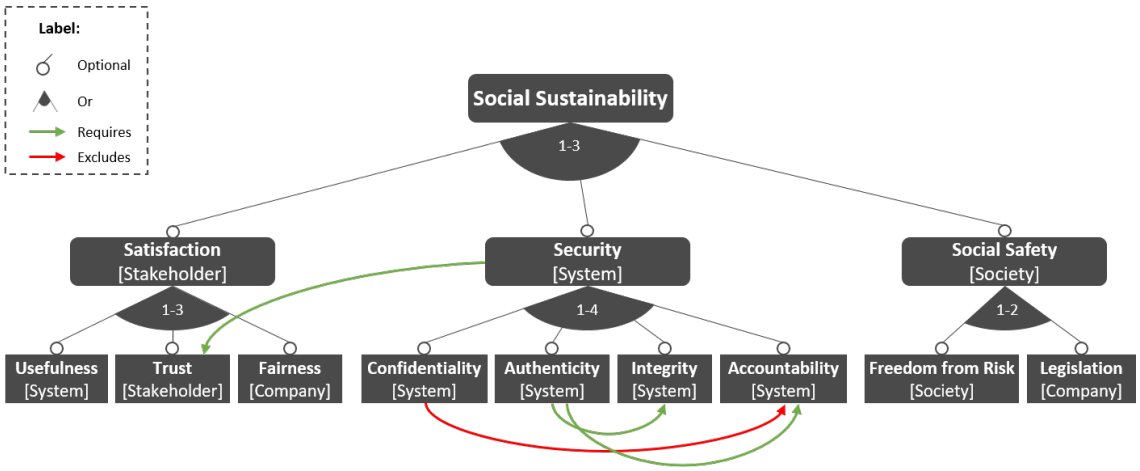


Figure 3.1: Social dimension feature model.

3.1.2 Economic dimension

The economic dimension covers financial aspects and business value. It includes capital growth and liquidity, investment questions, and financial operations [36]. It also takes care of budget constraints and cost as well as market requirements and long-term business objectives which can be translated into requirements for the system under consideration [38]. This dimension is related with notions such as: risk and cost analysis, fairness, marketing, working conditions, investment, development, infrastructure's, specifications and precision of the system. Translating these notions to non-functional requirements, we get: stakeholder satisfaction, efficiency, functionality and reliability of the system, and company economic safety [12]. A refinement and explanation of these non-functional requirements is presented below:

Satisfaction

Satisfaction in the economic dimension has the same definition as for the Social dimension (Section 3.1.1). However it presents some dissimilarities regarding to trust and fairness. Trust contributes to stakeholder satisfaction, not in a user perspective (as explained in 3.1.1), but in a company perspective. Thus, it relates to aspects such as the transparency and the good image of the company. Good usage of marketing, honest communication and transparent investments are all goals that the company should achieve in order to maximize the stakeholder trust [16]. Fairness in the economic dimension relates with being

fair to competitors and and all the economic environment, thus fairness relates with the economic safety of a company and not with the satisfaction of the stakeholder (3.1.2) [60].

Functionality

Functionality is how a system functions given a certain context. It can be how easy a specified task can be accomplished or how accurate a given result is. The set of functions a system has should cover all the specified tasks and user objectives [24]. In the economic side, if a system functions well in all given contexts, which contributes for the efficiency of such system, it will translate into user satisfaction, that ultimately helps the economic growth [12]. Functionality can be split in two different notions:

- **Functional appropriateness** → Functional appropriateness is the degree to which the functions of a software system facilitate the accomplishment of specified tasks and objectives [24]. When eliciting functional appropriateness requirements, one should consider aspects regarding ease of use, development process [13].
- **Functional correctness** → Functional correctness is the degree to which a product or system provides the correct results with the needed degree of precision [24]. When eliciting functional correctness requirements, one should consider aspects regarding accuracy, integrity and specifications (both from user and technical) [13].

Efficiency

Efficiency is how well a system can perform under certain circumstances, i.e, the extent to which the software system handles capacity, throughput, and response time [24]. This relates to the economic dimension since an efficient system will generate more value and less loss [12]. Efficiency can also be perceived as:

- **Effectiveness** → Effectiveness is the accuracy and completeness with which users achieve specified goals [1]. Efficiency requirements address the user concern for how fast the system functions, how efficiently the system takes in inputs and processes outputs, and how much can be processed at a time [34].

Reliability

Reliability is the extent to which the software system consistently performs the specified functions without failure [24]. If a system is reliable, i.e., is available during large periods of time, and in case of failure, it can re-establish its desired level of performance, it will have a lower risk associated with the economic segment [12]. Thus, we can refine reliability into two different qualities:

- **Availability** → Availability is the degree to which users can depend on the system to be up (able to function) during "normal operating times" [24]. The following needs

should be considered when eliciting availability requirements: downtime impact on the business, partial availability impact on the business, transparent unavailability, and minimizing unavailability [34].

- **Recoverability** → Recoverability is the degree to which, in the event of an interruption or a failure, a product or system can recover the data directly affected and re-establish the desired state of the system [24]. When eliciting recoverability requirements, one should consider system backups, failure procedures and preventive actions [34].

Economic safety

Safety in a economic context, is the degree to which a product or system mitigates the potential economic risk associated with the financial management of it [24]. But also about market and employee fairness [12]. Therefore, economic safety can be divided into two different aspects:

- **Economic risk mitigation** → The mitigation of the economic risk relates to the financial management and strategies of the company. There should be a meticulous analysis of investments and management of expenses. Economic indicators, such as ROI, should be adopted by the company [38].

- **Fairness** → A company must be fair to its competitors but also with their employees, so that there is no fraud whatsoever and for workers to be satisfied with company policies [60].

Relationships between economic requirements

In what regards relationships between economic requirements, the system's functionality helps its efficiency, since a functional system is an efficient one [12], and also the satisfaction of the stakeholder, the latter also receives the contribution of the system's reliability [12]. The system's functional appropriateness benefits from its usefulness [13], because functional appropriateness relates to how easy is to perform a specific system task, which directly relates to how useful the system can be to its user, which leads to an increase of satisfaction. Finally some strategies under the company's economic risk mitigation policy, such as investments analysis or good infrastructures, can be helpful for the system's effectiveness [38]. A feature model of the economic dimension, which depicts its decomposition into the main requirements and shows the various relationships among them, is shown in Figure 3.2.

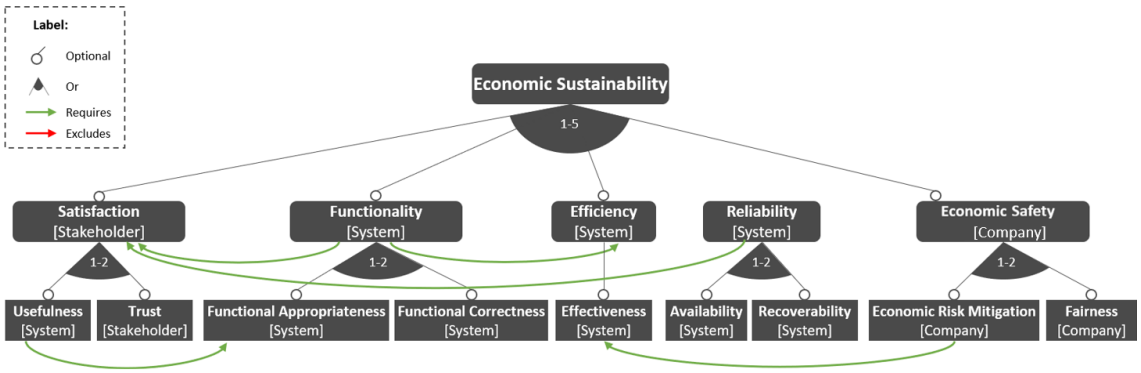


Figure 3.2: Economic dimension feature model.

3.1.3 Environmental dimension

Environmental sustainability seeks to improve human welfare by protecting natural resources. It includes questions ranging from immediate waste production and energy consumption to climate change concerns [36]. When we think about environmental sustainability and what relates to it, some general ideas come to mind: resources sharing and saving (reuse, reduce and recycle energy, materials, waste), conservation, restoration, preservation, renewability, and both environmental safety and health. The non-functional requirements that relates to these ideas are: maintainability, compatibility, and efficiency of the system, but also the environmental safety. An explanation of each of these non-functional requirements and further refinement is shown below [12]:

Maintainability

Maintainability is the ease with which a software system or component can be modified to change or add capabilities, correct faults or defects, improve performance or other attributes, or adapt to a changed environment [24]. Maintainability is important for the environmental dimension when it comes to modifiability and reusability. The reuse of resources or technology, leads to a decrease on resource consumption which has a positive impact on the environment [12]. An explanation of both these qualities is presented below:

- **Reusability** → Reusability is the extent to which a portion of the software system can be converted for use in another system [24]. When eliciting reusability requirements, one should consider aspects of feasibility of software reuse, possible areas for reuse, and development standards [34].

- **Modifiability** → Modifiability is the degree to which changes to a software system can be developed and deployed efficiently and cost effectively [24]. Modifiability requirements address the user concern for how quickly and cost effectively changes can be made to a software system [34].

Compatibility

Compatibility is the ability of a system to exist and work with other systems without problems or conflicts [24]. If a system can co-exist with more systems, while sharing resources, then, the total usage of resources will be lower, which is beneficial to the environment [12]. Compatibility involves these two qualities:

- **Adaptability** → Adaptability is the degree to which a product or system can effectively and efficiently be adapted for different or evolving hardware, software or other operational or usage environments [24]. An adaptive system is an open system that is able to get its behavior according to change in its environment or in parts of the system itself [34].
- **Interoperability** → Interoperability is the extent to which the software system is able to couple or facilitate the interface with other systems [24]. When eliciting interoperability requirements, one should consider aspects such as software testing, product engineering, industry partnership, standard implementation, and common technology [34].

Efficiency

Efficiency in the environmental dimension can be seen as the effectiveness of the system while using the strictly necessary amount of resources for each task [12]. Therefore, an efficient system is one that completes the tasks efficiently, without wasting resources.

- **Resource utilization** → Resource Utilization is the appropriate usage of memory, storage and other resources of a system [1], and takes a major role in environmental efficiency [12]. We should define the conservation and sharing of resources, but also the renewability of resources [13].

Environmental safety

Safety in a environmental perspective, is the degree to which a product or system mitigates the potential risk to the environment, without causing harming to general environment health. Thus, environmental safety relates to two related topics:

- **Freedom from hazardous risks** → Freedom from hazardous risks is the degree to which a product or system mitigates the potential risk to the environment relating with hazardous materials [24].
- **Environmental health** → If we mitigate the risk of hazardous accidents we will contribute to the general health of the environment. Another task that should be carried out is the monitorization of the system according to safety indicators. The use of renewable resources (energy, materials) is also a major contribute for the health of the environment [38].

Relationships between environmental requirements

With respect to relationships between environmental requirements, the system's interoperability helps its efficiency regarding the ease of incorporation of other systems or resources which ultimately leads to an increase of system performance [34]. A system's adaptability helps its modifiability because an adaptable system is one that is easily modifiable [34]. Finally the reusability helps the adaptability of a system, since if we can, successfully, reuse a system, or part of it, it proves that the system is highly adaptable to other environments [24]. A feature model of the environmental dimension, which depicts its decomposition into the main requirements and shows the various relationships among them, is shown in Figure 3.3.

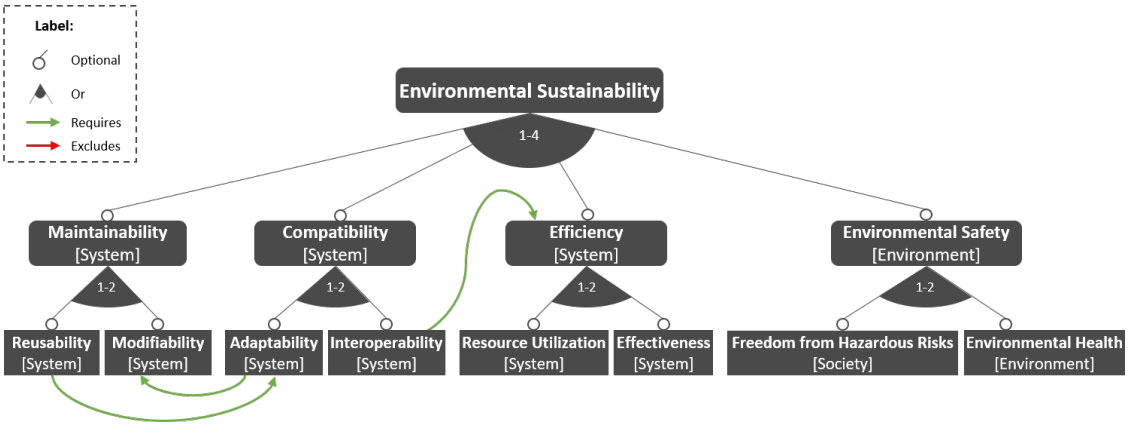


Figure 3.3: Environmental dimension feature model.

3.1.4 Technical dimension

Technical sustainability has the central objective of long-time usage of systems and their adequate evolution with changing surrounding conditions and respective requirements. It refers to maintenance and evolution, resilience, and the ease of system transitions [5]. Thus, the related non-functional requirements are the following: functionality, maintainability, compatibility and reliability, all of the system. They are defined as follows:

Functionality

In the economic dimension (Section 3.1.2), functionality was important since it helps the satisfaction of a user and the efficiency of a system. In the technical dimension, if a system is functional, and everything works as intended, then, the possibility of occurring internal errors and/or failures will decrease which leads to an increase in a system's longevity [12].

Maintainability

In the environmental dimension (Section 3.1.3), maintainability had the goal to decrease resource consumption by reusing resources. In the technical dimension, maintainability is

important when it comes to guarantee how well a system is maintained. A system that is checked periodically for internal errors, that is highly modular and has a multiple environment support will, generally, last longer than one that does not have this characteristics [12]. Beyond modifiability, which was already explained in the environmental dimension (Section 3.1.3), technical maintainability relates with these qualities:

- **Modularity** → Modularity is the degree to which a system's components may be separated and recombined, often with the benefit of flexibility and variety in use, while having minimal impact on other components [24]. When eliciting modularity requirements, one should consider aspects regarding architectural and interface design, code documentation, and modifiability [13].
- **Testability** → Testability is the degree of effectiveness and efficiency with which test criteria can be established for a system, product or component and tests can be performed to determine whether those criteria have been met [24]. When eliciting testability requirements, one should consider verification and validation techniques that might be used, possible inspection checks [1].

Compatibility

Compatibility in the environmental dimension (Section 3.1.3) is all about sharing resources by systems co-existing together, with the ultimate goal of lowering the total usage of resources. However, compatibility in the technical dimension mainly relates with adaptability of a system. If a system can adapt to constant changes, be them code, technology, or functionalities, then, it will have an adequate evolving process. This process will, consequently, result in an increase of the longevity of such system [12].

Reliability

The technical dimension is responsible for the resilience of the systems, thus, it must be reliable and available at all times, with little to no space for failures [5]. Such characteristics allows a software of performing specified functions for a longer period of time. Technical reliability relates with the same qualities as economical reliability (see Section 3.1.2), however, it has one extra quality [12]:

- **Fault tolerance** → Fault tolerance is the ability of a system or component to continue normal operation despite the presence of hardware or software faults [24]. When eliciting fault tolerance requirements, one should consider robustness and exception handle algorithms [1].

Relationships between technical requirements

About the relationships between technical requirements, system's adaptability helps its modifiability the same way as discussed in Section 3.1.3. If a system is robust and has

a good component of fault tolerance, then it will perform its tasks normally which leads to an increase of its availability [24]. If we define a set of criteria, being them functional or performance wise, for the system to met and we will help the system to function properly and as desired [1]. Finally if we correctly maintain a system, in what regards the correct usage of its components, it will lead to an increase of its reliability resulting in a longer, healthier system [12]. A feature model of the technical dimension, which depicts its decomposition into the main requirements and shows the various relationships among them, is shown in Figure 3.4.

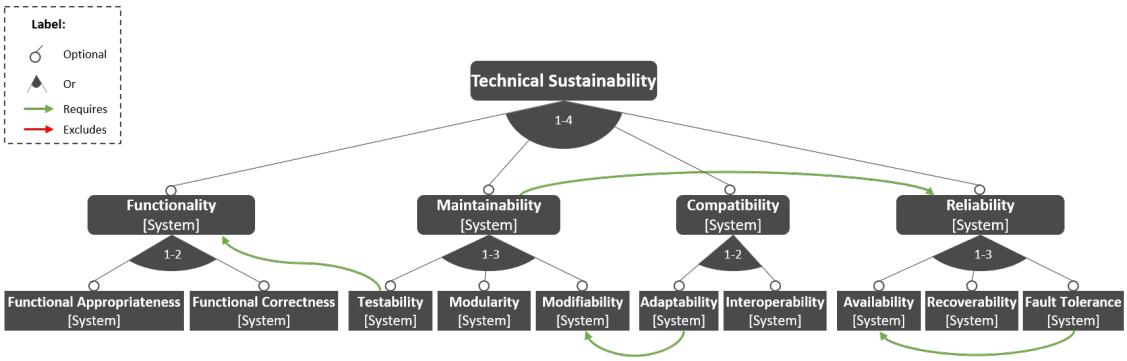


Figure 3.4: Technical dimension feature model.

3.2 Relationships between sustainability dimensions

The sustainability dimensions are inter-dependent [5]. This means that they can relate to each other, either in a positive, or in a negative way, i.e., an action in one dimension may affect the others. Thus, we can say that each dimension has a certain effect on the others. These effects exist across all combination of dimensions, and its study must be carried out. Furthermore, the dimensions also have some key properties in common. These common properties are the non-functional requirements that are part of each dimension [12], [38], [43]. Regardless of whether these requirements may have different refinements, or different contributions to other requirements of the dimension in which they are inserted, the inclusion of these requirements should be something to consider, since we will contribute to the sustainability of two or more dimensions by having that requirement. However, it is possible that this requirement may have negative contributions with other requirements of the same, or other, dimensions. This is why the trade-off analysis of each decision must be meticulous and well performed. In conclusion, the effects between the various dimensions can be seen as a high-level relationship, where we relate complex, and sometimes abstract, notions, while in what regards common requirements, its more of a concrete relationship. Both the analysis of effects and the common requirements will be explained in the following sections.

3.2.1 Effects

In an ideal world, we would want a fully sustainable project, covering all the sustainability dimensions. However, in reality, the best we can do is, in many situations, to try to maximize a subset of dimensions by combining some properties of some dimensions to achieve partial sustainability. We must understand the effects that each dimension can have in the remaining ones, be them beneficial or prejudicial. For each combination of dimensions we will discuss the respective effects they have on one another. A description of each combination of dimensions is discussed next. This discussion was based on the combination of different discussions, and different notions, performed in [12], [38], [43].

Social & Economic

Society is organized by individual humans and institutionalized by means of a government and its infrastructure. This infrastructure also enables economy. Therefore, social sustainability is a prerequisite for economic sustainability. On the other hand, economy again interacts with society by providing wages for its employees. If the society is happy with a product and the needs are fulfilled, then, it will prosper, which can be perceived as a positive effect on one another.

Social & Environmental

A software system that mitigates the potential risk to the environment, will have also a positive impact on the users that interact with such software system (social sustainability). Also, without natural resources, society does not exist. Therefore, environmental sustainability can be seen as the basis for social sustainability. However, the fact that we may use materials which are more energy efficient but more prejudicial for people's health can be seen as a problem and, therefore, a negative effect.

Social & Technical

If a product has diverse functionalities, is reliable and provides interoperability, it will represent an increase of user satisfaction. This satisfaction is the base for social sustainability. The society can also have a positive effect on the technical side of a product by providing feedback and suggest new functionalities. The constant and ever evolving needs of the society can be seen as one of the main booster of technology in a general way, which will ultimately result in better and more advanced products.

Economic & Environmental

The economic dimension should align with the environmental one in terms of resource savings (energy, materials, waste). However, and unfortunately, that's not the case in the reality we live. The global environment is in a degraded state almost solely because of economic interests. To add more on the negative side, they may also conflict when it

comes to additional certifications and turning to more expensive alternative solutions that are more environmentally friendly.

Economic & Technical

These are the two dimensions that are more strongly correlated. The longevity of systems and services (technical sustainability) is a fundamental element of the economy. However, technical systems are developed within a regulated economical context. Too much money spent can be prejudicial in terms of economic sustainability, but also too little spent can be prejudicial in what regards technical sustainability, which leads to a problem on the economic side. Therefore, economic and technical dimensions depend on each other to a certain extent. Economy motivates research and development and vice-versa. This can be seen as neither a positive or negative effect, but more like a symbiosis effect.

Environmental & Technical

If a software system can co-exist with other independent software in a common environment by sharing resources, it will have a good contribution to the environmental sustainability dimension (for instance, less energy consumption). The technical dimension will be also affected positively because the software can efficiently perform its functionality, while using less resources.

Social & Economic & Environmental

If a software system can efficiently perform without a high cost budget, while being environmentally friendly, it will have a positive effect on its users, and consequentially on the revenue of such system/product. However, in order to aid the environment, a shift in general thinking has to be done, both on the social and economic side. We, as a society, must understand that we are the responsible of how we use a product, we must be environmental conscious. Such consciousness must be present on the economic side as well. Only then can we truly have a positive effect on the environment.

Technical & a pair of Social/Economic/Environmental

The technical dimension is responsible for the maintenance, endurance and longevity of systems, as well as for their functionality and compatibility. If we combine the technical dimension to any of other pairs of dimensions, we can achieve better systems while impact positively across all dimensions. Let's take as an example, the environmental & economic & technical ternary relationship. We want a system that is economically and environmentally sustainable, however, the solution available is not reliable, either because of the environmental side or the economic one (low functionality, low longevity, high maintenance cost, etc.). By adding the technical dimension to the mix, we can be more conscious of what's need to be done, so that the solution can become more reliable and efficient, and,

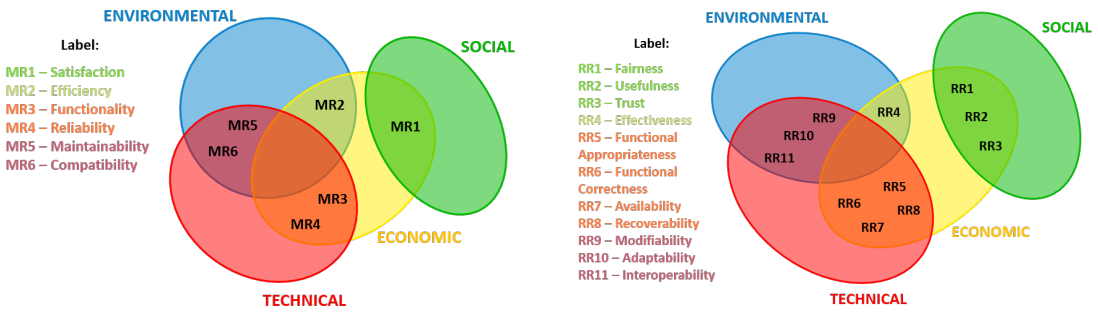
therefore, achieve its goals on all dimensions. The same happens with the social dimension, since by combining the technical dimension we can have an increase of user satisfaction by having a functional and reliable system.

Social & Economic & Environmental & Technical

By considering sustainability in all dimensions, when developing our software, or product, we will achieve a fully sustainable state. However, this notion of fully sustainable software is, somewhat of unrealistic in the real world, since it is not possible to contribute positively in all dimensions, there are always trade-offs. Nevertheless, by making a thorough trade-off analysis, in relation to our domain, we can maximize the sustainability of the system.

3.2.2 Common non-functional requirements

Each dimension of sustainability is made up of several non-functional requirements. These requirements may be unique, or common to more than one dimension. They have been already presented in the previous Sections (3.1.1, 3.1.2, 3.1.3, 3.1.4), as well as a brief discussion of some common requirements across different dimensions. However, these requirements may have different contributions, and refinements as well. Let's take two requirements as an example: **safety** and **maintainability**. **Safety** is the only requirement that is common to more than two dimensions, and each one has a different definition and a different refinement. However, **maintainability**, despite having the same definition in the two dimensions to which it belongs, contributions and refinements differ on certain points. Another important aspect is the contributions that a requirement can make to one or more other requirements. Having a particular requirement in one system may be detrimental to another requirement, but it may be beneficial to another, even if we are talking about different dimensions. The Venn Diagram in Figure 3.5(a) shows the shared main requirements between dimensions, while Figure 3.5(b) shows the shared refined requirements (MR stands for Main Requirement, and RR for Refined Requirement).



((a)) Main non-functional requirements.

((b)) Refined non-functional requirements.

Figure 3.5: Venn diagram of the common non-functional requirements.

3.3 Conclusions

We understood the main requirements of each dimension, as well as their refinements. These requirements are complex, and offer a more detailed and clear view of what each dimension has to offer for sustainability. We also presented the relations between dimensions, on a general perspective, but also in a requirement level one. With the conceptualization of the sustainability catalogue fully done, we are now ready to start implementing it.

CHAPTER 4

Sustainability Catalogue Implementation

To successfully implement our catalogue, we started by choosing an appropriate framework for its representation and a base working environment where we could develop some supporting tool or tool extension. We chose the iStar framework to specify our catalogue and we implemented it as an extension of the piStar tool. We adopted a bottom-up development strategy, starting by developing each individual dimension catalogue, and then combining these into the total and final artifact, the sustainability catalogue. The catalogue has an array of functionalities that will be presented in this chapter and can be used in a vast range of projects. Finally, we applied our catalogue to three different real-world scenarios.

4.1 Chosen specification framework & supporting tool

The first step for implementing our sustainability catalogue was to choose the appropriate technology. We needed a framework capable of representing, or specifying the sustainability requirements, while explicitly presenting the relationships among them. The framework chosen was iStar. But this was not enough, since we also needed a tool capable of supporting our catalogue implementation. The implementation environment we chose was *piStar* [1].

4.1.1 Mapping feature model elements to iStar concepts

We need an appropriate framework to support our catalogue needs, such as, good visually representation of elements, complete semantics, support for trade-off analysis. In Section 3.1 we presented four feature models (see Figures 3.1, 3.2, 3.3, and 3.4), each describing characteristics of each of the four dimensions addressed in this work. Despite being the starting point of our implementation, which will be further discussed in Section 4.2,

[1] https://www.cin.ufpe.br/ jhcp/pistar/

they are not enough to explicitly represent all our needs. First, the feature model does not let us make a clear separation between elements (they are all features), nor does it support the various types of intra- and inter-relationships required. Furthermore, although the feature model can represent some elements to support trade-off analysis, there's no difference between two different positive or negative contributions (as discussed in Section 3.1, there are certain type of requirements that have a stronger contributions among them than others). Therefore, the feature model neither presents a complete semantics for our domain nor does it support the extent of trade-off analysis we need. For these reasons we chose the iStar framework (see Section 2.2.1.3). Therefore, it can present, in an understandable manner, all the properties and relationships of the various sustainability dimensions. Furthermore, it has four different types of contribution links: break, hurt, help, and make. These links are suited to represent the respective trade-off analysis. Finally the iStar framework is suitable for the early stages of the development process [15].

We started by mapping the various elements of the feature model (*the source model*), which represent the notions of sustainability discussed in 3.1 into concepts, or elements, of the iStar framework (*the target model*). Thus, the source of this mapping process are the feature elements, and the target are the iStar elements. This mapping is shown in Table 4.1.

Table 4.1: Mapping of the feature model elements into iStar elements

Source (feature elements)	**Target (iStar elements)**
Main Feature	Quality
Sub-feature	Quality
Optional Link	Help or Make
Requires	Help or Make
Excludes	Hurt or Break

We considered sustainability as a quality and not as goal because we perceive sustainability as a quality of a system (as it happens with security or functionality, for instance), as our goal is for systems to *be* sustainable. In addition to this mapping, qualities were further refined in iStar model, so that operationalizations could be defined. Such operationalizations use other elements of the iStar language, such as *goal*, *task* and *resource*, together with their respective links (see Figure 2.4). In summary, the qualities, goals, tasks, and resources (which are iStar elements) can represent different properties of each sustainability dimension:

- **Qualities** → The iStar qualities are the non-functional requirements elicited in Section 3.1. For instance, under social sustainability, system security is a quality, but also its confidentiality, authenticity, integrity, accountability (which are the refined qualities from security). These qualities will always be linked by contribution links (break, hurt, help, make), and the value of such link will be directly related to the relationship of the two elements. The information related to these contributions was collected

from existing requirements catalogs, and also from the results of the systematic mapping study.

- **Goals** → In our catalogue, they represent the aspects needed to achieve a certain quality. These aspects were also elicited in Section 3.1. Given the example of security, but more in concrete, integrity, we can have as a goal, the "system data backup". Goals will, generally, need a set of tasks to be achieved [15].

- **Tasks** → Tasks represent actions that we want to be executed, with the purpose of achieving some goal or quality. Continuing with the same example, we have the system data backup as a goal of integrity, to achieve this goal we should perform two tasks: have our backup in multiple locations and set various routines and procedures. These tasks link to a goal via refinement links, *AND* or *OR*, depending whether the goal needs all the respective tasks to be achieved (*AND*), or only some of them (*OR*) [15].

- **Resources** → A resource is a physical or informational entity that we require in order to perform a certain task. Finishing the example, in order to have our system data backup stored in multiple locations, we should have secure data and protocols to make sure the data is not compromised. These resources will always link to tasks using the *NeededBy* link [15].

4.1.2 The piStar tool

The piStar tool is an open-source goal modelling tool for iStar, conforming to the iStar 2.0 standards. The iStar models created with this tool are visually appealing, and are suitable for including any type of documents, be them academic, research, or industrial ones. Furthermore, the tool is "architected" to be easily extended and customized by developers. One can create a built-in plugin to execute a certain task. piStar runs on a web-browser without requiring additional installation or complications [42]. On the main page of piStar we are presented with a large canvas to draw our models, and four different tabs, each one responsible for some functionalities of the tool. Figure 4.1 shows the main page of the piStar tool, which contains an iStar model describing some functionalities of the tool.

- **File** → This tab is responsible for the management of the model. We can create a new model, save the model as text file and also as an image. The text file will contain all the information related to the current model, we can then, use that same file to load the model into the tool.

- **Add** → Here we can select all the elements that exist in the iStar framework to add them on our model. Each element has the associated semantics that must comply with the iStar 2.0 standards. This means that we can only build models that are

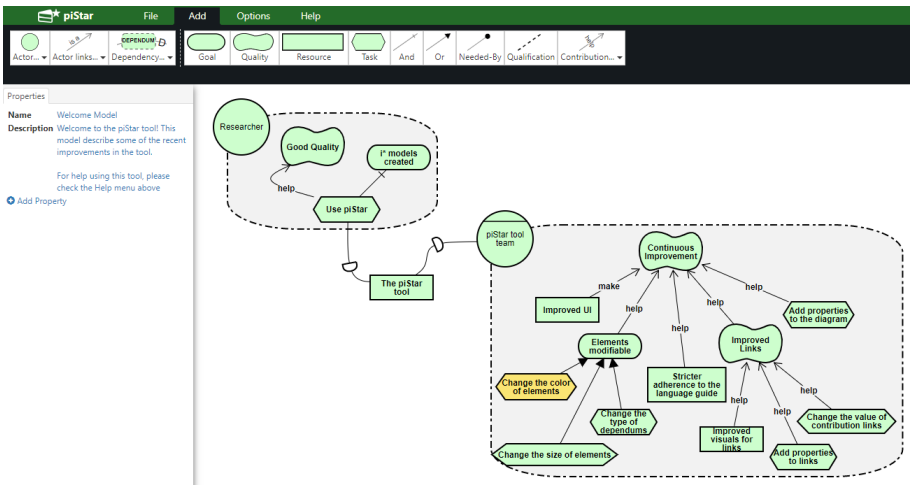


Figure 4.1: piStar tool's main page.

according to the semantics of the language. To add an element, we simply select the element we want and click on the respective canvas position. To delete, we click on the element and press *delete*. We can also drag and move around all the elements of the model.

- **Options** → The "Options" tab let us alter some characteristics of our model, such as: the size of the model, straighten all the links of it and make them pixel-perfect, or toggle to full-screen.

- **Help** → The last tab helps the user if s/he has any doubts on how to use the tool. We can check some model examples or read a brief tutorial on the functionalities of the tool. Furthermore we can read the iStar language guide or check works that used and referenced the piStar tool.

Besides this, piStar also has an improved user interface and lets the user modify the elements. All the elements have two different aspects that we can customize: its properties (such as, name, description, or a custom one) and its color. This tool was chosen since it presented the three main qualities we were looking for to represent in our catalogue: (1) compliance with iStar 2.0 standards, (2) visually appealing models and (3) extendability and customability (which will be discussed later, in Section 4.3).

4.2 Development methodology

As observed in Figure 3.5, there are many qualities shared among dimensions. We also know that there are intra- and inter-relationships between dimensions (see Section 3.1 and 3.2, respectively). To avoid redundancy and reduce the possibility of forgetting something intrinsic to a dimension, we should first draw the catalogues respective to each dimension before combining them all. This can be seen as a bottom-up approach.

It follows the merging design approach which combines the smaller sub-models into the larger, more complex model. The models are built from simple components connected. In our work, the components are the properties of each dimension (such as the requirements and relationships), the sub-models are the dimensions catalogues, and the complex model is the sustainability catalogue. Furthermore, this approach is ideally used on experimental and research projects [54].

4.2.1 Construction of the dimensions catalogues

Each dimension catalogue is an SR (Strategic Rationale) iStar model that complies with the iStar 2.0 standards (see Section 4.1.1) [15]. The base for the development of each catalogue were the feature models from Section 3.1. We used the same approach for all dimensions. Further, we chose a color for each dimension, so that it would be easier to identify in the final catalogue (Social: Green; Economic: Yellow; Environmental: Blue; Technical: Red). This color code will be further explained in Section 4.2.2. We perceived the catalogues itself as the actors of our models, and as the central and main element, the sustainability of each dimension (a quality), which was split in various main qualities, as it happens in the feature models (see Figures 3.1, 3.2, 3.3, 3.4). In addition, these main qualities relate to a set of other qualities, goals or tasks, depending on the context, and can be further refined. Such refinement was discussed in Section 3.1. However, we settled a four-level refinement as a maximum, to maintain a simple and understandable catalogue. The final step was to add possible resources needed to complete certain tasks. Therefore we can divide our dimensions catalogue into four different levels: (1) Dimension Sustainability, (2) Main Qualities, (3) Refinement, (4) Resources. Between the developing of each level, we applied all the intra-relationships of the various elements (which were discussed in Section 3.1), i.e., we linked the elements accordingly the iStar 2.0 standards [15] (see Figure 2.4).

To better understand the methodology, lets take as an example, the social dimension and some of its requirements. The central quality is social sustainability. As discussed in Section 3.1.1, social sustainability can be divided into three non-functional requirements: satisfaction, security and safety. These non-functional requirements, which in the iStar model are qualities, all link to the social sustainability via contribution links. These are our two first levels. Now on the third level, lets focus on the satisfaction (of the stakeholder), for instance. We know that satisfaction relates with the usefulness of the system, the stakeholder's trust and the fairness of the company. All of the previous are qualities, and from different entities. Given each refinement, we should look for possible relationships, an example is that the system's security helps the stakeholder to gain more trust on the system, therefore, we should draw such relationship with the appropriate link. Continuing, we can further refine the qualities previously mentioned (usefulness, trust, fairness). To assist this process, we applied known information about these refinements, which are presented in other NFR catalogues. Considering usefulness as an example, a useful system should

accomplish its proposed functionalities, in the most functional way possible [24]. Thus, we can have as a goal, the "accomplishment of proposed functionalities (of the system)", and the "ease of use" and "ease of learning" are possible tasks in order to achieve such accomplishment.

Finally, the usage of "user review metric" are resources needed to understand how the users perceive our system, and how we can improve it [1]. Thus, one can say that the usage of user review metrics are needed to perform tasks, such as easiness of use or learning, and both are needed to accomplish the proposed functionalities of the system, which positively contributes to its usefulness, and further, to the satisfaction of the stakeholder. As always, we should connect the elements with the appropriate link and check possible relationships between other elements. A visual representation of the previous example is shown in Figure 4.2. The complete catalogue of each dimension is shown on the following Figures: Social - B; Economic - C; Environmental - D; Technical - E.

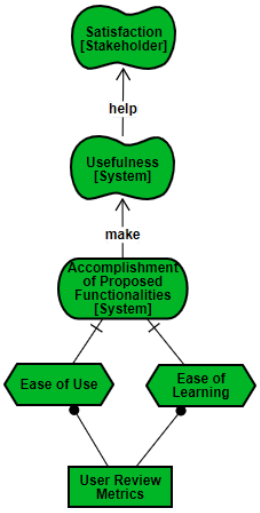


Figure 4.2: Refinement of the usefulness quality.

4.2.2 Composition of the sustainability catalogue

With the dimensions catalogues fully done, we were ready to start developing the sustainability catalogue. The sustainability catalogue was made by composing the catalogues of the various dimensions together with the study previously carried out on the various inter-dimensional relationships (see Section 3.2). The sustainability catalogue was developed in a way that the user can easily separate the various elements of each dimension. This was achieved by grouping, as much as possible, elements of the same dimension and by having a color scheme. As discussed in Section 4.2.1, each element of a dimension has a respective color, this color scheme was passed on to the sustainability catalogue so that the identification of a certain element would be easier to the user. If an element relates to two or more dimensions, its color will result from the mixing of the colors of each dimension

that he relates. Furthermore, a color label was created for the catalogue, but we will discuss that in Section 4.3.1.

The developing of the sustainability catalogue used a similar approach to the one discussed in Section 4.2.1, however, for simplicity, we excluded both the tasks and resources, thus remaining only with qualities and goals. Regardless of that, and in order not to lose valuable information and to make the catalogue as complete as possible, each element has a clear description (that the user can, at any time, read). This description explicitly details the element, and furthermore, it explains how it relates to its dimension(s) or to its "parent" element, and how we can achieve it. For instance, using the same example as in the Section 4.2.1, "accomplishment of proposed functionalities", in the sustainability catalogue we do not have the tasks "ease of use" or "ease of learning", however, in the description of that element we refer them. Figure 4.3, shows a zoomed-out view of the catalogue with the four major elements zoomed-in, which represent the sustainability of each dimension. All these four elements link to one single element, sustainability. A clear view of the catalogue can be consulted in the appendix F.

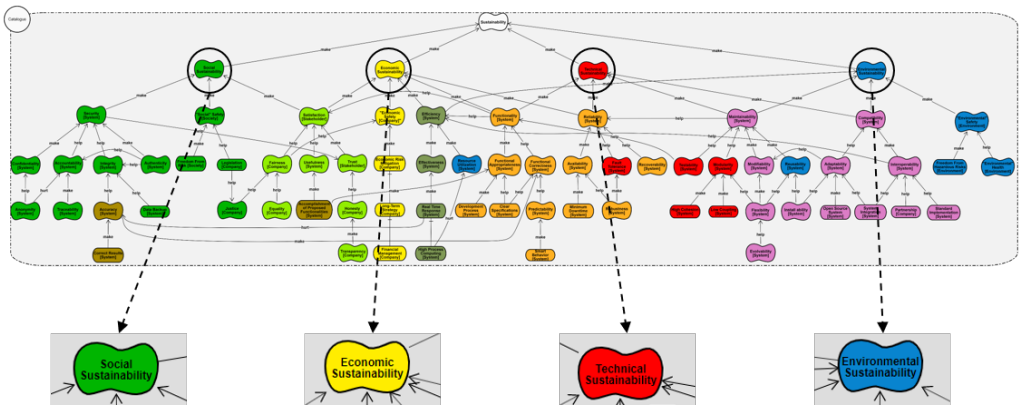


Figure 4.3: The major elements of the sustainability catalogue.

Each of these dimensions has associated, by contribution links, the main qualities, or (subset of) non-functional requirements (see Section 3.1). Furthermore, these qualities are refined into other qualities or goals, also linked by contribution links, as happened with the dimensions catalogues. While on the dimensions catalogue we had four different levels, on the sustainability catalogue we can divide it into three levels: (1) Dimension Sustainability, (2) Main Qualities, (3) Refinement. We kept the same maximum four-level refinement used on the dimensions catalogue, for the sake of simplicity and understandability. To better understand this, let's take as an example the economic dimension, which is detailed in Figure 4.4.

On the **first level** we have the respective dimension, which is linked by the main qualities via contribution links, regarding sustainability. On the **second level**, we have the main qualities, they can have contribution between themselves as well or relate to other dimensions. For example, the functionality of the system helps its efficiency, and the satisfaction of the stakeholder contributes to economic sustainability. Finally, on the

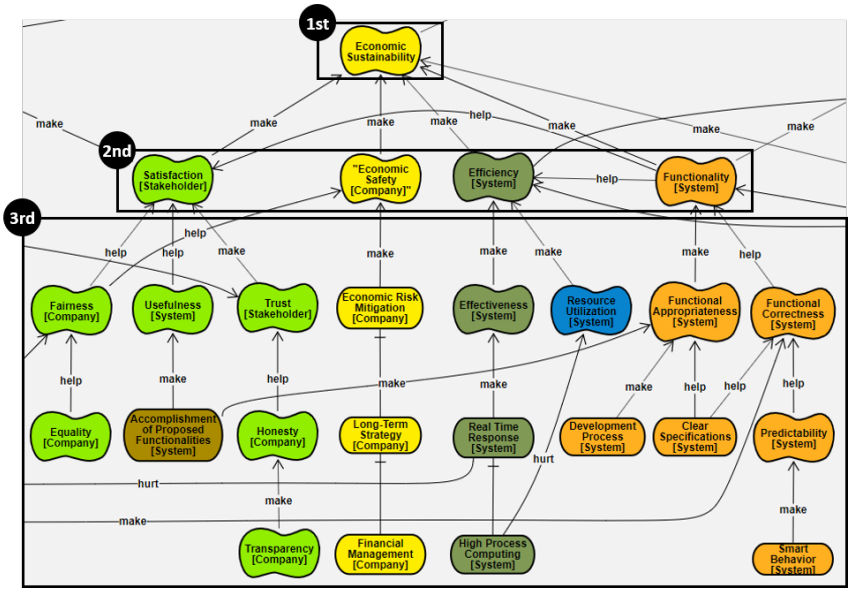


Figure 4.4: Economic dimension.

third level, we have all the refined elements, be them qualities or goals. For instance, the economic risk mitigation of the company contributes for its economic safety, but for that, the company should have a long term strategy, which in turn needs some type of financial management, done by the company.

4.2.3 Elements properties

Since we are dealing with complex notions, a simple name referring is not enough to transmit everything associated with it. For instance, if we just put the element "security" on the catalogue, without any associated explanation or anything that defines it, the user will be confused and will not know what to do, so that he can, effectively, have security on his system, nor will he know what security relates to. To prevent this from happening, each element of our catalogue (quality or goal) has three properties. Each property has a specific function and the combination of the three makes the user understand what the element really refers to. These properties are:

- **Name and the surrogate stakeholder** → The name alone is not enough for us to clearly differentiate an element. Even less, if we don't have a notion of what it is about, it can create ambiguity and confusion. For this, we put, in straight parentheses, the entity to which it refers. This entity can be seen as a surrogate stakeholder and can be, for instance, the system (software or hardware), a company or even the society. For example, when we talk about safety, we have to be clear to what we are referring to, whether it is the "company", "environment" or even "social" safety.

- **Description** → In order for the catalogue user to have a more in-depth knowledge of the various elements, we created a detailed description of each one. This description

varies a little bit depending on the element we are referring to. For instance, the description of the main qualities gives a succinct explanation and also how they relate to the dimension, or dimensions, that they relate. For the refined elements, we present an explanation, but also possible methods and tasks to achieve such qualities or goals.

- **Dimension** → In addition to the color scheme, we have a dimension property. The color scheme is more useful in a broad view of the catalogue, but if we want to clearly understand an element of the catalogue, its useful to have something that indicates us which dimension, or dimensions, it relates, without having to remember the color code.

Furthermore, the user can add custom properties on its will. This functionality will be further explained in Section 4.3.2. In Figure 4.5 we can see an example of the properties of an element of the catalogue, in this case, the system's functionality, which is a main quality of the economic and technical dimensions.

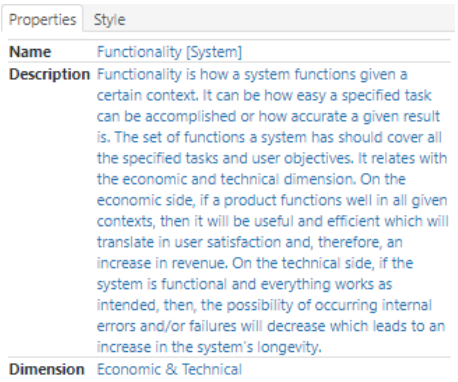


Figure 4.5: Properties of functionality [System].

4.3 Catalogue supporting environment

One of our objectives (see Section 1.3) was to have our catalogue able to adapt to different domains, being as generic as possible. The development of a static catalogue describing the various requirements for sustainability is not enough. Furthermore, sustainability is ever-evolving and is still an immature topic [7]. Therefore, the notions of sustainability can be different tomorrow. Thereupon, we came to the conclusion that our tool to handle the catalogue should have two major functionalities: configurability and modifiability. Configurability lets the user select a set of requirements across any combination of dimensions. In this way the user could only see the requirements that he needs for his domain, without unnecessary ones. We can see this functionality as a requirements filter. Modifiability lets the user modify the catalogue according to his needs or knowledge. In addition to this, the ability to save and load a custom catalogue is also a necessary functionality, otherwise the

user would need to make all the changes in a working session, which, most times, is not possible. Finally, we thought about possible functionalities that would enhance the user experience, such as labels. Some of these functionalities were developed by the author of this dissertation, in form of plugins, others were already implemented in the piStar tool, which was one of the reasons we select it as our base tool (see Section 4.1). Both the plugins and the native functionalities will be presented and discussed next.

4.3.1 Developed plugins (extensions of piStar)

The piStar tool allows the user to create and apply self-made plugins. Since piStar is an open-source project, one can study and understand the reasoning behind it, thus developing plugins compatible with the tool's working paradigm. A total of three plugins were developed for this work: (1) configurability of the catalogue (2) color label (3) element label. The programming languages used to develop them were *JavaScript* and *HTML*. These plugins, on the UI, are clickable buttons that when clicked, perform the specified function. Figure 4.6 shows the UI with those buttons.

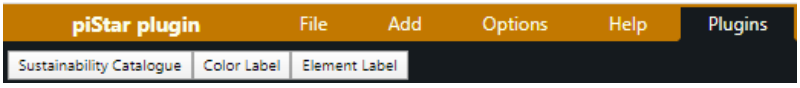


Figure 4.6: Plugin buttons on the UI.

As shown in Figure 4.6, we have three different buttons. The first, "Sustainability Catalogue", is responsible for the configuration of the catalogue, while the others, "Color Label" and "Element Label" are the two labels created to help the user. In this section, we will present these plugins, we will discuss why they were created, what their use is, and also show a visual representation of them.

Configurability

Since the catalogue tries to cover every aspect of the various sustainability dimensions and be as broad as possible, one might argue that this can pose the problem of information overflow for rather specific or small projects. We want our catalogue to be used in as many domains as possible (see Section 1.3) but we also want it to be useful and practical. For this reason, our catalogue is configurable, allowing the user full freedom to choose any set of the main requirements of each dimension. In this way, depending on the project domain and specifications, one can choose the more suited sustainability requirements and the catalogue will shape accordingly. For instance, if we have a project that only focuses on two dimensions of sustainability, we can "erase" the other two. In this way, we will have a more clearer view of what belongs and relates to what we want. We can configure and filter the catalogue according to these specific needs by clicking on the *Sustainability Catalogue* button (see Figure 4.6). Then it is presented a popup window containing four lists, each one associated with the dimensions. This window is shown in Figure 4.7.

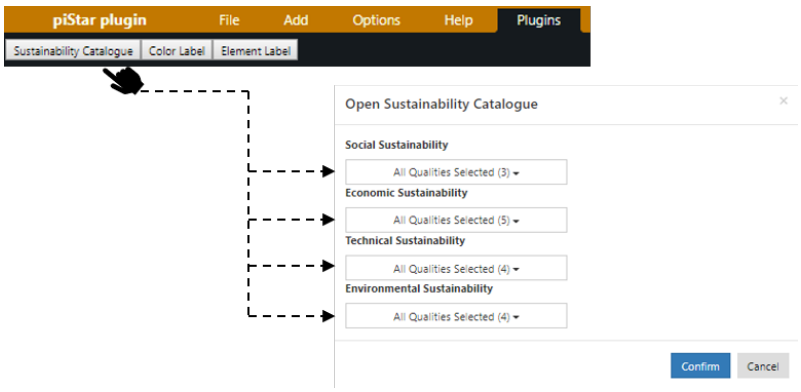


Figure 4.7: Sustainability catalogue window.

Each list is composed of the main qualities of each dimension, each one represented by a *checkbox*. We can select the qualities we desire of each dimension. The selected qualities will always be displayed to the user. For instance, in Figure 4.7 we can see that all qualities of all dimensions are selected. Upon confirmation, the result will be a custom catalogue based on them. All the possibilities and combinations of requirements are supported so that the catalogue is as general as possible. Figure 4.8 shows a possible combination and the steps required to do so.

Labels

For an easier understanding of the catalogue, two labels were created. A color label and an element label. The purpose of these two labels is purely for consulting. The user can, at any time, consult them and check the color typology of the catalogue (for the color label) or the semantics of each of its elements (for the elements label). It's important to notice that our color label is static, i.e., if an user has an element of his catalogue with a color, or colors, different from the ones that are depicted in the label, the label will, therefore, be outdated. We will discuss this gap in Chapter 6. Both labels are shown in Figure 4.9.

4.3.2 piStar native useful functionalities

As stated in Section 4.1.2, the piStar tool already offers an array of native functionalities. These functionalities proved to be extremely useful for the application and usability of the sustainability catalogue. The ones we will focus on are: the ability to modify any element, and to save and load models. In addition to this, the piStar tool offers complete freedom of choice on where to put the elements. The user can drag an element all around the canvas and put it where he wants. The modifiability of the tool and the ability to save and load models will be detailed in the following sections.

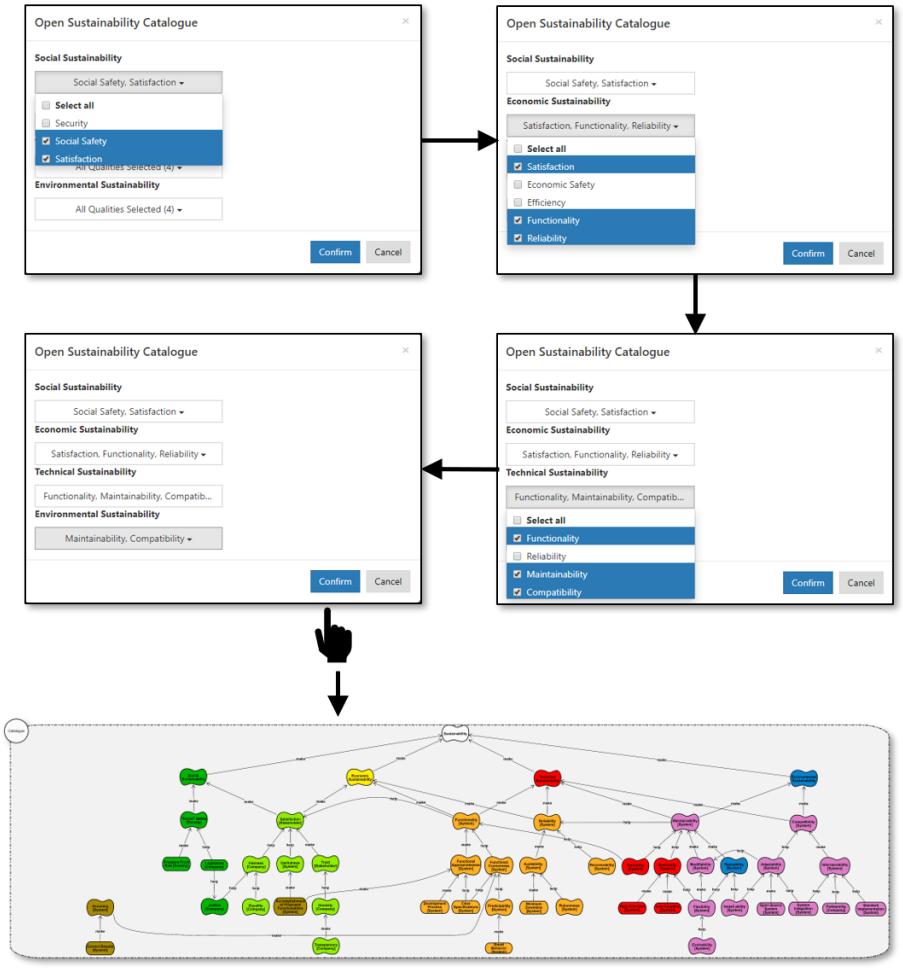
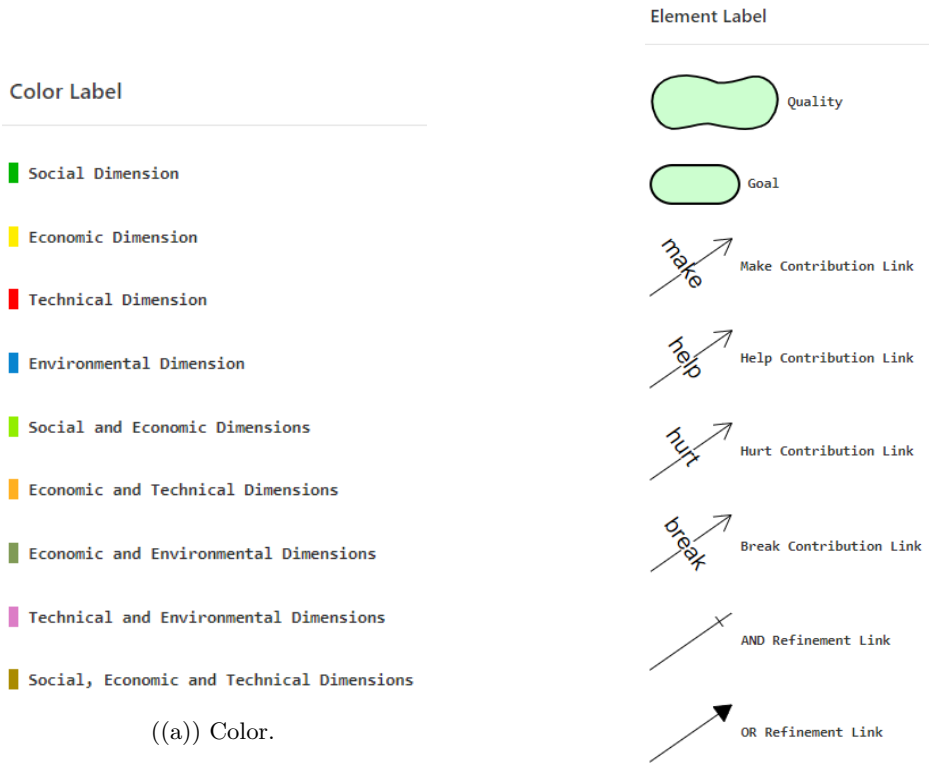


Figure 4.8: Configuration example.

Modifiability

Since sustainability is still an immature topic and is ever expanding on its concept [7], it is imperative to allow some type of update on the catalogue. Furthermore, depending on certain project needs, other requirements should be implemented. However, this modification is superficial and individual, that is, the catalogue in its origin is not modified, and its always available by clicking on the "Sustainability Catalogue" and selecting all the qualities. Furthermore, the user should always be careful when modifying the catalogue, such decision must be well-informed and validated. There are three different things we can do to modify the catalogue, according to our needs or knowledge:

- **Add** → The first most thing we should let the user do, is to add any element to the catalogue. The piStar tool lets the user add any element of the iStar 2.0 framework standards (see Section 4.1.2). To add an element, we simply select the element we want, drag it to the right position and link it, if needed, to another element on the catalogue. When adding an element, the user should not forget to update its information accordingly, like its properties or color. Figure 4.10 shows the add tab



((a)) Color.

((b)) Element.

Figure 4.9: Labels.

presented on the piStar tools with all the elements we can add to the catalogue.

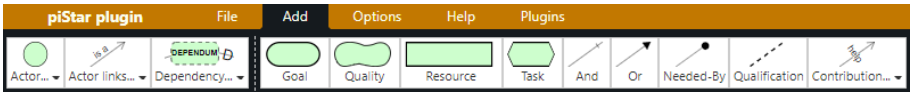


Figure 4.10: Add tab.

- **Delete** → Just as is important to let the user to add elements, it is also important to have the ability to delete them. Although the catalogue configurability already helps removing the requirements a user does not want, it only lets the user select the main qualities, thus staying with all the corresponding refined qualities. There can be some refined qualities that are not needed for the context of a domain, hence, the user should be able to delete them. The process to delete an element is simple: we select the element we want and press the *delete* button of our keyboard.

- **Update** → This lets the user update any information of the element. To update an element, we must select it, then change anything about its properties. The properties of an element are presented in a label, as seen in Figure 4.5. We can also add new properties or delete existing ones. Furthermore we can change the elements color, however, we advise to choose the color respective of each dimension (or combination of dimensions), otherwise our color label will be outdated (see Section 4.3.1).

Save & load work

Since it is not feasible to develop a catalogue, or change an existing one, according to project needs, in a unique work session, the ability to save and load existing work, is extremely useful and, hence, needed. The piStar tool lets the user to save its custom sustainability catalogue as an image file or as a text (*TXT*) file. A full-view image can be useful to gain a complete overview of the catalogue and can also be used to present it to others and discuss about it. However, when the user saves as a text file, the tool will promptly download such file. This file will contain all the information related of the catalogue (links, elements, relative position, properties) and can be later used to load it on the tool, so one can continue its work.

4.4 Application examples

Using our catalog in some real-world-inspired examples will allow us to get a sense of how useful or impacting it can be. Our aim, from the beginning, was to develop a sustainability requirements approach to contribute to a sustainable software development process. In this section we will use our catalogue in two different examples. The first example is a representation of a toll gate system, the *Via Verde* [2]. In this example we will focus on two different scenarios. In the first one, instead of relating what already exists in this system to our requirements, we will take our main requirements and list measures to make this system as sustainable as possible, via a thorough analysis of the system. In the second scenario, we will look to a perspective of the existing system and we will explain what are the required steps to implement a new quality to it. The second example is a more abstract one. Given the United Nations Sustainability Development Goals (SDGs) [3], we will explain which ones our catalogue can accommodate and, therefore, support.

4.4.1 Toll gate system

This example relate to a toll gate system, a simplified version of the *Via Verde* toll collection system, in use since 1991 on the Portuguese highways, and later, also in the Spanish tolls. In this system, drivers of authorized vehicles are charged at toll gates automatically, when passing in special lanes, known as green lanes. For this, drivers must register to receive a gizmo, which then must be activated using an ATM, associating the gizmo identifier with the vehicle owner bank account number for direct debits. The gizmo is read by the toll gate sensors of the special lanes, and the information is stored by the system and used to debit the vehicle owner account. The amount paid depends on three factors: distance traveled, the type of vehicle (a truck pays more than a regular car), and the highway concession. Upon passage, a light is shown to the driver. Green indicates that everything

[2] Via verde url: https://www.viaverde.pt/particulares/

[3] United Nations SDGs url: https://www.un.org/sustainabledevelopment/sustainable-development-goals/

went okay. However, if an exception is detected (e.g., the vehicle is non-authorized, the gizmo identifier is invalid, or the vehicle's class does not correspond to the registered one) a yellow light is turned on, an alarm sounds and the plate number is photographed.

First scenario

The goal of the first scenario is to show a general vision of what is needed for this system to be sustainable (we know that the *Via Verde* system already contains some of the measures we will present). Our sustainability catalogue has ten main non-functional requirements: security, social safety, satisfaction, economic safety, efficiency, functionality, reliability, maintainability, compatibility and environmental safety. For each of these requirements, we will provide a set of possible measures that should be used to make the service sustainable. Since we will analyze the system through all the main requirements of the catalogue, we will, thus, consider the four dimensions of sustainability we focused on this work: social, economic, environmental and technical. From a given dimension, we will analyse a main quality applied to a surrogate stakeholder (system, gizmo, driver, company, society, environment). Everything that we will discuss next is retrieved from analyzing the various elements of the sustainability catalogue. Each catalogue element in the discussion uses a typewriter font (`Courier`), to facilitate traceability.

- **Social dimension: `Security [Gizmo & System]`** → So that the *Via Verde* system is secure, both the driver's and vehicle's data (plate number, insurance, owner) must be confidential (refers to `confidentiality` in the sustainability catalogue) and the communication channels must have strict security protocols. Furthermore, for an increase of protection, the data should be encrypted by using strong encryption algorithms. The `integrity` of the data is also of an extreme importance, since the system shall never, in any moment, mistake the data of two different drivers and thus, debit an amount in the wrong bank account. Wrong payments amounts shan't happen as well. All the process of data capture and payment must be extremely reliable and accurate. This can be achieved by having a strong authentication (`authenticity`) of the data, but also by having efficient and complex data analysis algorithms, and a well structured database system.

- **Social dimension: `Social Safety [Society]`** → If there's any legal problem related with the service or with the debit payment, the company should always be available to help the driver solve the problem. There should not exist an abusive posture by the company (`justice`). The terms and conditions of the service must be available to the driver, and should not be ambiguous (`legislation`). Furthermore, the gizmo or the policies of the company must never propose a risk to the integrity of the user (`freedom from risk`).

- **Social & Economic dimensions: `Satisfaction [Driver]`** → There should be no problems with the proposed features. The use of metrics associated with the use of

the product must be adopted, a good example, are satisfaction questionnaires. In addition to this, the satisfaction of the user can also be increased by adding new functionalities, or making an existing one more simple or appealing (`usefulness`). Additionally, the company should adopt an honest and transparent posture to the users (`honesty & transparency`). By doing this, the user's trust on the service will increase (`trust`). An honest and trustworthy company is also an attractive one for new customers to acquire its services.

- **Economic dimension: `Economic Safety [Company]`** → For the *Via Verde* to be economically viable and safe, the company must have well-defined market and financial strategies to mitigate the potential economic risk, and know where to spend resources to improve its service (`economic risk mitigation`). This can be achieved by improving infrastructures (such as servers or database services), having a strong marketing department (with appealing and clever advertisements) or with an in-depth market study (such as the adoption of business intelligence). The fairness of the company in what relates to the employees management is also very important (`fairness`). There should exist good working conditions and fair wages. Social gatherings, clean working environment, and a flexible and available posture of the executives should be implemented.

- **Economic & Environmental dimensions: `Efficiency [Gizmo & System]`** → Since the snapshot of the vehicle's plate number is in real-time, the system must be efficient for, first, to not overcharge and, second, for all cases to be successfully resolved. This can be achieved by having high processing computing techniques, such as high-bandwidth communication channels, in-memory databases and a large storage capacity (`effectiveness`). However, these measures can present a large investment by the company, thus, a meticulous analysis of the implications must be carried on. Furthermore, the usage of more resources is also prejudicial for the environment (`resource utilization`), this can be counteracted, in part, by having resource sharing across different system modules. Possible partnerships with another companies and system's integration can also increase the efficiency of a system while using less resources, this is the basis of `interoperability`. Finally, speed restriction when passing through the toll gate can also be a measure to consider, yet, it can decrease the user satisfaction.

- **Economic & Technical dimensions: `Functionality [Gizmo & System]`** → The gizmo must be functional, i.e., it must accomplish the proposed functionalities (`functionality`). This can be achieved by having a suitable development methodology allied with clear and concise project specifications (`functional appropriateness`). Beside this, the system should not present errors when operating (`functional correctness`). Such errors can translate into a decrease of user satisfaction and/or in higher costs for the company. They can be avoided,

or at least mitigated, by having strong verification and validation techniques, and regular and methodical inspection checks (`testability`). Likewise, the usage of `smart behaviour` (such as artificial intelligence) so the system knows how to behave according certain situations can be something to consider.

- **Economic & Technical dimensions: `Reliability [Gizmo & System]`** → In case of a system failure, it must be able to recover as fast as possible to mitigate possible loss of information or cause a complete rupture of it (`recoverability`). This issue can be accomplished by having a robust (`robustness`) system with a strong `fault tolerance`. The system should have data backups and failure procedures, so it knows how to behave in case of a failure. The limitation of resources in the event of a failure is also an aspect to consider. Furthermore, such robustness leads to an increase of system's up-time (`availability`), which, in conclusion, translates in an increase of user satisfaction. Less system errors directly relates to the potential revenue of the company.

- **Environmental & Technical dimensions: `Maintainability [Gizmo & System]`** → The gizmo should be of easy maintenance (`maintainability`), i.e., if it has any problem, it should not be necessary many actions by the user. For instance, the battery of the gizmo should be detachable so the user can easily replace it. The materials used should also be reliable (`reliability`), the gizmo should be resistant, having an hardened case is a possible solution. However this can present a prejudicial contribution for the environment. In addition, the gizmo should alert the user when its malfunctioning, so that it does not reach a state of no repair. The good maintenance of the gizmo leads to an increase of its functionality.

- **Environmental & Technical dimensions: `Compatibility [Gizmo & System]`** → In terms of `compatibility`, there's one concern that stands out. If a driver changes his vehicle, should he get another gizmo? That would not be an optimal solution, neither for the user nor for the environment. When exchanging a vehicle, the system must be able to update the gizmo data to the new one (`adaptability`). Therefore, the gizmo can be exchanged, freely, to another vehicle, given correct and legal information. Moreover, the gizmo installation should be easy and support any type of vehicle (`install ability`). Finally, the system must be able to successfully connect the gizmo with the toll gates, but also with the respective banks (so that it can debit in the driver's bank account) (`system integration & partnership`). Thus, the system should be adaptable (different technologies supported and synchronized exchange of data) and there should exist a strong `interoperability`, among all the existing systems. For that, industry `partnerships` and `system integration` are possible measures to study.

- **Environmental dimension: `Environmental Safety [Environment]`** → Last but not least, in what relates to the `environmental safety`, the gizmo should

not contain noxious substances, or materials, to the environment (`freedom from hazardous risks`). Further, the materials should be, if possible, recyclable to reduce possible waste (`environmental health`). However, the usage of such materials can increase the cost of the gizmo's manufacturing (`financial management`). The ability to use the same gizmo in new vehicles (of a registered driver), as seen previously, is also a major factor to improve the preservation of the environment, since it is not need the fabrication of a new gizmo every time there's an exchange of a vehicle (`reusability & environmental health`).

Second scenario

In this second scenario we will look to a perspective of the existing system and we would like to increase its sustainability, by adding a new feature to it. We will detail the process, to consider such modification to the existing system, using our catalogue. Let's take as an example the possibility of an upgrade of the toll gates with better recognition technology software. Such upgrade would let to scan the vehicle's plate number directly without the need of an additional gizmo. This would be great for environmental sustainability since it would cease the production of gizmos, hence, less waste. For us to consider such change, we must first select a set of requirements that relate to this goal. We want to adapt the system to behave in a different environment while remaining efficient. Thus, we need adaptability and efficiency. Also, apart from those two specific qualities, the system should be functional but also well maintained. Now looking in a dimension perspective, we should consider the environmental one, since we are looking to eliminate the need of a gizmo. However, because we are dealing with an upgrade of the technical side of the system, we should also consider the technical dimension, since we want a functional and well-maintained system, for it to be durable and efficient.

Using our catalogue, we can select these qualities according to each dimension. We select "Functionality" under the "Technical Sustainability", "Efficiency" under the "Environmental Sustainability" and since "Maintanaibility" relates to both, we select from both dimensions (so we know how can we maintain a system to be as environmental-friendly possible). However, we can not select adaptability, since the catalogue configuration only lets us choose the main qualities, and not refined ones (this inability will be discussed in Chapter 6). Therefore, we must check what's the main quality responsible for the system's adaptability. With a quick check on the catalogue, we see that adaptability relates with compatibility. Thus, we need to also select "Compatability" from the check boxes. The result of this configuration will be a custom catalogue trimmed to address the needs of this specific problem. To understand what are the implications, and needs, of the implementation of such feature we must analyze the resultant catalogue.

4.4.2 United nations sustainability development goals

The united nations sustainable development goals (SDGs), also known as the global goals, were adopted by all United Nations Member States in 2015, as a universal call to action to end poverty, protect the planet and ensure that all people enjoy peace and prosperity by 2030 [52]. There are a total of 17 SDGs, as depicted in Figure 4.11.

Figure 4.11: United nations SDGs, from [52].

By providing a list of requirements that each sustainability dimension has and a detailed explanation of them, as well, as their relationships, being them intra- or inter, we can achieve a sustainable development. Since our sustainability catalogue focus on the development of software systems under the guidance of four sustainability dimensions (social, environmental, economic, technical), it can contribute to the following SDGs: (2) zero hunger, (3) good health and well-being, (7) affordable and clean energy, (8) decent work and economic growth, (9) industry, innovation and infrastructure, (11) sustainable cities and communities, (12) responsible consumption and production, (13) climate action, which makes a total of 8 goals, out of 17. For instance, more sustainable systems related to agriculture and harvesting can help decreasing the scarcity of food, hence contributing to SDGs (2, 3). An increase of environmentally friendly systems combined with the usage of clean energy, will have a positive impact against the global warming (3, 7, 13). Software systems that are sustainable on the economic and technical side will be responsible for an economic growth (3, 8, 9). If the development of the systems is developed in a sustainable way and the result is a sustainable system, then, it will lead to a sustainable world (11, 12). Our catalogue does not directly relates to these goals, since it is applied to software systems, regardless of, such software systems can be more sustainable by using our catalogue, hence, our catalogue indirectly contributes to the various SDGs we mentioned.

4.5 Conclusions

We were able to develop a sustainability catalogue that covers four, out of five, dimensions of sustainability while depicting numerous relationships among its elements. The iStar framework and the piStar tool proved to be the right choice in what regards the supporting technology. Further, we developed four dimensions catalogue (one for each dimension) where each one, can be seen as a subset of the sustainability catalogue. The catalogue itself has a set of functionalities, and characteristics, that aim to help the user to use it in an efficient way, while making the catalogue more generic and complete. Finally, we applied the catalogue to different use cases scenarios, and explained how we should use the catalogue to accomplish certain development goals.

CHAPTER 5

Evaluation

The work developed was assessed using a questionnaire sent to 89 participants and also a guide for the sustainability catalogue. The results collected were analysed using the Tableau software and a discussion of the results and analysis, together with a description of the entire evaluation process is offered by this chapter.

5.1 Tools

The first step of the evaluation process was the creation of a questionnaire and a guide that introduces the catalogue to the participant (helping answering the questionnaire). The questionnaire has a set of questions related to the usability and importance of the catalogue. The answers from the respondents form the basis of the evaluation. The guide serves as a background for our work. The questionnaire and the guide were sent together to 89 people, and 50 answers were obtained, which will be discussed in Section 5.2.

5.1.1 Questionnaire

The questionnaire was the main tool to assess our work [1]. Therefore, it had to be complete and explicit. It was made using *Google Forms*, which is an application developed by Google with the purpose of letting anyone to create online surveys for personal use. We selected this application since it presented a great usability and customization features, it is well known and standard, and lets the user to export the answers in various formats [19]. We grouped the questionnaire into five sections:

- **Introduction** → This section defines the objective of the questionnaire, gives a small scope of the work, informs that reading the guide is mandatory (Section 5.1.2),

[1] The questionnaire is available in the following link: https://docs.google.com/forms/d/e/1FAIpQLSfElPovoEHZF5hf77kh48pWrDHGrTD173aC_GpqTlPcktTPLQ/viewform?usp=sf_link

retrieves the email of the respondent, and, finally, informs the respondent that both, his answers and his email are confidential, i.e., only anonymised answers were used in the final results.

- **Personal data** → This section contains questions about the personal data of the participants: age, academic training, degree, and experience (in years) of software development. These questions serve for us to understand the heterogeneity of participants, which will be used as a filter in the discussion of the results (see Section 5.3).

- **Guide questions** → This section aggregates a set of questions about the guide. Since our guide was mandatory, the answers to these questions would allow us to know if both the explanations and illustrations were readable and understandable.

- **Catalogue questions** → This section contains the questions about the catalogue itself, grouped as follows:

  - Three questions about the quality of the catalogue, in terms of readability, interest and utility. These questions have the same possible answers, a number, from 1 to 5, where 1 is equivalent to very weak, 2 is weak, 3 is neutral, 4 is good, and 5 is very good
  - Two questions enquiring if the concepts of the catalogue are clear, and if the catalogue provides a general and concise idea about sustainability. The scale used has 5 values, ranging from strongly disagree to strongly agree.
  - Two free-style questions to understand about the strong and weak points of the catalogue, one for each topic, where a text box is offered for the respondents' opinion.
  - One question asking if the participant would use the catalogue as a basis, for his/her future projects related to sustainability. This asks about the true usefulness of our work. The choices to answer are: "yes", "no", and "perhaps".

- **Qualitative questions** → The purpose of these questions is to allow participants to share the thoughts about our work. These questions relate to positive or relevant aspects of the catalogue that the respondents could point out, but also aspects that should suffer a change or are less positive. The respondent can also leave some final comments or suggestions.

The set of questions we selected covers a different array of aspects of our catalogue, from usability to utility, but also asking if it provides a concise notion of what is sustainability and how it can be addressed and, most important, if it can be used in projects related to sustainability.

5.1.2 Guide

So that our participants could answer the questionnaire with sufficient information about the catalogue, a guide was created, offering a background of what is the sustainability catalogue and what we can do with it. The guide was made using *Microsoft PowerPoint* and contains both explanatory text and illustrative images across 10 pages. It contains two main sections:

- **Catalogue** → This section provides an overview of the catalogue, summarizing the information discussed in Section 4.2.2.
- **Functionalities** → We explain all the functionalities of the catalogue, from the labels to the configurability. The information presented here is identical to the one seen in Section 4.3.

5.2 Participants selection

We wanted the set of participants to cover a wide scope of people, from different backgrounds and with distinct experiences. With this, it is possible to have a broader view of the impact of our work. Thus, our set of people who answered our questionnaire can be divided into two groups:

- **Bibliography authors** → We invited all the authors from our bibliography list related to the topic of software sustainability. These are experts, that have worked first hand on our topic of interest, so they are capable of provide us well-informed and experienced feedback.
- **Academic personnel** → We also wanted feedback from academics, both staff and students, some of which with less knowledge and experience with the subject. Furthermore, we extended our set outside computer science to include people with other type of degrees, such as: mechanical, chemical and biochemical, and electrical engineering's. This was due to the fact that we wanted to know how people outside the scope of this work view our work and how it could impact them.

A total of 41 bibliography authors and 48 academic personnel were selected, making a total of 89 people. We sent an email to each person with an introductory formal text, accompanied with the abstract of this dissertation, followed by the link of the questionnaire and the guide as an attachment. We settled the time period to gather answers as 2 months, after which we would start the result analysis (see Section 5.3). We had 15 questionnaire answers from the first group and, and 35 from the second, in a total of 50 answers. Figure 5.1 shows the number of authors who answered the questionnaire versus the number of ones that did not answered. Figure 5.2 shows the same but with respect to academics.

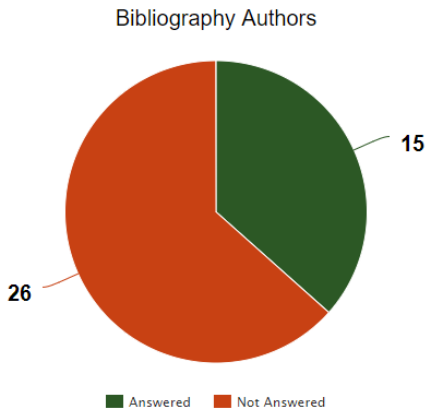


Figure 5.1: Number of answers from authors of sustainability-related topics.

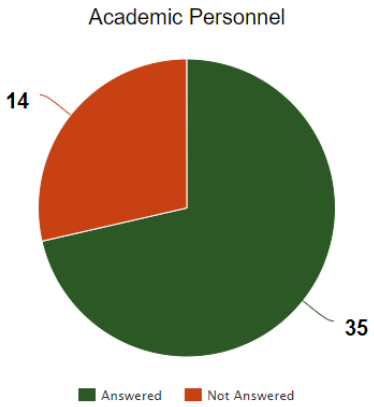


Figure 5.2: Number of answers from academics.

5.3 Discussion of the results

Having the answers, we could begin to analyze them to ascertain and discuss the results. In order to extract the relevant information we used two tools: *Microsoft Excel* and *Tableau*. Tableau is an interactive data visualization tool, that lets the user connect, modify, visualize and share data. The first step in getting the data ready for analysis, was to extract it from the Google Forms to an Excel spreadsheet. We also performed data cleansing to make sure that all the data was homogeneous, and that there wasn't incorrect data. An example of this data cleanse, was to identify participants that were from Computer Science, who instead of answering "Computer Science", answered a similar option, like "Computer Engineering", referring to the degree of the participant. From there, we loaded that same spreadsheet into Tableau as our data source. Our data source, which is a large table of data, had as columns the questions, and as rows the answers. Figure 5.3 shows a part of our data source.

# Sustainability Catalogue What is your age?	Abc Sustainability Catalogue What is your level ...	Abc Sustainability Catalogue Degree
24	Graduation	Computer Science
24	Graduation	Chemical and Bioche...
23	Graduation	Computer Science
23	Master's Degree	Chemical and Bioche...
25	Master's Degree	Micro and Nanotechn...

Figure 5.3: Tableau data example.

With the data ready, we decided to evaluate it in a general way and in a specific, filtered one, to make our study as complete as possible. The first takes into account all responses from all participants and we discuss the results based on the comparison between each of them, as a whole. In the second way, we filter the responses by groups

of participants, with similar characteristics, previously chosen, and compare the results between the various groups. General appreciation is quite important but we also wanted to understand what groups of people, with the same characteristics, think about our work and how the differentiate from other similar groups. For each question, we performed a respective query, in our data. In the next two sections we will show the results to our questions and discuss them.

5.3.1 General questions

As previously discussed, our general questions relate to the totality of our data set. We grouped them in four different groups: personal data, ratings, main question and qualitative questions. Each group and respective questions will be discussed next.

Personal data

The first thing we wanted to know was the characteristics of our respondents. For that we defined three queries:

- **Number of participants per degree** → The aim of this query was to know how was the distribution of degrees among all participants and how many different degrees there were. There was a total of 6 different degrees with computer science being the dominant one with 31 participants. The rest of the degrees makes a total of 19. The results can be seen in Figure 5.4.

Degree	
Biomedical	2
Chemical and Biochemical	2
Computer Science	31
Electrical and Computer	6
Mechanical	5
Micro and Nanotechnology	4

Total Number of Participants: 50

Figure 5.4: Number of participants per degree.

- **Percentage of participants per academic training level** → We only had three possible answers in what regards the training level of a participant, being them: graduation, master's degree and PhD. PhD was the dominant group, with a total of 40% of the participants. Figure 5.5 shows the percentage of each academic training level across our data set.

- **Years of experience in software development** → On this query we had a lot of different answers, since we let the respondents answer the exact number of years of their experience. We had values ranging from 0 to 30. However, we grouped these values into three groups: 0 years to 5 (excluded), 5 years to 10 (excluded), and 10

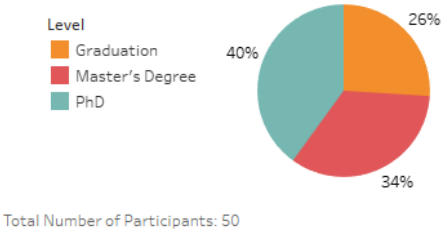


Figure 5.5: Percentage of participants per academic training level.

years or more. The reasoning behind this grouping will be later discussed in Section 5.3.2. We had an evenly distribution among all groups. Figure 5.6 shows the number of participants of each group.

0-5	19
5-10	13
10+	18

Total Number of Participants: 50

Figure 5.6: Number of participants per group of years of experience in software development.

As discussed in Section 5.2, we wanted a heterogeneous set of people. As shown in the previous queries, we achieved this goal, since we have participants from 6 different degrees, distinct academic training levels, and with distant years of experience in software development.

Ratings

Moving now to concrete data for evaluating our work, both the guide and the catalogue. To have a general idea about the quality of our work, we decided to make an average, across all answers, of the range of the values that we inquired. We had five possible values: 1 - Very Bad; 2 - Bad; 3 - Neutral; 4 - Good; 5 - Very Good. We posed a total of three queries:

- **Average rating of the aspects of the guide** → We asked our respondents to rate our guide in two aspects, explanation and illustration. We have an average rating of a little more than 4 in both aspects, which is quite positive. The results are shown in Figure 5.7.
- **Average rating of the aspects of the catalogue** → Likewise, we wanted to know the rating of the aspects of our catalogue. We have three different aspects that we used to rate it: readability, interest and utility. Readability was the aspect with the lowest average rating, which can be explained due to the fact that we made the catalog in an external tool and not in a native one, which could improve the visualization

Guide Explanation	4.0408
Guide Illustration	4.2000

Rating goes from 1 (Very Bad) to 5 (Very Good)

Figure 5.7: Average rating of the aspects of the guide.

of the catalogue (see Section 6.2). Both utility and interest had an average rating above 4, with interest ranking a bit higher. This also makes sense since software sustainability is an emerging topic and there is an urgent need to address it [10]. Figure 5.8 shows the average rating of all aspects.

Readability	3.9200
Interest	4.1400
Utility	4.0800

Rating goes from 1 (Very Bad) to 5 (Very Good)

Figure 5.8: Average rating of the aspects of the catalogue.

- **Rating of the catalogue as a whole** → The sustainability catalogue is a complex tool, offering different functionalities, therefore, we wanted to know how the participants rated our catalogue as a whole. We had a total of 84% of positive answers, 30% correspondent of a rating of 5 and 54% of a rating of 4. We had a mere 2% of negative answers and a total of 14% of neutral answers. Figure 5.9 shows the percentage of participants for each rating value.

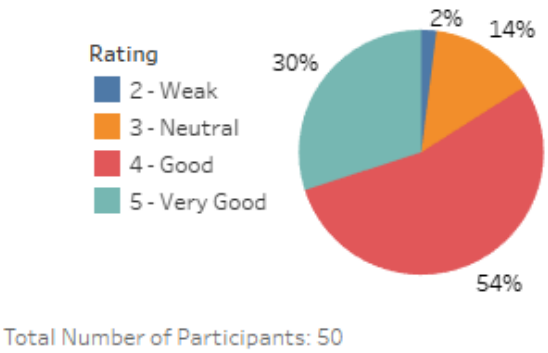


Figure 5.9: Rating of the catalogue as a whole.

We would like to emphasise some relevant points to keep in mind. In all results obtained in the rated answers, either guide or catalogue, we did not get any answer with rating of 1

(Very Weak), and only one answer with rating of 2 (Weak). We perceive these results as very positive, which further proves the utility of our work.

Main questions

We defined two questions about the general appreciation of our work. These questions can be seen as the most important in our assessment, as they provide a final view of what our work represents and how useful it can be in future sustainability projects. These are:

- **Does the catalogue give a more general and concise idea of sustainability requirements?** → Regardless of whether our catalog may be useful, or well built, one of our main objectives was to give a clearer idea of what sustainability is, and its requirements, so that it could be well addressed when developing software. Given our answers, we had a total of 84% of positive ones, 16% neutral and 0% negative. The dominant answer was "agree" with a total of 64%. The results can be seen in Figure 5.10.

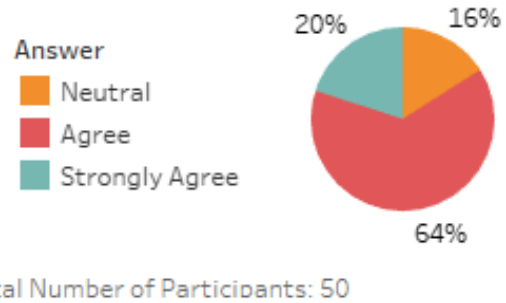


Figure 5.10: Answers on whether the catalogue gives a more general and concise idea of sustainability requirements.

- **Would you use the catalogue, as a basis, for future sustainability projects?** → Our final question is based on whether people would use our catalogue, or not, for future sustainability projects. In order for the results to be concise, we only gave three different possible answers, "yes", "no", and "perhaps". As with all other questions, the results were quite positive. We had a whopping 50% of "yes" answers and only 2% of "no", the rest, 48%, belongs to "perhaps". The results are shown in Figure 5.11.

Qualitative questions

As told in Section 5.1.1 we wanted to know what our respondents thought about our work, in a qualitative manner. Since any participant could answer whatever they wanted, we had many different responses, but we could see some kinds of patterns between them. Regarding the positive aspects of our work, those most mentioned by the respondents were: understandability, simplicity, configurability and completeness. About the aspects

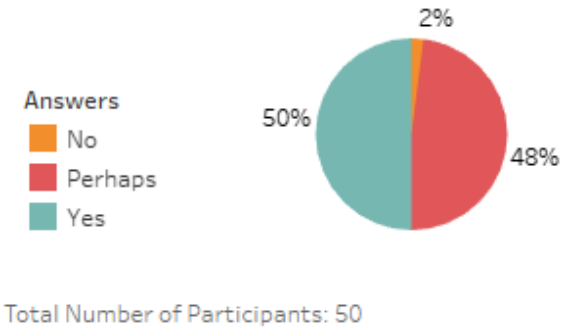


Figure 5.11: Answers on whether the participant would use the Catalogue, as a basis, for future sustainability projects.

to modify, the color palette and the need for a use case or an example, were the most cited ones. At last, on final comments we had compliments about the importance and completeness of our work, but also some suggestions to improve it. We even had some respondents asking if they could access the final work once finished, and various suggestions to make our work fully open-source and accessible to anyone. Some of the participants quotes are presented next.

- "The catalogue itself is a really helpful tool to be used in "real life" companies, since nowadays sustainability is a "hot topic" and its interest is ever growing. Not only is it very easy to use, it is very complete and possibly adaptable for each end goal and field."

- "The catalog is a great idea that will assist engineers in creating sustainability goals for their projects. I liked the idea of structuring the problem in multiple layers and establishing intra/inter relationships. I do believe this will easy the life of those engineers looking for pragmatic guidance to add sustainability concerns to their requirements models. It is indeed a promising field and I look forward to hearing from you the next chapters of this research."

- "The catalog is a great idea that will assist engineers in creating sustainability goals for their projects. I liked the idea of structuring the problem in multiple layers and establishing intra/inter relationships. I do believe this will easy the life of those engineers looking for pragmatic guidance to add sustainability concerns to their requirements models. It is indeed a promising field and I look forward to hearing from you the next chapters of this research."

- "Lots of work was put into this thesis and it looks really amazing! I hope this can cause impact on software development and make sustainable development a more standard thing, since from what I see in the industry, there is little thought in it."

- "Maybe I would flag a symbol (adding a *, for example) when it has been changed by the user and deviates from the original purpose. Also, I would create some mechanism that allows engineers extending the catalog for interrelationships."

5.3.2 Filtered queries

As discussed previously in this section, we wanted to evaluate our work in a filtered way, taking into account groups of people with similar characteristics, so that we could have a more structured assessment. Although we already had two groups of people in our set, bibliography authors and academic personnel (see Section 5.2), we did not used them to filter our results. The reason is that we do not know, in concrete, the experience that the people inside these groups have, or not, regarding sustainability or software development, or both. There may be a person from the bibliography authors with less experience in these topics, than an academic person, and vice-versa, we simply do not know.

Therefore, in order to analyze the results in a more specific way, we filtered the results by having two accurate aspects. The degree and the years of experience in software development. The degree of a person was chosen as a filter since a person from Computer Science will likely have more experience with software development and requirements, than a person from Chemical Engineering, for instance, and will see our tool with a different perspective as well. The second filter, years in software development, was chosen because we wanted to know how people with different years of experience perceive our catalogue as a useful tool to be used in projects. We excluded the academic training level because that data can be ambiguous, i.e., a person can have a PhD and have less experience with the covered topics, than a person with a Graduation level.

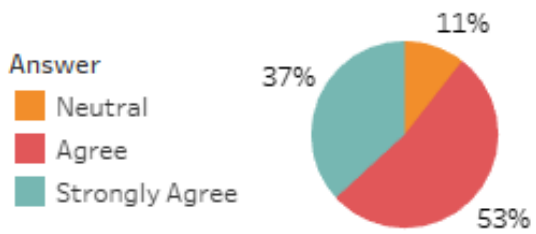
Between these two filters, we wanted to compare the results between Computer Science people and non-Computer Science, and between certain groups of years in software development (see Figure 5.6). For these filtered queries we focused on the main questions (see Section 5.3.1) since, as the name indicates, they are the core part of our assessment. The ratings, despite being great indicators, were excluded since the results among the different types of groups were similar to the ones discussed in 5.3.1, and the disparities of results between the different filters were very small, which makes them irrelevant for the context of this section. Both filters are discussed next.

Degree

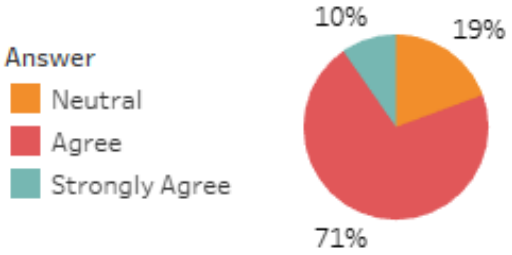
Our goal is to clearly analyse the difference of responses from Computer Science and non-Computer Science participants. These two groups may have a very different interest in our work since they come from different backgrounds and experiences. Our results shows that our catalogue is more useful, regarding the notions of sustainability requirements, for non-Computer Science people. We had approximately 90% of positive answers on whether the catalogue gives a more general and concise idea of sustainability requirements, split in 37% for "strongly agree" and 53% for "agree", for non-Computer Science participants.

For Computer Science ones, we had 71% for "agree" and only 10% for "strongly agree", making a total of 81% positive responses. The rest of the participants answered "neutral".

About using the catalogue for future sustainability projects, the results were reversed, that is, we had more positive responses ("yes") on Computer Science people, 55%, than on non-Computer Science, 42%. These results may indicate that people from Computer Science have a better perception of what are software requirements, and how to deal with them, thus they are more keen to use our catalogue. Such results make sense, since people of this scope have, since early on, contact with different software tools that provide a variety of possibilities, making them more fit to use others. The results comparison can be seen in Figure 5.12, for answers on whether the catalogue gives a more general and concise idea of sustainability requirements, and Figure 5.13 for answers on whether one would use the catalogue for future sustainability projects.

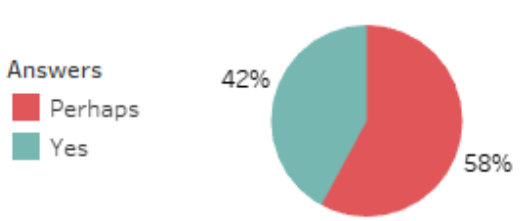


((a)) non-Computer science.

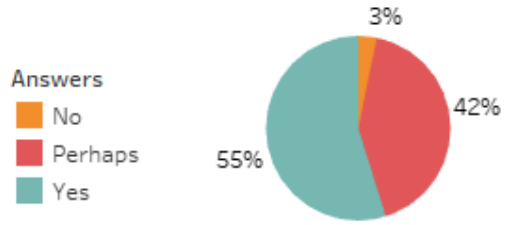


((b)) Computer science.

Figure 5.12: Degree - Result comparison on whether the catalogue provides a more general and concise idea of sustainability requirements.



((a)) non-Computer science.



((b)) Computer science.

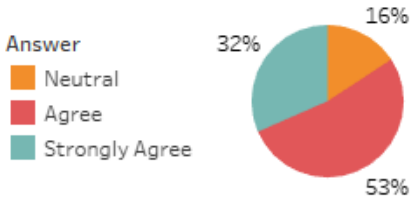
Figure 5.13: Degree - Result comparison on whether the participant would use the catalogue for future sustainability projects.

Years of experience in software development

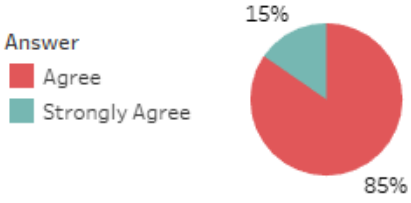
In our discussion of results, we put people with different years of experience in software development in three different groups: 0 to 5 years, 5 to 10 years, and 10 years or more (see Figure 5.6). We can see these groups as junior, intermediate and senior, respectively, regarding software experience. The critical thinking of each group is different since, with more years of experience, the knowledge and insight of software tends to be higher.

The first thing we did, was to see how our catalogue helped to give a more concise idea of sustainability requirements across the different groups. The senior group, was the one with the least positive answers, having a total of 72%, 61% agreed and 11% strongly agreed with this topic. The group with the most positive answers was the intermediate group with 100%, however it was on the junior group we had the most "strongly agree" answers, 32%. This clearly shows that people with more experience already have a more concrete idea of what sustainability requirements are, unlike people with less experience.

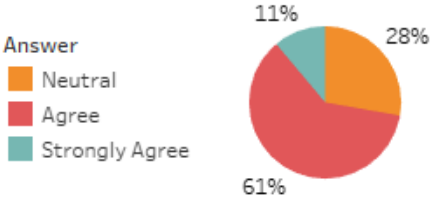
It was on the possible use of the catalogue that we had the most interesting results. The intermediate group was the one that is evidently more avid to use our catalogue, with a total of 62% participants answering they would use it in future projects related to sustainability. The junior group had the worst responses, with only a total of 42% "yes", while the senior one had 50%. The only participant that answered "no" on this topic, is a senior one. People with few years of software development experience may not have enough knowledge and cunning to apply the catalogue to real-world software problems. On the other hand, the seniors developers, even if they perceive our work as a useful and interesting one, may already have standard software tools and protocols that they are used to. Finally, the intermediate ones have the necessary insight and predisposition to apply different tools to their working methods, in order to improve them, which makes them the keenest group for the use of our catalogue. The results comparison can be seen in Figure 5.14, for answers on whether the catalogue gives a more general and concise idea of sustainability requirements, and Figure 5.15 for answers on whether one would use the catalogue for future sustainability projects.



((a)) Junior.

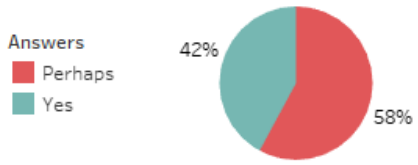


((b)) Intermediate.

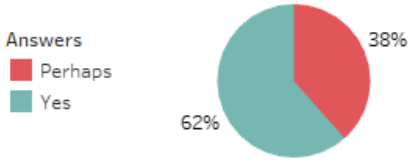


((c)) Senior.

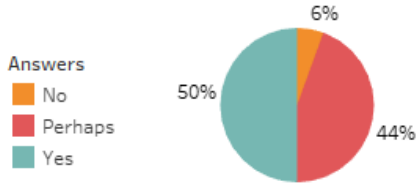
Figure 5.14: Software Experience - Result comparison on whether the catalogue provides a more general and concise idea of sustainability requirements.



((a)) Junior.



((b)) Intermediate.



((c)) Senior.

Figure 5.15: Software Experience - Result comparison on whether the participant would use the catalogue for future sustainability projects.

5.4 Threats to validity and reliability

There are always threats to the validity and reliability of any evaluation process. Despite this, we are able to mitigate such threats. A threat to our questionnaire is that we might not ask the correct questions, or it may have ambiguous questions. To mitigate this, a segmentation of the questionnaire was performed so we could clearly separate different evaluation topics and we were very careful on the wording and structure of the questions. Furthermore, the participant may not have enough knowledge to answer such questions. Thereunto, we constructed a guide for our work. However, it may be too complex for the participant, or it may fail passing the information that we want. We tried to make our guide as succinct as possible and easily readable with the aid of visual illustrations. The method behind the analysis of the results can also pose a major threat to the validity, and reliability, of our work, since incorrect data analysis methods may wield wrong results, thus invalidating the analysis. To lessen the possibility of this threat, we cleansed the data to have an homogeneous data set and we performed the analysis through a set of distinctive queries, where each one corresponds to a question of our questionnaire, thus the discussion of the results are directly related to the participants' answers. We also performed two different analysis, a general and a filtered one. Finally, we used a well-known, and efficient data analysis tool (*Tableau*) that provides an array of functionalities and graphs to make the analysis as complete as possible. Despite all the efforts that we took, we can not truly know how reliable a participant answer can be, since the answer is directly linked to his/her predisposition, and effort to answer each question. This is an external factor of our evaluation, where we do not have any control.

5.5 Conclusions

With the aid of various tools, *Google forms*, *Excel*, *Tableau*, we did a meticulous and complete evaluation of our work. Such evaluation was only possible because of the answers we had from our questionnaire, that was sent to 89 people, where 50 answered. So that our respondents could have a greater insight of our work and to help on answering the set of questions we had, we made an illustrative and detailed guide. The feedback we received, from the 50 participants, was positive across all aspects. We even had some bibliography authors eager to check the final results of our work.

CHAPTER 6

Final Conclusions and Future Work

It is time to draw our final conclusions with regard to all the work done for this master's thesis. What were the main obstacles we encountered. We learned that sustainability is, every day, more important in our world, and in the development of new technologies. This work, the sustainability catalogue, aims to contribute to such a need. We developed, and presented, a catalogue able to instantiate four dimensions of sustainability accommodating its properties (requirements and relationships). However, the improvement of the sustainability catalogue should be of extreme importance, so it can be as complete, and as useful, as possible. In this chapter, we start by mentioning the final conclusions about our work, and then we present some ideas, and guidelines, for possible future work.

6.1 A brief history about our work

Our work, as a whole, can be split in four main activities: (1) idealization, (2) state of the art, (3) development, and (4) evaluation. In each of these activities we faced numerous decisions and obstacles in order to achieve the respective goals. Before drawing the final conclusions of our entire work, we will first discuss each of the mentioned activities, so that, we are able to have a more careful and detailed analysis of our work. We will also revisit the challenges that were defined in Section 1.2.

At the beginning of this dissertation the contribution goal was the sustainability of software development. We knew that sustainability was, and is, a very important topic in our society, but also in our technology. Thus, we wanted to develop a tool, a framework, an approach, to help software development to become more sustainable than it is now. We envisioned a solution, that would combine requirements engineering and sustainability requirements. Since sustainability is such an important topic, we must address it as early as possible during the software development process, so that we can assure its

implementation. Additionally, we wanted a generic solution that could be applied to a wide range of projects and systems' domains. However, we didn't know what requirements could relate to sustainability, nor what requirements engineering techniques could be used, to assure a maximum efficiency. All these problems and challenges were presented, and discussed, in Section 1.2. With an envisioned solution in mind and a clear focus of what to search for, we started to analyze the current state of the art.

The second activity, the analysis of the state of the art, was one of great time consumption. We wanted to make sure that we researched what was needed, and also that this study was as complete as possible so that the loss of relevant information was minimal. That was our first, and main obstacle on this activity. To solve it, we decided to carry out a systematic mapping study (Challenge 1 solved). Such study uses a combination of search techniques allied with a meticulous work process. We searched on two main topics: requirements engineering activities (elicitation, prioritization, and trade-off analysis) and sustainability requirements. Additionally, we wanted to know whether some type of approach was already developed, and if so, which one(s). From this study we were able to elicit numerous requirements engineering techniques (Challenge 2 solved), but also knowing what type of requirements contribute to making software development more sustainable, and how important sustainability is for software development (Challenge 3 solved). With the study carried out, we were sure that our approach could be one of relevant importance for this subject. The next step was to transform what was an idea into something tangible and functional.

The development of our solution was, with no doubts, our hardest and most complex activity. The first challenge was to find a suitable solution for such problem. What should we do, or what could we do. We performed various brainstorming sessions until we got our final idea, a sustainability catalogue. This catalogue would contain four (social, economic, environmental, and technical) out of the five sustainability dimensions (we excluded the individual dimension, see Section 3.1) and their respective properties, such as their relevant requirements and relationships, be them inter- or intra-dimensional. The next obstacle was to know how to develop it, which techniques and/or tools could be appropriate, but also, what development methodology to embrace. We selected the iStar framework combined with the piStar tool allied with a bottom-up development approach. Such selection was based on a set of characteristics, and qualities, that we would like to have. With the catalogue done, we were still left with one last obstacle. Could our catalogue be as generic as possible, i.e., could our catalogue be applied into different domains while remaining useful and practical? To solve this issue we developed a plugin in the piStar tool. This plugin lets the user to configure and adapt the catalogue to his needs by letting him choose any set set of requirements across any set of sustainability dimensions. Furthermore, we added a detailed description of each and every element of the catalogue. With this, the catalogue can be applied in a wide range of projects, thus, making it generic, and also complete (Challenge 4 solved). With our catalogue, and its functionalities, fully developed, it was time for its evaluation.

Our evaluation was conducted through two major groups of people: bibliography authors (that related to sustainability in software development) and academic personnel (students, teachers, researchers). An early challenge that we came across was how to contact these people. One very important thing that was essential, is that we needed to summarize the content of our work, in an understandable and visually appealing way, since it would be our only way to show it. Thereunto, a ten page visual guide, explaining the sustainability catalogue in its entirety, was built. Furthermore, we also needed a questionnaire with a careful set of questions, to present to the respondents, so they could evaluate our work according to the aspects we considered important. These aspects was the readability, interest and usefulness of the catalogue in software development, but also if it provided a clearer notion of sustainability, and if they would use the catalogue in future projects, related with the subject. We had a total of 50 answers, from different ages, degrees, and academic experience. The average rating of the catalogue based on the aspects was positive (rating 4.1 out of 5) and the majority (98%) of the participants would use, or at least consider using, the catalogue in future projects. We were quite happy with the evaluation results and, in a way, we got the sense that we had accomplish our initial goal.

In conclusion, we started with a concept, an idea to reach a goal, and we are sure that we have achieved it. Our work, a sustainability catalogue, is one of its kind, capable of adapting to numerous project domains while remaining loyal to its purpose, and always providing a complete and detailed analysis of the needs of sustainability in its various dimensions. But mainly, we hope that this work has made people aware of the need to incorporate sustainability, not only in our lives, but also in our technological systems. We encourage everyone to adopt sustainable measures, and contribute to this goal that belongs to all of us.

6.2 Future work

Although we have finished our catalogue, with regard to this MSc dissertation, there are some characteristics that can be improved. Some were not implemented due to lack of time and some complexity constraints, while others are too elaborate and their development alone is sufficient for another work of this kind. These constitute part of future work.

In a first instance, we should incorporate the individual dimension into the catalogue. As explained in Section 3.1, we left out the individual dimension for simplicity due to extensive work already needed to handle four dimensions. Having said this, we did incorporate part of it into the social dimension. Nonetheless, having the individual dimension would enrich the content of our catalogue, and further relationships could be analyzed. Furthermore, the catalogue's configurability could be more extensive. In other words, would be interesting if we could chose the refined qualities as well, rather than just the main ones. Now in regard to more complex and elaborate perspectives, would be of great interest to have a native web application, to support the sustainability (and the dimensions)

catalogue alone. This would serve as a portal for anyone to access the catalogue in a much impactful and interesting way. This web-application could open the doors to an array of new functionalities, such as: (1) adaptive labels, (2) working sessions, (3) ease of look-ups, (4) better and more information about any element, (5) sharing option.

Bibliography

[1] *A collection of quality characteristics | Arborosa.* (Accessed on 08/26/2019). URL: https://arborosa.org/2015/07/15/a-collection-of-quality-characteristics/.

[2] T. Abdou, P. Kamthan, and N. Shahmir. "User Stories for Agile Business: INVEST, Carefully!" In: Oct. 2014.

[3] M. Abul Kalam Azad, M. Islam, and M. Ismail Hossin. "Generation of electronic-waste and its impact on environment and public health in Malaysia." In: *Annals of Tropical Medicine and Public Health* 10.5 (2017), pp. 1123–1127. DOI: 10.4103/ATMPH.ATMPH_349_17. eprint: http://www.atmph.org/article.asp?issn=1755-6783;year=2017;volume=10;issue=5;spage=1123;epage=1127;aulast=Abul;t=6. URL: http://www.atmph.org/article.asp?issn=1755-6783;year=2017;volume=10;issue=5;spage=1123;epage=1127;aulast=Abul;t=6.

[4] R. H. AL-Ta'ani and R. Razali. "Prioritizing Requirements in Agile Development: A Conceptual Framework." In: *Procedia Technology* 11 (2013). 4th International Conference on Electrical Engineering and Informatics, ICEEI 2013, pp. 733 –739. ISSN: 2212-0173. DOI: https://doi.org/10.1016/j.protcy.2013.12.252. URL: http://www.sciencedirect.com/science/article/pii/S2212017313004064.

[5] C. Becker, R. Chitchyan, L. Duboc, S. Easterbrook, B. Penzenstadler, N. Seyff, and C. C. Venters. "Sustainability Design and Software: The Karlskrona Manifesto." In: *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*. Vol. 2. 2015, pp. 467–476. DOI: 10.1109/ICSE.2015.179.

[6] C. Becker, S. Betz, R. Chitchyan, L. Duboc, S. M. Easterbrook, B. Penzenstadler, N. Seyff, and C. C. Venters. "Requirements: The Key to Sustainability." In: *IEEE Software* 33.1 (2016), pp. 56–65. ISSN: 0740-7459. DOI: 10.1109/MS.2015.158.

[7] C. Becker, R. Chitchyan, L. Duboc, S. Easterbrook, B. Penzenstadler, N. Seyff, and C. C. Venters. "Sustainability Design and Software: The Karlskrona Manifesto." In: *Proceedings of the 37th International Conference on Software Engineering - Volume 2*. ICSE '15. Piscataway, NJ, USA: IEEE Press, 2015, pp. 467–476. URL: http://dl.acm.org/citation.cfm?id=2819009.2819082.

[8] C. Calero and M. Piattini. "Puzzling out Software Sustainability." In: *Sustainable Computing: Informatics and Systems* 16 (2017), pp. 117 –124. ISSN: 2210-5379. DOI: https://doi.org/10.1016/j.suscom.2017.10.011. URL: http://www.sciencedirect.com/science/article/pii/S2210537916301676.

[9] D. Carrizo, O. Dieste, and N. Juristo. "Systematizing requirements elicitation technique selection." In: *Information and Software Technology* 56.6 (2014), pp. 644 –669. ISSN: 0950-5849. DOI: https://doi.org/10.1016/j.infsof.2014.01.009. URL: http://www.sciencedirect.com/science/article/pii/S0950584914000202.

[10] R. Chitchyan, C. Becker, S. Betz, L. Duboc, B. Penzenstadler, N. Seyff, and C. C. Venters. "Sustainability Design in Requirements Engineering: State of Practice." In: *2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)*. 2016, pp. 533–542.

[11] L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos. *Non-Functional Requirements in Software Engineering*. Vol. 5. Jan. 2000. DOI: 10.1007/978-1-4615-5269-7.

[12] N. Condori-Fernandez and P. Lago. "Characterizing the contribution of quality requirements to software sustainability." In: *Journal of Systems and Software* 137 (2018), pp. 289 –305. ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2017.12.005. URL: http://www.sciencedirect.com/science/article/pii/S0164121217302984.

[13] *Confidentiality - Requirements Tracker / TYRE - PSNC Wiki*. (Accessed on 08/27/2019). URL: https://confluence.man.poznan.pl/community/display/REQ/Confidentiality.

[14] *CORE Rankings Portal - Computing Research & Education*. URL: http://www.core.edu.au/conference-portal (visited on 2019).

[15] F. Dalpiaz, X. Franch, and J. Horkoff. "iStar 2.0 Language Guide." In: (May 2016).

[16] T. Deelmann and P. Loos. "Trust Economy: Aspects of Reputation and Trust Building for SMEs in eBusiness." In: *Eighth Americas Conference on Information Systems* (Jan. 2002).

[17] S. M. Easterbrook. "Climate Change: A Grand Software Challenge." In: *Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research*. FoSER '10. Santa Fe, New Mexico, USA: ACM, 2010, pp. 99–104. ISBN: 978-1-4503-0427-6. DOI: 10.1145/1882362.1882383. URL: http://doi.acm.org/10.1145/1882362.1882383.

[18] G. Elahi and E. Yu. "Comparing alternatives for analyzing requirements trade-offs – In the absence of numerical data." In: *Information and Software Technology* 54.6 (2012). Special Section: Engineering Complex Software Systems through Multi-Agent Systems and Simulation, pp. 517 –530. ISSN: 0950-5849. DOI: https://doi.org/10.1016/j.infsof.2011.10.007. URL: http://www.sciencedirect.com/science/article/pii/S0950584911002242.

[19] *Google Forms: Free Online Surveys for Personal Use.* (Accessed on 02/05/2020). URL: https://www.google.com/forms/about/.

[20] M. S. Hasan, A. Al Mahmood, M. Jahangir Alam, S. Md. Nahid Hasan, and R. Farin. "An Evaluation of Software Requirement Prioritization Techniques." In: *International Journal of Computer Science and Information Security* (Jan. 2010).

[21] H. Heerkens, L. van der Wegen, and B. van der Heijden. "Designing and assessing a course on prioritization and importance assessment in strategic non-routine requirements engineering processes." In: *Requirements Engineering* 21.4 (Nov. 2016), pp. 505–520. ISSN: 1432-010X. DOI: 10.1007/s00766-015-0230-6. URL: https://doi.org/10.1007/s00766-015-0230-6.

[22] F. Hujainah, R. B. A. Bakar, M. A. Abdulgabber, and K. Z. Zamli. "Software Requirements Prioritisation: A Systematic Literature Review on Significance, Stakeholders, Techniques and Challenges." In: *IEEE Access* 6 (2018), pp. 71497–71523. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2018.2881755.

[23] *IPCC — Intergovernmental Panel on Climate Change.* URL: https://www.ipcc.ch/ (visited on 2019).

[24] *ISO/IEC 25010:2011(en), Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models.*

[25] M. Kassab and N. Kilicay-Ergin. "Applying analytical hierarchy process to system quality requirements prioritization." In: *Innovations in Systems and Software Engineering* 11.4 (Dec. 2015), pp. 303–312. DOI: 10.1007/s11334-015-0260-8. URL: https://doi.org/10.1007/s11334-015-0260-8.

[26] J. Khan, I. Ur Rehman, Y. Hayat Khan, I. Javed Khan, and S. Rashid. "Comparison of Requirement Prioritization Techniques to Find Best Prioritization Technique." In: *International Journal of Modern Education and Computer Science* 7 (Nov. 2015), pp. 53–59. DOI: 10.5815/ijmecs.2015.11.06.

[27] S. U. R. Khan, S. P. Lee, M. Dabbagh, M. Tahir, M. Khan, and M. Arif. "RePizer: a framework for prioritization of software requirements." In: *Frontiers of Information Technology & Electronic Engineering* 17.8 (Aug. 2016), pp. 750–765. ISSN: 2095-9230. DOI: 10.1631/FITEE.1500162. URL: https://doi.org/10.1631/FITEE.1500162.

[28] B. Kitchenham. "Procedures for Performing Systematic Reviews." In: *Keele, UK, Keele Univ.* 33 (Aug. 2004).

[29] N. Kukreja, B. Boehm, S. S. Payyavula, and S. Padmanabhuni. "Selecting an appropriate framework for value-based requirements prioritization." In: *2012 20th IEEE International Requirements Engineering Conference (RE).* 2012, pp. 303–308. DOI: 10.1109/RE.2012.6345819.

[30] X. F. Liu, Y. Sun, C. S. Veera, Y. Kyoya, and K. Noguchi. "Priority assessment of software process requirements from multiple perspectives." In: *Journal of Systems and Software* 79.11 (2006). Software Cybernetics, pp. 1649 –1660. ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2006.03.012. URL: http://www.sciencedirect.com/science/article/pii/S016412120600077X.

[31] Q. Ma. "The Effectiveness of Requirements Prioritization Techniques for a Medium to Large Number of Requirements : A Systematic Literature Review." In: 2010.

[32] *Microsoft Word - column4.doc*. URL: http://www.jot.fm/issues/issue_2004_09/column4/column4.pdf (visited on 2019).

[33] S. Naumann, M. Dick, E. Kern, and T. Johann. "The GREENSOFT Model: A reference model for green and sustainable software and its engineering." In: *Sustainable Computing: Informatics and Systems* 1.4 (2011), pp. 294 –304. ISSN: 2210-5379. DOI: https://doi.org/10.1016/j.suscom.2011.06.004. URL: http://www.sciencedirect.com/science/article/pii/S2210537911000473.

[34] *Nonfunctional Requirement Examples - Requirements Quest*. (Accessed on 08/27/2019). URL: https://requirementsquest.com/nonfunctional-requirement-examples/.

[35] *Our Common Future ('Brundtland report')*. Oxford Paperback Reference. Oxford University Press, USA, May 1987. URL: http://www.bne-portal.de/fileadmin/unesco/de/Downloads/Hintergrundmaterial_international/Brundtlandbericht.File.pdf?linklisted=2812.

[36] S. Oyedeji, A. Seffah, and B. Penzenstadler. "A Catalogue Supporting Software Sustainability Design." In: *Sustainability* 10.7 (2018). ISSN: 2071-1050. DOI: 10.3390/su10072296. URL: https://www.mdpi.com/2071-1050/10/7/2296.

[37] C. Pacheco, I. García, and M. Reyes. "Requirements elicitation techniques: a systematic literature review based on the maturity of the techniques." In: *IET Software* 12.4 (2018), pp. 365–378. ISSN: 1751-8806. DOI: 10.1049/iet-sen.2017.0144.

[38] B. Penzenstadler. "Infusing green: Requirements engineering for green in and through software systems." In: *CEUR Workshop Proceedings* 1216 (Jan. 2014), pp. 44–53.

[39] B. Penzenstadler and H. Femmer. "A generic model for sustainability with process- and product-specific instances." In: *Workshop on Green in/by software engineering*. 2013, pp. 3–8.

[40] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson. "Systematic Mapping Studies in Software Engineering." In: *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*. EASE'08. Italy: BCS Learning & Development Ltd., 2008, pp. 68–77. URL: http://dl.acm.org/citation.cfm?id=2227115.2227123.

[41] M. Petticrew and H. Roberts. *Systematic Reviews in the Social Sciences: A Practical Guide*. Vol. 11. Jan. 2006. DOI: 10.1002/9780470754887.

[42] J. Pimentel and J. Castro. "piStar Tool – A Pluggable Online Tool for Goal Modeling." In: *2018 IEEE 26th International Requirements Engineering Conference (RE)*. 2018, pp. 498–499. DOI: 10.1109/RE.2018.00071.

[43] A. Raturi, B. Penzenstadler, B. Tomlinson, and D. Richardson. "Developing a Sustainability Non-functional Requirements Framework." In: *Proceedings of the 3rd International Workshop on Green and Sustainable Software*. GREENS 2014. Hyderabad, India: ACM, 2014, pp. 1–8. ISBN: 978-1-4503-2844-9. DOI: 10.1145/2593743.2593744. URL: http://doi.acm.org/10.1145/2593743.2593744.

[44] K. Roher and D. Richardson. "A proposed recommender system for eliciting software sustainability requirements." In: *2013 2nd International Workshop on User Evaluations for Software Engineering Researchers (USER)*. May 2013, pp. 16–19. DOI: 10.1109/USER.2013.6603080.

[45] G. Ruhe, A. Eberlein, and D. Pfahl. "Trade-off Analysis for Requirements Selection." In: *International Journal of Software Engineering and Knowledge Engineering* 13 (Aug. 2003), pp. 345–366. DOI: 10.1142/S0218194003001378.

[46] S. Sharma and S K. Pandey. "Revisiting Requirements Elicitation Techniques." In: *International Journal of Computer Applications* 75 (Aug. 2013), pp. 35–39. DOI: 10.5120/13166-0889.

[47] A. Silva, P. R. Pinheiro, A. Albuquerque, and J. Barroso. "Evaluation of an approach to define elicitation guides of non-functional requirements." In: *IET Software* 11.5 (2017), pp. 221–228. ISSN: 1751-8806. DOI: 10.1049/iet-sen.2016.0302.

[48] I. Sommerville and P. Sawyer. *Requirements Engineering: A Good Practice Guide*. 1st. New York, NY, USA: John Wiley & Sons, Inc., 1997. ISBN: 0471974447.

[49] N Subramanian. "Sustainability - Challenges and solutions." In: 81 (Dec. 2007), pp. 39–50.

[50] *Sustainability Reporting*. URL: https://www.globalreporting.org/information/sustainability-reporting/Pages/default.aspx (visited on 2019).

[51] *Sustainability Standards and Protocols | NSF International*. URL: http://www.nsf.org/services/by-type/standards-publications/ncss/sustainability-standards-protocols-stnd (visited on 2019).

[52] *Sustainable Development Goals | UNDP*. (Accessed on 03/25/2020). URL: https://www.undp.org/content/undp/en/home/sustainable-development-goals.html.

[53] R. Thakurta. "A framework for prioritization of quality requirements for inclusion in a software project." In: *Software Quality Journal* 21.4 (Dec. 2013), pp. 573–597. ISSN: 1573-1367. DOI: 10.1007/s11219-012-9188-5. URL: https://doi.org/10.1007/s11219-012-9188-5.

[54] *Top-Down And Bottom-Up Design Approach - EmbHack*. (Accessed on 01/22/2020). URL: https://www.embhack.com/top-down-and-bottom-up-design-approach/.

[55] M. Vestola. "A Comparison of Nine Basic Techniques for Requirements Prioritization." In: 2010.

[56] *Water crises are a top global risk | World Economic Forum*. URL: https://www.weforum.org/agenda/2015/01/why-world-water-crises-are-a-top-global-risk/ (visited on 2019).

[57] *Water Scarcity | Threats | WWF*. URL: https://www.worldwildlife.org/threats/water-scarcity (visited on 2019).

[58] *What is sustainability*. URL: http://www.globalfootprints.org/sustainability/ (visited on 2019).

[59] *What is the Kyoto Protocol? | UNFCCC*. URL: https://unfccc.int/process-and-meetings/the-kyoto-protocol/what-is-the-kyoto-protocol/what-is-the-kyoto-protocol (visited on 2019).

[60] F. van Winden. "Affect and Fairness in Economics." In: *Social Justice Research* 20.1 (2007), pp. 35–52. ISSN: 1573-6725. DOI: 10.1007/s11211-007-0029-9. URL: https://doi.org/10.1007/s11211-007-0029-9.

[61] C. Wohlin. "Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering." In: *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*. EASE '14. New York, NY, USA: ACM, 2014, 38:1–38:10. ISBN: 978-1-4503-2476-2. DOI: 10.1145/2601248.2601268. URL: http://doi.acm.org/10.1145/2601248.2601268.

[62] *World Energy Consumption Statistics | Enerdata*. URL: https://yearbook.enerdata.net (visited on 2019).

[63] A. Yrjönen and J. Merilinna. "Extending the NFR Framework with Measurable Non-Functional Requirements." In: 553 (Jan. 2009).

[64] E. Yu. "Social Modeling and i*." In: Jan. 2009, pp. 99–121. DOI: 10.1007/978-3-642-02463-4_7.

[65] M.-x. Zhu, X.-X. Luo, X.-H. Chen, and D. Wu. "A non-functional requirements tradeoff model in Trustworthy Software." In: *Information Sciences - ISCI* 191 (Jan. 2011). DOI: 10.1016/j.ins.2011.07.046.

APPENDIX A

Sheet template

Data Extraction	
1. Paper Basic Information:	
1.1 Paper id:	PRIO11
1.2 Paper title:	Comparison of Requirement Prioritization Tecnhiques to Find Best Prioritization Tecnhique
1.3 Paper conference/journal/workshop:	Modern Education and Computer Science, Volume 11, p. 53-59
1.4 Paper year:	2015
1.5 Paper number of citations:	20
1.6 Digital library:	Research Gate
1.7 Core Rank:	A
2. Assessment	
2.1 What is the approach regarding assessment proposed by the	None presented.
3. Prioritization	
3.1 What is the approach regarding assessment proposed by the paper?	This paper provides a detailed description of various RP techniques. It compares them based on different criteria, such as, "easiness", accuracy, time consumption, scalability. Finally it concludes which ones are the best, AHP is considered the best one, while Planning Game comes in second. The validation of the paper was done solely by doing a systematic literature review of the different requirement prioritization techniques.
4. Trade-off analysis	
4.1 What is the approach regarding trade-off analysis proposed by	None presented.
5. Sustainability Requirements	
5.1 What is software sustainability?	None presented.
5.2 What are the requirements that relate or are important to	None presented.

Figure A.1: Example of a sheet template for one study.

APPENDIX B

Social Dimension Catalogue

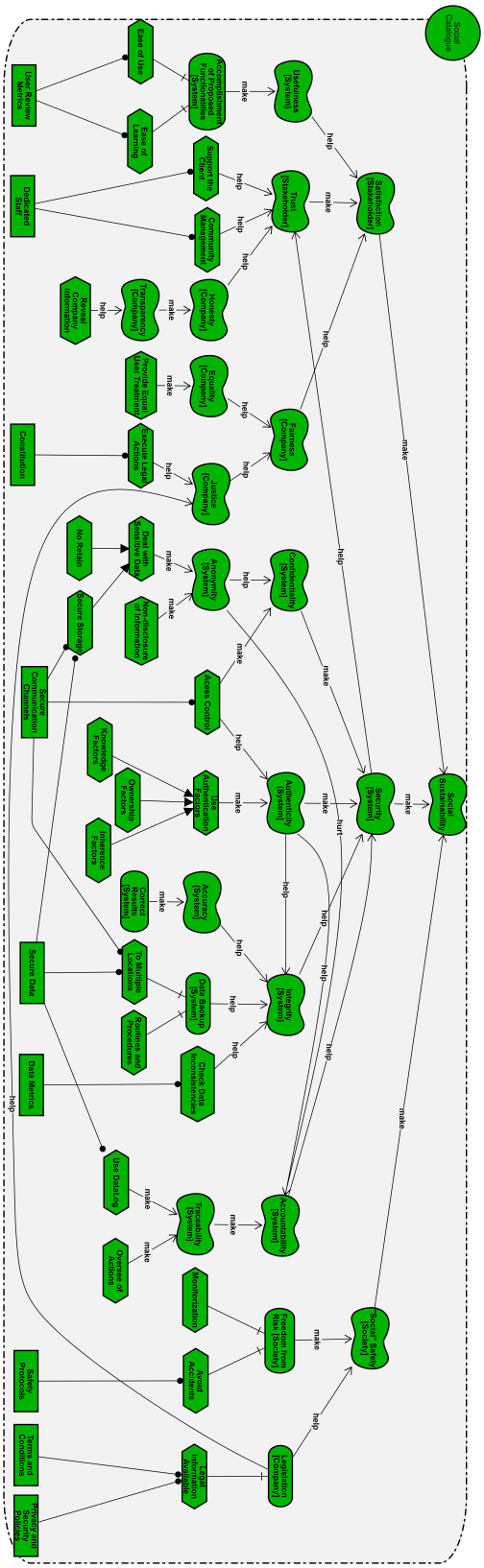


Figure B.1: Social dimension catalogue.

APPENDIX C

Economic Dimension Catalogue

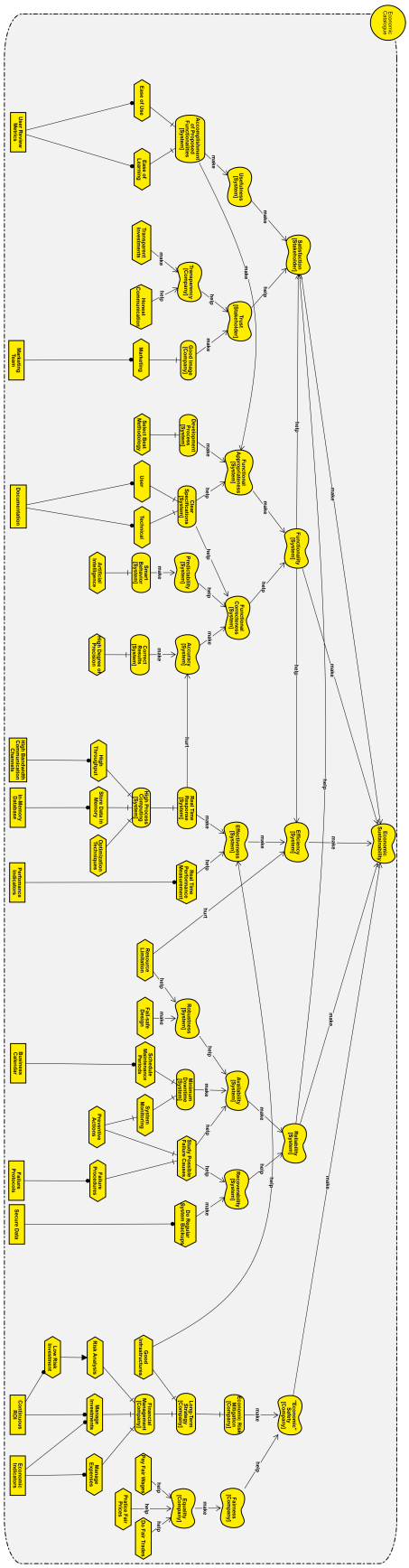


Figure C.1: Economic dimension catalogue.

APPENDIX D

Environmental Dimension Catalogue

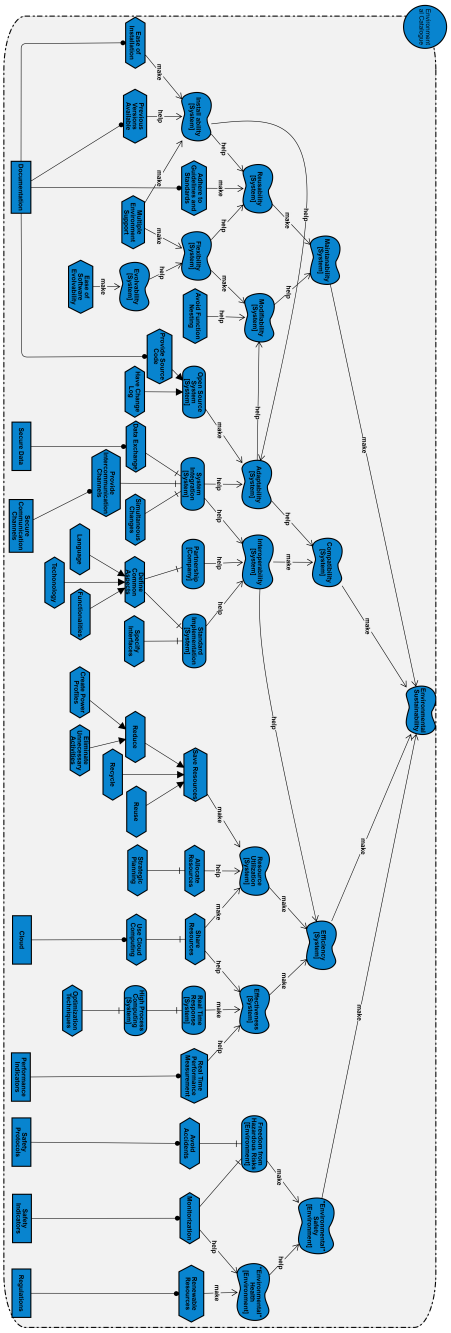


Figure D.1: Environmental dimension catalogue.

APPENDIX E

Technical Dimension Catalogue

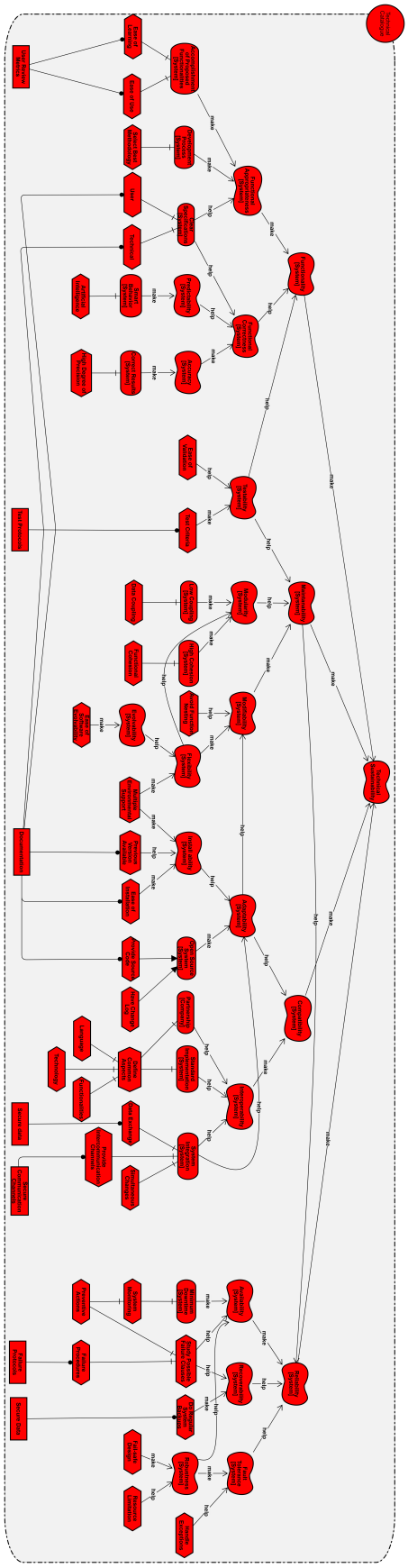


Figure E.1: Technical dimension catalogue.

APPENDIX F

Sustainability Catalogue

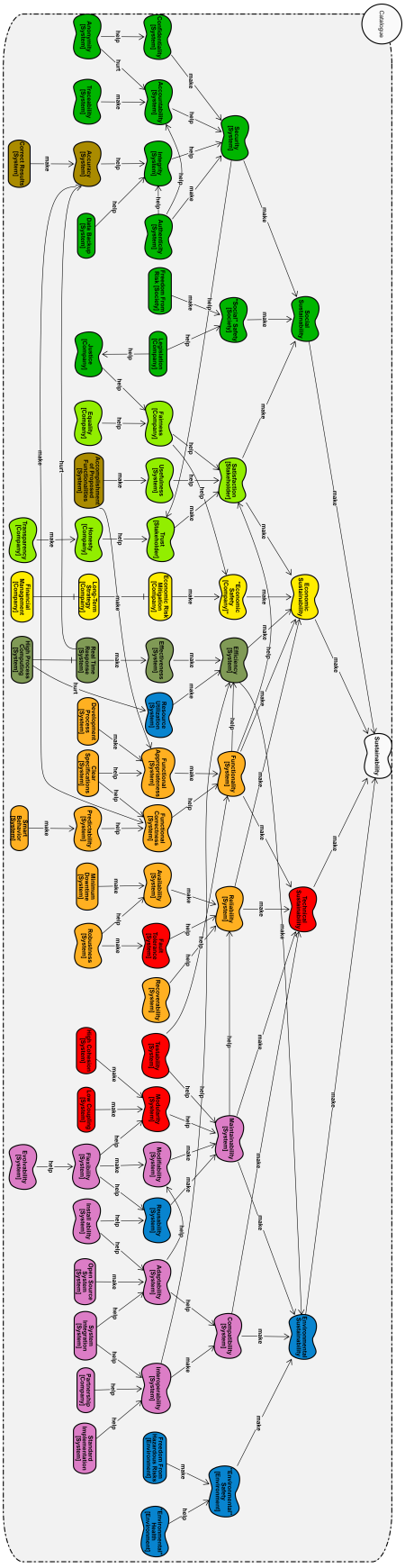


Figure F.1: Sustainability catalogue.

ANNEX I

Elicitation Techniques

Table I.1: Elicitation techniques with respective characteristics and recommended conditions, from [37], [46], [9].

Technique	Characteristics	Recommended Conditions
Interviews	Adequate for eliciting relevant aspects from the stakeholders' needs and understanding problems in existing systems	Expert stakeholders in the domain and available time. They are also accessible, expressive, and an expert interviewer supports them
	Able to recognize errors or clarify misunderstandings in requirements, through recapitulation and feedback	
	Effective in quickly obtaining complete information (structured knowledge, detailed requirements), clarify ambiguities, and process improvement	The requirements engineer should have an open mind and patience to listen to stakeholders
	Provides opportunity to explore topics in depth	
Surveys and Questionnaires	Collection of information in an effective and rapid way	Diverse stakeholders and involved
	Inexpensive way to reach a large number of people	Stakeholders geographically dispersed
	Allow respondents to take time to consider their responses carefully without interference's from others, such as an interviewer	Questionnaires must be clear, defined and concise along with the domain knowledge
		Development of not complex and long systems
Scenarios	Competent in analyzing the interaction between the software and user,and reducing discrepancies and ambiguities in the task's data flow by combining diverse communications channels	Limited schedule and low budget
	Effective for the requirements refinement	The set of actions must be clear and transmit real needs of the system
Workshop and Brainstorming	Useful because it consider all the needs of multiple stakeholders	Multiple stakeholders that need collaboration to obtain a complete picture of the requirement
	Capable of providing a complete set of diverse types of requirements and eliciting requirements from big and complex systems	Expressive stakeholders
	Effective for early requirements elicitation through direct communication with Stakeholders	The stakeholders do not know the requirements, or they provide these incorrectly
	Useful for generating a variety of ideas and system expectations (i.e. requirements)	Development of large and complex systems
Focus Group	Effective for conflict resolution within a group and its subsequent monitoring	Diverse and expressive stakeholders are involved
		Expert interviewer presented
	Adept for promoting a collaborative environment	A relaxing and collaborative ambient is needed to maximize the efficiency
Prototyping	Efficient in obtaining a graphical and functional representation of the requirements	Stakeholders are unable to express their needs
	Effective in understanding the interactions with the system and capturing enough details of the graphical user interface	The development of a new product or the improvement of an existing one
		Complex systems development

ANNEX II

Prioritization Techniques

Table II.1: Prioritization techniques with respective strengths and weaknesses and recommended size of requirements set accompanied of a brief description, from [22], [31], [26], [20].

Techniques	Brief Description	Strengths	Weaknesses	Size of Set
Nominal Scale (Numerical Assignment, MoSCoW)	Performs grouping of requirements into categories	Relatively straightforward Groups the requirements into smaller groups which leads to better understanding	Subjectivity of the respective categories to stakeholders Not well suited for small number of requirements Does not permit evaluation of relative priority	Medium to Large
Bubble Sort	Ranks the requirements according to their priority on an ordinal scale	It can be used as a sorting algorithm for other techniques	It provides a simple ranking and does not assign priority values to requirements	Small to Medium
Binary Search Tree	It is a tree in which each node has at most two children. Requirements are compared to each other until all have been inserted into it.	Implementation is simple It is usable for large number of requirements	It provides a simple ranking and does not assign priority values to requirements	Large
Hundred Point Method	Performs ranking of requirements based on a fictitious distribution of 100 points on requirements	Simplicity of approach Is fast	Does not permit evaluation of relative priority difference among the requirements Not well suited for large number of requirements	Small
AHP	It is a structured technique for organizing and analyzing complex decisions based on pairwise analysis of the alternatives	Ability to resolve conflicting objectives Produces reliable results on a ratio scale	Not well scalable for large number of requirements Can be a bit complex	Small to Medium
Minimal Spanning Tree	It is a subset of the edges of a connected, edge-weighted non-directed graph that connects all the vertices together, without any cycles and with the minimum possible total edge weight.	Eliminates redundancy Suitable for large sets of requirements if reliability and fault tolerance are not important	It is sensitive to judgment error It is unreliable and has poor fault tolerance	Medium to Large
Planning Game	It combines numerical assignment and ranking technique to prioritize requirements	Suitable for innovative development models because of its flexible behaviour Straightforward	It does not identify the priority value of each requirement	Medium to Large