Leibniz
Universität
Hannover

Requirements Engineering for Explainable Systems

Von der Fakultät für Elektrotechnik und Informatik

der Gottfried Wilhelm Leibniz Universität Hannover

zur Erlangung des akademischen Grades

Doktorin der Naturwissenschaften

(abgekürzt: Dr. rer. nat.)

genehmigte Dissertation von Frau

M. Sc. Larissa Chazette

2023

Betreuer: Prof. Dr. Kurt Schneider

1. Referent: Prof. Dr. Jörg Dörr

2. Referent: Prof. Dr. Markus Dürmuth

Vorsitzende der Prüfungskomission: Prof. Dr. Michael Rohs

Tag der Promotion: 15. Dezember 2022

Contents

List of Figures

List of Tables

Acronyms

AI	Artificial Intelligence
ER	Explainability Requirement
GDPR	General Data Protection Regulation
GPS	Global Positioning System
HCI	Human-Computer Interaction
IEEE	Institute of Electrical and Electronics Engineers
KBS	Knowledge-Based System
ML	Machine Learning
NFR	Non-Functional Requirement
OBNS	On-Board Navigation System
RE	Requirements Engineering
RFID	Radio-Frequency Identification
RQ	Research Question
SE	Software Engineering
SIG	Softgoal Interdependency Graph

SLR	Systematic Literature Review
UCD	User-Centered Design
UML	Unified Modeling Language
XAI	Explainable Artificial Intelligence

To all the selves I have been during the creation of this piece…

…and to my husband and my mother, who supported and loved all of them.

Acknowledgments

Behind every written production is an adventure. My adventure (the biggest one of my life up until now) started an ocean away from where I write: when I decided to leave Brazil to pursue a PhD in computer science in Germany. I set off alone on this adventure, leaving all my treasures behind: my mother, my dog, my friends, and my hometown. All I carried with me was a suitcase and a heart full of courage and dreams. The path from this day until here was full of subtleties and far from easy, and I seriously considered giving up several times. But I didn't. And it would be foolish of me to suggest that I succeeded or found strength on my own. For even though I came alone, and even though I felt utterly alone at times, I now know that I was never alone.

So now is the moment to be thankful for it and to thank all that helped me to be where I am now. First and foremost, God, my unending source of wisdom and hope. My mother, the strongest woman I know, for her unending love, support, daily prayers, and lifelong efforts to get me where I am now. To the greatest gift Germany and academic life have given me: my dear husband, for his partnership and support on both good and bad days. To my research group colleagues, particularly Nils Prenner and Maike Ahrens, for being the best office mates I could have asked for. To all the co-authors of the works that make up this work, but especially to Timo Speith for showing me the value of interdisciplinary research and for his amazing sense of humor. And, of course, to my supervisor or, better saying, "Doktorvater", Prof. Kurt Schneider, for his excellence, loyalty, and kindness: A seed needs a good gardener to flourish.

Abstract

Information systems are ubiquitous in modern life, and are powered by evermore complex algorithms that are often difficult to understand. Moreover, since systems are part of almost every aspect of human life, quality in interaction and communication between humans and machines has become increasingly important. Hence the importance of explainability as an essential element of human-machine communication; it has also become an important quality requirement for modern information systems.

However, dealing with quality requirements has never been a trivial task. To develop quality systems, software professionals have to understand how to transform abstract quality goals into real-world information system solutions. Requirements engineering provides a structured approach that aids software professionals in better comprehending, evaluating, and operationalizing quality requirements. Explainability has recently regained prominence and been acknowledged and established as a quality requirement; however, there is currently no requirements engineering recommendations specifically focused on explainable systems.

To fill this gap, this thesis investigated explainability as a quality requirement and how it relates to the information systems context, with an emphasis on requirements engineering. To this end, this thesis proposes two theories that delineate the role of explainability and establish guidelines for the requirements engineering process of explainable systems. These theories are modeled and shaped through five artifacts. These theories and artifacts should help software professionals 1) to communicate and achieve a shared understanding of the concept of explainability; 2) to comprehend how explainability affects system quality and what role it plays; 3) in translating abstract quality goals into design and evaluation strategies; and 4) to shape the software development process for the development of explainable systems.

The theories and artifacts were built and evaluated through literature studies, workshops, interviews, and a case study. The findings show that the knowledge made available helps practitioners understand the idea of explainability better, facilitating the creation of explainable systems. These results suggest that the proposed theories and artifacts are plausible, practical, and serve as a strong starting point for further extensions and improvements in the search for high-quality explainable systems.

Zusammenfassung

Informationssysteme sind im modernen Leben allgegenwärtig und werden von immer komplexeren und oft schwer verständlichen Algorithmen gesteuert. Da Systeme in fast allen Bereichen des menschlichen Lebens zum Einsatz kommen, wird die Qualität der Interaktion und Kommunikation zwischen Mensch und Maschine immer wichtiger. Daher ist die Bedeutung der Erklärbarkeit als wesentliches Element der Mensch-Maschine-Kommunikation auch zu einer wichtigen Qualitätsanforderung für moderne Informationssysteme geworden.

Der Umgang mit Qualitätsanforderungen war jedoch noch nie eine triviale Aufgabe. Um Qualitätssysteme zu entwickeln, müssen Software-Experten wissen, wie sie abstrakte Qualitätsziele in reale Informationssystemlösungen umsetzen können. Das Requirements Engineering bietet einen strukturierten Ansatz, der Software-Experten dabei hilft, Qualitätsanforderungen besser zu verstehen, zu bewerten und zu operationalisieren. Erklärbarkeit ist in letzter Zeit wieder in den Vordergrund gerückt und als Qualitätsanforderung anerkannt sowie etabliert worden. Allerdings gibt es derzeit keine Requirements Engineering-Empfehlungen, die sich speziell auf erklärbare Systeme konzentrieren.

Um diese Lücke zu schließen, wurde in dieser Arbeit die Erklärbarkeit als Qualitätsanforderung und ihr Bezug zum Kontext der Informationssysteme untersucht, wobei der Schwerpunkt auf dem Requirements Engineering liegt. Zu diesem Zweck werden in dieser Arbeit zwei Theorien aufgestellt, die die Rolle der Erklärbarkeit beschreiben und Richtlinien für den Requirements Engineering-Prozess von erklärbaren Systemen aufstellen. Diese Theorien werden durch fünf Artefakte modelliert und gestaltet. Diese Theorien und Artefakte sollen Software-Experten dabei helfen, 1) ein gemeinsames Verständnis des Konzepts der Erklärbarkeit zu vermitteln und zu erreichen; 2) zu verstehen, wie Erklärbarkeit die Systemqualität beeinflusst und welche Rolle sie spielt; 3) abstrakte Qualitätsziele in Design- und Evaluierungsstrategien zu übersetzen; und 4) den Software-Entwicklungsprozess für die Entwicklung erklärbarer Systeme zu gestalten.

1
Introduction

We are living in the algorithmic age, where algorithms and systems are ubiquitous in our society. When describing the concept of ubiquity in computer science, Mark Weiser famously said that “the most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it” [9]. For computer science, this ubiquity represents the incorporation of computers into everyday products and activities, culminating in a seamless integration between virtual and physical worlds. When Weiser wrote about it in 1991 for the *Scientific American*, he described a hypothetical scenario: a day in the life of someone in the future who lives in a world surrounded by computing, from coffee machines to smart offices.

Today, more than 30 years later, we are living the dawn of this anticipated future, no longer hypothetical. Algorithms are already shaping our experiences, transforming every facet of life. In fact, information systems are deeply integrated in many aspects of our society, having a range of applications such as advertising, communication, healthcare, sales, navigation, financial, entertainment, and the list can go on indefinitely, as it is almost impossible to imagine a domain that is not yet supported by information systems [10]. As these systems become more widespread, complex, and crucial to our society and lives, it becomes more urgent and critical to focus on their quality.

Software quality is frequently defined as the field of research and practice that specifies the desired attributes of software products. Requirements engineering is tied to software quality since the one of the key roles of requirements engineering is to transform abstract quality attributes (that are based in real-world conceptualizations and ideas) in concrete software and system capabilities. The desired quality characteristics for the systems of today go way beyond producing software that is free of bugs, vulnerabilities, and other errors [11]. As time passes, new quality aspects emerge and become more important in response to real-world needs. Explainability is an example of a quality aspect that has gained prominence in recent years as a result of the increasing ubiquity of information systems, emphasizing the importance of focusing on the quality of human-machine communication. Incorporating explainability in a system can mitigate software opacity [12], thereby helping users understand why the system produced a particular result and supporting them in making better decisions. Explainability also has an impact on the relationship of trust in and reliance on a system [13], it may avoid feelings of frustration [14], and thus leads to greater user acceptance [15].

The challenge of dealing with quality aspects (or "non-functional requirements") is also well-known and primarily the result of a communication issue caused by the absence of definitions, the difficulty of operationalizing these abstract concepts into concrete functionality, and the difficulty of determining how operationalizations affect quality [16, 17]. The development of explainable systems requires a thorough understanding of explainability's definition, taxonomy, interactions with other system quality factors, and of concrete aspects tied to explainability. In order to support the creation of high-quality explainable systems, software and requirements engineers need methodologies that help them analyze, define, and evaluate explainability requirements.

However, since explainability has only newly been established as a quality aspect, there is still little guidance for practitioners in this matter. Therefore, the overall goal of this research is to bridge this gap and support requirements engineering for explainable systems, both with theory and with hands-on artifacts that can support practice. This goal is subdivided in two other goals: **Goal 1** focuses on the theoretical background that is needed to comprehend explainability as a quality requirement. **Goal 2** focuses on investigating ways of supporting the requirements engineering process in the development of explainable systems.

1.1 Scientific Approach

The general approach taken by this thesis is rooted in Hevner's model [4] and in the framework for software engineering (SE) theories [3]. Hevner's model emphasizes that information systems' research should help to create theories and artifacts that are both grounded in 1) real-word needs and challenges from practitioners to be *relevant* and in 2) considering existing scientific and domain knowledge to be *rigorous*. This thesis offers two theories that are modeled through five supporting artifacts, including:

- A definition for explainable systems.
- A catalogue and a conceptual model of the impact of explainability on system quality and its relationship with other quality requirements.
- A process reference model for the requirements engineering process of explainable systems.
- A quality framework to support the analysis, specification, and evaluation of explainability requirements.

Different scientific methods were used to create the theories and artifacts. The methods were chosen pragmatically based on the respective study goals and research questions to achieve scientific rigor. More detail on the research approach will be given in chapter 5.

1.2 Thesis Structure

The thesis provides a chronological description of the findings. My first step in this research was to conduct a user survey to understand users' perceptions of software-embedded explanations and to evaluate the relationship between explainability and software transparency. The results of the user survey helped me identifying four research needs. These research needs shaped the rest of the research, motivating the research goals and questions that resulted in the proposed theories and artifacts. The first theory is an analytical-explanatory theory that discusses the role and impact of explainability on the quality landscape. This theory is linked to three artifacts that serve as models for this theory and should make it easier to discuss and apply it in practice, making the concept of explainability more understandable for practice. The second theory is a design and action theory that is linked to two artifacts that are meant to provide guidelines for practitioners as they develop explainable systems, with a primary emphasis on requirements engineering. Figure 1.1 depicts the structure of this thesis.

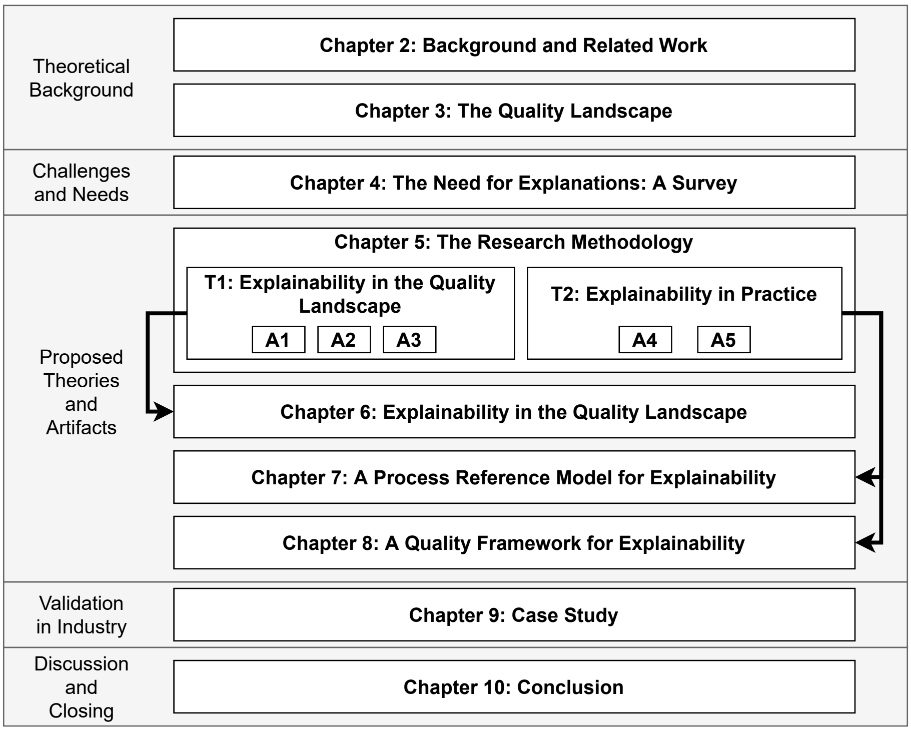

Theoretical Background
Chapter 2: Background and Related Work
Chapter 3: The Quality Landscape
Challenges and Needs
Chapter 4: The Need for Explanations: A Survey
Proposed Theories and Artifacts
Chapter 5: The Research Methodology
T1: Explainability in the Quality Landscape
A1
A2
A3
T2: Explainability in Practice
A4
A5
Chapter 6: Explainability in the Quality Landscape
Chapter 7: A Process Reference Model for Explainability
Chapter 8: A Quality Framework for Explainability
Validation in Industry
Chapter 9: Case Study
Discussion and Closing
Chapter 10: Conclusion

Figure 1.1: Structure of the thesis

Chapters 2 and 3 will cover the necessary theoretical background to comprehend this thesis. Based on the results of a user survey, I looked at the needs and challenges associated with explainability in chapter 4. The research methodology for this thesis is presented in chapter 5, where I link the two proposed theories and associated artifacts to the identified research needs. The first theory (T1) is discussed in chapter 6, while chapter 7 and chapter 8 discuss the second theory (T2). In chapter 9, I discuss an industry case study that helped validate one of the theories. Finally, in chapter 10, I conclude this thesis with an overall discussion.

2
Background and Related Work

Quality is connected with different aspects of any kind of product. Quality, in general, is the degree to which a set of inherent characteristics of an item fulfills its requirements [18]. A sofa, for example, can be considered high quality if it fulfills a customer's requirements by, for example, being comfortable and having a beautiful design. Given that a system can be thought of as a product as well, focusing on the quality of systems may seem fairly obvious. In systems and software engineering, quality is the degree to which a system satisfies stated and implied needs of its stakeholders [18]. Nevertheless, "in normal life, you only think about software quality when there is something wrong with it"[*] [19]. In fact, modern life is full of small and large examples where the lack of quality in a system has caused consequences for users and companies. The list of consequences caused by poor software quality goes on and on, from problems with thermostats to prison breaks, and safety issues [19, 20, 21].

Establishing and ensuring that software development processes and practices produce a system with the appropriate quality to meet its requirements is the foundation of software quality. Software quality assurance is a process that connects various aspects and activities of SE. Some may argue that testing is all that is required to ensure software quality, and that the only reason issues arise is because testing is insufficient or inadequate. Testing, though an important component of the intricate landscape of software quality, is only one step in a longer

[*] "An Softwarequalität denkt man im normalen Leben eigentlich nur, wenn etwas nicht stimmt.", freely translated

process that starts with defining and identifying a system's quality requirements. The landscape of software quality includes a variety of dimensions, from those that relate to the quality of the code to those that relate to how users view the system (the "dimensions" concept will be better discussed in chapter 3). For instance, the ISO 25010 standard [22] specifies eight quality aspects for systems: functional suitability, reliability, operability, performance efficiency, security, compatibility, maintainability, and transferability. However, there are numerous other quality aspects that influence the quality landscape. Explainability, for example, is an essential quality aspect that will be discussed and investigated in this thesis.

This chapter will cover the terminology and concepts necessary for understanding this thesis as I discuss the fundamentals of requirements engineering and explainability. In addition, I will discuss some works in the literature that either explore the concept and meaning of explainability or propose artifacts that aid in the development of (explainable) systems. I will start by covering the topic of requirements engineering. Next, I will dive into the quality aspect that is the focus of this thesis: explainability (and related concepts). Since this thesis proposes five artifacts, I will also discuss works in the literature that propose similar artifacts that are either directly related to explainability or, if not, artifacts that are related to other quality aspects.

2.1 Requirements Engineering

Requirements Engineering (Requirements Engineering (RE)) is a branch of systems and software engineering, whose ultimate goal is *to deliver some system behavior to its stakeholders*, according to their goals [23]. Even though RE tends to be traditionally associated with the waterfall process, RE can also occur in agile projects, even if it is not officially called "requirements engineering".

To understand RE, some definitions are of the utmost importance. The first one is the definition of systems. I consider those systems *information systems*[†]:

Definition 2.1.1 (*Information Systems*)

Information systems are combinations of hardware, software, and telecommunications networks that people build and use to collect, create, and distribute useful data, typically in organizational settings [24]. A software system consists of several separate computer programs and associated configuration files, documentation, etc., that operate together and compose information systems [25].

[†] For simplicity, I will refer often to them simply as "systems".

Another important definition is the one of *stakeholders*:

Definition 2.1.2 (*Stakeholder [18]*)

A person or organization who influences a system's requirements or who is impacted by that system. End users, customers, operators, and managers are typical examples of stakeholders.

Software engineers need to be able to apply a variety of abilities in order to translate stakeholders' goals into system functionality.

Definition 2.1.3 (*Software engineer [26]*)

A software engineer is a person involved in the specification, design, construction, deployment, evolution, and maintenance of software systems. Requirements engineers, architects, developers, coders, and testers are examples of common roles for software engineers.

More specifically, according to the International Requirements Engineering Board (IREB) [27], RE is a systematic and disciplined approach to the specification and management of requirements with the following goals:

1. knowing the relevant requirements, achieving a consensus among the stakeholders about these requirements, documenting them according to given standards, and managing them systematically;

2. understanding and documenting the stakeholders' desires and needs;

3. specifying and managing requirements to minimize the risk of delivering a system that does not meet the stakeholders' desires and needs.

Börger et al. [1] proposed a reference model for RE which divides the RE process into two major areas: requirements analysis and requirements management. Each of these areas includes activities that are carried out in order to achieve a specific goal during the process. This thesis is primarily concerned with the requirements analysis process, and activities related to requirements management, such as tracing and change management, are not particularly covered. Below, I describe both areas with a focus on requirements analysis activities, primarily using the RE terminology glossary [18].

2.1.1 Requirements Analysis

The process of analyzing the elicited requirements in order to understand and document them is known as requirements analysis. It is also frequently used as a synonym for requirements engineering. According to Rupp et al. [28], success in a software project is defined as meeting the agreed-upon costs and time goals while delivering a finished product or specific project

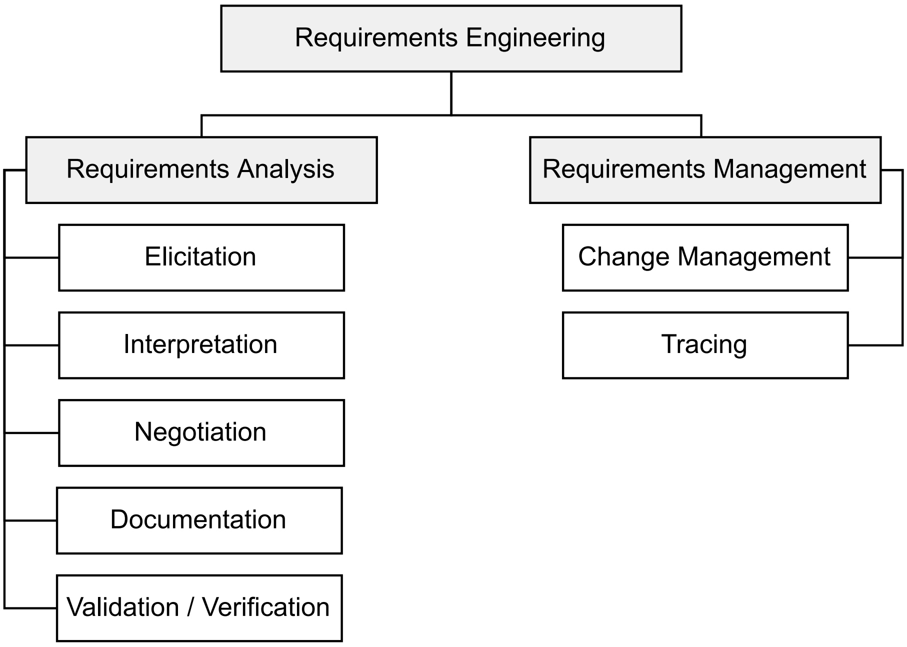


Figure 2.1: Requirements Engineering reference model, adapted from Börger et al. [1]

outcome that meets user expectations in terms of functionality and quality. The majority of errors and risks during system development are closely related to requirements. They occur primarily during the requirements analysis phase and result in significant financial outlay [28]. Therefore, emphasis should be placed on requirements analysis, a stage of system development that often determines whether a project succeeds or fails.

The activities that can be performed during requirements analysis are: elicitation, interpretation, negotiation, documentation, and validation/verification.

Elicitation

Elicitation is the process of seeking, capturing and consolidating requirements from available sources such as stakeholders, documents, existing systems, and observations. Requirements engineers need to collaborate closely with stakeholders to obtain and understand relevant information. During this activity, functionality needs and quality expectations are also defined. Preliminary requirements (known as raw or high-level requirements) can be generated.

Interpretation

Interpretation is the process of analyzing and categorizing raw (or high-level) requirements into categories (e.g., business goals, functional and non-functional requirements). Potential gaps, ambiguities, conflicts, and dependencies are identified in advance for negotiation.

Negotiation

Negotiation is the process by which stakeholders work together to resolve identified gaps, ambiguities, and conflicts. All requirements are prioritized during this activity, and compromises are reached among all stakeholders.

Documentation

Documentation is the process of systematically and persistently defining requirements. This activity results in the creation of a written or graphical specification or requirements document.

Validation/Verification

Validation is the process of confirming that the documented requirements match the stakeholders' needs; in other words: whether the right requirements have been specified. Verification is the process of confirming that the requirements have been documented properly and satisfy the quality criteria for requirements; in other words, whether the requirements have been specified right.

2.1.2 Requirements Management

Requirements management is the process of managing existing requirements-related work products, including the storing, changing and tracing of requirements. It includes all measures necessary to document, change, and track requirements and requirements-related artifacts.

2.1.3 Requirements

A requirement is a single documented physical or functional need that a specific design, product, or process aims to satisfy [18]. According to the Institute of Electrical and Electronics Engineers (IEEE) [29], a requirement is:

Definition 2.1.4 (*Requirement*)

1. A property or capability required by a user (person or system) to solve a problem or achieve a goal. 2. A property or capability that a system or subsystem must meet or possess in order to comply with a contract, standard, specification, or other formally specified document. 3. A documented or conveyed representation of a property or capability as defined in (1) or (2).

There are also various types of requirements and different levels of abstraction for requirements. In the traditional engineering approach, sets of requirements are used as inputs during the design phase in product development. Because tests must be traceable back to specific requirements, requirements are an important input into the verification process. When agile or iterative software development techniques are used, system requirements are developed incrementally alongside design and implementation. Requirements are typically classified into types that are generated at various stages of the development process. Below, I list some of the most common types (as defined in [30]):

Business requirements High-level statements of the goals, objectives, or needs of an organization. They usually describe opportunities that an organization wants to accomplish or problems that they want to solve. Often stated in a business case.

User (stakeholder) requirements Mid-level statements of the needs of a particular stakeholder or group of stakeholders. They usually describe how someone wants to interact with the intended solution. Often acting as a mid-point between the high-level business requirements and more detailed solution requirements.

Architectural requirements These requirements explain what has to be done by identifying the necessary integration of systems structure and systems behavior, i.e., systems architecture of a system. In SE, they are called architecturally significant requirements, which is defined as those requirements that have a measurable impact on a system's architecture.

Quality (non-functional) requirements A non-functional requirement is an attribute of or a constraint on a system [31]. They are usually detailed statements of the conditions under which the solution must remain effective, qualities that the solution must have, or constraints within which it must operate. Examples include: reliability, testability, maintainability, availability.

Functional (solution) requirements Usually detailed statements of capabilities, behavior, and information that the solution will need. Examples include formatting text, calculating a number, modulating a signal. They are also sometimes known as capabilities.

♦

Quality requirements, also known as non-functional requirements (Non-Functional Requirements (NFRs)), have emerged as a crucial focus of study within the discipline of RE. Numerous approaches to NFRs have been developed, and numerous empirical studies have been conducted to assess different facets of these approaches [32]. The terms *NFR*, *quality* *aspects*, and *quality goals* are used throughout this thesis. The term *quality aspects* refers both to *NFRs* and to *aspects* that relate to or compose NFRs. For ease of understanding, I consider quality goals as the quality aspects that are agreed upon for system quality within a project, which can be stated and refined as NFRs.

2.1.4 Shared Understanding

RE is not only a process of discovering and specifying requirements, it is also a process of facilitating effective communication of these requirements among different stakeholders [23]. In heterogeneous groups, shared understanding is both a key factor and a challenge since group members might be using the same words for different concepts or different words for the same concepts without noticing [33].

If individuals have equivalent mental models and can agree on how to understand something, then they have a shared understanding of it [34]. Shared understanding between stakeholders and software engineers is a crucial prerequisite for successful development and deployment of any system, being critical for efficient communication and for reducing the risk of stakeholder dissatisfaction and rework [26].

Shared understanding can be 1) explicit, when the explicit documentation (e.g., requirements, manuals, conceptual architecture) is understood in the same way by all persons; or 2) implicit, when there is a common agreement regarding non-specified knowledge, such as assumptions, opinions, needs, goals, and values. Achieving shared understanding by explicit documentation should be done as far as needed and relying on implicit shared understanding should be done as far as possible [26].

According to Glinz and Fricker [26] some practices tied to artifacts can help to create and improve shared understanding. Two of these practices were chosen because they serve as the foundation for the five artifacts proposed in this thesis:

- **Models** (of domains, problems, or solutions) help infer and properly interpret non-modeled or only coarsely modeled concepts and increase the probability of interpreting them correctly, thus contributing to the creation of proper implicit shared understanding.

- **Glossaries and ontologies** provide explicit definitions of terminology for the system to be built and its domain. As this constitutes again a conversion of implicit shared understanding into explicit shared understanding, the same arguments as given for models apply: explicitly shared terminology reduces the probability of misunderstandings when concepts using this terminology are not specified or only loosely specified.

Three artifacts can be considered models: the conceptual model, the knowledge catalogue, and the process reference model. The definition is classified as a glossary because it relates to terminology, and the framework can be classified both as a model and an ontology. The proposed artifacts help to achieve a shared understanding by transforming implicit shared understanding into explicit shared understanding.

2.2 Explainability and Related Concepts

Incorporating explainability in a system can mitigate software opacity, thereby helping users understand why the system produced a particular result and supporting them in making better decisions. Explainability is going through a new wave of hype due to the recent surge in popularity of machine learning (Machine Learning (ML)) models and the increasing investment in artificial intelligence (Artificial Intelligence (AI)). More recently, terms like **interpretable machine learning** or **explainable artificial intelligence** (Explainable Artificial Intelligence (XAI)) have emerged and proposals aimed at improving the intelligibility of ML algorithms have become a trending topic [35], [36], [14].

However, explainability is not a novel concept and the idea of embedding explanations in systems is not new. In fact, explainability has been already intensely investigated in the domain of knowledge-based systems (Knowledge-Based System (KBS)) [37], and by the human-computer interaction (Human-Computer Interaction (HCI)) community [38]. According to Xu [39], the first work on XAI was found in the literature forty years ago ([40]). Since the beginning of AI research, scientists have argued that intelligent systems should explain the AI results, particularly when it comes to decisions. The use of explanations in this area typically focuses on understanding the mechanics of the learned models during decision making [41], [42], on visualizing the learned model [43], [44], [45] or, in the case of KBS, on supporting users to gain knowledge of a domain [46], [47].

The common terminological confusion surrounding explainability was one of the motivations for this work. As already discussed in subsection 2.1.4, NFRs are still challenging for practice and there is often ambiguity and confusion with respect to their meaning. This ambiguity is also present in the case of explainability, where there are different terms to define the same idea or different definitions and ideas for the term *explainability*. The terms *explainability*, *interpretability*, *understandability*, and *transparency* are often used interchangeably.

According to Lipton [48], *transparency* can be informally defined as the opposite of opacity or blackboxness. *Blackboxness* refers to the notion that some of these ML models can be as mysterious as black boxes to us [49], since their inner workings are opaque and difficult to interpret and explain. Therefore, transparency means “seeing through”, to understand the inner mechanisms by which an algorithm works or what was learned by a model (in the case of ML-based applications). But why is transparency important? The ubiquitous influence of such “black-box systems” has induced discussions about the transparency and ethics of modern systems [50]. Cysneiros et al. [51] discussed transparency as a modern requirement, stating that it is considered to be a key requirement for self-driving cars. They presented transparency as a requirement for developing more robust systems and increasing the rate of adoption of new technologies.

Interpretability is defined in different ways. It can be defined as the level to which the user understands and can make use of the explanations given by the system and the information provided [52]. It is also defined as the ability to explain or to present information in understandable terms to a human [53]. The terms *interpretability* and *explainability* are often used to refer to the same thing. This second definition of interpretability overlaps with the concept of explainability, which can be defined as the level to which a system can provide explanations for the cause of its decisions or outputs [52]. I consider the earlier definition of *interpretability* to be the correct one because it is more closely related to the subjective aspects of how users interpret the information that is being presented. Explanations, by conveying information, can influence the system’s interpretability and understandability, which are rather subjective factors.

Definition 2.2.1 (*Explainability*)

Explainability is the ability or act of disclosing information that is necessary for an addressee to understand a particular aspect of a system in a given context, which can be accomplished by providing explanations.

Despite this naive distinction, there is still a need to specify more precisely what explainability is, and what makes a system explainable, especially since it is an NFR that must be understood and specified in terms of a system’s functions. Therefore, a definition of explainable systems will be provided in chapter 6.

2.2.1 What are Explanations?

Explanations, in general, must serve a purpose. Logic or causal constraints have typically been imposed on what qualifies as an explanation by theories of explanation from the philosophy of science. Philosophically speaking, a common view is that an explanation is an answer to a predefined question of why something is or happens, comparing two or more possibilities [54, 55].

Explanations are also defined as arguments showing how what is being explained (the *explanandum*) is a deduction from natural laws and empirical conditions [56]. Explanations

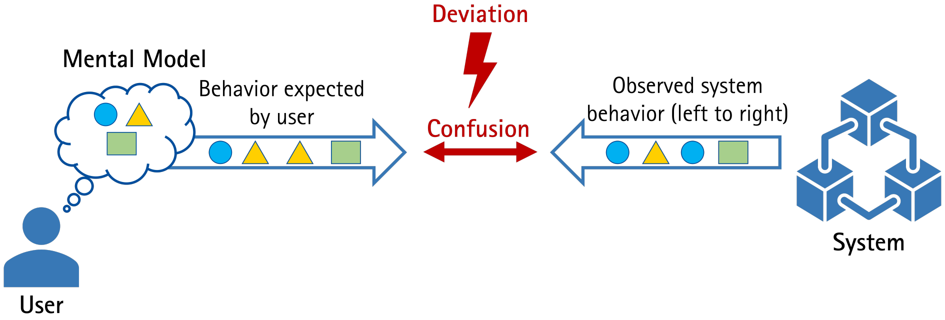

Deviation
Mental Model
Behavior expected by user
Confusion
Observed system behavior (left to right)
System
User

Figure 2.2: Expected behavior *versus* observed system behavior (extracted and adapted from [2])

either show the explanandum to be an example of a general pattern or they identify all or a portion of its causes [57, 58]. According to Schank [59], explanations fill in knowledge gaps and contribute to understanding by addressing states of confusion and deviation that occur when a person encounters a situation that does not conform to her internalized model of the system – her mental model (Figure 2.2). As a result, explanations can help to correct or adjust the user's mental model.

Achinstein [60] focuses on explanations as a process of communication between entities. Here, the purpose of an explanation is to fulfill a recipient's request for information from a designated sender. Therefore, according to this view, if an entitiy does not require a detailed explanation, then something does not necessarily need to be explained. This interpretation goes in the same direction of Rosenfeld and Richardson's [61] discussion on what makes a proper explanation. Explainability is viewed by Rosenfeld and Richardson as a passive process based on user understanding that also determines whether an explanation is a proper explanation. The authors give an example of a neural network that has been trained to compare two images and output a score, but most people do not understand how it arrived at this score. The authors contend that offering the user the images so they can examine them and determine whether they are similar can serve as an explanation. As a result, the very definition of an appropriate explanation depends on the recipient and the circumstances surrounding the sender and the receiver's interaction. These two perspectives ([60, 61]) are concerned with the same issue: the need for and usefulness of an explanation is dependent on the receiver and their understanding.

Köhl et al. [62] have a similar view and consider that the intended target group G, or the type of stakeholder, and the explanatory context C determine what qualifies as an explanation of what for whom. They build a definition based on the ideas of Achinstein [60] and Van Fraassen [54]: "E is an explanation of explanandum X with respect to aspect Y for target group G, in context C, if and only if the processing of E in context C by any representative R of G makes R understand X with respect to Y". This definition of explanation qualifies explanation based on the information receiver's understanding and relevance (depending on the context). As a result, for the purposes of this thesis, I condense the above viewpoints into the following concise definition:

Definition 2.2.2 (*Explanation*)

An explanation is a piece of information that contributes to the addressee's understanding of an explanandum in a given context.

2.3 Artifacts in Software and Requirements Engineering

According to Ghazi and Glinz [63], an artifact is "any kind of textual or graphical document with the exception of source code". Artifacts may have different forms, such as textual requirements documents, graphic models (including Unified Modeling Language (Unified Modeling Language (UML)) diagrams), glossaries, charts, frameworks, or quality models. Artifacts can also have any size and granularity, ranging from comprehensive documents to user stories, use cases, bug reports or diagrams. Artifacts can be both **used**, **reused** and **produced**. Requirements engineers typically use or reuse artifacts as guidance during software (or requirements) engineering activities. They may also create artifacts to gather knowledge that will be used and reused (e.g., catalogues), or they produce artifacts as a form of documentation (e.g., requirements specification or story cards).

This work proposes five different artifacts: a definition, a conceptual model, a knowledge catalogue, a process reference model, and a quality framework. These artifacts are all linked to one quality aspect: explainability. They were created with the intention of being reused to achieve shared understanding and to provide guidance during RE activities. In the remainder of this section, I will look at how artifacts can support RE for both "regular" or explainable systems.

2.3.1 Definitions

Definitions are the first important step in facilitating communication and providing support regarding a given topic, concept, or aspect in SE. Definitions help to delineate the scope of a given concept, often providing variables that represent components and actors that compose the concept. In SE, definitions provide software engineers with a preliminary guideline on the scope and elements of something. A definition of the meaning of a quality aspect, for example, aids software engineers in understanding the quality aspect and what makes it what it is during RE and quality assurance.

For instance, Penzenstadler [64] proposes a definition for sustainability for SE. The proposed definition highlights the important variables of sustainable systems in the context of SE. According to the author, these variables can help to define the scope of analysis and to guide discussion around requirements. Wixon [65] explores the importance of a definition for usability and discusses how essential it is for development teams to agree on its definition. For instance, for some developers, usability could mean ease of use; for others, it could mean long-term efficiency. Failure to reach an agreement on a definition results in wasted development time and products with no systematic design focus.

Definitions tend to evolve and change over time, as new insights about the concept to be defined emerge. Usability is an example of a quality aspect whose definition evolved over time [66]. Definitions also tend to vary according to the focus of the field of study. In the literature, definitions of explainability vary considerably: the field of AI might focus more on models' aspects, while philosophy might focus on types of explanations. As a result, one could argue that there is no one true definition of explainability, but rather several complementary ones. To achieve shared understanding, definitions should be agreed upon between individuals (e.g., software engineers) in a given context. In chapter 3, I will discuss in detail other aspects in which definitions of explainability differ.

2.3.2 Models

Models assist the RE process by allowing software engineers to gain a sufficient understanding of a subject (such as a domain, a quality aspect, or a process). Since a model is an abstraction of reality, it eliminates all but the most fundamental aspects of reality in order to serve the model's intended purpose [67]. Therefore, a model deliberately focuses on some of aspects while excluding others [68]. Models can be created from scratch or they can be modified versions of already existing models. This thesis proposes three kinds of models: a conceptual model, a reference model, and a framework. Although they can all be seen as models, I distinguish between the three in the context of this work.

Conceptual Models

Conceptual models are models that provide a high-level overview of a field or topic by categorizing it into broad categories. During requirements analysis, conceptual models can be used to understand the taxonomy or characteristics of a specific quality aspect. Conceptual models can be used as references in various projects because they document knowledge about a specific domain, concept, or NFR (in the case of this work, a conceptual model about the impact of explainability on system quality). Taxonomies are well-known examples of conceptual model. Typically, the knowledge required to develop conceptual models is derived from literature, prior experiences, and domain expertise.

Nunes and Jannach [69] propose a taxonomy for explanations in recommender systems based on results from a literature review. The taxonomy is composed of general facets associated with explanations such as their generation approach, and interface aspects such as explanation content and presentation form. The authors also investigated how explanations are generated, presented to the users, and evaluated. Arrieta et al. [70] proposed a taxonomy that maps machine learning models to the explanations they produce. The taxonomy should serve as an artifact that supports professionals in the implementation of AI methods in organizations. The authors also investigated the relationships between different concepts related to the field of XAI.

Reference Models

A reference model consists of a *minimal* set of unifying concepts, axioms and relationships within a particular problem domain, and is independent of specific standards, technologies, implementations, or other concrete details [71]. A reference model brings together the main entities of the knowledge area and the relationships between them, as well as related methods of research and practice [72] (in the case of this work, a process reference model that serves as a blueprint for a potential development procedure).

The Open Systems Interconnection model (OSI) [73] is a reference model that divides network protocols into seven abstraction layers. The layers help to separate concepts and network aspects into abstraction levels, helping to compartmentalize the development of network applications. The OSI model is widely used by network engineers to describe network architectures, even though it is informal and does not correspond perfectly to the protocol layers in widespread use. In fact, this is precisely the reason of their wide adoption: reference models can be seen as templates for understanding significant relationships among the entities of some environment or domain (e.g., computer networks or, in this thesis, the development process of explainable systems); this abstract nature gives them flexibility, making them easily adaptable. Reference models can also be used to standardize or describe processes [74]. A prominent example is the Requirements Engineering Reference Model (REM) [75] which pursues the goal of an efficient "RE organization and process".

Other works also focus on the development process of explainable systems. Mohseni et al. [76] propose an iterative process to develop explainable systems. Schoonderwoerd et al. [77] present a user-centered design (User-Centered Design (UCD)) approach with reusable patterns for explanations. These patterns cover domain analysis, requirements elicitation, the design of the system, and the evaluation of interactions. While both works recommend some techniques, the process reference model proposed in this thesis specifically proposes practices and associated techniques.

Frameworks

A framework is more detailed and contains all the elements that must be observed. A framework will outline the steps or phases that must be followed in order to implement a solution without going into specifics about what activities are performed in each phase [78].

Simkute et al. [79] propose a framework to support the design of explainable systems based on the cognitive needs of expert users. It pinpoints the three key dynamics that have the greatest impact on the style and nature of the explanations needed and offers illustrative ideas for how designers might approach the problems. The framework is domain-independent and can be used to inform explainability researchers and guide interface designers by outlining important decision-making processes and associating relevant contextual factors with the appropriate explanation strategies.

Sokol and Flach [80] propose a framework that compiles the criteria and desiderata that other authors have proposed or implicitly used in their research. The framework links explainability algorithms to criteria that are used to guide their development and evaluation methods. The framework is in form of fact sheets that should enable researchers and practitioners alike to quickly grasp capabilities and limitations of a particular explainability method, guiding the development of new explainability approaches. It also allows to compare and contrast explainability approaches, helping to understand their capabilities and to identify discrepancies between theoretical characteristics and properties of implementations.

More related work will be presented when I discuss the framework in chapter 8. Although there are different frameworks for the integration of explanations, most of the publications are focused on specific domains and do not address explainability for information systems in general. To the best of my knowledge, no contribution has yet been made that specifically addresses RE.

2.3.3 Knowledge Catalogues

Catalogues document knowledge about a given topic (e.g., a specific domain or about quality aspects, in the case of information systems). Some researchers developed catalogues for specific domains based on the premise of the NFR framework. Serrano and Serrano [81] developed a catalogue specifically for the ubiquitous, pervasive, and mobile computing domain. Torres and Martins [82] propose the use of NFR catalogues in the construction of Radio-Frequency Identification (RFID) middleware applications to alleviate the challenges of NFR elicitation in autonomous systems. They argue that the use of catalogues can reduce or even eliminate possible faults in the identification of functional and non-functional requirements. Carvalho et al. [83] propose a catalogue for invisibility requirements focused on the domain of ubiquitous computing applications. They emphasize the importance of software engineers understanding the relationships between requirements in order to select appropriate strategies to satisfy invisibility and traditional NFRs. Furthermore, they discovered that invisibility might impact other essential NFRs for the domain, such as usability, security, and reliability.

Leite and Capelli [84] discussed and defined transparency in the context of system engineering, but also as a broader concept, applied to processes and organizations. The authors created a graph to illustrate how transparency interacts with other quality aspects. The graph comprises 33 quality aspects (called softgoals) arranged in three levels according to the dependency relationship between the nodes. Transparency occupies the higher level of decomposition, and the second level consists of five derived quality aspects which influence directly on the satisfaction of a degree of transparency, being those: accessibility, usability, informativeness, understandability, and auditability. One way to achieve transparency is by satisficing these quality aspects at the lower level of decomposition.

Mairiza et al. [17] conducted a literature review to identify conflicts among existing NFRs. They constructed a catalogue to synthesize the results and suggest that catalogues can assist software developers in identifying, analyzing, and resolving conflicts between NFRs. Carvalho et al. [85] identified 102 NFR catalogues in the literature after conducting a systematic mapping study. They found that the most frequently cited NFRs were performance, security, usability, and reliability. Furthermore, they found that the catalogues are represented in different ways, such as in the form of softgoal interdependency graphs, matrices, and tables. The existence of so many catalogues illustrates their importance for RE and software design. Although these catalogues present knowledge about 86 different NFRs, none of these works addresses explainability.

By making knowledge available in artifacts such as definitions, models, and catalogues, software engineers can 1) draw on know-how beyond their own fields and use this knowledge to meet the needs of a particular project, and 2) have a comprehensive picture of the system's requirements. Essentially, software engineers can use these artifacts to facilitate the RE and software design process.

This chapter discussed concepts and terminology that are needed to understand this thesis. Although some of the related work on artifacts approach SE, RE, or explainable systems, none of them is specifically focused on SE or RE *for* explainable systems.

3
The Quality Landscape

Software quality is a multidimensional concept and, like any concept, has levels of abstraction, and can be conceptualized in a broad or specific way [86]. I introduce four concepts in this chapter to help readers better understand the significance and complexity of system quality: quality landscape, quality spectrum, stakeholders in the landscape, and quality dimensions. In order to understand the proposed theories and artifacts, these concepts will be useful to this thesis both in terms of jargon and in terms of theory, so they will be used throughout the remainder of this text.

3.1 The Composition of the Quality Landscape

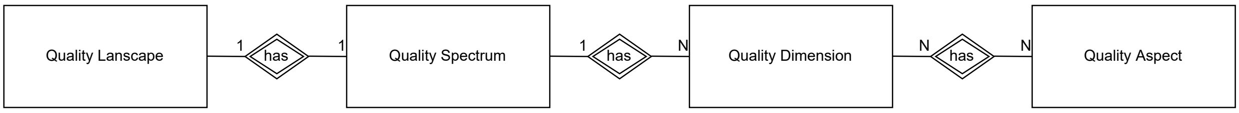


Figure 3.1: The composition of a quality landscape

The research described in this chapter is partially based on two publications ([12] and [87]), performed in collaboration with three researchers: Wasja Brunotte, Timo Speith, and Kurt Schneider.

Boruszewski [88] defines requirements *landscape* as the structure of a project with respect to its requirements artifacts and their links. This definition focuses on elements and their relationships. I apply and adapt this concept, viewing landscape as a collection of links and roles of elements between system quality aspects. A quality landscape is composed by a quality spectrum, quality dimensions, and quality aspects. These concepts form the landscape composition, presented in Figure 3.1. Besides, stakeholders are the actors that create, change, and experience this landscape (cf. section 2.1).

Therefore, to understand the quality landscape, the first and most basic definition is that of quality aspect:

Definition 3.1.1 (*Quality Aspect*)

> *A quality aspect is an abstract real-world aspect that reflects the perceived or desired quality of a system by a specific stakeholder. A quality aspect can be translated into quality goals and stated and refined as NFRs.*

Different quality dimensions need to be taken into account during the requirements engineering process in order to successfully identify the quality aspects of a system, how they translate to requirements, and to comprehend how these requirements affect system quality. The word *dimension*, per definition, is "a way of considering something", or "one of the elements or factors making up a complete entity"[89]. Therefore:

Definition 3.1.2 (*Quality Dimension*)

> *A quality dimension is a conceptual layer in the quality spectrum that groups quality aspects that make up a system. A quality dimension also represents a perspective from which to consider the quality of the system.*

The external/internal quality concept based on the ISO 25010 [22] and proposed by Freeman[90] is another important factor to understand the concept of quality dimensions. One way to assess the quality of systems is by using software measures that evaluate the quality from two different perspectives: 1) internal quality, referring to "static" quality aspects of the software related to its architecture (e.g., number of lines of code and level of coupling); 2) external quality and quality in use, referring mostly to quality evaluated according to the perceived system's behavior or to how well the software meets user needs in a specific context (e.g., efficiency and satisfaction).

I consider the *external* quality characteristics as the ones which are more related to the users or the *quality in use*, and the *internal* as the ones which are more related to the developers or the *system itself*. This concept brings the idea of a continuous spanning from internal to external quality. Therefore, in the classical sense, the dimensions represent "positions" or "directions" in a quality spectrum that ranges from internal to external.

Definition 3.1.3 (*Quality Spectrum*)

A spectrum is "a range of different positions between two extreme points" [91]. In this work, the system quality spectrum is a range between two extreme points: internal and external quality.

As pointed out by McConnel [92], the difference between internal and external characteristics is not completely clear-cut and affects several dimensions. Therefore, I do not assign clear-cut internal or external dimensions to the dimensions and quality aspects of the conceptual model, but rather acknowledge a continuous shift from external to internal.

Finally, the aforementioned concepts compose a quality landscape:

Definition 3.1.4 (*Quality Landscape*)

A quality landscape is a collection of the existing quality aspects, their relationships (interactions with other quality aspects), and their taxonomy (dimensions and position on the spectrum), being that on a real system project or on a conceptual level.

For instance, it is possible to discuss the quality landscape of a specific system or domain, concerning all the identified quality aspects for this system or domain; or the quality landscape conceptually speaking, concerning all existing quality aspects. In the context of this work, I will present the quality landscape formed by the quality aspects that are influenced by explainability. This quality landscape is intended to be conceptual and generic, representing the potential impact of explainability on system quality in general. The landscape is represented by a conceptual model that summarizes the quality aspects influenced by explainability, which are divided into seven dimensions (that are summarized in four), each of which is discussed in relation to its position on the spectrum and in relation to the relevant stakeholders.

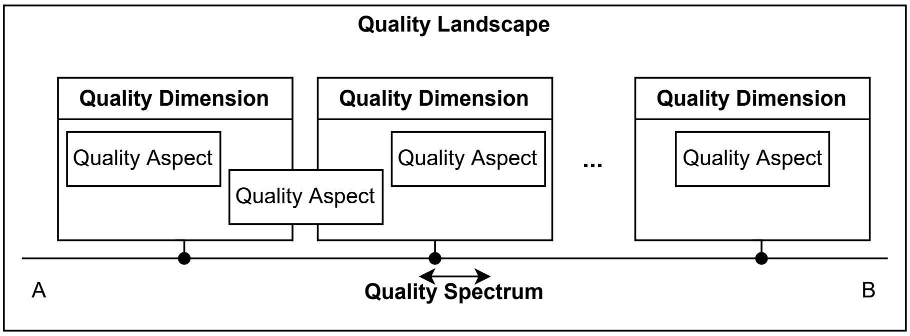


Figure 3.2: A conceptual illustration of a quality landscape

Figure 3.2 illustrates the concept of quality landscape and all related concepts: A quality landscape has a spectrum that ranges between two extreme points (A, B). Quality aspects compose quality dimensions that are situated in given points of the spectrum. Quality aspects can sometimes belong to different dimensions at the same time.

3.2 Stakeholders in the Landscape

A person or organization that is affected by a project, service, or decision is referred to as a stakeholder. Therefore, stakeholders are also those who are impacted by a system's quality. End users, project sponsors or clients, architects, developers, testers, quality engineers, project managers, regulators, product managers, operators, and maintainers are examples of typical stakeholder roles in an average software project [93].

Individuals differ in their background-knowledge, values, experiences, and many further respects. Accordingly, they also differ in what is required for them to understand certain aspects of a system. Langer et al. tackle explainability from the perspective of the persons who demand explanations [94]. They hold that some persons are more likely to be interested in a certain quality aspect than others. For instance, a developer may be more interested in the maintainability of a system than a user.

They categorize quality aspects that are influenced by explainability according to so-called *stakeholder classes* and distinguish the following ones for the case of explainable systems: *users*, *developers*, *affected parties*, *deployers*, and *regulators*. According to them, these classes should serve as a reference point when it comes to implementing explainability since the interests of those different stakeholder classes may influence the quality goals in a system or even conflict [94]. As a result, stakeholders have an impact on a quality landscape in terms of both the quality aspects to be addressed and how these quality aspects interact and are implemented.

3.3 Seven Quality Dimensions

Quality aspects are not mere descriptions of the quality characteristics of the system. They are central to understanding how these quality characteristics translate into functional requirements and constraints that must prevail and are fundamental aspects for the design of a system [95]. According to Cysneiros et al. [96], dealing with NFRs from the start of software development and integrating this knowledge with functional conceptual models leads to cost savings and higher customer satisfaction.

Quality aspects have three main characteristics that should be considered during requirements engineering: They can 1) be subjective in nature, because some solutions aimed at achieving a quality aspect may be considered accomplished by some people but not by others; 2) be relative in nature, because the degree to which they are perceived as met also varies depending on the person and the context; and 3) be interacting, because attempts to achieve one quality aspect can harm or help the achievement of another [97].

In addition to these challenges, various factors contribute to system quality and must be considered during the RE process. These factors can be seen as quality dimensions that will have an impact on the analysis process, either while a requirements engineer is still considering the relevance of a given quality aspect within a system, or later when making design decisions toward the necessary operationalizations to achieve this quality aspect. To identify relevant quality dimensions, I manually reviewed the existing literature by searching for papers on quality aspects published in two key requirements engineering sources: the Requirements Engineering Journal and the proceedings of the IEEE Requirements Engineering Conference. For a better coverage, I also looked for papers of interest in the listed references. I combined the findings of this search with the findings of the survey (that will be discussed in chapter 4) and knowledge from the explainability literature.

I identified seven dimensions that affect the elicitation and analysis of quality aspects [12]: the *needs and expectations of users*, *cultural values*, *corporate values*, *laws and norms*, *domain aspects*, *project constraints*, and *system aspects*. These dimensions influence the identification of relevant quality aspects within a system, the impact of a quality aspect on system quality, and design choices toward the operationalization of a given quality. In the text that follows, I will discuss the seven dimensions through the lens of explainability.

Users' Needs and Expectations

Quality is shaped by social and human factors as well as context-dependent aspects. Different groups of users have different expectations, experiences with technical systems, personal values, preferences, and needs. Such aspects also mean that individuals can perceive quality differently and have different needs and expectations with regard to explanations.

These individual expectations need to be balanced with a number of other factors: cultural values, corporate values, laws and norms, domain aspects, as well as with project constraints and system aspects [98, 99].

Cultural Values

Culture is the collective mindset that distinguishes the members of one group of people from another. Different cultures might require different types of information, process it differently, and require different designs of information systems [100]. Systems are influenced by the respective national environments in which they are deployed, such that the respective culture strongly influences their design [101].

Cultural values refer to the *ethos* of a group or society [102]. They influence the need for a given system quality and how it should be operationalized [101]. The European *Ethics guidelines for trustworthy AI* [103] are an example of values that represent the common vision of a group. Such values vary between cultures. For instance, while some cultures are more concerned with data privacy and the ethics of systems, the largest proportion of internet users in some emerging economies claim to trust the internet [104]. Hence, the need for explanations and what is expected in terms of explanations can be perceived differently among cultures.

Corporate Values

Corporate values refer to the strategic vision and values of the organization [105]. With the omnipresence of information systems and the age of AI, the importance of integrating human values into software becomes more evident. Organizations have made considerable efforts to define their public values statements, including values such as corporate integrity, respect and honesty [106]. However, integrating such values in the systems we produce is not a task embedded in the everyday routine of a project. As Whittle [106] observes, even in cases in which companies consider values during software development, the approach is limited to creating a values-driven culture instead of having it integrated in software.

While explainability can be a means of providing greater transparency in information systems, if there are no clear laws requiring the company to satisfy a certain level of explainability, having explainability as an NFR would depend on the company's own values and interests. In the case of self-driving cars, according to Cysneiros et al. [51], different car manufacturers are likely to prioritize NFRs according to specific selling points of their brands. For instance, the decision of achieving explainability within a system may stem from a corporate interest in achieving more transparency, to improve the users' trust in the system, or to provide a better user experience.

Laws and Norms

Cultural values resonate in the conception of laws and norms. The European Union has adopted a specific data protection legislation (General Data Protection Regulation (GDPR)) [107] to protect citizens' data. A certain degree of explainability is also required under this law. Other countries either have their own separate data protection law, or no specific legislation.

Laws and norms may impose constraints that have to be met despite the corporate values. Information systems are required to comply with them, or face sanctions. These laws, regulations, and policies need to be analyzed and accommodated during the definition of requirements for a new system [108]. They have a direct impact on how companies prioritize quality goals and can influence the system architecture [109].

Domain Aspects

Domain aspects are an essential aspect on the analysis of NFRs, since needs change depending on the domain [110]. Each domain has quality characteristics that are of particular importance and may deserve more attention. The relative priority among NFRs may change as the characteristics of the environment in which the system operates differ [111].

Domain aspects determine whether explanations are more urgent and how they should be designed. Explainability may be an optional requirement in some areas, being a way of improving user experience, whereas it may be a fundamental quality in others. The degree and extent to which an explanation is required to support or justify a decision varies with the domain's criticality. When compared to non-critical systems, critical systems impose a different set of NFRs. In more critical domains such as medical, financial, and autonomous systems, explainability may be more urgent. In the case of a simple navigation system, for example, explanations that inform the user about the chosen route or the reasons for a route change can be viewed as an additional feature that influences user experience and product satisfaction. However, in the case of systems that support medical diagnosis, the lack of explanations of the reasons for a given diagnosis can have a much more dramatic impact, with ethical consequences [112].

Project Constraints

Project constraints are more practical aspects (also known as *non-technical aspects* [113]), such as available resources (e.g., time, money, technologies, manpower). Such aspects have a strong influence during RE, and may take precedence over others [114].

During the elicitation and analysis of explainability, the project constraints must be considered. In the end, all dimensions need to be balanced considering these non-technical aspects. Every project has resource constraints that must be balanced against the desired level of quality. The need for explainability and the effort required to meet it must be balanced against what it represents in terms of resources for the company. Excessive quality can lead to unnecessary costly design of the information system, unnecessary use of the resources needed to operate the system, and trade-offs where other important quality goals are negatively affected [115].

System Aspects

This dimension is related to quality aspects of the code or the system itself, such as efficiency and testability. These aspects are also related to specific characteristics of the system: physical, architectural, logical. The system aspects are related to the practical and technical aspects in which the requirements are implemented. This influence many decisions because some of these aspects restrict the range of possible solutions or imply a lot of work. System aspects are similar to the project constraints in terms of practicality and are also more related to the internal quality and the way that the system is constructed.

Some requirements can impact, for example, the efficiency or the testability of a system. In the case of explainability, an algorithm to provide explanations could have a negative impact on system efficiency, or the architectural decisions for an explainable system could have a negative impact on code complexity. Alternatively, explanations may aid in the detection of bugs, thereby facilitating debugging, and having a positive effect on the system itself.

The concept of *quality landscape* is essential for understanding software quality and the research presented in this thesis. In the following chapter, I will look at the first step toward understanding the role of explainability in the quality landscape: a survey of end users to identify challenges and needs related to explainable systems.

4
The Need for Explanations: A Survey

The incorporation of explanations into software has frequently been discussed as a solution to provide more transparency and reduce system opacity. But, before proposing any new solutions or delving deeper into this topic, it was critical to first hear from users and understand their perspectives on explainability.

As common in scientific research, my research questions were driven by real-world questions based on anecdotal evidence. I started researching explainability in an effort to comprehend it from the standpoint of RE, but I began to doubt the value of carrying out more research. I was also seeking to find out if, from a users' point of view, the premise that explainability would promote software transparency would hold true. When discussing the subject with other academics, it was commonly debated as to whether users would find receiving explanations to be really interesting or useful, or if they would find it annoying and useless. It is important to note that the "renaissance" of the explainability trend* started with a total focus on providing explanations for software developers and data scientists who needed to understand the outputs of ML models, perceived as way more urgent and complicated. The focus until then was neither on *any* user nor in *any* software.

*Although the topic of explainability is not new, there has recently been a revival in interest in it, which is why I use the term renaissance (cf. section 2.2).

The research described in this chapter was performed in collaboration with two researchers: Oliver Karras and Kurt Schneider. The results were published in [116], on which this chapter is based. As a result, I occasionally use the pronoun "we".

Therefore, it was clear for me that the first step in this research was to comprehend user requirements in terms of explainable software behavior: Are users really interested in software transparency and can explanations be considered as an appropriate way to achieve it? To get more insights into these questions, I conducted a survey with 107 participants to assess their opinion on the current level of transparency in software systems and what they consider to be the main advantages and disadvantages of embedded explanations. Based on their answers, I assessed the relationship between explanations and transparency and analyzed its potential impact on software quality. More specifically, I obtained an initial overview of the interrelationships between explainability and some other quality aspects in the quality landscape. More precisely, understandability, usability, auditability, and the relationship between user and system. Furthermore, with the help of this survey, it was possible to identify four research needs that guided the rest of the research and culminated in this thesis.

4.1 Research Design

Explainability has been addressed as a key requirement for software-supported decisions and a means of promoting transparency [117]. Although some studies investigated the impact of explanations on user experience [37, 118] and others have investigated in which situations some kinds of explanations are more appropriate [119, 47], there was a lack of studies investigating the relationship between explanations and transparency at the level of NFRs. Furthermore, it was not clear whether end users actually see explanations as a way to better understand a system and improve transparency.

Since users are an essential source of requirements, it is fundamental to understand their views and what they expect from explanations. Therefore, the goal of this study was to investigate and understand the users' views and expectations, as these are important steps towards addressing explainability as an NFR. I also wanted to explore the interaction of explainability with NFRs related to transparency, and what has to be considered to meet the users' needs. This knowledge can assist software engineers in determining potential trade-offs, costs, and implications of integrating explanations into software systems.

I addressed this by asking users about the need for explanations in applications that they use on a daily basis. I conducted an online exploratory questionnaire with 107 participants. To analyze the data and understand whether and how explanations affect transparency, I investigated whether aspects of participants' responses could be linked to transparency-related NFRs described in Leite and Capelli's Transparency Softgoal Interdependency Graph. (Softgoal Interdependency Graph (SIG)) [84]. We used this graph in the later iterations of the coding process to identify requirements in the participants' discourse.

I applied the goal definition template by Wohlin et al. [120], to formalize the goal of this research:

Goal definition: We *analyze* end-users' perspectives about the need for explanations in software *for the purpose of* investigating which non-functional requirements are impacted by explanations and how they relate to software transparency *from the point of view of* end-users *in the context of* an online questionnaire.

This goal was derived into three research questions:

Research Question A

What do end users currently think about the need for explanations?

This question was meant to assess users' perceptions about the importance of explanations. Particularly if users perceive a need for explanations in scenarios involving confusion and deviation from expected behavior (cf. subsection 2.2.1), such as when users expect something from their interactions with the system but do not receive it. A navigation scenario was provided to address this issue (cf. Appendix A). Participants were asked whether they would be interested to receive an explanation in the hypothetical situation. Those who showed interest were asked for suggestions on what would be a good explanation in this situation. The goal was to discover what information users deem relevant while obtaining explanations, which is essential knowledge for the operationalization of explainability. A question was also posed to determine whether explanations were generally required. The participants were then asked when explanations should be provided: never, whenever they requested it, only when something unusual occurred, or both.

Research Question B

Which non-functional requirements related to transparency can be impacted by the problems described by the participants?

This question was designed to find out whether and how participants experienced issues with the software applications they frequently use. Participants were asked to report situations where they were unable to understand something while using an information system. Analyzing their responses helped to understand how current concerns can have a negative impact on transparency and which NFRs are affected by the reported issues.

Research Question C

> *What are the advantages and disadvantages of receiving explanations and how do they relate to transparency?*

This question was designed to examine the perceived advantages and disadvantages of receiving explanations. This was useful for determining: 1) the NFRs affected by explanations; 2) the benefits and drawbacks to be taken into account when considering explainability as a quality goal; and 3) whether the identified NFRs can have a positive or negative impact on system transparency.

4.1.1 Survey Design

Based on the research questions, an online questionnaire was designed using LimeSurvey. The questionnaire contained 16 questions (11 multiple choice, five open-ended): three on demographics, one self-assessment question on software skills, four on software use, three on problems with software use, three on explanation needs, one on the frequency and one on the presentation of explanations.

Figure 4.1 summarizes the process of survey design and data analysis. The questions were reviewed beforehand, using the checklist provided by Lessmann [121]. This was followed by four rounds of pilot-tests, two of these with members of the target population and two with research colleagues (indicated by the different symbols in Figure 4.1). In each pilot-test, the respective participant completed the survey and we discussed how the questionnaire could be improved. The full instrument can be found in Appendix A.

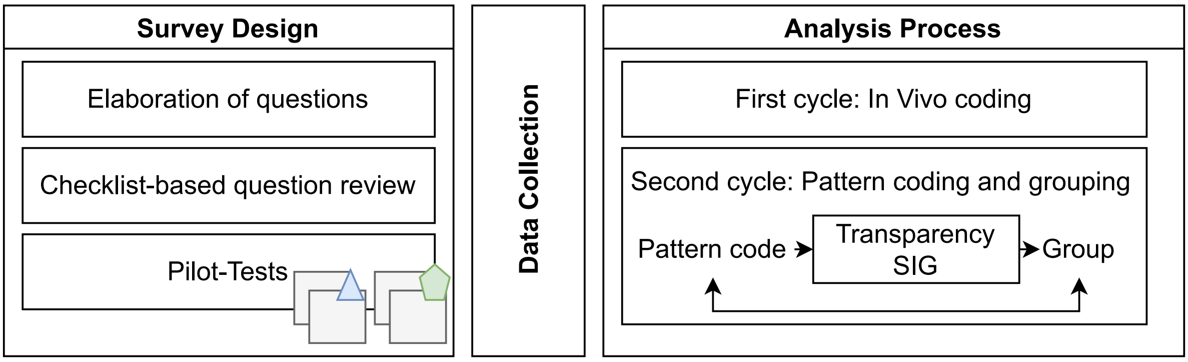


Figure 4.1: Survey design and data analysis process

4.1.2 Data Collection

In late 2018, I shared the questionnaire using different channels: LinkedIn, Twitter, Facebook, and academic mailing lists. The questionnaire was widely disseminated via social connections. In order to reach more people, I also encouraged my colleagues to share the questionnaire with their connections. Since the goal was to learn what end users from various backgrounds had to say about the topic, the target population included adults of all ages and occupations.

Based on this sampling strategy, I expected the majority of participants to be from Brazil and Germany. As a result, the questionnaire was available in three languages: Portuguese, German, and English. Participants had to sign an informed consent form and indicate that they were at least 18 years old.

From the 171 that started the survey, 107 completed it. Only responses from those who completed the questionnaire were analyzed. Because the qualitative questions were optional, some respondents did not respond to all of them. In this situation, the replies to the qualitative questions that were answered could still be analyzed to gain insight into the research questions.

4.1.3 Analysis Process

I examined the open-ended questions utilizing Saldaña's open-coding approach [122]. It involves two successive coding cycles for a qualitative data analysis. Each cycle consisted of three phases, that can be iterated. For the first coding cycle, I employed *in vivo coding* [123], a first cycle coding technique that is known for preserving participants' points of view in the codes. In their answers, I have noted the essential components pertinent to the research questions. Depending on the length and significance of the response, a single response could yield more than one code. A second researcher reviewed the codes independently to mitigate subjectivity.

In the second cycle, we grouped the initial *in vivo codes* according to their similarities. Next, we applied the Pattern Coding approach by Miles and Huberman [124]. This approach is a way of grouping the initial codes into a smaller number of themes or constructs. The main theme expressed by the codes in each group was used to create categories. During this stage, my colleague and I collaborated to identify the themes and discuss their implications. We tried to keep the essence of what the participants said during the categorization process. So, even though some categories appear to be quite similar to one another and could thus belong to the same category, we kept them separate to preserve their original connotations.

After that, groups were created from the categories. If there was any similarity between the category and an NFR in the Transparency SIG, the category would be designated as belonging to this NFR, forming a group. In the absence of a match with a quality aspect listed in the SIG, a new group would be created based on the meanings and connections of the codes. During this phase, we coded the data independently into the final categories to guarantee consistency. After this, we compared the two coded data sets, discussed differences in the coding process and reached an agreement about the codes. The degree of agreement was calculated using the Cohen's Kappa statistic [125]. The calculated value of $\kappa = 0.88$ indicates an almost perfect agreement [126].

4.2 Demographics

From the 107 valid responses, 90 (84.11%) came from Brazil and 17 (15.88%) from Germany. On average, participants reported a good proficiency in the use of software systems. To assess this proficiency, we included a self-assessment question in which respondents had to indicate which of the tasks they were able to complete. It also included tasks corresponding to certain skill levels. The majority of respondents claimed to have a high level of proficiency, including participants who work with computers, are programmers or have programming skills. This shows that, in general, participants are comfortable with the use of software systems. A clear limitation is that our results do not represent the population of unskilled users or people who struggle with technology. In any case, our demographic represents a representative subset of all information system users: end users of various ages who are technologically literate.

When asked whether they use software applications more for work or for personal reasons, 43% of the respondents affirmed they use an equal amount for both work and private life, while 43% responded that they use them more for work, and 14% more in private life. This highlights the ubiquitous nature of software systems in the lives of these individuals.

4.3 Need for Explanations

Three aspects of the need for explanations were examined: the questions that need to be answered by explanations, the situation-specific need based on a hypothetical scenario with variable context, and the overall need for explanations.

4.3.1 Situation-specific need

A hypothetical scenario was presented based on the device and applications that the participant identified as being used frequently. In the hypothetical scenario, users would utilize a navigation system while driving on a familiar route, and the system would suggest a different route than normal. We asked participants whether they would like an explanation for the route change. This question had two variations to assess the problem from two separate angles, where the need for explanations may differ: 1) one in which the user is in a car using an on-board navigation system (On-Board Navigation System (OBNS)) to navigate, and 2) another in which the user is a pedestrian utilizing public transportation and relying on the navigation system in the smartphone to consult better routes and alternative transportation.

The goal of this question was to examine the need for explanations in scenarios where there is a mismatch between the users' goals and what the system presents, such as when the users' expectations are not satisfied. It is thus a need in a specific context: if what the user expects differs from what is displayed in the system. This notion is also addressed in Doshi-Velez and Kim's work [53]. The authors examine why and when explanations are essential and beneficial. They claim that an explanation need stems from a lack of completeness in the information presented, and that people may require explanations to fill gaps in understanding. This break of expectations is one of the triggers for the explanation need. This trigger will be discussed in chapter 7.

All participants answered the question with the smartphone scenario, since they indicated that they had access to this type of device. 20 of the 107 participants also indicated using OBNS and, therefore, also answered the question corresponding to this scenario. Of the resulting 127 responses, 71.6% (91) answered that they would be interested or extremely interested in receiving an explanation about the obscure situation. There was a variation in the expressed degree of interest according to the type of device: 95.0% (19) of OBNS users answered they would be interested or extremely interested, while one would not be interested at all. 67.3% (72) of smartphone users answered they would be interested or extremely interested, 18.7% (20) slightly interested, 11.2% (12) indifferent and 2.8% (3) not interested at all. This could imply that OBNS users have a higher demand for explanations, tied to the type of system they use: When driving, users have less time to make a decision, resulting in a greater need for explanations.

4.3.2 Questions that should be answered by explanations

We asked participants to suggest a good explanation to be presented in the hypothetical situation. We looked for specific elements in the responses, such as references to specific data (e.g., time, route) or the level of abstraction of the suggested explanation. 87 participants completed this question, which resulted in 103 codes. During the coding process, we identified three main questions to which the participants want answers: *what*, *why*, or *how*.

35.0% (36) of the codes refer to the *what* question. The what question is related to the *explanandum*. Participants expressed desire in knowing which specific piece of information supported and influenced the decision: data-related aspects contained in that information and used during decision-making. Some code examples include *"which information led the software to take this decision"*, and *"which variables are influencing the choice"*. Some answers referred to specific data, such as in this participant's statement: *"If there is some kind of incident, show me the route, time, possible accidents, etc."*.

In 11.6% (12) of the codes, participants refer to the *how* question. It reflects the users' desire to understand the inner reasoning process of the algorithm. Users also expressed the wish to be able to audit or verify the behavior of the system or to find out more about the internal model the system built about the user. Some examples are *"evaluate the capacity of the system of generating better routes"* and *"to see how the algorithm detected the changes"*.

In 53.4% (55) of the codes, participants refer to the *why* question. Participants expressed willingness to understand **why** something happened, i.e., to better understand the reasons behind a decision or event, or the existing policies. Answering this question usually requires knowledge about **what** data are considered and **how** the information was inferred. To understand exactly how this question must be answered, the level of abstraction of the explanation needs to be assessed. Explaining why something happened may either need a more specific answer, considering many variables **or** a very general answer, with a higher abstraction level. Some code examples are *"why the route is not being suggested"*, and *"benefits of the new route when compared to the usual"*.

These results resonate with the empirical investigation of Kuhnke [127] about how end users express the need for explanations in app reviews. Kuhnke identified that the explicit need for explanations frequently manifests itself in connection with the use of one or more of these words: who, what, when, where, why, and how (the author refers to these words as the *Five Ws and How (5W1H)*). The author analyzed 253 sentences from app reviews and discovered that at least one of the 5W1H words occurs with a frequency of approximately 52.2% in the analyzed sentences.

4.3.3 Overall Need

We asked the participants about when explanations should be presented. 66.4% (71) answered that explanations should be presented only on demand. This reflects that users are interested in explanations, but want to have total control about when to receive it. It also imposes a new challenge, since systems must have the ability to explain much of their behavior in order to provide explanations by request. 28.0% (30) answered that explanations should be shown just when something exceptional happens (e.g., in the case of mismatched objectives). 3.7% (4) answered that they would like to receive explanations in both situations (automatically, when something exceptional happens and by request) and only 1.9% (2) answered that explanations should never be presented.

Answering RQ A (Finding 1)

Results indicate that a number of factors need to be properly understood and transformed on requirements in order to design relevant explanations. These factors include the overall need for an explanation (if and when an explanation is needed), the user's circumstance, and the type of question that has to be answered.

4.4 Perceived Problems in Understanding Software

To answer this question, we asked respondents if they remembered having trouble understanding the behavior of any software they had previously described as being in regular use. Then we inquired if they could report an incident like this. 24 participants answered the open-ended question, which resulted in 19 valid answers. We interpreted each answer and tried to identify whether they could be associated with a quality characteristic listed in the NFR framework.

68.4% (13) of the responses were related to *usability*, with explicit correspondences to a perceived impact on its sub-dependencies *uniformity*, *simplicity*, *intuitiveness*, *adaptability*, and *user-friendliness.* 31.6% (6) were related to *informativeness*, impacting on the sub-dependencies *clarity*, *completeness*, *correctness*, and *consistency.* These sub-dependencies are NFRs that need to be fulfilled in order to achieve the requirements in the higher level, according to the Transparency SIG [84].

While *usability* refers to the quality of presentation and interaction between the user and the system, *informativeness* refers explicitly to the information presented. It is already well-known that usability problems prevent a user from successfully completing a task. These problems may have negative consequences and may result in the abandonment of the use of the system [128]. Participants also reported problems while trying to obtain information from a system. They reported situations where they could not understand the information presented, being by lack of *completeness*, as evidenced in the following quote from one of the participants: *"The system was not explicit about the public transport routes, nor if it was necessary to take more than one line"* or *clarity*, as in *"I tried to identify which transport I could take. The information usually comes a bit muddled. I can not always get the information I need".*

Answering RQ B (Finding 2)

The issues pointed by the participants are related to usability and informativeness. The majority of the issues found are usability-related.

4.5 Advantages and Disadvantages of Explanations

We asked participants to list three advantages and three disadvantages of receiving explanations. The question about the advantages of receiving explanations produced 214 valid responses from 91 participants. As one answer can generate more than one code (as explained in section 4.1), their responses resulted in 231 codes. The disadvantages question was answered by 85 participants, producing 164 valid responses. After coding, their responses yielded 177 codes.

Table 4.1 and Table 4.2 list the advantages and disadvantages, respectively, organized in groups and the underlying categories. Each category is followed by the number of codes and its respective percentage relative to the total number of codes.

NFR	Category	Number of Codes	Percentage (n = 231)
Informativeness Understandability	Facilitate understanding	47	20.35%
	Reduce obscurity	15	6.49%
	Support decision making	12	5.19%
	Others	4	1.73%
Usability	Facilitate the use	25	10.82%
	Guide the use	16	6.93%
	Proficiency in system operation	13	5.63%
	Time efficiency	12	5.19%
	Prevent mistakes	5	2.16%
	Others	2	0.87%
Relationship	Positive feelings	30	12.99%
	Trust	17	7.36%
	User in control	9	3.90%
Auditability	Technical aspects	8	3.46%
	Data transparency	7	3.03%
Others		9	3.90%

Table 4.1: Advantages of explanations according to the participants

By analyzing the participants' responses, we could identify associations with the *usability*, *informativeness*, *understandability* and *auditability* requirements in the Transparency SIG. These associations show how explainability can affect these NFRs and hint at explainability's influence on transparency. To compile all categories of responses in which participants

NFR	Category	Number of Codes	Percentage (n = 177)
Informativeness Understandability	Unnecessary information	48	27.12%
	Impair understanding	24	13.56%
	Add obscurity	3	1.69%
Usability	Impair the use	28	15.82%
	Time consuming	16	9.04%
	Use of computational resources	6	3.39%
Relationship	Negative feelings	35	19.77%
	Loss of control	11	6.21%
Others		6	3.39%

Table 4.2: Disadvantages of explanations according to the participants

expressed their individual thoughts regarding the potential impact of explanations on their relationship with the system, either positively or negatively, we created the *relationship* group.

4.5.1 Informativeness and Understandability

These two qualities are grouped together because their concepts occasionally overlap. Informativeness is related to the quality of the provided information and can also be defined as the ability to provide or convey information in order to help with understanding. To be understandable, this information must be properly stated (in understandable language).

Advantages

Participants mentioned that explanations can be a way to **facilitate the understanding** of a system by conveying information. This understanding can be either specific, related to the current situation or a piece of data; or more general, related to the whole context of the system. Some sample quotes are: (understanding) *"how the software works"*, *"what is being shown"*.

Participants also mentioned that explanations can **reduce obscurity or clarify doubts**. This category encompasses responses where participants explicitly see explanations as a way of mitigating system's obscurity, providing clearer information. Some participants also believe that explanations can **support during decision making**. Both codes could be identified in this participant's statement: *"(Explanations) allow my decisions to be made on the basis of clear information"*. The results illustrate how participants believe that explanations can better convey information and lead to a better understanding of system elements. Explanations may have a favorable impact on a system's *informativeness* and, therefore, on its transparency level.

Disadvantages

Participants believed that explanations could only be a source of **unnecessary information**. They affirmed that explanations may be too lengthy, repetitive, irrelevant or useless. One participant stated: *"Explaining what is already known makes information boring and irrelevant"*.

Participants were also concerned that explanations may actually, rather than facilitate, **impair understanding**. This can be the case if explanations are poorly worded or not given in a language appropriate to the user. This can be noted in the following statement: *"If the explanation comes in a very technical language, the user may not understand it"*. Some participants expressed concerns that explanations could **add obscurity** instead of reducing it. All these categories have a direct impact on understandability since information must be concise and comprehensible in order to be fully understood.

4.5.2 Usability

Advantages

Participants considered receiving explanations as a way to **facilitate the use** of a system: *"(explanations) increase the usability of the device"*. Some participants believe that explanations are a way to **guide the use** of a system, enabling faster familiarization or working as a tutorial. Some participants referred to the possibility that explanations help the user to become **proficient in system operation**, knowing all available features and mastering its operation. Participants also express the belief that explanations may support **time efficiency**, assisting the user in making faster decisions or having more agility while operating the system. Some also mentioned the possibility that explanations might be a way to **prevent users from making mistakes**, supporting them during decision-making situations.

Disadvantages

Participants expressed worry about explanations **impairing the use** of a system. Users were concerned that the user interface becomes polluted with the excess of explanations or notifications, with the interruption of the workflow, and with explanations being too distracting. Some participants expressed concern with the **use of computational resources** when incorporating explanations into a system, consuming storage space, memory and CPU resources, or data volume. The risk that receiving explanations would be **time consuming** or "a waste of time" was also brought up by some participants, as users may have to invest time to consume explanations.

4.5.3 Relationship

Advantages

Participants also expressed their **positive feelings** toward explanations. Some stated that explanations improve the experience of using a system and avoid frustrations. Some stated that receiving explanations may help to **establish a relationship of trust** with the user. A participant affirmed that explanations may help to *"increase confidence in software and its developers"*. Other participants see explanations as a way to **put the user in control**.

This argues in favor of the beneficial effects of explanations for users. Explanations might help users to feel more comfortable and satisfied in using a system. They can also be used to foster a feeling of trust in the system by, for example, revealing the rationale behind a decision. Some participants affirmed that explanations could allow them to decide whether the system decision can be accepted. The responses also indicated the desire of the participants to have control during system use (*locus of control*). This is a phenomenon in psychology [129] that is often taken into account by usability designers to give users a sense of control. It is also an important aspect on the human-computer interaction that impacts on the perceived quality of the system [130].

Disadvantages

Some of the responses reflected **negative feelings** about receiving explanations in a system. Explanations may harm the relationship with the users and trigger negative emotions (e.g., feeling annoyed). A poor relationship with the system may prompt them to stop using it. For instance, participants expressed concerns about explanations being annoying, inconvenient, tiring, or boring.

Participants also stated that receiving explanations may result in **loss of control**. They perceive as negative if they do not have the option to disable explanations when these are not desired. This can be observed in the following statement: *"It would be interesting if I could request the explanation just when I wanted. If it is not requested, makes it inconvenient"*. This conclusion contrasts with the feeling of being in control, previously discussed. It is also supported by the findings presented in subsection 4.3.3, when participants expressed the desire to receive explanations whenever they request them.

4.5.4 Auditability

Some responses address the possibility of explanations leading to a better understanding of the **technical aspects of the system**. This category includes answers in which participants relate explanations with the ability to understand the internal process of the algorithm, as well as being a way to check its behavior. They also consider explanations as a way to find out more about the internal model that the system has created of the user. Some participants also associated explanations to more **data transparency**.

Seven of the 11 codes are about data transparency. Participants expressed concern about what is happening to their data. To mitigate this, explanations could be used to inform users about how their data is being processed and why it is being collected. Some participants explicitly mentioned **transparency** as an advantage. This is yet another strong indicator of the significance of explainability in achieving transparency.

Answering RQ C (Finding 3)

> *Explanations can have an impact on quality aspects that are related to transparency. Impacts on* usability, informativeness, understandability, *and* auditability *were identified.*

4.6 Limitations and Threats to Validity

There were some limitations as a result of the participant selection strategy. To reach more respondents, the questionnaire was made available online. Therefore, participants have likely a good level of technologically literacy to respond to an online survey. This was confirmed when respondents indicated that they had a high level of technological proficiency. Furthermore, we only got responses from Germany and Brazil which also threatens the global generalizability of our findings.

The results might not reflect the needs and perceptions of those who have trouble using software systems and are more likely to have different requirements on explanations. Due to these facts, neither Brazil's nor Germany's overall populations are represented by this sample. Clearly said, it only represents a small portion of the technologically literate end users. Although I do not claim generalizability of the findings beyond the group of participants, they represent the perspectives of a part of this population, which may give us a hint about the overall perspective.

Due to sample size restrictions, some of the findings may have been compromised. Even though there were over a hundred participants, there could have been more to yield more trustworthy results. Some of the conclusions also might have been affected by limitations due to a small sample size. Although over a hundred participants provided a substantial body of responses, an even higher number could have led to more reliable results. However, the analysis process involved labor-intensive coding. Therefore, a sample size of 107 participants is a good place to start when examining this subject. To explore the need for explanations in the context of people with low or no technological literacy, however, more research needs to be done. The participants did, however, express a need for explanations despite having a high level of technological knowledge.

The use of a questionnaire as an instrument, as well as the use of qualitative analysis, causes a mono-method bias. The impact of the questionnaire on this bias is that it is a single source of data and allows only a limited explanation of the findings. Further experiments need to be conducted, where the same questions can be evaluated in the context of a deployed system. Regarding the qualitative analysis, its subjective nature may have affected the findings. To mitigate this, we used *in vivo coding* to adhere closely to the language of the respondents. In addition, during the second cycle, two researchers coded the data into the final categories to ensure consistency. Both coders compared and discussed the results of their coding process in order to reach an agreement on the assigned codes, thus increasing the reliability of the findings.

Another potential threat was the use of a hypothetical scenario to ask questions. It may have resulted in reactions that do not reflect what people would do in the same situation in the real world. To mitigate this, the questions had a high level of psychological realism and the participants faced a scenario that they would most likely encounter in their daily lives [131]. The questions focused on common everyday applications because the goal of this study was to understand the needs of the average end user. As a result of the applications we discussed, another concern is that the findings may not reflect the effect of explanations in more sensitive scenarios (e.g., using software recommendations to make business decisions or decisions with critical/ethical consequences). In fact, the results cannot be generalized to the complex context of such systems. However, the desire for explanations even in the context of lightweight systems suggests that they can be useful in more complex contexts as well.

To guarantee good question wording and good instrumentation, layout guidelines for survey design were followed and pilot-tests were conducted. Yet, the order of questions in the questionnaire may have had an impact on the participants' understanding of whether the questions were about the need to receive explanations in a general context or related to the more specific contexts of previous questions. However, this was used to help participants who may have trouble imagining other scenarios in which they might need explanations.

4.7 Usability through and despite Explanations

There was a time when usability was a secondary concern: when computers were so expensive and used by only a small amount of people who mostly performed very specialized tasks. The popularization of computers shifted this perception, as then all sorts of consumers had access to personal computers. Nowadays, user interfaces are a major way to differentiate products in the market and it is what adds value to a software product [5].

Usability is a quality aspect that assesses how easy user interfaces are to use [132] and is rated as one of the most important NFRs by practitioners [114]. In an empirical study which investigated the importance of quality requirements in the industry, usability was among the top five in importance for product managers, project leaders and developers, for all types of projects in various domains [133]. According to a study by Groen et al. [134], it was one of the most frequently identified software qualities in the feedback of users from app stores. Usability also has an impact on trust since it fosters a better understanding of the contents and tasks that the user must do in order to reach a goal and minimizes the risk of error [135]. According to ISO/IEC 25010 [22], trust is regarded as a critical aspect of user satisfaction and has a significant impact on system acceptability and continuity of use.

In this survey, it was possible to see that, on the positive side, explanations may potentially help to improve key quality aspects such as usability and understandability. However, if not correctly elicited and analyzed, explanations may have a negative impact on the same quality aspects. Therefore, I looked through the participants' responses regarding the disadvantages of explanations and selected the problems they mentioned in order to better understand the negative impact of explanations and how to avoid them. More specifically, my goal was to identify the most prevalent problems and determine whether existing principles could help to avoid these problems.

I found that, by using well-known usability heuristics, as suggested by Nielsen [5], most of the negative effects reported by the participants can be prevented. This finding is summarized in Table 4.3. The perceived disadvantages are classified as a problem class. Then, each class is associated to a usability heuristic that can minimize this problem.

		Negative Effect	Usability Heuristic [5]
Informativeness / Understandability	Unnecessary Information	Explanation: Repetitive, Unnecessary, Inopportune, Lengthy, Lack of Objectivity	Simple and Natural Dialogue
		Information: Unnecessary, Excessive, Redundant, Useless, Irrelevant	
	Hinders Understanding	Confusing, Misinformation	Speak the Users' Language
		Language: Difficult, Complicated, Technical, Incomprehensible	
	Adds Obscurity	Vague, Obscure, Unclear, Ambiguous	
Usability	Impairs the Use	Distractive, Lack of Focus	Simple and Natural Dialogue
		Disrupts Flow, Interruption	
		Excessive Information on Screen, Polluted user interface	
	Time Consuming	Loss of Time, Time Consuming	
		Decreases Dynamism and Speed	

Table 4.3: Negative effects identified in participants' answers and the correspondent usability heuristics that may mitigate them, following Nielsen's usability principles [5]

Usability Heuristic "Simple and Natural Dialogue": The participants mentioned both the possible negative characteristics of the explanations and the undesirable aspects of the information provided by the explanations. Explanations may be repetitive, unnecessary, inopportune, long, and verbose; while information may be redundant, useless, irrelevant, and excessive.

These characteristics are contrary to the idea of presenting only **essential content** to the user, as recommended by the principle of *simple and natural dialogue.* According to this principle, user interfaces should be simplified as much as possible presenting exactly the information the user needs, when it is needed. One of the elements included in this principle is the concept of *less is more.* This concept recommends to identify the information that is really important for the user and that will help the user to perform the task, avoiding unnecessary extra information.

Participants also pointed to some aspects that may negatively impact usability. They mentioned that receiving explanations may be distracting, interrupting, or may pollute the user interface with excessive notifications. In this case, the interface should be kept as clean as possible, avoiding cluttered information. The amount of information will also have an influence on user performance. Any piece of information is something that users will have to look at while navigating, so their performance will be slowed down and they can perceive it as being time consuming.

Usability Heuristic "Speak the Users' Language": Participants also mentioned negative aspects of explanations that may hinder understanding. In their perspective, the information presented may lead to confusion or misinformation, while the language in which the information is presented may be too difficult, complicated, technical, or incomprehensible. These aspects suggest that the language used is not an **accessible language**, something that is recommended by the usability principle of *speaking users' language.*

The identification of an appropriate vocabulary for the interface must take into account aspects such as the needs and expectations of different users, cultural factors and the vocabulary used in the domain. Following this heuristic, it is necessary to avoid the use of technical terms that may be unfamiliar to the user and to pay attention to how the explanation is designed so that there is no ambiguity or vagueness.

4.8 Summary

This study helped to have a first overview of explainability as a quality aspect. By investigating the need for explanations, the first finding was that there are many factors that need to be better understood to design relevant explanations (*Finding 1*). Therefore, there is a need for a more in-depth analysis of how the design of explainability would work in practice and the challenges involved, as well as on how to specify requirements for explanations. Plus, studies need to be conducted to evaluate the effect of explanations in deployed systems.

Although impacts on usability, understandability, auditability, and the relationship with the user could be identified (*Finding 3*), further research is still required to fully comprehend the relationships between explainability and other quality requirements, as well as their interactions and taxonomy. Usability is an essential NFR for system quality and strongly influences factors such as user experience, user satisfaction, and trust. The results indicate that usability is a common source of problems when understanding software (*Finding 2*) and explanations could be a way to mitigate these problems.

This discovery led me to make one assumption: that explainability and usability are inextricably intertwined, and explainability can have both a positive or negative impact on usability. To avoid the negative impact of explanations on usability, I investigated the relationship between explainability and usability in further depth and recommend the use of well-known usability heuristics for reconciling explanations with usability. These heuristics are suitable for preventing the potentially negative effects of explanations in a system without the need for unknown new methods. Therefore:

Assumption A**:** Since explainability and usability are strongly intertwined, existing usability-related practices and techniques could also be useful for explainability.

Summing up, this chapter discusses the results of a survey that aimed at understanding the perception of end users on explainability. It provides a first analysis of explainability as a quality aspect and of its impact on the quality landscape, shedding light on research needs that must be explored. In the next chapter, I go into detail about how the findings up to this point contributed to the identification of research needs, and how these needs translated into research goals, questions, and the theories proposed in this thesis.

5
The Research Methodology

The theories proposed in this thesis have an empirical foundation. An *empirically-based theory* is a theory that uses empirical means through which one may generalize analytically to explain processes or phenomena [136]. Empirically-based theories enable generalization from situations in which statistical generalization is not desirable or possible, such as from case studies, across populations, and from experiments in the social and behavioral sciences, with which experiments in empirical software engineering often share essential features [3].

Theories can support both research and practice (Figure 5.1). In a way, every theory is also practical because it interprets a phenomenon, realizes a phenomenon, or distinguishes one phenomenon from another. For research, theories offer common conceptual frameworks that allow the organization and structuring of facts and knowledge in a concise and precise manner, thus facilitating the communication of ideas and knowledge. In the industry, theories can give input that is useful to decision-making with respect to choice of technologies, approaches, or to help understand and predict phenomena in a given setting [137].

Gregor [137] has classified theories into four main types: *analysis*, *explanation*, *prediction*, and *design and action*. In the context of this research, three of these types are relevant:

1. *Analysis* theories: include descriptions and conceptualizations of *what is*. Some examples are taxonomies, classifications, and ontologies.

2. *Explanation* theories: theories that explicitly explain *why something is or happens*.

3. *Design and action* theories: describe *how to do* things, that is, they are prescriptive.

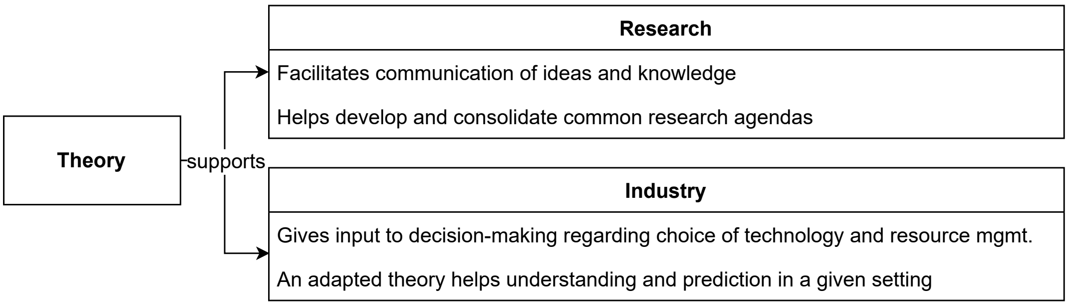

Theory
supports
Research
Facilitates communication of ideas and knowledge
Helps develop and consolidate common research agendas
Industry
Gives input to decision-making regarding choice of technology and resource mgmt.
An adapted theory helps understanding and prediction in a given setting

Figure 5.1: Usefulness of theory for research and industry, according to Sjøberg et al. [3]

In this thesis, I combine elements of two frameworks for my research methodology: the *framework for SE theories* [3] and the *information systems research framework* [4]. The goal of the first framework is to support the development of theories in SE by offering an archetype model, guidelines, and a list of criteria to evaluate theories. The second framework provides a methodology for research execution and emphasizes that information systems research is based on real-world needs that are based on people, organizations, and their existing or planned technologies. The frameworks are used as follows:

Framework for SE theories: A theory that has at least one construct that is unique to SE is said to be an SE theory. An SE theory is supposed to analyze, explain, predict, or prescribe phenomena occurring in SE. In SE, an **[actor] applies [technologies] to perform certain [activities] on a (existing or planned) [software system]**. Sjøberg et al. [3] call these high-level concepts *archetype classes*. To help shape useful theories, the authors propose that the constructs of an SE theory should typically be connected to these archetype classes, their respective subclasses, or any attributes of those classes. Table 5.1 lists how I apply these archetype classes in the context of this work: *software engineers* **[actors]** applying the *definition, conceptual model, knowledge catalogue, process reference model,* and *quality framework* **[technologies]** to perform *activities of the requirements engineering or design process* **[activities]** on an (existing or planned) *explainable system* **[software system]**.

Framework for information systems research: As mentioned in chapter 1, according to Hevner's model [4], research that creates theories and artifacts to achieve a human goal must be integrated with both the application domain and the scientific and domain body of knowledge in order to be relevant and rigorous. This integration process, like the creation of the theories and artifacts themselves, can be iterative, alternating between theory and artifact building, and evaluation and incorporation of gathered knowledge.

Archetype class	Subclasses	In this work
Actor	Individual, team, project, organization or industry	Requirements engineers and software practitioners
Technology	Process model, method, technique, tool or language	Definition, conceptual model, knowledge catalogue, quality framework, reference model
Activity	Plan, create, modify or analyze (a software system)	Activities of the requirements engineering process and design choices
Software system	Classified according to size, complexity, application domain, etc.	Explainable systems (existing or planned)

Table 5.1: Framework for SE theories [3]: Archetypes and subclasses in the context of this work

Figure 5.2 shows how the two frameworks (*framework for SE theories* and the *information systems research framework*) were mixed and applied to this research. I used the archetype classes [3] to help define the environment constructs. The user survey and existing literature assisted me in identifying the practice's needs. The existing body of knowledge helped to identify relevant concepts in the fields of RE, software quality, and explainability to be applied to the research. All studies employ a multi-method approach that triangulates data from multiple sources to augment the reliability of the findings [138]. As a result, a wide range of methods, such as qualitative studies, user studies, quasi-experiments, systematic literature reviews, and quantitative data analysis are used to evaluate the proposed theories and artifacts. The specific methods are described along with the studies.

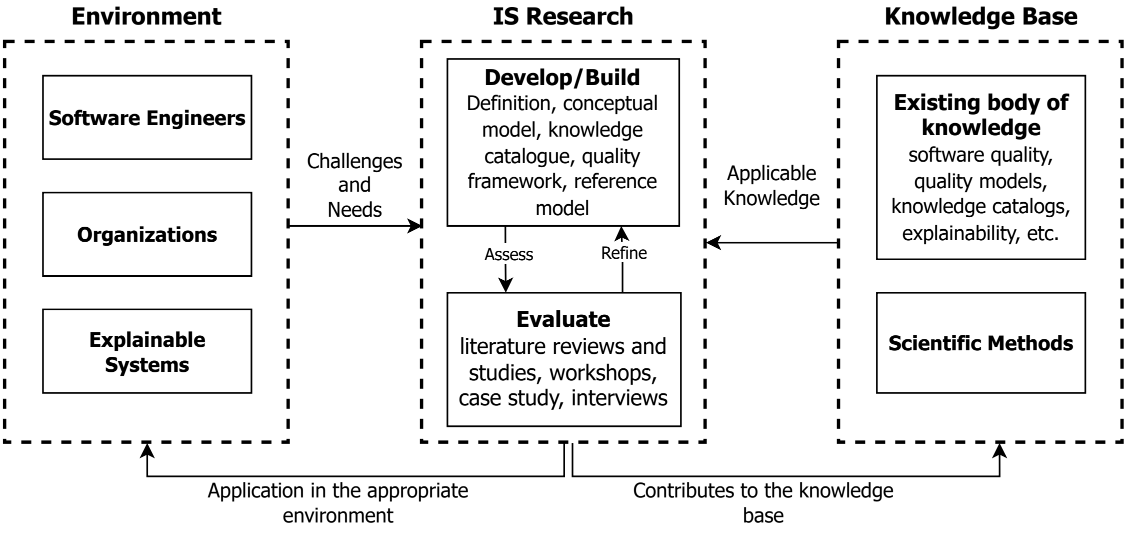

Environment
Software Engineers
Organizations
Explainable Systems
Challenges and Needs
IS Research
Develop/Build
Definition, conceptual model, knowledge catalogue, quality framework, reference model
Assess
Refine
Evaluate
literature reviews and studies, workshops, case study, interviews
Applicable Knowledge
Knowledge Base
Existing body of knowledge
software quality, quality models, knowledge catalogs, explainability, etc.
Scientific Methods
Application in the appropriate environment
Contributes to the knowledge base

Figure 5.2: Information systems research framework applied to the research in this thesis, adapted from [4]

More specifically, the previous chapters' research steps (knowledge gained from the literature on quality aspects and explainability, as well as the survey findings) aided in the identification of four explainability-related research needs that had to be addressed: (N1) The need for a deeper understanding of explainability in the context of software engineering;

N2 The need for an expanded view of the effects of explainability on software quality; N3 The need for recommendations on how to structure the software cycle for the development of explainable systems; N4 The need for guidelines to support practitioners during RE.

These research needs guided subsequent research, motivating the research goals that were pursued. To fulfill these needs, this thesis presents two empirically-based theories and proposes five artifacts to support (primarily) RE for explainable systems. These theories will be explored in detail in chapter 6, chapter 7, and chapter 8.

5.1 Research Needs

While explainability has become an important software quality concern, it was still under-specified [62]. In section 2.2, we saw that there is often terminological confusion regarding explainability and other similar concepts such as interpretability and understandability. This is usual for NFRs because they are typically not as well understood as the functional aspects of the software. Terminological inconsistency frequently exists with relation to the meaning and characteristics of a certain quality aspect, frequently resulting from differing interpretations of the same term [114]. In the case of explainability, a common terminology needs to be investigated in order to create shared understanding and simplify discussion and analysis of this NFR during RE and SE. Therefore, the following research need has been identified:

Research Need 1

> ***N1:*** *The need for a definition of explainability for the context of software engineering.*

Besides their terminological complexity, NFRs are also complex in other ways. In the survey study, I explored whether explainability can help to achieve transparency, and what are the possible advantages and disadvantages of built-in explanations according to the end user perception. It was possible to observe that explainability might not only be a means of achieving transparency and building trust, but is also linked to other important NFRs. The survey results helped to identify possible impacts of explainability on *usability*, *informativeness*, *understandability*, and *auditability*.

However, even though I was able to pinpoint these extremely relevant relationships between explainability and other NFRs, these were just preliminary findings. Going beyond transparency, there was still a need to look into the relationships between explainability and other requirements more thoroughly in order to comprehend their interactions and taxonomies. More importantly, it was crucial to comprehend the ways in which explainability can affect quality and its different quality dimensions (cf. section 3.3). Therefore, the following research need has been identified:

Research Need 2

> ***N2:*** *The need for an expanded view of the effects of explainability on software quality.*

Usability is one of the NFRs that can be either positively or negatively affected by explainability and it is an essential NFR for system quality that strongly influences factors such as user experience, user satisfaction and trust. Therefore, given the importance of usability for software quality, I explored the relationship between explainability and usability in more detail. I found that existing usability engineering techniques can help to prevent negative effects of explanations on usability (cf. section 4.7). I assumed that existing usability-related practices and techniques could be helpful for explainability and could guide software engineers through the development process of explainable systems while also assisting them in establishing requirements that are in line with the defined quality goals, users' needs, and context. Because doing so would "only" require the incorporation of current methods and approaches, it could ensure that the designed explanations adhere to essential usability principles while also avoiding the traditional issue of aligning research and practice. This assumption, however, required to be studied further in order to determine whether these practices are seen as appropriate by software professionals. As a result, the following research need has been identified:

Research Need 3

> ***N3:*** *The need for recommendations on how to structure the software lifecycle for the development of explainable systems.*

Furthermore, each of the discussed NFR interactions emphasizes the importance of paying close attention to the requirements for explainability. For instance, the survey results pointed to the possibility of receiving unnecessary information as a disadvantage. This indicates that software engineers must pay attention to what is to be explained, how it needs to be explained, and if the user really needs to receive an explanation about it. This highlights the value of RE because requirements can not only assist in identifying needs and comprehending a context, but they can also direct and influence how explanations are designed and implemented, affecting whether explainability has a positive or negative effect on the system. As a result, a more in-depth investigation of the specifics of RE for explainable systems is required. The key purpose is to develop more comprehensive guidelines for practice. Therefore, the following research need has been identified:

Research Need 4

> ***N4:*** *The need for guidelines to support practitioners during requirements engineering.*

These research needs guided the research goals, questions, and the scientific methodology employed, which will be further explored in the following sections.

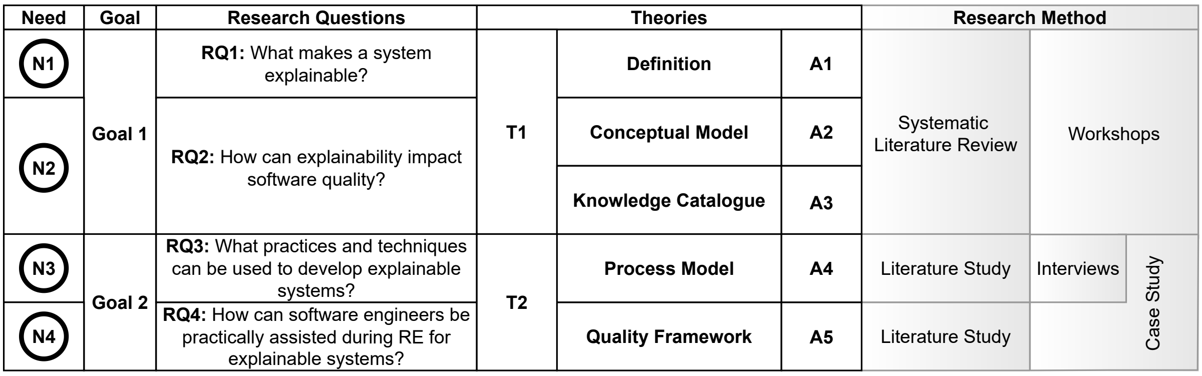
5.2 Research Goals

Understanding explainability better and coming up with strategies to support the development of explainable systems is crucial given its growing relevance. Therefore, the main goal of this thesis is **to guide the requirements engineering process for explainable systems**. This goal is refined into two other goals in accordance with the four identified research needs:

Goal 1 – Define and understand explainability as a quality requirement: addresses N1 and N2, focusing on the theoretical background that is needed to comprehend explainability as a quality requirement. This information makes it easier to communicate, contributes to the development of a shared understanding of explainability, and supports the analysis and specification of explainability requirements.

Goal 2 – Assist practitioners during the requirements engineering process for explainable systems: addresses N3 and N4, investigating how to assist software engineers (mainly) during the requirements engineering process when developing explainable systems.

To achieve the two aforementioned goals, each research need was translated into a corresponding research question, resulting in four main research questions (RQ): **RQ1**–**RQ4**. Next, each research question was investigated using a research method. **RQ1** and **RQ2** were elaborated to achieve **Goal 1** and address N1 and N2, respectively. **RQ3** and **RQ4** were elaborated to achieve **Goal 2** and address N3 and N4, respectively. The answers to these research questions helped to build the two theories ($T1$ and $T2$) and five artifacts ($A1$ - $A5$) that result from this research and address the identified research needs ($N1 - N4$). Figure 5.3 depicts the correlation between research needs, research goals, research questions, resulting theories (and respective artifacts), and the applied research methods.



Need	Goal	Research Questions	Theories			Research Method		
N1	Goal 1	**RQ1:** What makes a system explainable?	T1	**Definition**	A1	Systematic Literature Review	Workshops	
N2		**RQ2:** How can explainability impact software quality?		**Conceptual Model**	A2			
				Knowledge Catalogue	A3			
N3	Goal 2	**RQ3:** What practices and techniques can be used to develop explainable systems?	T2	**Process Model**	A4	Literature Study	Interviews	Case Study
N4		**RQ4:** How can software engineers be practically assisted during RE for explainable systems?		**Quality Framework**	A5	Literature Study		

Figure 5.3: Correlation between research needs, goals, questions, methods, and theories

5.3 Research Questions

RQ1 - The Meaning of Explainability in the Context of Information Systems

To understand the role of explainability for information systems, a first important step is to define explainability in the context of information systems and, consequently, for the software engineering context. Although definitions of explainability exist in other domains, it is essential to define explainability particularly in the context of software engineering. Moreover, to be able to define requirements for explainable systems, it is important to first define what constitutes an explainable system. Hence, **RQ1** was formulated to address N1 and achieve **Goal 1**:

Research Question 1

What makes a system explainable?

Since other disciplines have a long history working on explainability, **RQ1** focuses on harnessing the work of other sciences in the field of explainability to compile a definition that is useful for the area of software and requirements engineering. A definition facilitates the discussion around the topic, contributing to a shared understanding among stakeholders. The answer to this research question resulted in a definition for explainable systems (*A*1).

RQ2 - Impact of Explainability on Software Quality

During requirements engineering, there are key steps that need to be taken toward a successful outcome in terms of quality. One of these steps is to understand how exactly a defined quality goal can impact the quality of a system (e.g., if and how explainability impacts the system's external quality). To do this, it is important to identify inter-dependencies as well as potential conflicts and trade-offs between quality aspects that need to be identified, negotiated, and solved. Therefore, in addition to a definition, the following research question (**RQ2**) was formulated to investigate the potential impact of explainability on software quality, addressing N2 and achieving **Goal 1**:

Research Question 2

How can explainability impact software quality?

As such, this question focuses on understanding the impact of explainability on software quality and on achieving a detailed overview of the quality aspects that might be impacted by explainability. The answer to this research question was documented in two artifacts: a conceptual model (*A*2) and a knowledge catalogue (*A*3). Artifacts such as conceptual models [139], knowledge catalogues [17], and quality models [140] help to analyze and specify quality requirements [141] and are usually adopted by requirements engineers to deal with quality aspects during RE. Such artifacts compile and document knowledge about specific quality aspects (and might also document their interactions with other quality aspects), helping to make decisions and to solve conflicts between requirements. In addition, two challenges in terms of quality aspects can explain their relevance: 1) information concerning quality aspects is rather tacit, distributed, and based on experience [139, 142], which poses difficulties to standardization and reuse; 2) because there are so many different quality aspects, it is challenging to keep track of all the potential interactions and trade-offs.

RQ3 - Development Process of Explainable Systems

When discussing explainability with software engineers, they frequently voiced concerns about the costs associated with developing these systems, as well as the deadlines and novel methodologies that would need to be implemented into the software lifecycle. This anecdotal evidence was combined with the motivation behind N3 and motivated the research on how to proceed in this matter to achieve **Goal 2**: Are new activities really necessary? What practices and techniques are appropriate? These questions are summarized in RQ3:

Research Question 3

> *What practices and techniques can be used to develop explainable systems?*

RQ3 thrives for an overview of activities and practices that can be helpful in the development of explainable systems and should be incorporated into the development process of explainable systems. A literature study and interviews with software engineers were conducted to investigate **RQ3**. *A*4 emerged as a result of this investigation: a process reference model consisting of a set of practices and techniques that can be integrated into the lifecycle for the development of explainable systems.

RQ4 - Explainability Requirements

Because explainability is an emergent NFR, there is little guidance and information to support RE for explainable systems, and guidelines and frameworks that support it are especially lacking. Therefore, in addition to a definition, the following research question (**RQ4**) was formulated to address N4 and achieve **Goal 2**:

Research Question 4

> *How can software engineers be practically assisted during requirements engineering for explainable systems?*

The RE process goes from abstract to concrete, since requirements originate first as an abstract vision and are subsequently refined until the level of concrete and measurable requirements. According to Boehm [140], mapping abstract aspects to quantifiable elements facilitates the evaluation of software products and helps to propose concrete solutions (i.e., operationalizations) for achieving a desired quality goal. The answer to this research question formed a quality framework (*A*5) that consists of three abstraction levels and links the dependencies (abstract), characteristics (concrete), and evaluation paradigms (measurable) for explainability requirements. To construct each of the levels appropriately, **RQ4** is divided in three further research questions, each corresponding to one of the abstraction levels:

Research Question 4.1

> *What external factors influence the requirements for explanations?*

The goal of this research question was to identify external factors that might influence the requirements on explanations. These external factors are "real-world" factors such as cultural values, domain aspects, and project constraints that need to be considered during the RE process [12]. These factors either influence the consideration of explainability as a necessary NFR within a system, or the design choices toward its operationalization.

Research Question 4.2

> *What characteristics of explanations need to be observed in the design of explanations?*

The goal of this research question was to identify the characteristics of explanations tied to design options. These characteristics are an essential part of understanding how high-level requirements can be "translated" into concrete explanations' attributes or aspects such as aspects of demand, content, and presentation.

Research Question 4.3

> *How to evaluate the impact of an explanation on system quality?*

The goal of this research question was to identify existing evaluation methods for measuring or estimating the impact of explanations on the quality of a system, with regard to different quality aspects (e.g., usability, security). It also helped to compile a list of possible strategies for the evaluation of explanations and explainable systems.

5.4 Two Proposed Theories

By studying and investigating the research questions, I devised two theories and five associated artifacts (depicted in Table 5.2).

- The first theory, *T*1, is a mix of two types of theory: on the one hand, it is an *analysis theory* that describes and conceptualizes explainability as a quality requirement, and on the other hand it is an *explanation theory* that explains the role of explainability as a quality requirement based on the collected empirical evidence. *T*1 comprises three artifacts: a definition of explainability (*A*1) a conceptual model (*A*2) and a knowledge catalogue (*A*3).

- The second theory, *T*2, is more practical, being a *design and action theory* aimed at providing not only purely theoretical concepts, but also recommendations and guidelines to more actively support RE for explainable systems. This theory comprises two artifacts: a process reference model (*A*4) and a quality framework (*A*5).

Scientific theories are collections of models that include assertions about how various worldly objects adhere to the models' assumptions. Models are thus frequently essential to theory, are the theory itself, or the two terms are used interchangeably [143]. The artifacts are a way of modeling the theories. The artifacts, in fact, shape the theories. The theories and artifacts contribute to the RE process for explainable systems by assisting in scope delimitation, facilitating shared understanding, and guiding RE activities.

Theory type	Theory	Artifacts
Analysis theory Explanation theory	T1: Explainability in the quality landscape	Definition Conceptual model Knowledge catalogue
Design and action theory	T2: Explainability in practice	Process reference model Quality framework

Table 5.2: Theories and artifacts in this work classified according to theory type

5.4.1 T1: (The Role of) Explainability in the Quality Landscape

*T*1 is a theory that explores the role and impact of explainability on the quality landscape of systems. *T*1 emerged from the answers to **RQ1** and **RQ2**. A systematic literature review (Systematic Literature Review (SLR)) and two workshops were conducted to answer these two research questions. **RQ1** resulted in a definition for explainable systems (*A*1) that aims at facilitating communication between software professionals and delineating the scope of explainable systems. **RQ2** resulted in a conceptual model (*A*2) and a knowledge catalogue (*A*3) that provide knowledge on the role, taxonomy, and effect of explainability, helping to achieve shared understanding and identify requirements and trade-offs between requirements.

5.4.2 T2: Explainability in Practice

*T*2 focuses on guidelines to support explainability in practice, focusing on *how* to develop explainable systems, primarily through RE. *T*2 emerged from the answers to **RQ3** and **RQ4**. Literature studies, interviews, and a case study were conducted to answer these two research questions. As discussed earlier in this chapter, design and action theories are prescriptive. More specifically, the process reference model (*A*4) provides recommendations on how to organize the requirements engineering process, including practices and techniques that can be integrated to the classic activities. The quality framework (*A*5) goes one step further and structures this knowledge to provide a more systematical support for the elicitation and interpretation of requirements, negotiation, and the planning of strategies for verification and validation.

In summary, this chapter discussed the research structure of this thesis, including the connection between research needs, research questions, methods applied, as well as the resulting theories and artifacts. The research that produced the two theories will be presented and discussed in the chapters that follow.

6
Explainability in the Quality Landscape

Since quality aspects are grounded in the real world, they have an abstract essence that has to be well understood in order to be correctly specified. To properly develop a system that satisfies a specific quality aspect (e.g., a usable system, an explainable system), the software team must share a common understanding of the meaning, taxonomy, and value of that quality aspect, and how this quality aspect will impact the quality landscape of the system to be developed. This chapter investigates the meaning and role of explainability in the software quality landscape. I describe a multi-method research approach that consisted of an SLR and two workshops. This research allowed me to answer research questions **RQ1** and **RQ2**, forming theory *T1*, that explores the meaning and role of explainability in the quality landscape, and includes three artifacts that aim to advance the field and foster shared understanding among stakeholders: a definition, a conceptual model, and a knowledge catalogue.

In both traditional and agile development projects, shared understanding is accomplished by eliciting requirements and formalizing them in requirements specifications, stories, upfront test cases, or other design artifacts [26]. Clarity about possibilities and challenges, as well as the resolution of communication problems, are supported by a shared understanding of a given requirement. One way to establish shared understanding among and between

The research described in this chapter was performed in collaboration with two researchers: Wasja Brunotte and Timo Speith. The results were published in [87], on which this chapter is based. As a result, I occasionally use the pronoun "we".

stakeholders and software developers is to ensure that all parties involved perceive the terms and jargon being used in the same way [144]. In the case of quality aspects, this means that all stakeholders should understand the same when they mean, e.g., that the system should be usable or explainable. Therefore, the definition is one of the proposed artifacts. To answer **RQ1**, I harnessed the work of other sciences in the field of explainability to compile a definition that is useful for the area of software and requirements engineering. This definition should contribute to better understand the concept of explainability in practical terms and to provide a shared understanding of the topic for software engineers.

Another way to establish shared understanding is to gain a deeper knowledge of a quality aspect, both theoretically, by understanding the taxonomy of an aspect and its interdependencies, and practically, by understanding its technical implications and trade-offs. So, in addition to the definition and as an answer to **RQ2**, I propose a conceptual model and a knowledge catalogue. Existing works propose to build such artifacts to capture and structure knowledge that is scattered among several sources [17, 139, 81, 85]. The explainability conceptual model presented in this chapter is a taxonomy of explainability's impact on software quality and shows how explainability affects other quality aspects in the quality landscape. This knowledge is supplemented by the knowledge catalogue, which documents the polarity of explainability's influence (if positive or negative) on other quality aspects. Software engineers can use conceptual models and catalogues in negotiations with stakeholders, while discussing the necessary quality requirements for the system [145], and as a support during trade-off analysis. Furthermore, this theoretical foundation enables the knowledge generated about the relevance and role of explainability in software quality to be documented and structured for reuse and knowledge sharing.

6.1 Methodology

The research used a multi-method approach, combining two qualitative techniques to increase the reliability of the data: systematic data collection and qualitative data analysis through an SLR and through workshops. I collaborated with two other academics to carry out the research that is discussed in this chapter. For the data collection, we conducted an interdisciplinary SLR that resulted in a total of 229 papers. The gathered data was coded by using an open coding approach [122] (the same approach used in chapter 4). The resulting codes were analyzed for definitions of explainability (**RQ1**), and for relationships between explainability and other quality aspects (**RQ2**). To validate and complement the findings, a second qualitative method was employed: two workshops with experts. Finally, we framed the obtained knowledge in a model by structuring and grouping the quality aspects impacted by explainability along four dimensions, and documented the polarities in the catalogue. An overview of the research design is depicted in Figure 6.1.

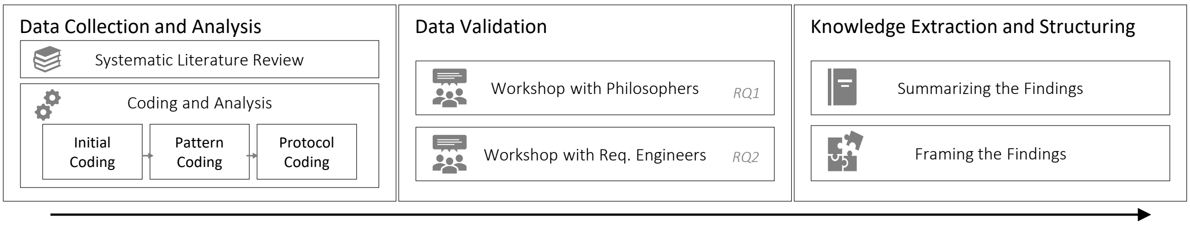

Data Collection and Analysis
Systematic Literature Review
Coding and Analysis
Initial Coding
Pattern Coding
Protocol Coding
Data Validation
Workshop with Philosophers
RQ1
Workshop with Req. Engineers
RQ2
Knowledge Extraction and Structuring
Summarizing the Findings
Framing the Findings

Figure 6.1: Overview of the research design related to RQ1 and RQ2

6.1.1 Systematic Literature Review

To conduct the SLR, guidelines from Kitchenham et al. [146] and Wohlin [147] were followed. The search strategy for the SLR consisted of a manual search followed by a snowballing process. Both the manual research and the snowballing processes comprised two selection phases that entailed observing and applying pre-formulated inclusion and exclusion criteria.

Table 6.1 shows the inclusion (IC) and exclusion (EC) criteria that were applied to filter and select the publications. For a publication to be included, all inclusion criteria had to be met. A publication was rejected if at least one of the exclusion criteria was met. Our selection procedure was divided into two phases. We chose candidate publications in the first round based on title, abstract, and keywords. Plus, we examined the conclusion section in cases where the aforementioned elements did not provide enough information. EC_3 did not apply in this phase, since at this point the text was not fully analyzed. In the second phase of the selection process, we thoroughly examined the content and considered EC_3.

Table 6.1: Inclusion (IC) and Exclusion Criteria (IC)

	Criterion	Description
Inclusion	IC_1	Publications that address one or more of our research questions
	IC_2	Publications that were published between 01/1984 03/2020
	IC_3	Publications that are a peer-reviewed journal, conference, or workshop paper
Exclusion	EC_1	Publications that are not written in English
	EC_2	Tutorials, Proposals, and other non-peer reviewed publications
	EC_3	Publications exploring or proposing raw algorithmic techniques without further discussion about the theoretical background of explainability

The time period investigated in the literature review is from 1984 to 03/2020 (when we started the SLR). We defined 1984 as the start of the search period since it was the year that the first major work on explainability was published (namely, [148]). Despite the fact that 36 years is a long time, we wanted to get as broad an overview of the subject as possible. When it comes to explainability, it is important to remember that this topic was already important in the 1980s [149].

The manual search also took into account sources from other disciplines because the objective was to conduct an interdisciplinary SLR. In addition to computer science, the disciplines of philosophy and psychology were also considered. Philosophy is concerned with the meaning of concepts and terminology, as well as the nature of knowledge and reality, whereas psychology is concerned with human cognitive elements. As a result, both disciplines are essential to explainability. Moreover, both disciplines have decades of experience in explanation research.

To help us identify the most relevant sources, we consulted researchers in the relevant areas. The limitations of this approach include the limited number of disciplines chosen and the sources evaluated, as the list of disciplines and sources was not exhaustive, resulting in a limited and partial picture of the topic. However, we believe that the snowballing strategy allowed us to broaden the scope of our review and find relevant publications from a variety of fields and sources. A detailed list of all the sources and the number of retrieved publications is presented in Appendix B (Table B.1).

Manual Search

During manual searches, after identifying the relevant sources giving the research questions and purposes, an investigator starts by retrieving all the publications in specific sources such as proceedings or journals. This first step in the manual search yield a total of 19595 publications. We filtered these publications by title, abstract, and keywords, selecting a total of 217 publications. After thoroughly examining the content of these 217 publications, 104 publications were considered relevant.

We used Fleiss' Kappa statistics [150] to assess the reliability of the selection process, i.e., to inspect the agreement rate between all three researchers during the inclusion and exclusion of publications. The calculated value of $\kappa = 0.81$ showed an almost perfect agreement [126] between me and my colleagues.

Snowballing

After the manual search, we used snowballing to supplement the search results. The snowballing process includes backward and forward snowballing as described by Wohlin [147]. The references of the chosen publications (in our case, 104 publications from the manual search) and the publications that cite the chosen publications were examined, respectively, through backward and forward snowballing. The review process that we applied is partially based on a grounded theory approach for literature reviews proposed by Wolfswinkel et al. [151]. The goal of using this approach to reviewing the literature is to reach a detailed and relevant analysis of a topic, following some of the principles of grounded theory.

According to Wolfswinkel et al. [151], a literature review is never complete but at most saturated. This saturation is reached when no new insights or categories can be drawn from the data (i.e., the publications that were inspected). We adopted this strategy to determine when to end the snowballing process. While analyzing the last publications of the first snowballing iteration, we noticed that no new insights were being gained, and we were rather finding confirmations of the insights and concepts we already had found before. We observed that the insights were repetitive throughout the beginning of the second iteration. Since the partial second iteration generated no new insights or thoughts, only one snowballing iteration was completed. As a result, we only took into account the publications extracted during the first completed iteration.

This snowballing iteration was also independently conducted by myself and my two colleagues. The backward and forward snowballing process yielded a total of 11173 publications. We filtered these publications by title, abstract, and keywords, resulting in a total of 223 publications. After thoroughly examining the content of these 223 publications, 125 publications were considered relevant. The calculated value of $\kappa = 0.87$ showed, once more, an almost perfect agreement. Overall, combining the selected publications from the manual search and the snowballing, our SLR yielded a total of 229 publications. An overall summary of the number of publications inspected and selected in the different phases of the SLR is shown in Figure 6.2.

Coding and Analysis

We followed an open-coding approach [122] for the qualitative analysis of the papers we selected. This approach consists of up to three consecutive coding cycles. For the first coding cycle, we applied *Initial Coding* [123] to preserve the views and perspectives of the authors in the code. In the second coding cycle, we clustered the initial codes based on similarities, using *Pattern Coding* [124]. This allowed us to group the data from the first coding cycle into categories. We then debated these categories until we came to a consensus as to whether or not they accurately reflected the meaning of the codes. We were able to organize the data using these categories in order to analyze and find patterns in the data.

To answer **RQ2**, we conducted a third coding cycle to further classify the categories into quality aspects. We applied *Protocol Coding* [152] as a procedural coding method in this cycle. For this method, we used a pre-established list of NFRs from Chung et al. [139]. If any correspondence between a category and an NFR was found, we assigned the corresponding code. In the specific cases where we could not assign a corresponding NFR from the list [139] to the

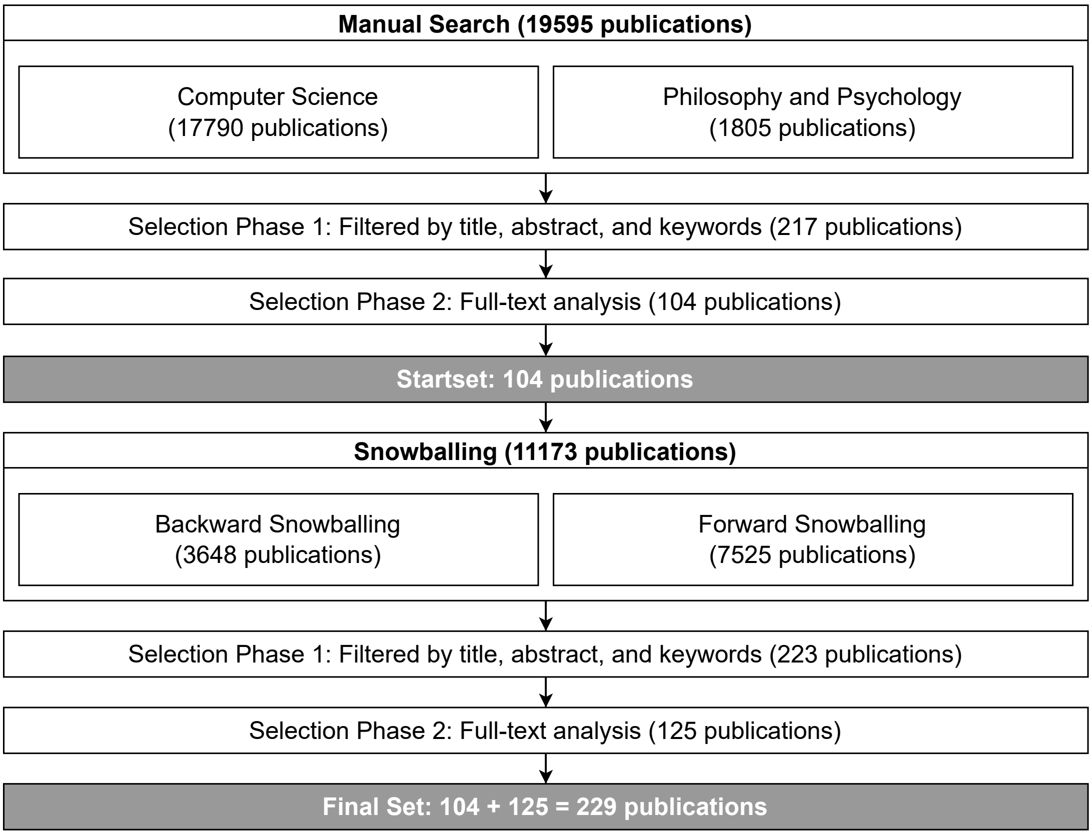

Manual Search (19595 publications)
Computer Science
(17790 publications)
Philosophy and Psychology
(1805 publications)
Selection Phase 1: Filtered by title, abstract, and keywords (217 publications)
Selection Phase 2: Full-text analysis (104 publications)
Startset: 104 publications
Snowballing (11173 publications)
Backward Snowballing
(3648 publications)
Forward Snowballing
(7525 publications)
Selection Phase 1: Filtered by title, abstract, and keywords (223 publications)
Selection Phase 2: Full-text analysis (125 publications)
Final Set: 104 + 125 = 229 publications

Figure 6.2: Overview of the SLR

data, we talked about it and decided on a quality aspect that would adequately describe the concept presented in the text fragment. My two colleagues and I worked independently on all coding processes. We had regular consensus sessions to discuss discrepancies. A list of all codes is available in Appendix B (section B.3).

6.1.2 Data Validation: Two Workshops

We held two workshops to augment and to validate the knowledge gathered during data collection: one exclusively with philosophers and psychologists, and one exclusively with requirements engineers. The structure of the workshop is depicted in Table 6.2. Each of the workshops lasted four hours. A week before the workshop started, we gave each participant in both workshops preparation exercises to work on individually in order to get ready for the discussions. We talked about the categories and other pertinent data that were found during our coding in both workshops. We extracted various definitions of explainability from the literature and divided them into categories for **RQ1**. For **RQ2**, the categories were the identified quality aspects that are related to explainability, as well as the impact of explainability on each of the extracted quality aspects.

Workshop with Philosophers and Psychologists

We validated the data related to **RQ1** in a workshop with philosophers and psychologists (two professors, one postdoc, three doctoral candidates). All scholars except for one doctoral candidate do research in the field of explainability or explanations, one professor and the postdoc even as a focus. Scholars in these fields have a long history of investigating explanations and, thus, explainability. We decided on an open discussion after consulting with experts from these disciplines on the workshop design.

Table 6.2: Structure of the two workshops

Workshop with...	Preparatory Exercises	Workshop Activities
Philosophers and Psychologists	Give a definition for explainability	1. Open discussion on presented categories from the SLR. 2. Compare presented categories with definitions from the pre-workshop task. 3. Discuss important quality aspects related to explainability.
Requirements Engineers	Select quality aspects based on provided scenarios and list of quality aspects. Suggest other quality aspects related to explainability.	1. Enter positive and negative impacts of explainability on other quality aspects. 2. Compare results with the findings from the SLR. 3. Cluster the found quality aspects into groups.

Preparatory Exercise The philosophers and psychologists were asked to write down a definition of explainability that took into account their own background knowledge on the subject. The plan was to collect these definitions prior to the discussion so that we could compare them and avoid bias from our preliminary results and the debate.

Workshop Activities There were three activities in the workshop. In the first activity, we presented the **RQ1**-related categories found in the literature for discussion. We debated whether these categories accurately reflected the participants' perceptions on the meaning of explainability. The idea behind the second activity was to compare the definitions found in the literature with the participants' own definition of explainability, which was submitted prior to the workshop. To reach a consensus, we compared the definitions and identified and discussed the differences. Finally, we briefly discussed the inter-dependencies between explainability and other quality aspects.

Workshop with Requirements Engineers

We validated the data related to **RQ2** in a workshop with requirements engineers (three professors, two postdocs, one practitioner, one doctoral candidate). All three professors do research in the field of RE and two of them also in direct relation to explainability, as do the two postdoctoral researchers. The practitioner works as a product owner in an international company, and the doctoral candidate studies the interaction between RE and agile development. Two RE experts with experience in the topics of NFRs and software quality were consulted about the workshop design, which is also depicted in Table 6.2.

Preparatory Exercise We asked participants to list quality aspects that can be impacted by explainability. To assist them with the task, we created four hypothetical scenarios in which explainable systems should be designed and sent them a list of quality aspects resulting from our coding process (without the identified polarities, to avoid bias). The scenarios took the form of brief stories that described a domain and a business problem connected to the need for explainability. The goal was to help participants better understand situations in which explainable systems might be considered necessary. We asked participants to specify desirable quality aspects for each system based on their expertise, and to analyze how explainability would interact with each of these identified aspects, based on the scenarios (positively or negatively). The four hypothetical scenarios are described in the appendix (cf. Appendix B, section B.4). We also welcomed participants' suggestions for additional quality aspects related to explainability that were not included in our list.

Workshop Activities This workshop also included three activities, each of which lasted about an hour. In the first activity, we gave the participants a list of quality aspects without polarities and asked them to set the polarities. We devised a strict structure for this activity, in which each participant would first define the polarity, then justify their decision, and then all participants could discuss each other's choices at the end of the round. The goal was to spark debate and reach a consensus. In the second activity, we compared the polarities provided by participants to the results of our coding process. Again, we compared the results and held an open discussion to discuss differences and reach consensus. Experts could agree or disagree on all polarities that they had not previously mentioned but were discovered in the literature. In the third activity, we collaboratively clustered the quality aspects based on their relationship and discussed their effects on the system.

Knowledge Structuring

The last step of our research consisted of making sense of and structuring the knowledge collected in the previous stages.

Summarizing the Findings: Definition and Knowledge Catalogue To produce a definition, we used Kohl et al. [62] definition as a basis, and included insights from the literature and the first workshop. This definition was refined through several iterations and discussed with other peers until we arrived at the final version, presented in the next section.

Additionally, we compiled the findings regarding the polarity of the impacts in a knowledge catalogue. Overall, we extracted 57 quality aspects that could be influenced by explainability. We list these quality aspects and how explainability affects them in Figure 6.5. In addition, for each positive and negative influence listed in the catalogue, a representative example from the literature is presented in section 6.3 to demonstrate how this influence may occur. These examples also help to demonstrate the semantic meaning of various quality aspects.

Framing the Findings: Conceptual Model Next, we built a conceptual model to frame the knowledge catalogue. This model illustrates the impact of explainability on several quality dimensions in the quality landscape (see Figure 6.4). During the workshop with requirements engineers, we discussed possible ways to classify the different quality aspects. The participants made some suggestions in this area. We searched the literature and found three promising concepts to help classify the results (more details section 6.3). These three concepts are similar to the suggestions made by workshop participants and helped us develop the conceptual model.

6.2 Explainable Systems: A Definition

The domain of software engineering does not need a mere abstract definition of explainability, but one that focuses on requirements for explainable *systems*. Before requirements engineers can elicit the need for explainability in a system, they have to understand what explainability is in a system context and what makes a system explainable.

Explainability is tied to disclosing information, which can be done by giving explanations. In this line of thought, Köhl et al. hold that what makes a system explainable is the access to explanations [62]. More specifically, the authors propose the following definition for explainable systems: *A system S is explainable by means M with respect to aspect Y of an explanandum X for target group G in context C, if and only if M is able to produce an E in context C such that E is an explanation of X with respect to Y, for G in C.*

Köhl et al.'s definition, however, includes many variables and is still rather complex. The definition also leaves it unclear as to what needs to be explained and who or what will explain it. As a result, the definition presented here focuses on the most significant and noteworthy components discovered during the research. Based on Köhl et al.'s definition, the definitions we found in the literature, and results from our workshop with philosophers and psychologists, we were able to develop a definition of explainability applied to the context of explainable systems that can be adjusted according to project or field of application.

Answering RQ1

> A system S is explainable with respect to an aspect X of S relative to an addressee A in context C if and only if there is an entity E (the explainer) who, by giving a corpus of information I (the explanation of X), enables A to understand X of S in C.

The definition above summarizes the main characteristics of explainable systems in a simpler and more concise way. The proposed definition also focuses on the explanation receiver and their understanding, making it the essence of what makes a system explainable. The literature provided numerous concrete examples for the important characteristics of explainable systems (or *constituent elements*). These characteristics were grouped and summarized in the definition. In other words, the definition focuses on the important constituent elements of an explainable system that are relevant for requirements and software engineers: *aspects* of a system that should be explained, *contexts* in which to explain, the *entity* that does the explaining (the *explainer*), the *information* that constitutes an explanation, and the *addressees* that receive the explanation.

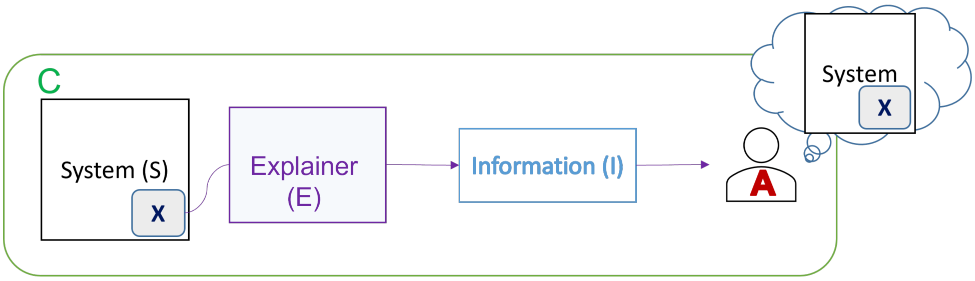


Figure 6.3: The definition of "explainable system", illustrated

In conclusion, the definition provides a summary of the factors that should be considered in explainable systems. Understanding these elements and their meaning is critical for requirements engineers to elicit the appropriate level of explainability for a project and specify the appropriate requirements on explanations. In particular, the use of our definition to support requirements engineering in practice is demonstrated in chapter 7 and chapter 9.

6.2.1 Aspects to be explained

The aspects of a system that must be explained are the object of explanation, and what lies behind the need for explanation. Simply put, it is what needs to be explained. Identifying the aspect to be explained is fundamental so that the information presented is relevant. The following examples of aspects that frequently require explanation were discovered in the literature and discussed in the workshop with philosophers and psychologists: the system in general (e.g., global aspects of a system) [153], and, more specifically, its reasoning processes (e.g., inference processes for certain problems) [154], its inner logic (e.g., relationships between the inputs and outputs) [62], its model's internals (e.g., parameters and data structures) [155], its intention (e.g., pursued outcome of actions) [156], its behavior (e.g., real-world actions) [157], its decision (e.g., underlying criteria) [50], its performance (e.g., predictive accuracy) [158], and its knowledge about the user or the world (e.g., user preferences) [157].

6.2.2 Contexts and Explainers

A context is set by a situation consisting of the interaction between a person, a system, a task, and an environment [159]. Plausible influences on the context are time-pressure, the stakes involved, and the type of system [94].

Explainers refer to a system or specific parts of a system that supply its stakeholders with the needed information. For instance, in the definition of Köhl et al., an explainer is "a means *M* to produce an explanation of some aspect *Y* and does not have to be part of the system *S* but may be provided by someone or something detached from *S*". In short, this definition makes the boundaries of what can be considered an explainable system very abstract and fuzzy. Semantically, the definition presented in this thesis also allows this fuzziness by referring broadly and simply to an *explainer*, which could also imply that explainers do not have to be technical entities or components of the system itself (such as algorithms or hardware elements). In this sense, an *explainer* could also be an intermediate instance, a kind of external *mediator*. This mediator acts as an interface between the system and the addressee, explaining something and helping the addressee to understand the aspect of the system [94]. Although the individual applying the definition may be the one who determines where the boundaries of an explainable system are set, **the focus of this work is on self-explainable systems: systems that are able to explain themselves directly to the intended user or addressee**.

Consider the following example. A patient (addressee) is in a hospital and has been examined by a physician using a medical diagnosis system (context). The medical findings (aspects) are processed by the system and presented directly to the patient in an electronic dashboard. Patients, however, are unable to interpret and understand these findings because they lack the necessary medical domain knowledge. Therefore, the physician intervenes as a mediator and explains the results of the examination to the patient in a way that is understandable for the patient. This system could be considered explainable following Köhl et al.'s definition since it communicates results to the physician, who understands them, and the physician, in turn, is able to explain the output of the system to the patient based on the received explanations.

However, because the focus is on **self-explainable systems**, the system in the example above is only deemed explainable if the physician is the intended end user and addressee for explanations. In the case that the patients are the intended addressees, the system would be considered explainable if the system explains itself directly and comprehensively to the patient (since the patient is the intended end user) without the need for a physician to intervene as a mediator. Thus, if necessary, no medical terminology may be used and the results of the examination must be presented in a way that is clear and understandable to laypersons. In this sense, I consider that the target audience of explanations determines whether or not a system is explainable. If a medical diagnostic system is designed for physicians (who are the end users in this situation), the delivered explanations should be understood by physicians so that they perceive the system as explainable.

6.2.3 Addressee's Understanding

To reiterate: the proposed definition emphasizes the explanation receiver and their comprehension, which is the essence of what renders a system explainable. A vast number of papers in the literature make reference to the addressee's prior understanding as important factor for the success of explainability (e.g., [35, 61, 70, 153, 160]). Framing explainability in terms of understanding provides the benefit of making it measurable, as there are established methods of eliciting a person's understanding of something, such as questionnaires or usability tests [12].

6.3 Explainability as a Means to an End

Explainability can be seen as an *enabler* or as *a means to an end*, since it can be a way to achieve particular quality aspects in a system, which means that other quality goals can be satisfied by specifying and fulfilling explainability requirements.

RQ2 focuses on providing an overview of the quality aspects that may be impacted by explainability, much like the work of Leite and Capelli [84], who looked into the interaction between transparency and other quality aspects. This section will examine the concept of explainability as a means to an end, presenting its influence on the quality landscape through favorable and unfavorable interactions with different quality aspects, and discussing why this is the case. The conceptual model presented in this thesis represents the impact of explainability on quality aspects in the quality landscape, considering three concepts: the concept of the perspective of the persons who demand explanations (i.e.*stakeholders*, presented in section 3.2); the concept of the seven dimensions (presented in section 3.3); the concept of the quality spectrum (presented in chapter 3).

The seven dimensions were partially grouped and formed four dimensions in the conceptual model: 1) *user's needs*, 2) *cultural values & laws and norms*, 3) *domain aspects & corporate values*, and 4) *project constraints & system aspects*. Each dimension is linked to classes of stakeholders: 1) *users*, 2) *regulators & affected parties*, 3) *deployers & domain experts*, 4) *developers*. The quality aspects identified in this study were framed along these four dimensions. During this framing process, it was also possible to identify quality aspects that are present in all dimensions. In particular, there are three quality aspects that form a foundation for the four dimensions: transparency, informativeness, and understandability. This means that, without the influence of explainability on these foundational qualities, many other quality aspects would not be influenced. Furthermore, it was also possible to identify so-called superordinated quality aspects. These superordinated quality aspects are influenced by all other aspects, sometimes being described as the *ultimate goals* of explainability (e.g., trust, system acceptance).

The dimensions and their respective quality aspects (presented in alphabetical order) are illustrated in Figure 6.4. I will go over the catalogue and the model's quality aspects in the paragraphs that follow. To this end, I will analyze them, whenever possible, based on the three concepts that were considered in the conceptual model: the stakeholders involved, the quality dimensions, and the quality spectrum. Some of the findings presented here overlap with and are consistent with those from the survey, further validating the survey findings.

6.3.1 Foundational Qualities

Explainability can influence three quality aspects that have a crucial role: **informativeness**, **transparency** and **understandability**. These quality aspects provide a foundation for all four dimensions, thereby having an influence on the other aspects inside these dimensions. Giving explanations about a system, its processes and outputs can contribute to the informativeness of the system and, consequently, facilitate understanding on many levels [161]*. Furthermore, explanations contribute to a higher system transparency [162]. Informativeness, understandability, and transparency are required, for example, both on an external dimension so that

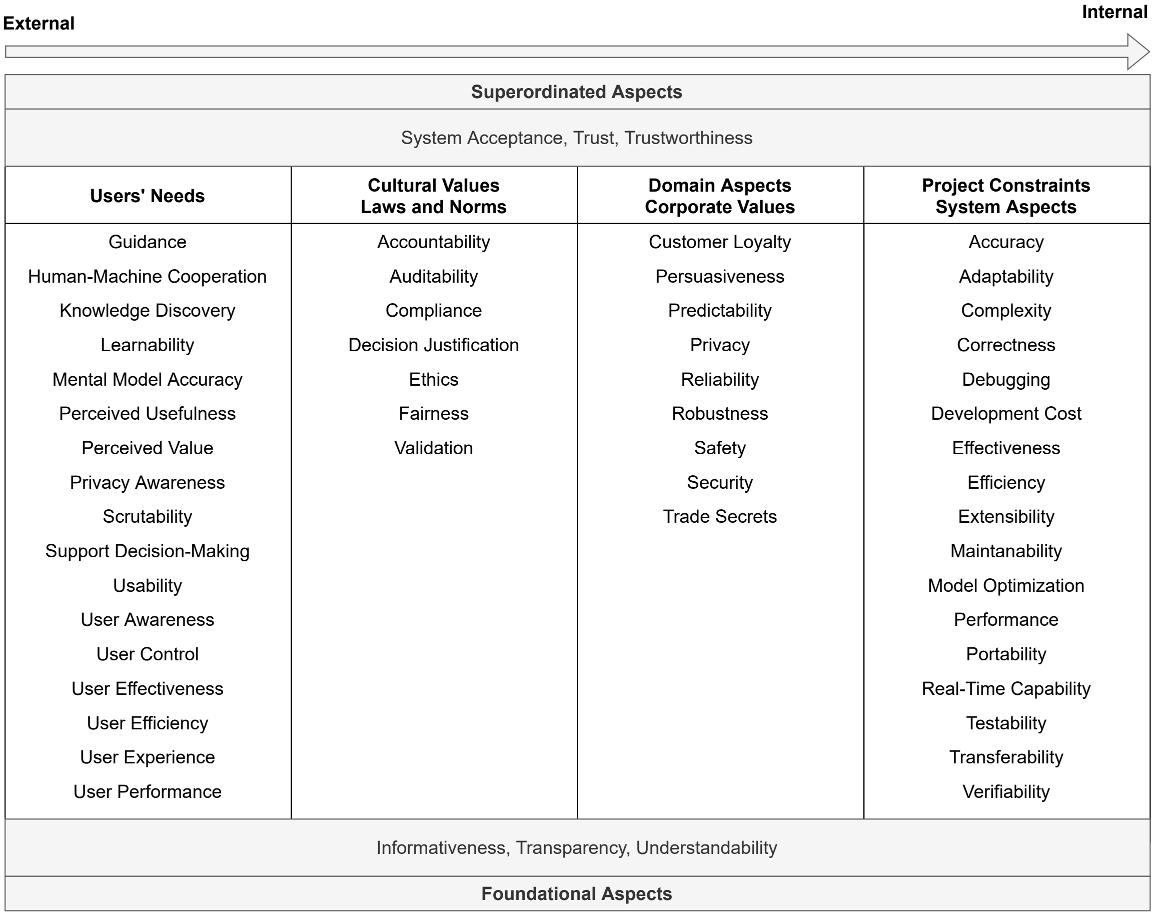
*Informativeness can be seen as the ability to provide useful information. I consider that understandability aspects are tightly related to informativeness (and vice versa), since understandability is the ability to understand the provided information.

users understand the outputs of a system, which may positively impact user experience, and on an internal dimension, where they can contribute to understanding aspects of the code, facilitating debugging and maintainability.

Essentially, by laying this foundation, it is possible to influence all other quality dimensions of the system, thereby influencing a variety of other quality aspects.

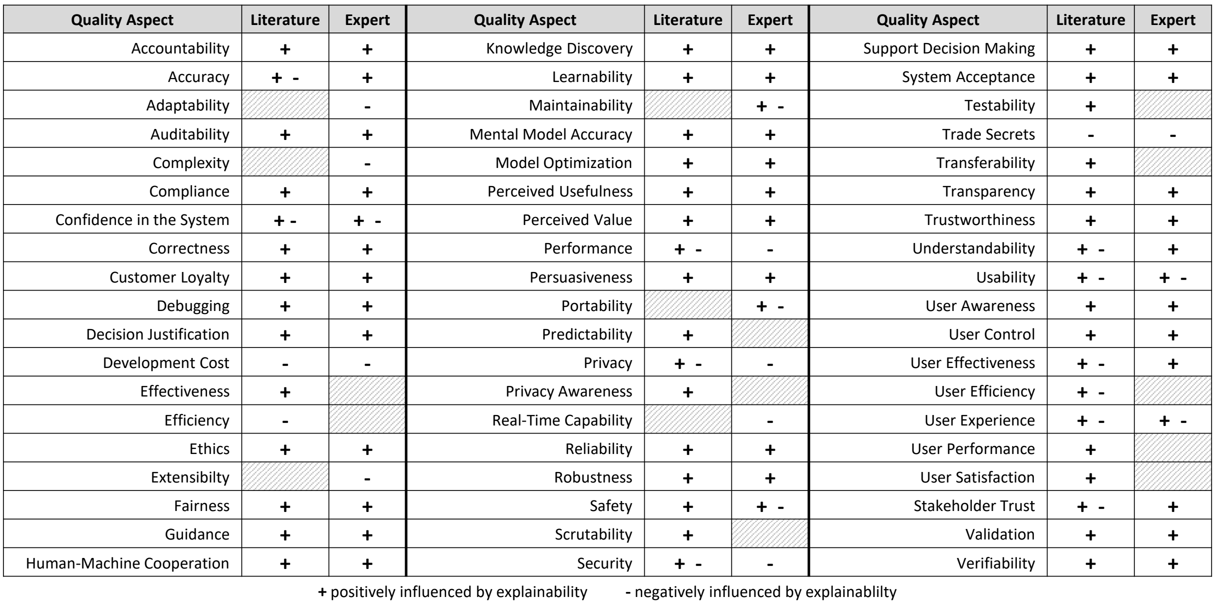
6.3.2 User's Needs

Most papers concerning stakeholders in XAI state *users* as a common class of stakeholders (e.g., [70, 163, 164]). This, in turn, also coincides with the view from requirements engineering, where (end) users also count as a common class of stakeholders [93]. Users take into account recommendations of systems to make decisions [165]. Members of this stakeholder class can be medical doctors, loan officers, judges, or hiring managers. Users can differ in many levels: their level of expertise, in their cultural background, in their goals and physical needs. Hence, different user groups will undoubtedly have different needs, expectations, personal values, and



Superordinated Aspects			
System Acceptance, Trust, Trustworthiness			
Users' Needs	**Cultural Values Laws and Norms**	**Domain Aspects Corporate Values**	**Project Constraints System Aspects**
Guidance	Accountability	Customer Loyalty	Accuracy
Human-Machine Cooperation	Auditability	Persuasiveness	Adaptability
Knowledge Discovery	Compliance	Predictability	Complexity
Learnability	Decision Justification	Privacy	Correctness
Mental Model Accuracy	Ethics	Reliability	Debugging
Perceived Usefulness	Fairness	Robustness	Development Cost
Perceived Value	Validation	Safety	Effectiveness
Privacy Awareness		Security	Efficiency
Scrutability		Trade Secrets	Extensibility
Support Decision-Making			Maintanability
Usability			Model Optimization
User Awareness			Performance
User Control			Portability
User Effectiveness			Real-Time Capability
User Efficiency			Testability
User Experience			Transferability
User Performance			Verifiability
Informativeness, Transparency, Understandability			
Foundational Aspects			

Figure 6.4: The conceptual model illustrating the impact of explainability across different quality dimensions



Quality Aspect	Literature	Expert	Quality Aspect	Literature	Expert	Quality Aspect	Literature	Expert
Accountability	+	+	Knowledge Discovery	+	+	Support Decision Making	+	+
Accuracy	+ -	+	Learnability	+	+	System Acceptance	+	+
Adaptability	▨	-	Maintainability	▨	+ -	Testability	+	▨
Auditability	+	+	Mental Model Accuracy	+	+	Trade Secrets	-	-
Complexity	▨	-	Model Optimization	+	+	Transferability	+	▨
Compliance	+	+	Perceived Usefulness	+	+	Transparency	+	+
Confidence in the System	+ -	+ -	Perceived Value	+	+	Trustworthiness	+	+
Correctness	+	+	Performance	+ -	-	Understandability	+ -	+
Customer Loyalty	+	+	Persuasiveness	+	+	Usability	+ -	+ -
Debugging	+	+	Portability	▨	+ -	User Awareness	+	+
Decision Justification	+	+	Predictability	+	▨	User Control	+	+
Development Cost	-	-	Privacy	+ -	-	User Effectiveness	+ -	+
Effectiveness	+	▨	Privacy Awareness	+	▨	User Efficiency	+ -	▨
Efficiency	-	▨	Real-Time Capability	▨	-	User Experience	+ -	+ -
Ethics	+	+	Reliability	+	+	User Performance	+	▨
Extensibilty	▨	-	Robustness	+	+	User Satisfaction	+	▨
Fairness	+	+	Safety	+	+ -	Stakeholder Trust	+ -	+
Guidance	+	+	Scrutability	+	▨	Validation	+	+
Human-Machine Cooperation	+	+	Security	+ -	-	Verifiability	+	+

+ positively influenced by explainability   - negatively influenced by explainablilty

Figure 6.5: The knowledge catalogue for explainability: how explainability impacts other quality aspects.

preferences regarding explainability. As a result, quality can also be perceived differently by each individual.

This dimension occupies a more external position in the spectrum. The quality aspects associated with users are mostly external. To be more precise, they depend mostly on the expectations and the needs of the person who uses the system.

On a general level, the **user experience** can both profit and suffer from explainability. Explanations can foster a sense of familiarity with the system [166] and make it more engaging [167]. In this case, user experience profits from explainability. On the other side, explanations can cause emotions such as confusion, surprise [168], and distraction [160], harming the user experience. Furthermore, explainability has a positive impact on the **mental-model accuracy** of involved parties. By giving explanations, it is possible to make users aware of the system's limitations [158], helping them to develop better mental models of it [168]. Explanations may also increase a user's ability to predict a decision and calibrate expectations with respect to what a system can or cannot do [158]. This can be attributed to an improved **user awareness** about a situation or about the system [15]. Furthermore, explanations about data collection, use, and processing allow users to be aware of how the system handles their data. Thus, explainability may be a way to improve **privacy awareness** [70]. Explainability can also positively impact the **perceived usefulness** of a system or a recommendation [169], which contributes to the **perceived value** of a system, increasing users' perception of a system's competence [170] and integrity [171] and leading to more positive attitudes toward the system [172]. Finally, all of this shows that explainability can certainly positively impact **user satisfaction** with the system [168].

Explainability can also influence the **usability** of a system. On the positive side, explanations can increase the ease of use of a system [69], lead to more efficient use [15], and make it easier for users to find what they want [173]. On the negative side, explanations can overwhelm users with excessive information [174] and can also impair the user interface design [12]. Explanations can help to improve **user performance** on problem solving and other tasks [171]. Another plausible positive impact of explainability is on **user effectiveness** [175]. With explanations, users may experience greater accuracy in decision-making by understanding more about a recommended option or product [176]. However, user effectiveness can also suffer when explanations lead users to agree with incorrect system suggestions [13]. **User efficiency** is another quality aspect that can be positively and negatively influenced by explainability. Analyzing and understanding explanation takes time and effort [177], possibly reducing user efficiency. Overall, however, the time needed to make a judgment could also be reduced with complementary information [175], increasing user efficiency. Furthermore, explanations may also give users a greater sense of **control**, since they understand the reasons behind decisions and can decide whether they accept an output or not [61]. Explainability can also have a positive influence on **human-machine cooperation** [35] since explanations may provide a more effective interface for humans [178], improving interactivity and cooperation [85], which can be especially advantageous in the case of cyber-physical systems.

Explainability can have a positive influence on **learnability**, allowing users to learn about how a system works or how to use a system [176]. It may also provide **guidance**, helping users in solving problems and educating them about product knowledge [179]. As these examples illustrate, explanations can **support decision-making** processes for users [69]. In some cases, this goes as far as enabling **scrutability** of a system, that is, enabling a user to provide feedback on a system's user model so that the system can give more valuable outputs or recommendations in the future [69]. Finally, explainability can help **knowledge discovery** [61]. By making the decision patterns in a system comprehensible, knowledge about the corresponding patterns in the real world can be extracted. This can provide a valuable basis for scientific insight [158].

6.3.3 Cultural Values & Laws and Norms

Although I distinguished *Cultural Values* and *Laws and Norms* as two separate dimensions and Langer et al. [94] did the same for the related stakeholder classes (*regulators* and *affected parties*), the concepts were combined into one dimension because they are complementary and influence each other. The dimensions form a kind of symbiosis since, e.g., legal foundations are grounded, among others, on the basis of the cultural values of a society. This combination of dimensions that form a symbiosis also happen in subsection 6.3.4 and subsection 6.3.5.

The same symbiosis concept applies for the related stakeholders: Regulators commonly envision laws for people who could be affected by certain practices. In other words, regulators stipulate legal and ethical norms for the general use, deployment, and development of systems. This class of stakeholders occupies an extraordinary role, since they have a "watchdog" function concerning the systems and their use [94]. Regulators can be ethicists, lawyers, and politicians, who must have the know-how to assess, control, and regulate the whole process of developing and using systems.

The restrictive measures by regulators are necessary, as the influence of systems is constantly growing and key decisions about people are increasingly automated – often without their knowing [94]. Affected parties are (groups of) people in the scope of a system's impact. They are stakeholders, as for them much hinges on the decision of a system. Patients, job or loan applicants, or defendants at court are typical examples of this stakeholder class [94].

In this dimension, cultural values represent the ethos of a society or group and influence the need for specific system qualities and how they should be operationalized [102, 101]. These values resonate in the conception of laws and norms, which enforce constraints that must be met and granted in the design of systems. Explainability can influence key aspects on this dimension.

A clear attribution to the position in the spectrum is not possible. Rather, the quality aspects appear to occupy a hybrid position. Whether or not they are present is rather dependent on general conventions (e.g., legal, societal) that are in place.

On the cultural side, explanations can contribute to the achievement of **ethical** decision-making [180] and, more specifically, ethical AI. On the one hand, explaining the agent's choice may support ensuring that ethical decisions are made [61]. On the other hand, providing explanations can be seen as an ethical aspect itself. Furthermore, explainability may also contribute to **fairness**, enabling the identification of harms and decision biases to ensure fair decision-making [61], or helping to mitigate decision biases [158].

On the legal side, explainability can promote a system's **compliance** with regulatory and policy goals [181]. Explaining an agent's choice can ensure that legal decisions are made [61]. A closely related aspect is **accountability**. We were able to identify a positive impact of explainability on this quality that occurs when explanations allow entities to be made accountable for a certain outcome [182]. In the literature, many authors refer to this as *liability* [182] or *legal accountability* [183].

In order to guarantee a system's adherence to cultural and legal norms, regulators and affected parties need several mechanisms that allow for inspecting systems. One quality aspect that can help in this regard is **auditability**. Explainability positively impacts this quality aspect, since explanations can help to identify whether a system made a mistake [13], can help to understand the underlying technicalities and models [156], and allow users to inspect a system's inner workings to judge whether it is acceptable or not [184]. In a similar manner, **validation** can be positively impacted, since explainability makes it possible for users to validate system knowledge [176] or assess if a recommended alternative is truly adequate for them [69]. Exactly the latter aspect is essential for another quality that is helped by explainability, namely, **decision justification**. On the one hand, explanations are a perfect way to justify a decision [182]. On the other hand, they can also help to uncover whether a decision is actually justified [50].

6.3.4 Domain Aspects & Corporate Values

People who decide where to employ certain systems (e.g., a hospital manager decides to bring a special kind of diagnosis system into use in her hospital) are deployers. Other possible stakeholders in this dimensions are specialists in the domain, known as domain experts [70]. People have to work with the deployed systems and, consequently, new people fall inside the range of affected people [94].

This dimension is shaped by two aspects: 1) the corporate values and vision of an organization [105], and 2) the domain aspects that shape a system's design since explanations may be more urgent in some domains as in others.

This dimension is more internal in the spectrum, as it includes quality aspects related to the domain or the values of the corporation or team. In general, the incorporation of such aspects has an architectural impact on the design of a system.

Explainability supports the **predictability** of a system by making it easier to predict a system's performance correctly and helping to determine when a system might make a mistake [185]. Furthermore, explainability can support the **reliability** of a system [153]. In general, explainability supports the development of more **robust** systems for critical domains [186]. All of this contributes to a positive impact on **safety**, helping to meet safety standards [61], or helping to create safer systems [187]. On the negative side, explanations may also present safety risks by distracting users in critical situations.

Explanations are also seen as a means to bridge the gap between perceived **security** and actual security [154], helping users to understand the actual mechanisms in systems and adapt their behavior accordingly. However, explanations may disclose information that makes the system vulnerable to attack and gaming [49]. Explainability can also influence **privacy** positively, since the principle of information disclosure can help users to discover what features are correlated with sensitive information that can be removed [188]. By the same principle, however, privacy can be hurt since one may need to disclose sensitive information that could jeopardize privacy [15]. Explainability can also threaten model confidentiality and **trade secrets**, which companies are reluctant to reveal [70].

Explainability can contribute to **persuasiveness**, since explanations may increase the acceptance of a system's decisions and the probability that users adopt its recommendations [69]. Furthermore, explainability influences **customer loyalty** positively, since it supports the continuity of use [166] and may inspire feelings of loyalty toward the system [173].

6.3.5 Project Constraints & System Aspects

Individuals who design, build, and program systems are, among others, developers. They count as stakeholders, as without them the systems would not exist in the first place. Generally, representatives of this group have a high expertise concerning the systems and a strong interest in creating and improving them.

This dimension is shaped by two aspects: project constraints and system aspects. The project constraints are the non-technical aspects of a system [113], while system aspects are more related to internal aspects of the system, such as performance and maintainability.

The quality aspects framed in this dimension are almost entirely internal in the classical sense, since they correspond to the most technical aspects of a system or the process through which the system is built.

Explainability can have both a positive and negative impact on **maintainability**. On the one hand, it can facilitate software maintenance and evolution by giving information about models and system logic. On the other hand, the ability to generate explanations requires new components in a system, hampering maintenance. A positive impact on **verifiability** was also identified, when explanations can work as a means to ensure the correctness of the knowledge base [176] or to help users evaluate the accuracy of a system's prediction [15]. **Testability** falls in the same line, since explanations can help to evaluate or test a system or a model [61]. Explainability has a positive influence on **debugging**, as explanations can help developers to identify and fix bugs [50]. Specifically, in the case of ML applications, this could enable developers to identify and fix biases in the learned model and, thus, **model optimization** is positively affected [189]. Overall, all these factors can help increase the **correctness** of a system, by helping to correct errors in the system or in model input data [182].

The overall **performance** of a system can be affected both positively and negatively by explainability. On the one hand, explanations can positively influence the performance of a system by helping developers to improve the system [35]. In this regard, explainability positively influences system **effectiveness**. On the other hand, explanations can also lead to drawbacks in terms of performance [177] by requiring loading time, memory, and computational cost [12]. Thus, as the additional explainability capacities are likely to require computational resources, the **efficiency** of the system might decrease [50]. Another quality that is impacted by explainability is **accuracy**. For instance, in the ML domain, the accuracy of models can benefit from explainability through model optimization [189]. On the negative side, there exists a trade-off between the predictive accuracy of a model and explainability [50]. A system that is inherently explainable, for instance, may have to sacrifice predictive power in order to be so [155]. Explainability may have a negative impact on **real-time capability** since the implementation of explanations could require more computing power and additional processes, such as logging data, might be involved.

Adaptability can be negatively impacted, for example, if lending regulations in a financial software have changed and an explanation module in the software is also affected. Next, assume that a new module should be added to a system. The quality aspect involved here is **extensibility**, which in turn is negatively impacted by explainability. Merely adding the new module is already laborious. If the explainability is also affected by this new module, the required effort increases again. Depending on the architecture of the software, it may even be impossible to guarantee the system's explainability. Explanations affect the **portability** of a system as well. On the negative side, an explanation component might not be ported directly because it uses visual explanations, but the environment to which the system is to be ported to has no elements that allow for visual outputs. On the positive side, explainability helps **transferability** [190]. Transferability is the possibility to transfer a learned model from one context to another (thus, it can be seen as a special case of portability for ML applications). Explanations may help in this regard by making it possible to identify the context from and to which the model can be transferred [190].

Overall, the inclusion of explanation modules can increase the **complexity** of the system and its code, influencing many of the previously seen quality aspects. In particular, as an explainability component needs additional development effort and time, it can result in higher **development costs** [62].

6.3.6 Superordinated Qualities

There are aspects that are not specifically tied to any of the dimensions, but to all at the same time since they are commonly seen as some kind of superordinated goals of explainability. For instance, organizations and regulators have been lately focusing on defining core principles (or "pillars") for responsible or **trustworthy** AI. Explainability has been often listed as one of these pillars [70]. Overall, many of the quality aspects found in the literature contribute to trustworthiness. For instance, explanations can help to identify whether a system is safe and whether it complies to legal or cultural norms. Ideally, **trust** in a system originate solely from trustworthy systems. Although one could trust an untrustworthy system, this trust would be unjustified and inadequate. For this reason, explainability can both contribute to and hurt trust in a system [154, 15]. Regardless of the system's actual trustworthiness, bad explanations can always degrade trust [154]. Finally, all of this can influence the system's **acceptance**. Many quality aspects (e.g., trustworthiness, user experience) can lead to system acceptance and explainability is key to this [157].

6.4 The Double-Edged Sword Effect

One aspect that conveys the complexity and the challenges of dealing with quality aspects or NFRs is that they can be *interacting*. This means that the attempts to achieve one NFR can hurt or help the achievement of another [97]. The purpose of the study presented in this chapter was to better comprehend the role and influence of explainability in the quality landscape. The study showed that explainability can both help or hurt the achievement of other quality aspects, indicating the existence of a *double-edged sword effect*. It can act both as a synergistic or as an antagonistic quality aspect.

This double-edged sword effect was also discussed by Mairiza and Zowghi [17], who named the antagonistic effect as conflict. They identified three kinds of conflicts between NFRs: *absolute conflict*, when two NFRs are always in conflict; *relative conflict*, when a pair of NFRs are sometimes in conflict depending on factors such as stakeholders agreement and the architectural decision to operationalize the NFR; and *never conflict*, when a pair of NFRs never conflict. Having conflicts between NFRs mean that fulfilling one requirement can affect another's achievement. The conflict between usability and security is often pointed as a classic example of such conflicts [191, 192, 193]. A system module may require security mechanisms, which may increase its complexity and, consequently, make the interaction with the system more complex [194].

Explainability's double-edged sword effect indicates a general state of relative conflict between explainability and other quality aspects. The impact is determined by how explainability is refined to more fine-grained requirements and how they interact with other NFRs. On the positive side, explanations may potentially aid in the achievement of many crucial quality aspects. However, if explanations are not properly elicited and analyzed, they may have a negative impact on the same quality aspects. As a result, while considering the integration of explanations in a system, the intention may be to improve transparency, so improving system use and comprehension, but it may have the opposite effect. The outcome will be heavily influenced by the defined requirements as well as the design decisions made to operationalize these requirements. This highlights the need for careful requirements analysis and design.

Furthermore, explainability can exhibit an impact on nearly all traditional quality aspects that can be found in the ISO 25010 [22]: performance, efficiency, usability, reliability, security, maintainability, and portability. The significance of explainability must therefore be further acknowledged. The conceptual model and the knowledge catalogue are two important artifacts that can be used as checklists both for elicitation and trade-off analysis. They can help software engineers be aware of and avoid conflicts between quality aspects and choose the best strategies for achieving the desired quality outcomes.

Answering RQ2

> *Explainability has the potential to influence all dimensions of the quality landscape. Positive and negative influences on 57 different quality aspects could be identified. The impact of explainability on the quality landscape is depicted in a conceptual model (Figure 6.4), and the knowledge catalogue (Figure 6.5) documents the polarity of the impacts.*

The answer to **RQ2** brings to light the importance of explainability as a quality aspect and how it can impact the entire quality landscape. But why is that so? What makes explainability have this effect on system quality? In the next section, I will explore this matter to better understand the role of explainability.

6.5 Understanding the Role of Explainability

As information system is “an integrated set of components for collecting, storing, and processing data and for providing information, knowledge, and digital products” [195]. Therefore, as the name implies, information systems deal with **processing and providing information** to other entities (such as users or even other information systems).

Since explainability is a quality aspect linked to information, and information is a central aspect of information systems, explainability has a strong interaction and coupling with the quality of these systems, as we could see in this chapter. And that is precisely why the relationship between explainability, informativeness, understandability, and usability is even stronger, since there is a strong interdependency betweeen these four aspects.

I will get to usability later on, since the relationship between explainability, informativeness, and understandability is more evident and easier to delineate. As discussed in chapter 4, explainability is a way of improving the understandability of a system by conveying quality information (informativeness), thus influencing its understandability. Therefore, a system could be described as explainable if it is informative and understandable. The reverse is also true: a system can be considered understandable and informative if it is explainable. Since explainability is also about providing information, it is difficult to draw the boundary between explainability, informativeness, and understandability. Therefore, I argue that informativeness and understandability can be considered as components or attributes of explainability (cf. Figure 6.6).

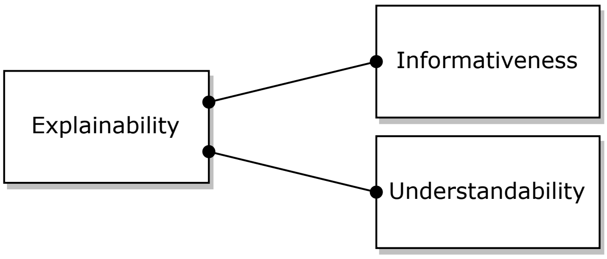


Figure 6.6: The strong interplay between explainability, informativeness, and understandability

The relationship between explainability and usability is more difficult to distinguish and occasionally causes confusion. If a system provides information that helps a user understand something, is the system explainable or is this just good usability? For example, if a navigation software application **informs users** via a symbol on the screen that the route is very congested today, is this system explainable or is this simply good usability? If a system is **easy to operate** because it provides information that assist the user in better operating the system, is this just good usability or is this system explainable? These questions all lead back to the same question: What makes a system explainable?

If we consider the definition given before, in simple terms, a system is explainable if the system, by providing a piece of information through an explanation, helps a given user to understand the system (or an aspect of it) in a particular context [87]. But where are the boundaries? Since the essence of information systems is to provide information: Are all information systems explainable because they provide information about aspects of the system? Therefore, to facilitate this delimitation of boundaries, one way to determine if a system is explainable is by assessing the explainability of the system in terms of certain attributes.

To understand better this concept, we can start by drawing a parallel with usability and ask the same question as before: What makes a system usable? According to Nielsen, usability is not a single, one-dimensional property of a user interface [5]. Usability has many dependencies and is traditionally associated with five quality (or usability) attributes: learnability, efficiency, memorability, low error rate, and satisfaction. These attributes are in essence, again, related quality aspects. By defining the abstract concept of usability in more concrete and measurable components (the attributes) it is possible to systematically approach usability in a system [5][†]. This is the same logic used in the Transparency SIG [84] (cf. subsection 2.3.3).

A system is considered usable if, among others, it is easily learnable, has error-proofing mechanisms, and assists the user in completing their tasks more efficiently [5]. Following the same train-of-thought, a system can be considered explainable if attributes of (or quality aspects related to) explainability are met. To help establish the attributes of explainability, I consider the work of Kahn et al. [7]. The authors propose a reference model for information quality (informativeness). According to this model, informativeness has many quality attributes that are grouped into four aspects: soundness, usefulness, dependability, and usability (Table 6.3).

Table 6.3: Attributes of information quality, according to Kahn et al. [7]

Information Quality Aspect	Dimensions/Attributes	Definition. "The extent to which...
Soundness	Completeness	information is not missing and is of sufficient breadth and depth for the task at hand"
	Concise Representation	information is compactly represented"
	Consistent Representation	information is presented in the same format"
	Free-of-Error	information is correct and reliable"
Useful Information	Appropriate Amount	the volume of information is appropriate for the task at hand"
	Relevancy	information is applicable and helpful for the task at hand"
	Understandability	information is easly comprehended"
	Intepretability	information is in appropriate languages, symbols, and units, and the definitions are clear
	Objectivity	information is unbiased, unprejudiced, and impartial"
Usable Information	Believability	information is regarded as true and credible"
	Accessibility	information is available, or easily and quickly retrievable"
	Ease of Manipulation	information is easy to manipulate and apply to different tasks"
	Reputation	information is highly regarded in terms of its source and content"
	Value-Added	information is beneficial and provides advantages from its use"
Dependable Information	Security	information is restricted appropriately to maintain its security"
	Timeliness	information is sufficiently up-to-date for the task at hand"

Since explainability and informativeness are strongly intertwined, I recommend that these attributes also be considered for explainability. So by defining explainability in these more concrete attributes that focus on providing good information, it is possible to systematically approach explainability in a system. After all, if an "explanation is an information about an aspect of the system", the explainability of a system can be assessed, among others, in terms of information completeness, in terms of its relevance for the task, or in terms of its accessibility.

Therefore, it is difficult to separate explainability, informativeness, understandability, and usability from each other, since they are all interrelated and influence one another. This strong interaction clarifies the role of explainability in the software landscape and helps to understand why explainability affects so many distinct quality aspects and, in essence, all dimensions of the quality landscape. This is because explainability impacts quality aspects

[†] This notion of decomposing one quality aspect in other related quality aspects or in concrete requirements is a common practice in RE and will be explored in more detail in chapter 8.

that are inextricably linked with the interface between the system and the human, as well as with human-machine communication, and it influences the very essence of communication: information.

Summing up, in this chapter I presented three artifacts that can contribute to better understanding the role of explainability in the quality landscape. Definitions, conceptual models and catalogues can help to abstract, understand, and communicate information. These artifacts may be used to turn explainability into a positive catalyst for other essential system qualities in modern systems. I also discussed the role of explainability and why it has such a broad impact on system quality. Explainability can have a positive impact on the quality landscape, but it is highly dependent on the defined requirements and the design choices made to operationalize these requirements. This highlights the need for careful requirements analysis and design. Furthermore, in order to accommodate the necessary steps toward creating explainable systems and to foster the positive impact of explainability on the quality landscape, the RE community needs to investigate what kinds of activities, methods, and tools need to be incorporated into the software lifecycle.

7

A Process Reference Model for Explainability

New quality requirements constantly bring with them new obstacles and doubts about how to proceed or adapt system development. Recent study has focused heavily on the importance of explanations and how they should be presented in a software application. (cf. [196, 94, 197]). However, it was yet unclear how explainable systems can and should be developed. That is, how does a software process look like that specifically supports the integration of explanations? Are existing processes, activities, methods, practices, and techniques sufficient for the development of these systems?

In this chapter, I discuss a study that aimed to provide more in-depth insights into how to develop such systems. This study combined the findings of a literature study, in which 80 papers were examined for activities, practices, and techniques for developing explainable systems, with the feedback from 19 practitioners from an interview study. These insights were synthesized into six practices that are crucial for the creation of explainable systems: 1) vision definition, 2) stakeholder analysis, 3) back-end analysis, 4) trade-off analysis, 5) explainability design, and (6) evaluation. Furthermore, the importance of user-centered practices could be

The research described in this chapter was performed in collaboration with three researchers: Merve Balci, Jil Klünder, and Kurt Schneider. The results were published in [198] and [199], on which this chapter is based. As a result, I occasionally use the pronoun "we".

observed since all participants advocated for a user-centered development strategy for the development of explainable systems, confirming a previous assumption (*A*, cf. section 4.8).

7.1 Methodology

The research design followed a multi-method approach that combined a rapid literature review and an interview study. While the literature study strives for a concise overview of already existing approaches to develop explainable systems, the interview study was intended to collect feedback on these existing ideas, which were synthesized in six practices. The overall goal of the research presented in this chapter is to *provide recommendations on activities, practices, and associated techniques to support the development of explainable systems*, answering **RQ3**.

7.1.1 Literature Study

Due to time constraints, we chose to conduct a rapid literature review rather than an SLR. Rapid reviews are a form of knowledge synthesis in which components of the systematic review process are simplified or omitted to produce information in a timely manner [200, 201]. We performed this review based *in parts* on the guidelines provided by Kitchenham et al. [202], but deviated to some extent given our constraints. This limits the generalizability of our findings when compared to more comprehensive reviews, but we deemed this type of literature review adequate to reach our research goal. Nevertheless, all steps required to get an overview of existing literature were performed, namely 1) definition of the search string, 2) definition of inclusion and exclusion criteria, 3) selection of the database(s), 4) definition of the termination criterion, 5) execution, and 6) data analysis. Details on the definition of the search string, the inclusion and exclusion criteria, databases, and the termination criterion are presented in Appendix C.

Execution

The literature search lead to 446 publications that were selected based on their titles. These papers were subjected to the exclusion criteria, which resulted in the removal of 366 publications: 300 publications that were not related to the focus of our study (EC_1), 11 publications that were not peer-reviewed (EC_2), 20 that were not accessible (EC_3), 4 that were neither written in German nor in English (EC_4), and 31 publications due to the missing relation to computer science (EC_5). 80 papers were deemed relevant for the study since all had undergone peer review and satisfied one of the two inclusion criteria.

Data Analysis

The information from the 80 relevant papers about the practices as well as the reason for implementing them during the development process was summarized in a concept matrix. This information was grouped into categories, in a process that consisted of three steps: 1) The extracted information was clustered to avoid duplicates and to produce a unique list of elements, with care taken to maintain the traceability between techniques, practices, and phases or its purpose throughout the development process. 2) The techniques and practices were considered more or less relevant based on the number of papers that mentioned them. 3) Throughout our weekly iterations, we discussed the set of practices and techniques until we came to a consensus on the final set.

The relevant data regarding the goals of the approaches and their use in the development process were examined in order to create the set of practices. The same three-step approach was used to categorize this information as before. The techniques were then divided into the corresponding practices using the information from the literature. Finally, we assigned each practice to the relevant steps in the software lifecycle or RE process. As a result, only information from the literature was used to build the first iteration of the six practices (and related techniques). For this first version, the practices were created using the most relevant techniques from this concept matrix, which can be found in 40 publications from the review (cf. Appendix C, Table C.2). The practices and techniques were given as a reference to the participants during the interview study.

7.1.2 Interview Study

We conducted an interview study with practitioners to get feedback on the applicability of the process resulting from the literature review. We elaborated an interview protocol with questions and tasks for the participants. The interviews were semi-structured and helped us to learn from the practitioners' experiences. The goal was to combine this feedback with the first version of the six practices, synthesized from the literature review.

Interview structure

One interviewer conducted the interviews. The interviews were exploratory and consisted of two predefined tasks and a set of predefined questions that could be adapted during the interview [*]. After asking the practitioners about the current software development process in their companies, a short introduction to the topic of explainability was given, since it cannot be assumed that everyone has the same understanding of what explainability means. Then, each

[*] A more detailed structure of the interviews is also presented in Appendix C

interviewee was asked to describe and draw the current development process in their company, including activities, methods, practices, and techniques. Using the think-aloud protocol, the interviewees were asked to explain their thoughts. Afterwards, the interviewer explained the concept of explainability in more detail by using the definition for explainable systems (cf. section 6.2). The interviewer presented a scenario and asked the participants to imagine themselves in the position of a process engineer working for a fictional company that is trying to create an autonomous vehicle. As process engineer, they should develop a process (including activities, practices, and techniques) that explicitly considers explainability for a system in a self-driving car. The interviewer asked practitioners to sketch the process and explain their rationale. A list of activities, practices, and techniques from SE and HCI was provided, including those from the literature review, to help with this step. After that, the interviewer inquired the participants about their thoughts on the suitability of the suggested process in the industry.

Participant selection

The goal of this study was to interview persons who have experience in software development. That is, we considered software engineers suitable interview participants. In addition, product owners and requirements engineers were invited, as the literature on the development of explainable systems put a strong focus on RE. 87 practitioners were contacted via different channels including LinkedIn (43 invitations), personal contacts (35 invitations) and via one contact person in a company (11 invitations). In total, 19 experts accepted the invitation, resulting in a response rate of 22%. The main reasons for not participating were time constraints, holidays, or workplace policy that prohibited participation.

Setting

All interviews were performed online via BigBlueButton[†] or Jitsi Meets[‡]. All participants agreed upon recording the interview. Based on the recorded data, the interviews were transcribed. In addition, the collaborative tool *Miro* was used as a virtual board for the tasks.

Pilot interviews

Three pilot interviews were performed with PhD students, which resulted in changes to the interview and tasks' structure.

[†] https://bigbluebutton.org

[‡] https://meet.jit.si

Data Collection

In total, 19 interviews were conducted with participants from seven companies. The interviews had an average duration of 60 minutes and were mostly conducted in German. Two interviews were conducted in English. A complete overview of the participants is presented in Table 7.1. The participants had an average age of 32.2 years (min: 23 years, max: 41 years, SD: 4.6 years) and an average of 6.6 years of experience (min: 2 years, max: 17 years, SD: 4.4 years). Almost half of them work in small companies with less then 50 employees, five work in medium-sized companies with less then 250 employees, and five work in large companies with more than 250 employees.

Table 7.1: Overview of the participants' demographics

ID	Role	Age	Years of experience	Company size
1	Requirements engineer	37	17	small
2	Product owner	32	5	small
3	Requirements engineer	27	2	small
4	Developer	23	3	small
5	Developer	32	7	small
6	Requirements engineer	31	5	small
7	Developer	27	2	small
8	Developer	33	9	small
9	Developer	25	4	small
10	Requirements engineer	35	8	large
11	Developer	34	10	medium
12	Developer	28	6	medium
13	Developer	35	3	large
14	Product owner	29	5	large
15	Product owner	41	16	large
16	Product owner	34	2	medium
17	Requirements engineer	35	8	large
18	Requirements engineer	35	2	medium
19	Product owner	39	12	medium

Data Analysis

The 19 interview transcripts resulted in almost 95K lines of text. The statements were deductively categorized based on the guidelines presented by Mayring [203] to systematically retrieve insights from qualitative data. Accordingly, we divided the interview data into these categories using activities, practices, and methodologies from SE or HCI in consideration of the study's overall objective. In addition, we examined the process sketches provided by the respondents and contrasted them with the research from the literature. Over the course of several sessions, we debated the distinctions and combined the data until we arrived at a final version of a process reference model consisting of six main practices and related techniques that are categorized under respective software lifecycle phases. I will explore this process reference model in section 7.3.

7.2 Practitioners' Feedback

To evaluate the insights of the literature review, we conducted an interview study. All participants stated that they use an agile software development process: The smaller companies use Scrum, the larger ones apply the V-model extended with SAFe. There is a time frame for testing or evaluation at the end of each iteration. Furthermore, at the conclusion of each loop, there is a time limit for testing or review. Participants stressed the significance of this feedback loop repeatedly. We compared the participant recommendations to the findings of the literature in order to validate the findings of the literature review.

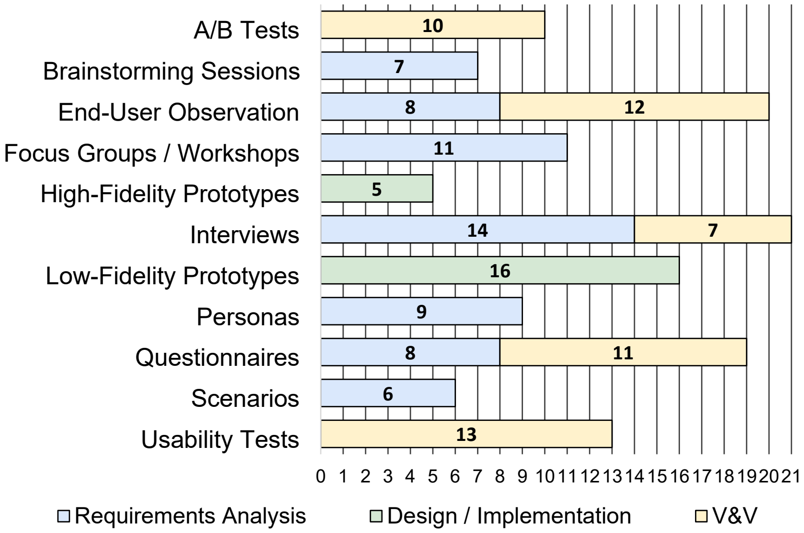


Figure 7.1: Recommended techniques per phase or activity (n=19)

Regarding the techniques that can be used for elicitation, interpretation, negotiation, and documentation [§], participants suggested: interviews (73.7%), focus groups and workshops (57.9%), personas (47.4%), questionnaires (42.1%), brainstorming with customers and colleagues (36.8%), scenarios (31.6%), and/or end user observations (42.1%). During design or implementation, participants suggested: low-fidelity prototypes (84.2%) (such as mock-ups

[§] For simplification purposes, I refer to these activities in this chapter as "requirements analysis" and paper prototypes) and/or high-fidelity prototypes (26.3%). Prototypes were often mentioned as a practice useful for requirements prioritization. For the verification and validation usability tests (68.4%), end user observation (63.1%), interviews (36.8%), questionnaires (57.9%) and/or A/B tests (52.6%) were suggested. Because our interview study used a small, unrepresentative sample, it is important to treat the results with caution because they cannot be generalized based on statistics. The outcomes of a study based on a representative sample might differ.

Overall, the suggestions made by the participants are consistent with the research's findings. The need for a comparison between users' expectations and the system's performance was stressed by the participants at various points, even though they did not explicitly mention mental models in their statements. In addition, the interviewees recommended practices such as brainstorming sessions with the team or with stakeholders, and also end user observation. The following sections discuss the main points raised by the interviewees. These points include the significance of using an iterative and user-centered approach, the impact of process type, how realistic the identified practices and techniques are, and the potential challenges of implementing them in practice.

7.2.1 Iterative and User-Centered Approach

In summary, the participants were unanimously in favor of developing explainable systems through an iterative process. However, it is important to keep in mind that all participants work in an agile environment and may not be familiar with plan-driven approaches, which may have influenced their recommendation. All participants recommend an agile development process because [sic] "agile procedures are iterative and allow changes to be made in an uncomplicated way". But since explainability is still relatively unknown and thus not explicitly addressed in existing software processes, participants advocate for a trial-and-error iterative procedure to develop explainable systems. This way, the users' understanding can be constantly assessed, and the design can be changed to improve the system's explainability.

One participant mentioned that the development of explainable systems is relatively independent of the software development process and that it is also possible within a waterfall development process, though it may take longer than agile approaches. Another participant agreed that the waterfall development process, due to its sequential nature, may be less suitable for developing explainable systems because a perfect solution is unlikely to be found upfront. One participant emphasized the importance of feedback loops in the process because they allow for design adaptation based on user feedback.

All participants recommended an end-user-centered process for the development of explainable systems. Since all participants stated that the development of explainable systems would be possible using the current development methodology in their companies, and often used the process as a base for their recommendations, we asked the participants if they see any need for optimization. All participants agreed that implementing more end-user-centered practices and techniques can aid in the capture of end-user requirements and the evaluation of design decisions. According to eight participants, including team members in the evaluation phase and making them aware of user feedback can help team members better understand current challenges and design concerns.

7.2.2 Realism and Challenges

Four participants stated that even the best development process for explainable systems would be useless if the team is unaware of the significance and value of explainability. As a result, the motivation for incorporating explainability should be clear to the team, and the team should be convinced and inspired by its value. Giving its importance, explainability must be kept in mind whenever changes or adjustments are made, just as security and usability are constantly considered.

Finally, participants were asked about the suitability of their recommendations in light of their company's context. As an example, the interviewer mentioned resources such as time, cost, or the number of employees in a company as possible factors influencing the practicability of the suggested activities and practices. All participants agreed that the activities and practices could be integrated. Three participants noted that the choice of the activities and practices depends on each company's resources, the team, and the product. Seven participants concurred that a trial-and-error process is necessary until an ideal strategy for the activities and practices that are most appropriate in the company's situation, as well as how to integrate them into the company's particular context, is identified.

7.3 A Process Reference Model

A process model is an abstract description of a process. A process is an instance of the process model. In contrast to the process itself, which is what actually occurs, a process model could be used to specify how things must/should/could be done [204].

A process model is essentially a prediction of how the process will or should look like in reality. During the actual system development, the process will be concretely determined [204]. Therefore, even if this thesis (or any other publication) offers a process model, the focus is rather on the identified practices and techniques. This offers more flexibility, since these practices and techniques can be more easily integrated into any software process that implements the software lifecycle, being waterfall or agile. One of the main distinctions between waterfall and agile methodologies is how the lifecycle is conducted. Agile is an incremental and iterative approach that repeats specific phases of the lifecycle; whereas waterfall is a linear and sequential approach that conducts the lifecycle once.

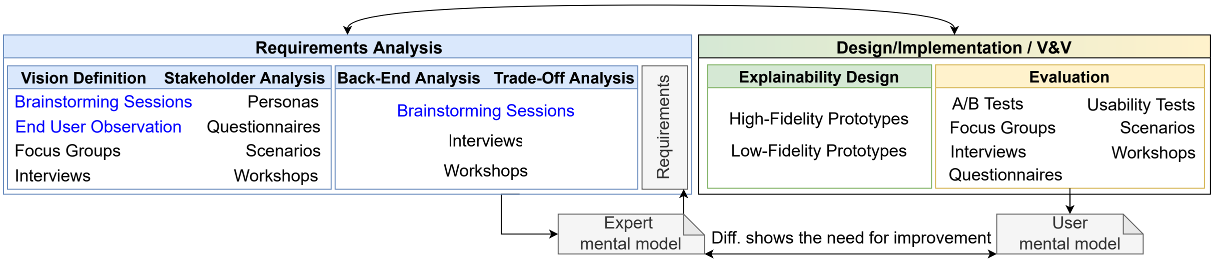


Figure 7.2: Six practices for developing explainable systems, related to the respective activities of the requirements engineering process or software lifecycle

In the figure, the activities of the requirements analysis process (elicitation, interpretation, negotiation, documentation, verification & validation (V&V)), and some activities of the common development process (design, implementation) are depicted. This is due to the fact that these practices can be integrated into both the traditional approaches (which often consider the classic RE process) and the agile process. This difference is especially noticeable in frame two (Design/Implementation/V&V). These three activities are present because explainability requirements can either be *designed* and *implemented* directly into the system, depending on the development stage, or be integrated in a prototype for *verification and validation* of requirements. The arrow at the top of Figure 7.2 indicates that the activities can be iterated as needed. The activities in blue were only mentioned during the interview study. In the following I will discuss each of these practices, drawing parallels with the respective activities in the RE (or software lifecycle) process.

7.3.1 Elicitation

The elicitation focuses on establishing a vision for the system to be produced, as well as gathering sufficient information by focusing on stakeholders' needs, goals, domain, and context. This information helps to formulate high-level requirements (or raw requirements) and to define the system boundaries to facilitate the requirements analysis process.

Vision Definition

Before requirements can be specified, broader visions are developed. A vision can refer to the capabilities, features, or quality aspects of a system [180]. A vision establishes the long-term objectives of a project, makes it easier to communicate, helps define the project's scope, and increases the likelihood that a successful system will be created. It is possible to clear up misunderstandings regarding the system's goals and stakeholders' expectations by creating a shared vision and getting a handle on the crucial quality aspects that the system must take into account [180, 26].

Therefore, the first thing that needs to be determined when establishing a vision is if and how explainability actually adds value for stakeholders. After all, explainability should only be considered if it is identified as a need [12] and if it aggregates value to the system since in some cases the cost of explanations might outweigh their benefits [118]. Qualitative techniques (e.g., brainstorming sessions, focus groups) can be used to support this discovery [205].

Stakeholder Analysis

Understanding the user is one of the main aspects that influence a positive impact of explainability in a system. Software engineers must understand the users, their cognitive behavior, their attitudes and the characteristics of the tasks that they must perform [206]. Once again, qualitative techniques such as interviews and ethnographic studies provide important information on the actual needs and expectations of users with regard to the system and the information to be received.

It is important to pay special attention to understanding the existing stakeholder groups in the case of explainable systems (cf. section 3.2). Identifying stakeholder groups is critical for determining which interests and needs the relevant stakeholder groups have in terms of explainability [23]. Different stakeholders have different goals and needs for the information system, as well as other requirements for explanations. Again, qualitative approaches such as interviews and end user observation provide valuable information on users' genuine needs and expectations regarding the system, where explainability may be required in the system, and how explanations should be designed [207, 208]. The most frequent techniques used to achieve these goals are interviews, personas, scenarios, questionnaires, focus groups, and workshops. After vision definition and stakeholder analysis, broad quality goals, explainability goals, or raw requirements may be set, which can be further refined over the subsequent practices and activities.

7.3.2 Interpretation and Negotiation

Interpretation helps to make sense of all the data gathered during elicitation and helps to achieve a deeper and more precise understanding of the raw requirements. This data is analyzed and classified into quality goals, functional requirements, design ideas, existing constraints, etc. The raw requirements are refined into requirements based on these categories. Gaps, ambiguities, conflicts, and dependencies are identified to be clarified during negotiation. Negotiation is used to resolve these conflicts and to prioritize the requirements based on the defined quality goals. The project constraints and system aspects also need to be considered within these activities, including what is feasible given budget, time, technology constraints and other limitations.

Back-End Analysis

During back-end analysis, the logic of the algorithm is assessed or planned in relation to the goals of explainability. The first step in this activity is to establish whether the system component that needs to be explained already exists (e.g., explainability must be integrated into an existing system) or if the algorithm still needs to be developed (e.g., a system developed from scratch). The first thing to think about is whether the algorithm that needs to be explained (from now on, referred to as "back-end algorithm") is interpretable [209]. Consider the case where the back-end algorithm is based on a ML model. Although ML models, particularly deep learning, can produce accurate system outputs, they are frequently referred to as "black boxes" because it is difficult to comprehend their rationale [70]. Even data scientists frequently have trouble understanding how algorithms produce the models and how the models generate the results [210]. In the ML domain, a model is considered interpretable when it is possible to determine why it produces a specific outcome. In a nutshell, the more interpretable a model is, the easier it is to explain its rationale and outcomes. When the back-end algorithm is comprised of an uninterpretable model, other explainability techniques (such as post-hoc explanations) must be used to explain it [70]. Another example of a factor to consider during back-end analysis is whether the desired explanations call for global or local interpretability, since the algorithmic strategy to be used will also depend on it. Local interpretability refers to explaining each individual prediction, whereas global interpretability frequently refers to understanding how a model functions in general [211].

Thus, it is crucial to assess and specify which explainability approach is necessary and feasible in light of the algorithm and the aspects that need to be explained [209]. Communication among team members is crucial for determining whether there are any technical constraints impeding or limiting the achievement of the explainability goals and for developing appropriate solutions [158]. Suitable techniques are workshops, brainstorming sessions and interviews with team members.

Trade-off Analysis

During trade-off analysis, it is important to evaluate how explainability interacts with other quality aspects. In chapter 6, I explored the interaction between explainability and other quality aspects and how explainability can have both a positive or negative impact on the quality landscape. This negative or positive impact depends, in the end, on the design decisions toward explainability. During this activity, knowledge catalogues (such as the one proposed in this thesis) are useful artifacts that can assist software developers in avoiding quality-related conflicts and determining the best techniques for achieving the intended quality outcomes [12]. As a result, practitioners should concentrate on design choices and interactions to make explainability a powerful catalyst for other essential quality attributes in contemporary systems. The same practices recommended for back-end analysis can be applied during this activity. Existing approaches and tools to support practitioners in the identification of conflicts between NFRs can be either experience, model or mathematically based [212, 213]. Requirements catalogues can be used during various phases of software development projects, including elicitation and architecture design [17], and also during trade-off analysis to identify the interdependencies between requirements [97].

For instance, it may be that what is needed to satisfy explainability is limited by the project constraints, or that providing a specific type of explanation is not in the interest of the company, as trade secrets may be revealed. Stakeholders such as executives, managers, developers, usability designers, law scholars and system architects may be heard during requirements negotiation in order to reconcile possible conflicts between stakeholder objectives and to prioritize requirements.

7.3.3 Documentation

The purpose of documentation is to persistently and efficiently represent and store the knowledge related to the requirements. The needs of the stakeholders are converted into written specifications and diagrams that can be read, reviewed, and used for this purpose.

Requirements and Mental Models

To define requirements, it is necessary to compare and understand the goals of the various stakeholders in the process, as well as how these goals result in what is expected from the system in terms of communication with the user. The concept of *mental models* [214] (which has long been studied and discussed in particular in the HCI community [215, 216]), refers to the mental image that users have of the behavior of a system. Mental models offer a deep understanding of people's motivations and thinking processes [217]. The closer the system is to the user's mental model, the better the usability and understandability of the system. Mental models can be used to capture expectations regarding the understandability of a system [218, 208, 219], supporting the definition of requirements.

Users typically have certain expectations for how a software system will behave when they use it. The users' implicit mental models of the system are formed by these expectations. Confusion arises whenever the system's actual behavior differs from the users' implicit mental models. This is critical because if users are dissatisfied with the results, they may be dissatisfied with the system as well. One way to mitigate this and plan around those expectations is to develop explicit mental models that foretell users' confusion in order to counteract it. By doing so, it might be possible to provide appropriate explanations when they are required [2].

Figure 7.3 illustrates the process of confusion and creating explicit mental models to counteract and predict confusion.

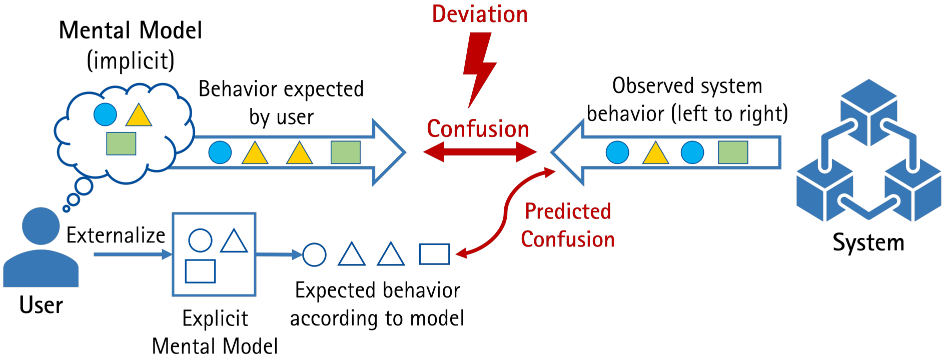

Mental Model
(implicit)
Behavior expected by user
Deviation
Confusion
Observed system behavior (left to right)
System
Predicted Confusion
Externalize
User
Explicit Mental Model
Expected behavior according to model

Figure 7.3: Implicit and explicit mental models (extracted from [2])

For instance, after back-end and trade-off analysis, an *expert mental model* can be defined. An expert mental model represents the planned behavior of the system, as defined by the designers (e.g., experts). Expert mental models can be used to capture how the system should be understood and how an explanation can aid in understanding. When defining or refining requirements, practitioners can use expert mental models as a reference. These expert mental models can then be compared to the *users' mental models* to assess the system's understandability or the quality of the explanations. Essentially, mental models can help define the right requirements and make the right design decisions.

Think-aloud and other traditional methods of usability testing [5, 220] should be put in place to evaluate the effect of explainability. In addition, templates for writing requirements, as well as frameworks and quality models can be developed to assist in the documentation and to assist in the understanding of the requirements. These artifacts will be explored in more detail in chapter 8.

7.3.4 Design, Implementation, V&V

The purpose of validation and verification is to examine the elicited and elaborated requirements formally and substantively. While verification examines whether elaborated requirements and elicited requirements are consistent, validation looks to see if the requirements are in line with the established quality goals and the needs of the stakeholders. The main goal of design and implementation is to translate these requirements into design solutions or in the corresponding code that can be implemented in the system.

Explainability Design

Software engineers must decide on the details of the explanations based on the requirements during this activity. Design elements that are crucial for explanations should be specified during this phase, including: whether explanations should be interactive or static, what language to use (e.g., formal or informal), when to present explanations (e.g., before or after an event), and how to present the information (e.g., audio, text, and other user interface aspects) [209, 69][¶]. Prototypes, both low-fidelity and high-fidelity, are helpful for presenting design concepts and different types of explanations during explainability design. This enables quick visualization and discussion of design concepts prior to implementation. Prototypes can be tested for efficiency and to see if they meet the explainability requirements. Once these design decisions have been made, the project's development culture can proceed as usual to implement them.

Prototype review has been especially effective in identifying usability issues and optimizing the design of users interfaces. Building prototypes allows the team to capture and validate assumptions about the desired software characteristics [115]. Prototyping is widely used in usability engineering and is a lightweight way to test a system [205]. It allows to collect feedback from stakeholders to validate whether the requirements meet their expectations. Prototypes can be validated with real user groups, assessing whether the explanations fulfill their needs or hurt usability. The use of mock-ups can help to identify design flaws and to assess the effect of explanations on user experience. It is also possible to compare whether the represented model of the system (e.g., how the system is presented to the user) matches the users' mental models.

Evaluation

The evaluation determines whether the system is explainable, that is, whether the explanations are adequate or should be improved. Because the effectiveness and quality of an explanation are subjective, the focus of this practice is on end-user feedback combined with prototype or version testing (A/B tests) [221]. Evaluating the explainability of a system is challenging, as each individual has different cognitive processes while understanding something [221]. Understanding the cognitive processes of end users is critical because explanations fill knowledge gaps. Mental models play, again, an important role, being an important aspect of the evaluation since they provide a good way to capture cognitive processes [208]. The users'

[¶] These elements will be discussed in-depth in chapter 8.

mental models and the expert mental model can be compared during evaluation. When the end-user model differs from the expert model, it indicates misconceptions or issues with the explanations or the requirements that need to be fixed by changing the system itself, the requirements, or the explanations that are given [208, 76].

7.3.5 Limitations and Threats to Validity

Since the findings are based on information from 19 interviews and 80 publications, they should not be overly interpreted. However, I believe the data is sufficient for this preliminary analysis. Future studies are required to expand on these findings and create more in-depth, precise results that increase the reliability of the conclusions.

The fact that the interviewer could think about questions on the fly may have jeopardized the validity of the interview study. Because questions can be posed in a way that effects the answer, a researcher bias could have been introduced. To reduce this risk, I examined and discussed the interview protocol with the interviewer before to the interviews to determine the appropriate interview strategy and minimize threats.

A rigorous review procedure was followed. While one of the writers prepared and carried out the two investigations, I reviewed each stage of the research. In addition, for the interview study, we conducted pilot interviews, which resulted in minor changes to the questions and interview structure. We carefully extracted and discussed relevant information from the publications and from the interview transcripts throughout data analysis.

Because there are still very few development teams with experience in creating explainable systems, the insights collected from the interviews are based on a hypothetical situation. As a result, it is probable that the concepts that arose from the interviews are not applicable in industry. Future studies should consider a real-world environment. All interviewees also work in an agile setting, which influences their views on plan-driven methodologies and limits the scope of potential solutions obtained from the interview study. Furthermore, the findings do not cover every stage of the software lifecycle equally.

7.4 Summary

The findings of a literature review were combined with the perspectives of practitioners gathered through an interview study to compile a set of practices and corresponding techniques for the development of explainable systems. Practitioners believe that existing user-centered approaches and practices are effective when dealing with explainability in a software project, which supports the findings of the literature. The participants also expressed familiarity with these practices and techniques. Based on this analysis, **RQ3** can be answered:

Answering RQ3

> *I propose six practices that should be considered when developing explainable information systems: vision definition, stakeholder analysis, back-end analysis, trade-off analysis, explainability design, and evaluation. These practices can be supported by user-centered techniques such as end user observation, interviews, questionnaires, personas, prototypes, focus groups, workshops, brainstorming sessions, scenarios, storyboards, A/B and usability tests.*

In section 4.7, I discussed how heuristics based on well-known usability principles can avoid some of the possible negative effects of explanations on usability and presented the assumption that usability-related practices and techniques (which are mostly user-centered) could help to prevent negative effects of explanations on usability and be useful for explainability in general. The results in this study support this assumption. This assumption had not yet been validated beyond that survey and, more importantly, the feedback of software experts on whether this approach is compatible with the reality of the industry was still missing. The research discussed in this chapter revealed that these user-centered practices and techniques are also used in the works examined in the literature for the development of explainable systems. Additionally, the feedback from experts in the software industry indicate that these practices and techniques are also applicable in practice, demonstrating alignment between research and practice.

In fact, established UCD best practices and usability engineering methods should be a core part of every software development activity [222]. Hehn and Uebernickel argue that usability engineering should be integrated into the RE process, combining the human-oriented aspect of the former with the more formal, technology-driven aspect of the latter [223]. Many authors advocate for the incorporation of a user-centered development strategy in the development of explainable systems (cf. [224, 219, 196, 12, 225]). By integrating more user-centered practices, there is a shift of focus in systems development towards putting the goals, needs, and wishes of the users or the addresses of explanations in the first place. There are two crucial reasons for integrating UCD practices into the development process of explainable systems: An explainable system offers explanations to fill in knowledge gaps that are highly personal to each addressee. Additionally, explanations serve as a channel of communication between humans and machines [226].

The results also point to the use of practices rather than methods. First, this goes in line with findings from recent research on software development processes showing that practices appear to be way more important than methods, as the use of practices does not depend on the selected development process [227, 228], making the practices and techniques applicable in either waterfall or agile development environments. Second, there is evidence that many of the research community's contributions have not been adopted by practitioners, because it is often difficult for practitioners to find a way to incorporate new research ideas into their busy workdays [114]. The major concern of software professionals is cost, both in terms of time and money [229]. As a consequence, embracing existing practices and techniques rather than introducing new ones that increase costs is a favorable option [12].

During the study, it was possible to identify a special emphasis on the requirements process and on validating these requirements, both in the literature and in the interviews. Therefore, I decided to focus on the RE activities. I contend that the development of explainable systems heavily relies on meticulous requirement analysis, as well as how those requirements are implemented and transformed into explanations, and how those explanations are displayed on the user interface. Therefore, requirements-related activities in the process (whether in a traditional or agile setting) should be given special attention.

In this chapter, I proposed six practices that support the development of explainable systems systems as well as user-centered techniques to support these practices. In the next chapter, I will explore how to facilitate these practices with help of a quality framework for explainability.

8
A Quality Framework for Explainability

Quality aspects are abstract ideas that are often difficult to realize and transform in software requirements. Since quality is so elusive, effort must be made to make it as tangible and measurable as possible. Therefore, according to Boehm [140], mapping abstract aspects to quantifiable elements during RE facilitates the evaluation of software products and helps to propose concrete solutions (i.e., operationalizations) for achieving a desired quality goal. In the case of explainability, these operationalizations essentially translate into explanations (the means to achieve system explainability). Thus, it is important to know what characteristics the explanations must possess and how to evaluate their effect on desired quality aspects. Since explainability is a relatively new NFR, there is presently little guidance to support RE for explainable systems. In particular, guidelines and frameworks that offer an overview of features of explanations are missing.

To bridge this gap, this thesis proposes a framework that links the dependencies, characteristics, and evaluation methods for integrating explanations in information systems, facilitating RE. The framework was constructed based on the results of a literature study, in which the following three aspects were identified 1) the influences on explanations (or dependencies), 2) the characteristics of explanations, and 3) forms of evaluating explanations. The framework

The research described in this chapter was partially performed in collaboration with three researchers: Florian Herzog, Verena Klös, and Kurt Schneider. Some of the results were published in [230] and [231], on which this chapter is based. As a result, I occasionally use the pronoun “we”.

should support the creation of quality models for explainability and guide the specification of explainability requirements for information systems.

Quality aspects may become concrete and operational through refinement [16]. By specifying requirements, one can determine what a system should and should not do, which allows software engineers to compare if a system is achieving the expected goals. According to Schneider [19], a step-by-step approach can be taken from abstract quality notions to measurable quality requirements to facilitate the specification of quality requirements. This approach considers three abstraction levels: abstract goals, concrete characteristics, and measures or indicators [19, 232]. The idea behind the approach is to work top down and closely involve customers in the process. By following the approach, it is possible to derive quality models, since the framework supports the modeling of quality aspects, the derivation of requirements, as well as the analysis and strategies to fulfill and measure these requirements. These "three abstraction levels" work as a foundation for the structure of the here proposed quality framework.

While there have been several proposed models, taxonomies or frameworks for the integration of explanations (cf. chapter 2), most of the publications are focused on specific domains and do not address explainability for information systems in general. Furthermore, there is still a lack of contributions regarding methods and concepts related to RE. The framework proposed in this thesis differs from others in the two following ways. First, there are no specific works focusing on quality frameworks for explainability. Second, the other works focus on specific types of systems such as AI or recommender systems. The goal of this work is to provide a quality framework that can be used as a basis and be generalized to more types of systems.

8.1 Methodology

The proposed framework is a structured collection of findings and recommendations from literature that focus on the design of explanations. To this end, seven works that present models, taxonomies or frameworks for the integration of explanations served as a base for the framework. The concepts were combined and restructured based on the data obtained from the other publications in the literature.

The research approach consisted of three steps, namely 1) the data collection through a literature study, 2) the structuring of the collected data in a framework, and 3) the application of the framework in a case study, combined with an experiment and a quasi-experiment (discussed in chapter 9). As a foundation for the framework, we used an existing structure that takes three abstraction levels into account: abstract, concrete, and measurable [19]. The goal of the study was to answer research question **RQ4**, and its corresponding sub questions 4.1, 4.2, and 4.3.

To identify influences on explanations, characteristics of explanations, and forms of evaluating explanations, a literature search in the fields of explainability and HCI was performed, with focus on relevant aspects for the design of explanations, their presentation on the user interface, and their evaluation. All steps required to get an overview of existing literature were performed, namely 1) definition of the search string, 2) definition of inclusion and exclusion criteria, 3) selection of the database(s), 4) definition of the termination criterion, 5) execution, and 6) data analysis. The literature study took place in May 2021. Four databases were searched, namely *ACM Digital Library, IEEE Xplore, Science Direct,* and *Springer Link*. The search led to 58 papers that were considered relevant with regard to our research questions. More details on the literature study are provided in Appendix D.

8.1.1 Literature Study

The framework is a structured collection of findings and recommendations from literature in the fields of explainability and HCI. To this end, we derived data from the following seven works that present models, taxonomies or frameworks for the integration of explanations, and added insights from the other literature that were examined. An overview of the aspects that were identified and their respective sources can be found in the list of references in Appendix D. Below, I discuss the seven works that were used as a theoretical base for the framework.

Nunes and Jannach [69] propose a taxonomy for explanations in recommender systems based on results from a literature review. The taxonomy is composed of general facets associated with explanations such as their generation approach, and interface aspects such as explanation content and presentation form. The authors also investigated how explanations are generated, presented to the users and evaluated. Ribera et al. [233], propose a four-step approach to integrate user-centered explanations into AI systems. The authors recommend that explanation designers ask themselves what is the reason for the integration of explanations (why?), define which part of a system should be explained (what?), and how explanations should be presented to users (how?).

Arrieta et al. [70] proposed a taxonomy that maps ML models to the explanations they produce. The taxonomy should serve as an artifact that supports professionals in the implementation of AI methods in organizations. The authors also investigated the relationships between different concepts related to the field XAI. Sokol and Flach [80] proposed a framework with five dimensions to compare and contrast explainability approaches, identify possible problems, and support in the choice of explainability methods for specific cases. They discuss key aspects such as the explainability goals and the target audience.

Chari et al. [234] propose an explanation ontology that models attributes of explanations, the role of explanations, user attributes, different literature-derived explanation types, and attributes visible in the interface. The goal of the ontology is to allow explanations to be assembled automatically by AI tasks. Kouki et al. [235] define seven distinct dimensions for the user interface design of explanations in recommender systems. These dimensions contain six generally applicable categories including the presentation (e.g., text- or graphic based), grouping, information density, abstraction level, and visualization. Van der Waa et al. [236] define three sets of practical recommendations to improve user evaluations for XAI. They categorize their recommendations into constructs, relations, use cases, experimental context, and measurements. The authors highlight the importance of appropriate use cases for user evaluations in combination with the experimental setting.

8.1.2 Data Analysis: Building the Quality Framework

To build the quality framework, three abstraction levels were defined based on the adopted structure: an *abstract* level that groups aspects that influence the requirements on explanations and help to elicit and define the desired quality goals and the existing constraints, one *concrete* level that groups the characteristics of the explanations themselves and shape the design choices toward explanations and the related functional requirements, and one *measurable* level that summarizes the evaluation methods and metrics that can be defined to assess if the concrete choices help to achieve the established quality goals. Next, the literature was analyzed based on the research questions. In a first iteration, a researcher created the categories in the model under my supervision, based on the concepts extracted from the seven publications listed in subsection 8.1.1. We discussed the categories, and I analyzed the ideas and the papers in parallel. We discussed disagreements until we reached a consensus.

The categories were organized in each level and the 58 papers were investigated to identify further information belonging to each of the three abstraction levels and refine the categories. A list of aspects based on the publications was compiled (cf. Appendix D). This list was discussed and refined over the course of several sessions until we produced a beta version of the framework.

We gathered feedback on this version during a workshop in a case study (chapter 9). Based on this feedback, we refined the framework once more, mostly changing the visualization and layout. Finally, the framework was adapted once more to include the concepts explored in chapter 6, chapter 7, and to include feedback from a focus group with three requirements engineers. During this focus group, I asked the participants to offer critical feedback on the framework after asking them to read the related publication (cf. [230]). Their comments were likewise mostly focused on visual elements. They suggested that the constraints should be visually separated from the objectives, since the constraints restrict objectives (this motivated me to separate both and highlight this influence with a red arrow.). They also suggested that each level be color-coded and related with the practices identified in the research. The two former versions of the framework are presented in section D.2 (Appendix D).

8.2 A Quality Framework for Explainability

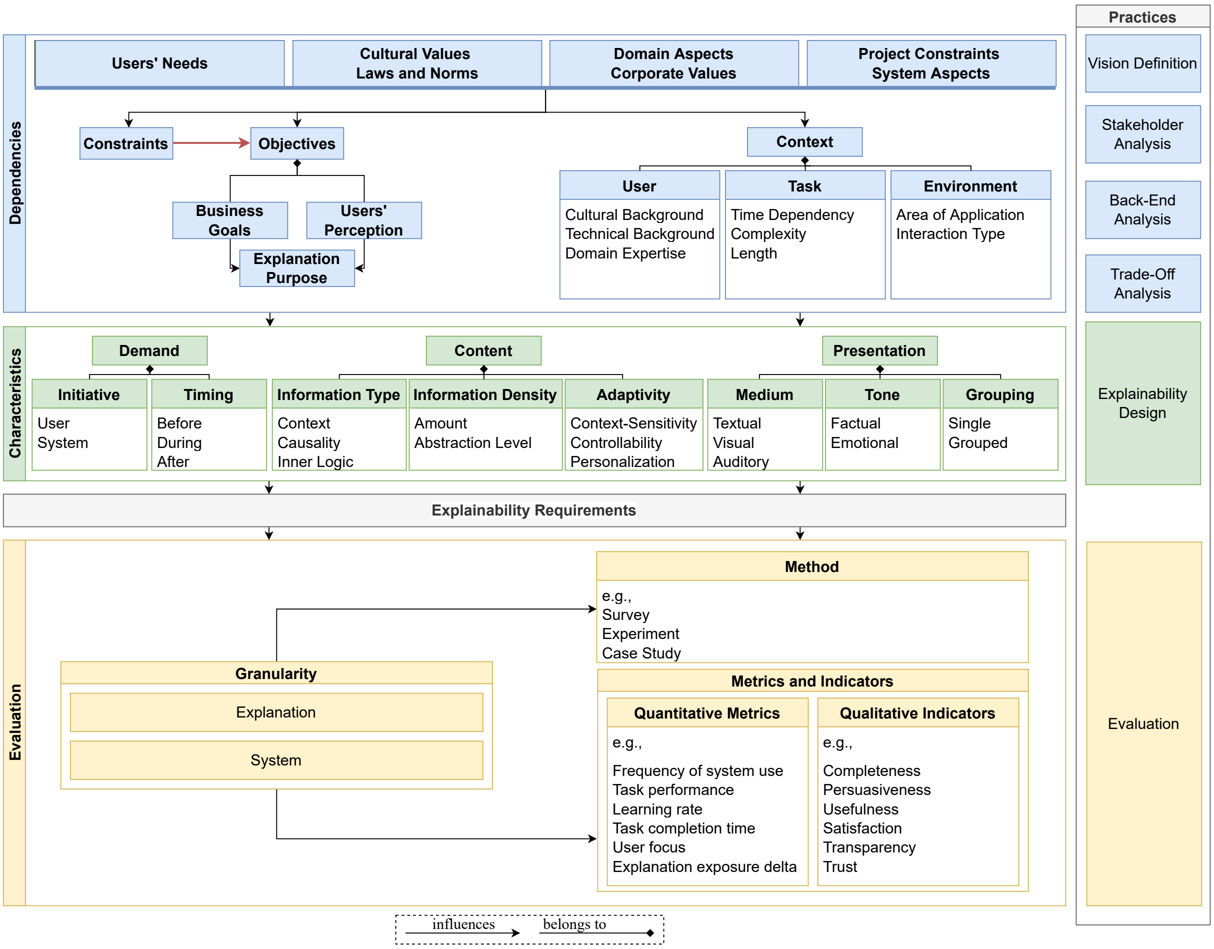


Figure 8.1: The explainability quality framework

Since the ability to translate real-world aspects into software requirements is essential to software quality, the framework (Figure 8.1) is thought to assist in this process. The practices covered in chapter 7 are located to the right of the framework. In the figure, it is possible to see how each level of the framework supports the suggested practices. In chapter 9 we will see how this happens in detail in practice, based on what was observed in the case study.

The framework can have several applications. The most general is to help requirements engineers identify relevant aspects for explainability during RE. A more specific application is to use it as a basis for building quality models for explainability that can be used to guide the specification of requirements from abstract quality aspects into more fine-grained constructs that can be measured in the lowest level of the model [237].

The framework is divided in three levels: *dependencies*, *characteristics* and *evaluation*. The former two levels are directly related to explainability requirements, while the latter summarizes methodologies for evaluating the quality of explanations and their impact on a system. Each level is further divided into categories that function as a collection of aspects that are essential for that level. Six categories of aspects connected to the dependency and characteristic levels were identified in the literature: the *context* where the system is used, the *objectives*, the possible existent *constraints* that limit the and influence the objectives, the *demand*, the *content* of explanations, and aspects of *presentation*. For instance, the demand, the content, and the presentation are essential to determine the characteristics of the provided explanations. Furthermore, the levels above have an effect on the levels below: dependencies influence the characteristics of explanations (which are the design decisions) and both influence the choice of evaluation methods. Moreover, the effects of explainability can be evaluated in two degrees of granularity: one can evaluate the *explanations* as such or measure the impact of the explanations on the *system*. To evaluate the quality, the method as well as the metrics or indicators should be defined.

Each category is explored in the following subsections. Note that some parts of the framework are not intended to be complete, but show typical examples from the literature study. In Figure 8.1, this is indicated by "e.g.".

8.2.1 Dependencies

Dependencies are abstract aspects that impact the design of explanations, such as the objectives, the constraints, and the context. They are related with "real-world" aspects that have an impact on the system and on the explanations. In addition, the dimensions presented in chapter 3 and applied in chapter 6 also have an influence on the objectives, constraints and context. The context, for example, will be influenced by users' individual needs and expectations, which will then influence the characteristics of explanations, such as the level of detail to be provided. Similarly, laws and regulations might impose constraints that must be observed and respected during system design and requirements definition.

Objectives and Constraints

Objectives are the pursued quality goals in a system. Constraints refer to a "requirement that constrains the solution space beyond what is necessary for meeting the given functional, performance, and specific quality requirements" [93]. Objectives are established according to the needs of each particular project and context, while constraints limit the solution space toward these objectives. These objectives and constraints help to define the quality goals and help to define the purpose of explanations in the system (i.e., explanation purpose). For instance, while an objective can be to increase the system's transparency through explanations, stakeholders can be hesitant to explain certain aspects, because of existing trade secrets (a constraint). The process of setting goals based on objectives and constraints involves a trade-off between what is desired and what is feasible.

The objectives are divided into three subcategories: *business goals*, *users' perception*, and *explanation purpose*. These three subcategories are an example of the aspects that should be in focus when considering explainability from an SE point of view: The goals associated with explainability are established by focusing on the business and on the user. In chapter 6, I summarized 57 quality aspects that can be affected by explainability. These aspects are examples of objectives or constraints for the integration of explanations in a system [87]. Some examples of the most common objectives for considering explainability in a system found in the analyzed publications are: *trust* [87, 238], *satisfaction* [69, 174], *scrutability* [239, 240] *effectiveness* [239, 241], *efficiency* [174, 173], *transparency* [239, 12] and *persuasiveness* [242, 243]. Please keep in mind that these are just examples based on the findings of the literature study, and I do not claim that they are the only or most important goals to pursue in explainable systems.

Business goals Business goals are the primary pursued quality goals in a system. In the case of explainable systems, it is either true that explanations must contribute to the achievement of these objectives, or explanations must at the very least not conflict with them. They are developed from a corporate viewpoint in collaboration with the client, and can, in essence, be any quality aspect that stakeholders want to meet or enhance and can be considered as the "general quality goals" [19]. Common business goals are *system acceptance* and *usage increase*.

Users' perception Quality is often defined in terms of the fitness of the product for its purpose, considering different stakeholders [244]. Perception can be seen as a belief or opinion, or the way that someone thinks and feels about a company, product, or service [245]. When we discussed the definition (cf. section 6.2), the importance of the interlocutors receiving the explanation became clear. Therefore, the same idea applies here. Quality goals should also be founded on the user's perception and on their needs. Explanations should have a positive impact on how users perceive the system. This subcategory includes quality objectives that are related to user perception and can be influenced by providing explanations. It considers the quality of the explainable system in light of the user's expectations and needs [246]. Therefore, user perception goals are those goals that are to be perceived directly by the end user of the system. They contribute to the achievement of business goals. As a result, this subcategory is linked to quality goals that are more relevant to users or quality in use (i.e., first dimension) and includes quality aspects that are more related to how the user should view the system in the case.

Explanation purpose Explainability can be seen as an enabler: a way to achieve given quality aspects in a system. In other words, explanations can be seen as a means to an end [87]. In this sense, the quality goals related to the *explanation purpose* are more specific than the other goals, and refer to the immediate purposes of explanations [239, 238], i.e., the aspects that shall be improved by explanations. They can be derived from business goals or goals related to the users' perception. For instance, explanations can be specifically designed to improve the understandability or transparency of a system, which are foundational aspects of explainability (as depicted in Figure 6.4, chapter 6). By improving the understandability of the system, it is possible to achieve a better user experience [12], which was previously defined as a business goal. Thus, by defining and achieving the specific goals of explainability (understandability and transparency), other quality goals (user experience) can be achieved.

Context

The context is the interaction between a user, a task, a system, and the environment [94, 87]. The context is tied to the in-depth thought processes of users and the underlying structure of their activities [247]. Furthermore, the context of an explanation is linked to the influences that directly affect the system and may be used to derive requirements for explanations. This category influences both the objectives and constraints as well as the design decisions. For instance, the aspects listed within each of the three subcategories (user, task, and environment) directly influence the need for, the content of, and the presentation of an explanation in the system. The quality of a tool can be estimated by how well users can use a tool in their environment to perform their tasks.

User Users are the persons who interact with the system and to whom the explanations must be tailored. In explainable systems, they are also the addressees of explanations [62, 87]. However, it is important to point out that the addressee can belong to different stakeholder groups, since a system can have different users (and different users need different kinds of explanations). The most frequently mentioned user-related characteristics in the literature are the cultural and technical background, as well as the domain expertise [239, 248, 87]. The *cultural background* does not merely mean culture in terms of nations or territories but also in terms of social groups [102]. Furthermore, *technical background* and *domain expertise* [248] refer to the different levels of background knowledge that the addressees might have that, in turn, contribute to different requirements on explanations. Users are generally considered either novices or experts (or something in-between) with respect to their technical backgrounds or on a given domain [5].

Task Tasks are actions that must be carried out as part of a process for the user to achieve their objectives with the aid of the system. To identify how explanations might help users achieve their intended goals, it is crucial to understand the structure of the tasks, the actions involved, and other specifics on how users perform their tasks to achieve their intended goals. Some aspects that are relevant in a task are the *time dependency* (when the task should be performed), and the *complexity* and the duration of the task (*length*). These three characteristics also influence the requirements for explanations [80, 249].

Environment The task to be performed is also closely related to its environment. The environment is defined by the external circumstances affecting the system. This includes the general *area of application* of the system which, among others, defines the criticality of a system. The *type of interaction* (e.g., per voice, haptic, etc.) between user and system also depends on the environment and has an effect on the type of explanation that users need to receive [250, 251, 80].

8.2.2 Characteristics

This level refers to the characteristics of explanations *per se*. It lists design aspects of explanations. Thus, it directly influences specific design choices and functional requirements related to explainability. It is important to note is that the options presented are not necessarily mutually exclusive. This means that an explanation does not have to fit only one property; it is also possible to have properties that fall somewhere in between two possibilities. It is also not necessary that an explanation fulfills all aspects, since not all categories apply to every context.

Demand

This category contains aspects related to the *initiative* and *timing*. The demand specifies the certain point in time when explanations are provided and who has the initiative of delivering or requesting explanations.

Initiative Initiative refers to the trigger for explanations in a system. The system either autonomously provides an explanation or explanations can be requested manually by the user. This reverberates the result of the survey (cf. subsection 4.3.3), when users were asked when they would like to receive an explanation. In the best case, the participants demonstrated that they would like to receive on demand explanations whenever they need them. Because this is not always practicable, software engineers must carefully choose when to offer explanations, taking into consideration dependencies such as the context factors (the user, the task, and the environment) to help stipulate the initiative.

Timing The timing refers to when an explanation is given. For example, explanations can be shown before, during, or after an event in the system [251, 80, 61]. Accurate timing is essential for human brain function and it shapes and supports brain function and how humans perceive and react to inputs [252]. Therefore, timing is also an important aspect of information processing and how users will make sense of the received information and react to it during system use. Timing gets even more important in the context of critical systems, when understandability is crucial for quick decision-making. This was also reflected in the survey results (cf. subsection 4.3.1), since participants seemed to have a greater need for explanations while driving.

Content

The content considers the information conveyed in the explanation. This category is divided into *information type*, *information density*, and *adaptivity*.

Information type Information type considers the aspects of a system that should be explained. The information found in the literature was classified into the following categories: about aspects of the context, causality information for an event or behavior, or about the system's inner logic. Context-specific information refers to the underlying data, namely the impacts and events associated with a certain context [253, 240]. Explanations about causality are more specific than context information, since they convey information about correlations between inputs and outputs in the system, as well as the specific reason for a certain system behavior or decision [242, 254]. Explanations about inner logic disclose details about how a system's algorithms work, as well as system processes, i.e., the system's rationale [234, 255].

Information density The information density is the amount of information that is presented [233, 235] or the abstraction (or granularity) level of this information [256, 257]. For instance, users may receive an explanation containing only partial information about an aspect that does not necessarily cover all details (e.g., all details of the system's rationale) so to not overwhelm them. Explanations can range from very detailed and specific to more general and abstract information.

Adaptivity Explanations can be context-sensitive (e.g., present information that corresponds to the current context) or static (e.g., about the general logic of an algorithm) [258, 259]. Explanations can also be controllable if the user has the power to change the explanation (e.g., expand an explanation) [242, 196]. The personalization is the ability of the explanation to adapt to the wishes or characteristics of a specific user [80].

Presentation

Presentation is the way that the information in an explanation is made available to users. This category is divided into *medium*, *tone*, and *grouping style*.

Medium The medium is "a method or way of expressing something" [260]. Therefore, the medium is the means to express explanations. The most popular media are textual [242, 259], visual (e.g., icons, videos) [69, 80], and auditory (e.g., tones, speech) [69, 251]. Since there are many forms of human-machine interaction, I do not discard the potential of giving explanations (or information in general) in different ways.

Tone The tone is "a quality in the voice that expresses the speaker's feelings or thoughts, often toward the person being spoken to" [261]. Hence, the tone of an explanation is related to the feelings or moods transmitted with the explanation. The spectrum of possibilities ranges primarily between very factual explanations, that focus merely on hard facts [262, 255]; and emotional explanations, that focus on building a sense of closeness with the addressee [242, 263].

Grouping The grouping determines how many explanation types are presented to users at the same time or in combination. For instance, graphics can be combined with text explanations or voice overs. Basically, there is the possibility to display one explanation at a time (single) or to group several explanations or explanation types (grouped). In the latter case, either explanations with different presentations but the same content can be combined [235] or several individual explanations can be displayed together [238].

8.2.3 Evaluation

To evaluate the quality of explanations or the explainable system, evaluation procedures and corresponding measurements (metrics or indicators) should be established at this level. These measurements help to determine whether the chosen design solutions are capable of meeting the specified requirements. For a successful evaluation, the degree of granularity at which the evaluation will take place must be decided, and the appropriate methods and metrics must be selected.

Granularity

Evaluation can occur in two granularity levels:

Explanation The quality of an explanation *per se* is evaluated by focusing on specific properties or qualities that the explanations should fulfill (e.g., comprehensibility, correctness).

System The quality of an explanation is evaluated in terms of how it impacts the perception of the user toward the system or in terms of the effects on a certain quality aspect that was established as an objective before (e.g., influence of explanation on system acceptance, performance, usability).

Method

Evaluations frequently focus on user feedback because the quality of an explanation is typically dependent on perception. As a result, qualitative methods are frequently used. In the analyzed literature, surveys (in the form of perception questionnaires or interviews) are the most widely used qualitative method for evaluating explanations since they allow for a great deal of flexibility, allowing quality to be assessed "by gathering information from or about people to describe, compare, or explain their knowledge, attitudes, and behavior" [264]. In principle, the evaluation strategies for SE recommended by Wohlin et al. [120] are appropriate for measuring the quality of explanations: surveys, experiments, and case studies. Experiments can be used to understand the effects of explanations on users in a controlled setting, while case studies can help to evaluate explanations in a real world context.

Metrics and Indicators

In general, evaluation distinguishes between focusing on subjective metrics (qualitative) and measurable metrics (quantitative). Even though quantitative approaches are used less frequently to evaluate explanations and their impact on systems [235], some authors also recommend to additionally perform a quantitative evaluation [236, 238, 251]. Van der Waa et al. [236] emphasize that, by combining both methods, "a complete perspective" on the effects of an explanation can be obtained.

Quantitative metrics or qualitative indicators help to assess if the functional requirements meet the established NFRs. They measure if the requirements were fulfilled, and help to measure the quality in both degrees of granularity: system and explanation. Metrics and indicators are intertwined, meaning that they can be dependent on or interrelated between each other. For example, the impact of explanations on system acceptability can be measured both quantitatively (by measuring the frequency of system use) as well as qualitatively (by observing users' attitudes with respect to the system acceptability).

Quantitative Metrics Quantitative metrics are tied to quantifiable measurements. Some examples of quantitative metrics are: the frequency of system use [173], users' task performance [236, 265], users' learning rate [173, 249], users' focus [238], task completion time [173], number of explanation requests [251, 238], and the explanation exposure delta [266, 69] (which measures the difference between a score given by participants before and after the presentation of an explanation).

Qualitative Indicators Qualitative indicators are tied to subjective assessments that help assess quality based on the impact of explainability on other quality aspects or on explanation properties. On the explanations granularity level, explanations can be evaluated by focusing on particular explanation properties*. Properties of explanations often mentioned in the literature are the comprehensibility of an explanation, its relevance, its correctness, its timeliness, its completeness, its persuasiveness, and its usefulness [243, 268, 267]. Alternatively, on the system granularity level, explanations can be evaluated in terms of how the user perceives their impact on a given quality aspect (e.g., satisfaction, transparency, trust).

8.3 Limitations and Threats to Validity

The conception of the framework was based on discussion among two researchers based on information extracted from the literature. As a result, we may expect a certain amount of

*Vilone and Longo compiled a list of explanation properties (cf. [267]).

subjective decision-making during paper analysis as well as category formation and structuring. Furthermore, because the literature review was not as thorough as a systematic review, the information gleaned cannot be deemed complete. To reduce subjectivity during paper analysis, weekly discussions were held to analyze the substance of the papers and clarify misconceptions. Feedback from practitioners and requirements engineers was used to improve the framework. However, the framework was not thoroughly validated. To mitigate this issue, we organized weekly reviews during which we addressed concerns and established agreements until a suitable version was acquired. We discussed the framework with other researchers to reduce the extent of researcher bias. Nonetheless, I am confident that the framework provides a good starting point to be further extended in future research.

In conclusion, this chapter presented a quality framework that aims to support RE for explainable systems by guiding software engineers through the process of identifying explainability-related quality goals, as well as specifying and evaluating explainability requirements. In the following chapter, I will look at how this framework was applied in a case study to help with the integration of explanations into an existing navigation system.

9 Case Study

The application of the proposed framework was demonstrated through a case study in a software company. The case study had two focus: 1) evaluate the framework by gathering feedback from practitioners in a workshop and during the development process and 2) evaluate the explanations that were designed with framework support.

The first two levels of the framework were used to support the requirements engineering process and the integration of explanations in an existing software product: a collaborative navigation software application, helping to examine the utility of the framework for software practitioners. Afterwards, the third level of the framework (*evaluation*) was used to plan and design an experiment to evaluate the quality of the newly integrated explanations. Moreover, to guide the process of defining NFRs and explainability requirements, a quality model was built with framework support. In this chapter, I discuss the case study in three phases: the elicitation of explainability requirements, the design choices for explanations, and the evaluation of the newly added explanations.

In short, the framework and the process reference model are complementary. Practices can be executed using the framework as a support to understand which aspects of explainability should be considered. Therefore, the categories in the framework supported the practices that

The research described in this chapter was performed in collaboration with three researchers: Florian Herzog, Verena Klös, and Kurt Schneider. The results were published in [230] and [231], on which this chapter is based. As a result, I occasionally use the pronoun "we".

were executed during the case study. In this chapter, I will indicate which practices were supported by each level in the framework with a color scheme* to connect the practices to the categories. I will also explain how the categories in the framework guided the practitioners during the practices in the case study.

9.1 Case Study Design

The case study happened in collaboration with the software company *Graphmasters GmbH.* Graphmasters is a small software company (52 employees) with an agile culture that focuses on navigation software products both for logistic companies and private end users. The development process and methods used are rather informal. As is typical of small software companies, the organizational structure has loosely defined roles, with members of a team frequently taking on multiple responsibilities [269]. For the case study, we focused on the company's main software product, *NUNAV Navigation*, which is a free navigation software application for smartphones based on swarm intelligence. The algorithm underlying the software application is based on the principle of "collaborative routing", which suggests a unique route for each member of the swarm and avoids traffic jams by dispersing cars to different routes.

The framework presented in chapter 8 was employed to support the definition of explainability requirements and the design of explanations. The framework assisted in defining abstract goals and in defining and analyzing requirements for explainability, which resulted in a quality model. It was also used to consider and design the method for evaluating the quality of the implemented explanations. The explanations were then incorporated into the system and tested in a two-week experiment and a subsequent quasi-experiment.

The five steps for case study research by Runeson and Höst [270] were followed: 1) The study was designed based on **RQ4** and its sub-questions. 2) Next, a protocol for the workshop was defined. 3) One researcher conducted the workshop and collected the data. 4) I and two fellow researchers analyzed and discussed the data. 5) The results were reported in a publication (cf. [230]).

9.2 Identifying Understandability Issues

As already explored in section 6.5, understandability is closely intertwined with explainability. In other words, user needs for more information are triggered by difficulties understanding

*the same color scheme introduced in chapter 7 and used in the framework in chapter 8, Figure 8.1

Topic	Questions
Collaborative Routing	What is collaborative routing / how does it work? Why do I need a permanent internet connection? Why are no standard routes displayed?
Navigation	Why is this route / detour taken? Why are the routes with this software application longer? Why is the position inaccurate?
Trust in the system	Where does the traffic / road closure data come from? What is happening with my data? Why do I need location services when starting the software application? How can I know if this route was well calculated?
Features	What does the coloring of the route show? How do I report a road closure? How do I enter my destination? How do I create favorites?

Table 9.1: User questions derived from reviews in the application stores and other support channels.

the system. This need for explanations can be addressed through explainability. The case company's stakeholders identified issues in the system's understandability, resulting in a desire for better system explainability.

For an initial overview of the existing understandability problems, we manually analyzed reviews in the Google and Apple application stores. Out of 304 inspected reviews, 46 were selected as presenting significant information about general issues. 33 of those reviews addressed understandability issues. In collaboration with the user support team, these understandability issues were examined and compared to the most frequently asked questions and problems reported by users through other feedback channels. The findings were summarized in a set of 14 questions that cover the most common understandability issues (Table 9.1). The questions were mostly about the working of the collaborative routing algorithm ("How does the collaborative routing algorithm work?"), about choices during navigation ("Why was this route chosen?"), about privacy, and about specific system functionality.

In order to define the goals and requirements for the explanations and to collect initial first implementation ideas, a workshop was held with seven participants (six software engineers and one customer support expert). This list was brought to the workshop with practitioners to be used as a foundation for prioritizing the understandability issues that explainability should address. A table that summarizes the workshop design can be found in Appendix D (Table D.3).

9.3 Dependencies

The first level of the framework is focused on gathering information that is relevant for the system design. As a result, this level is crucial (mainly) for elicitation, interpretation, and negotiation. The *objectives and constraints* were useful to guide the *vision definition* and *trade-off analysis*, when the quality goals and constraints were brainstormed and defined. The *context* category was useful during *stakeholder analysis* and *back-end analysis*, when we used personas to elicit and analyze context aspects. It is important to note that these practices occurred concurrently and intertwined rather than sequentially.

9.3.1 Objectives and Constraints

Vision Definition

The first step in using the framework is to define the vision to establish the objectives of the project that translate into quality goals. After the initial introduction on explainability, the results from the user feedback analysis were discussed. Three of 14 understandability issues were prioritized based on their relevance with respect to the business goals, the users' perception, and the explanation purpose. The selected issues concerned 1) how the collaborative routing algorithm works, 2) why a particular route was recommended, and 3) why the navigation is inaccurate.

First, to clarify the concept of explainable systems, the facilitator presented the definition proposed in this thesis (cf. section 6.2) and discussed concepts tied to *T*1. Next, the facilitator presented the framework and its categories and, by following the framework, each level of abstraction was discussed and derived into requirements in a quality model. The first level (dependencies) helped to think about what objectives and constraints existed, as well as the context of system use and stakeholders. These resulted in objectives that were translated into three quality goals which were refined into NFRs (after considering the constraints):

Objectives (Quality Goals): Usage increase, route acceptance, satisfaction.

Three explanation purposes were established based on the defined NFRs: algorithm transparency, route transparency, and route scrutability. These were refined into explainability requirements (Explainability Requirement (ER)1 - ER4). Finally, methods, metrics, and indicators were discussed and defined. Design ideas were also brainstormed and documented during the workshop.

Trade-off Analysis

Participants also identified two major obstacles. First, the explanations should be as little invasive on the system as possible, as stakeholders were concerned that the explanations might have a negative impact on the system. Because the system is a real-world commercial software application, stakeholders were rightly worried about the impact of explainability on customer loyalty. Second, due to a lack of human resources and strict project deadlines, stakeholders wanted to be cautious when allocating resources for the development of explanations. These obstacles translated into three constraints:

Constraints: System acceptability, human resources, complexity of implementation.

These constraints also illustrate the interaction between quality aspects. In a software project, several quality goals and constraints exist and interact with each other. Explainability can interact positively or negatively with other important aspects of the system (e.g., usability, development cost) [87]. This means that, during requirements analysis, quality models or other artifacts (e.g., SIGs or catalogues) must include the defined quality goals and constraints in order to identify and resolve conflicts between them. The scope of this study did not allow for a complete examination of trade-offs, which would have taken into account many other quality goals in the system that may interact with explainability. However, the framework described may be utilized to establish explainability requirements, and from there, other quality goals can be introduced to review and analyze the existing trade-offs.

9.3.2 Context

Stakeholder Analysis

To better understand the context aspects, two personas were created based on discussions with end users and on feedback analysis. Creating personas is a useful technique to identify and define different types of potential users for a system. It supports the identification, description and prioritization of user groups [206]. By modeling different personas, it is possible to model different user groups and consider each group's particular needs in terms of explanations. Personas may be used as a support during the identification and operationalization of requirements, to check whether the requirements meet the needs of the identified groups.

Modeling personas requires consideration of aspects such as user roles, level of user domain and technical expertise, and cultural values. Domain aspects also need to be understood in order to understand the particularities of a domain and what users need to know in order to perform their tasks within a system. Using personas, it is also possible to identify the main challenges, what elements of the process really need to be explained, and how to communicate information in a user-friendly manner. As discussed earlier, the domain language also needs to be considered and can be matched to specific user profiles.

The first persona was based on first-time users without experience with the system and its algorithm, while the second persona was based on frequent users. Furthermore, two use case scenarios were defined: 1) **outside active navigation**, which includes searching and selecting a destination, route overview, and starting navigation; 2) **during active navigation**, when the user is driving. The personas helped us to focus on understanding the users' backgrounds and their needs concerning explanations. By focusing on the aspects of the task, it was determined that time dependency is a relevant aspect for the system in case, as users often need relevant explanations immediately while driving in order to make a route decision. Aspects of the environment, such as the type of interaction, were also addressed. It was decided that explanations should be brief and interactions should be kept primarily hands-free.

Back-End Anaylsis

During back-end analysis, the architecture of the system is analyzed in order to understand how to provide the explanations. The practitioners also discussed what kind of data they would need to provide the explanations. *Nugraph* is the overall term in the architecture for the services which belong to the routing algorithm. The routing algorithm uses a predictive traffic model to calculate the individual fastest route between two points. The traffic on the route is predicted with the help of an ML model. All the data required for navigation (e.g. course and speed on the route) comes from *Nugraph*.

During this practice, the ideas were evaluated based on their feasibility. Because of the constraints, the simplest ideas were prioritized and chosen for implementation. Two of the chosen explanations are static, based on information already available on the company's website, and are not based on information provided by *Nugraph*. Therefore, only two explanations (cf. ER3 and ER4, subsection 9.4.1) are based on the algorithm.

Due to the established constraints, it was determined that the integrated explanations would be fairly simple and that none of the explanations would need access to the inner workings of the ML model. As a result, no interpretability approach was required to "open the black box". For one of the explanations, the current number of users on the route was needed. This information was simple to acquire through *Nugraph*. For the Global Positioning System (GPS) explanation, a simple algorithm was developed to estimate the accuracy of the GPS taking into account aspects like update time and hardware signals.

9.4 Characteristics of Explanations

The second level of the framework is focused on making sense of the information collected in the first level (that can be in form of high-level or raw requirements) and use it as a base to define more concrete requirements and meet design choices. As a result, this level is again crucial (mainly) for interpretation, negotiation, and also documentation. The categories in this level were all useful during *explainability design*. Low and high-level prototypes were some of the techniques used to support the design.

9.4.1 Writing and Documenting Requirements

Explainability requirements can be written in many ways. Rupp et al. [271] offer many different templates that can be used for writing good requirements. Balasubramaniam et al. [272] offer a template to write user stories in the context of explainable systems based on the definition proposed in this thesis: "As a <type of addressee>, I want to get explanation(s) on an <aspect> of a <system> from an <explainer> in a <context>".

In this case study, the explainability requirements (ERs) were formulated based on the Volere Template [273], since the software engineers were familiar with it:

ER1 The system shall explain the collaborative routing algorithm to the end user.

ER2 The system shall explain to the end user what information about traffic events (e.g., traffic flow, closures, roadwork, congestion) is taken into account by the algorithm.

ER3 The system shall inform the end user about the traffic volume on the current route during navigation.

ER4 If the accuracy of the positioning is unreliable, the system shall indicate to the end user that the positioning is currently unreliable.

It should be noted that these ERs are still in a relatively high-level form. They can be refined further to become functional explainability requirements that describe specific design choices that contribute to explainability. This is similar to the refinement of other NFRs to the concrete level of functional requirements. Security requirements, for example, are translated to functional requirements, which operationalize the established quality standards for system security. High-level NFRs, in general, can be modified by specifying quantitative indicators, human-checkable criteria, or operationalizing them into functional requirements. This is also true for explainability requirements.

We developed a quality model with framework support to assist practitioners in visualizing the choices made during the workshop and documenting the rationales for operationalizing requirements. Quality models are used to guide the specification of requirements related to abstract quality aspects, since the requirements defined in a quality model are broken down into more fine-grained constructs that can be measured in the lowest level of the model [237]. During the workshop, the facilitator presented the framework and its categories and, by following the framework, each level of abstraction was discussed and derived into requirements in a quality model.

Figure 9.1 shows the quality model that was created with framework support. Three NFRs were determined and further refined based on the objectives and constraints that were defined. Through explainability, stakeholders expected to promote quality aspects such as usage increase (NFR1), route acceptance (NFR2), and satisfaction (NFR3). The increase in system usage means that users must use the system more frequently for navigation. For route acceptance, participants determined that the recommended route should be accepted by users, meaning that users should adhere more closely to the suggested route. Satisfaction was mapped to users' satisfaction with the recommended route.

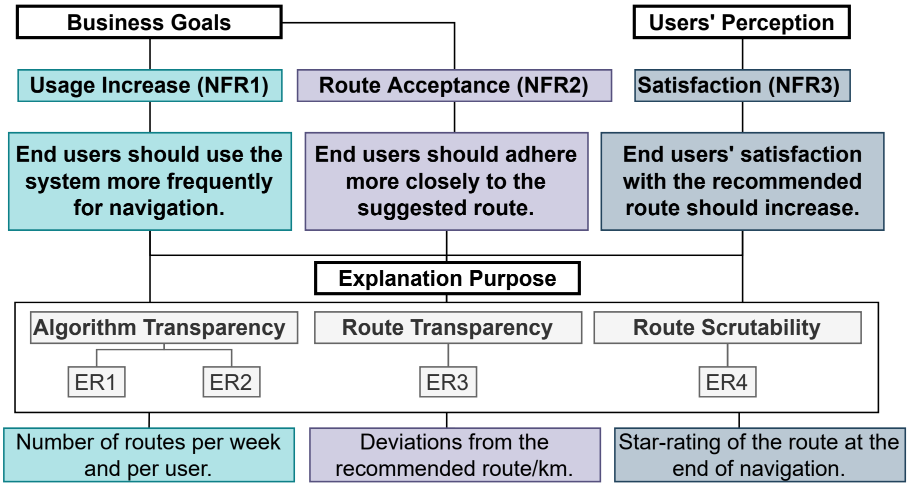

Business Goals
Users' Perception
Usage Increase (NFR1)
Route Acceptance (NFR2)
Satisfaction (NFR3)
End users should use the system more frequently for navigation.
End users should adhere more closely to the suggested route.
End users' satisfaction with the recommended route should increase.
Explanation Purpose
Algorithm Transparency
Route Transparency
Route Scrutability
ER1
ER2
ER3
ER4
Number of routes per week and per user.
Deviations from the recommended route/km.
Star-rating of the route at the end of navigation.

Figure 9.1: A quality model derived with support of the framework

9.4.2 Design Choices

Explainability Design

The *characteristics* level of the framework was used to brainstorm specific design choices that would help to refine the ERs into functional explainability requirements. In the end, each ER corresponds to one explanation (either single or grouped). Aspects of demand, content, and presentation were considered to guide design decisions. An overview of the ERs and the corresponding design choices, based on the framework, is shown in Table 9.2.

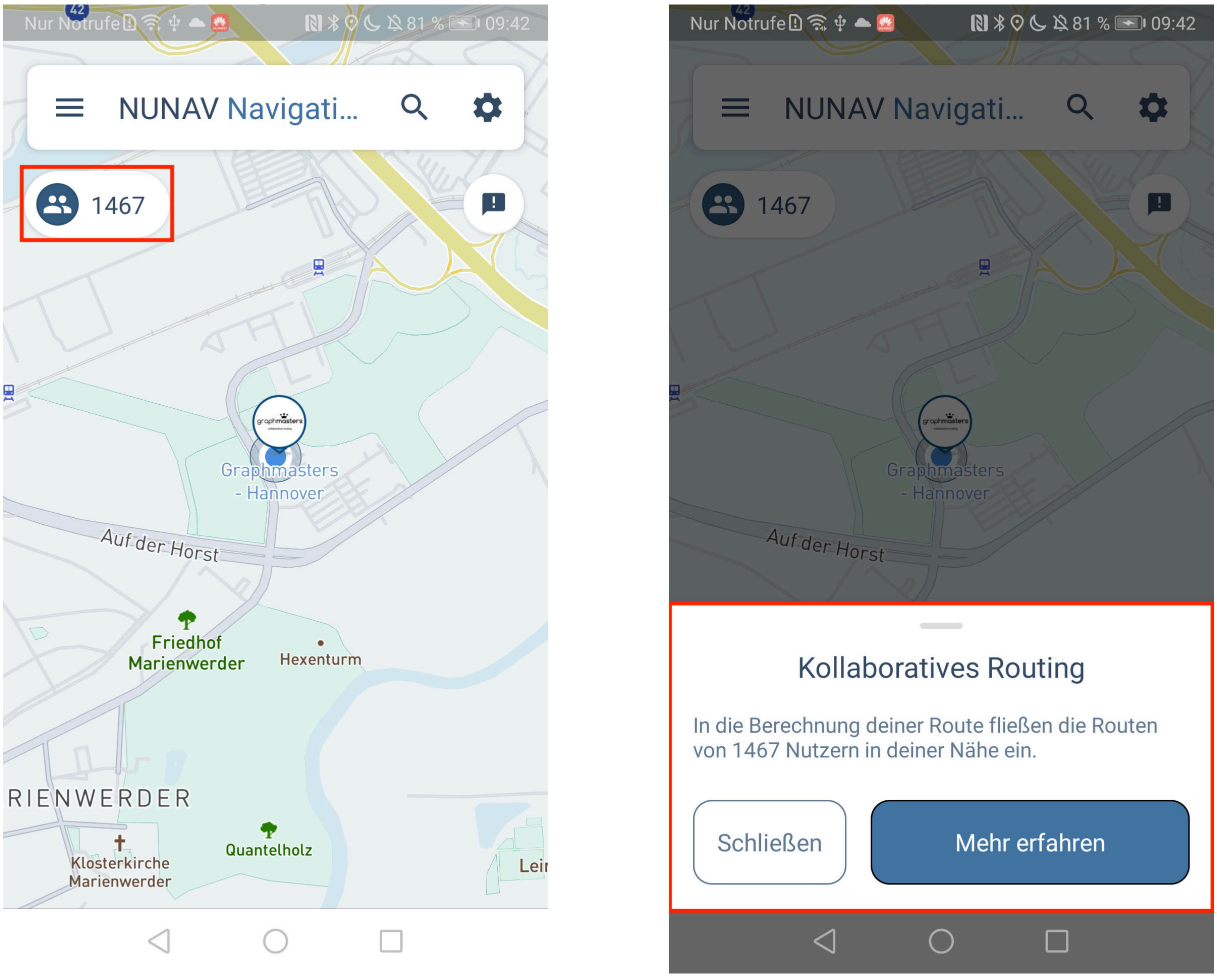
Table 9.2: Explanation requirements and design choices based on the characteristics in the quality framework

		Explainability Requirement			
Characteristics		ER1	ER2	ER3	ER4
Demand	Initiative	User	User	System	System
	Timing	During navigation	Before navigation	Before and during navigation	During navigation
Content	Inf. Type	Inner logic	Inner logic	Context, causality	Context
	Inf. Density	Brief and expandable	Long explanation	Short	Short
	Adaptivity	Static	Static	Context-sensitive	Context-sensitive
Presentation	Medium	Textual, visual	Textual	Textual, visual, auditory	Textual, visual
	Tone	Factual	Factual	Factual, emotional	Factual
	Grouping	Grouped	Single	Grouped	Grouped

The explanations were created using design mock-ups such as low and high fidelity prototypes, which were discussed and improved with the team over a month of iterations. When making design decisions for the explanations, it was especially important to follow the established constraints and ensure that explanations did not interfere with other aspects of quality, particularly usability. The following paragraphs explain how the characteristics level influenced the design decisions for each ER.

ER1 – Routing Algorithm

The goal of this explanation was to achieve more algorithm transparency. During navigation, an icon was shown with the total number of users influencing the calculation of the present route. By clicking on the icon, users could request an explanation about the algorithm's inner logic. Hence, two presentation medium approaches were grouped: visual and textual. The collaborative routing algorithm works by calculating each route depending on the routes of other users. To shed more light on this logic, the explanation of the routing algorithm has been implemented in two different levels of granularity. First, users could click on an icon with the number of users that would influence the routing. By clicking on it, a small window on the bottom of the screen appeared, containing a short explanation, a "close" button, and a "learn more" button. In a first version, the explanation was "In the last 15 minutes, we have collected anonymous traffic data from [number of users] users in your area.". When discussing



Kollaboratives Routing

In die Berechnung deiner Route fließen die Routen von 1467 Nutzern in deiner Nähe ein.

(a) Screen 1

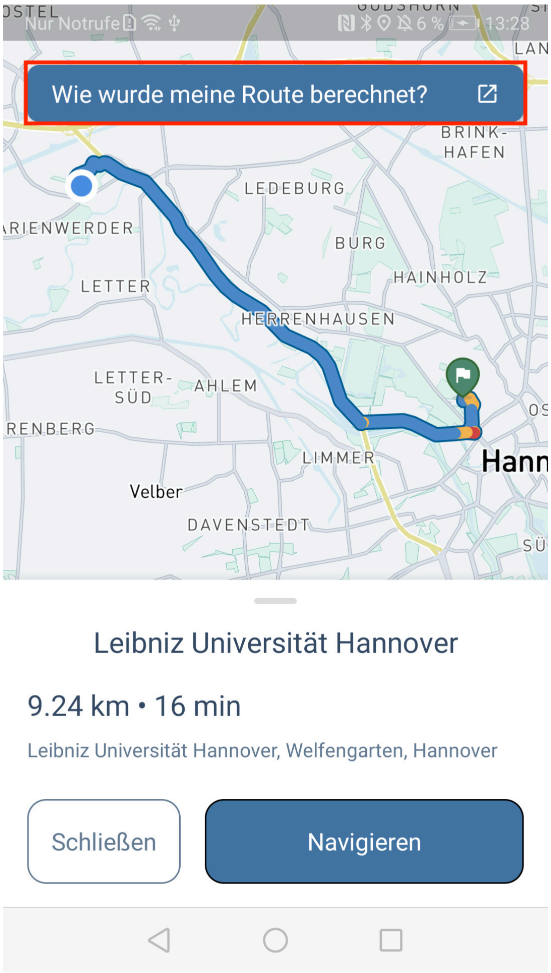
(b) Screen 2

Figure 9.2: Final design for ER1

this sentence with company stakeholders, it was concluded that it was too technical for the users. Therefore, we rephrased it to “The routes of [number of users] users in your area are included in the calculation of your route”, to make the relevance of this information to the users become more clear. As a further explanation option, users could click on “learn more” (“Mehr erfahren”) to jump to the corresponding help center article that would explain the details of the collaborative routing algorithm. A link from the software application directly to the help center did not exist in the application before this work.

ER2 – Influence of Traffic Events

This explanation also aimed to increase algorithm transparency. Before beginning navigation, users could request an explanation of the traffic events relevant to the algorithm (the algorithm’s inner logic). By clicking on the question “How was my route calculated?”, users were redirected to a page with a long textual explanation. In this case, a single static and factual explanation was presented. Closures, road works, and traffic jams were already signalized on the map, which is the main interaction element of the software application. According to user feedback, this contextual information was not understandable enough. Therefore, a new help center article was created as part of this work on this topic. This article is suitable for users who are using the application for the first time. Additionally, the option to access this article was made available before the navigation began. The option to access this explanation was integrated into the route preview in order to avoid clogging up the default map view with multiple ways to access help center articles. During route preview, a button with the question "How was my route calculated?" was provided at the top of the window. The user would be redirected to the corresponding article if they clicked on it. Several prototypes were designed to gather feedback from the business stakeholders and other team members until the final design was approved.



Leibniz Universität Hannover

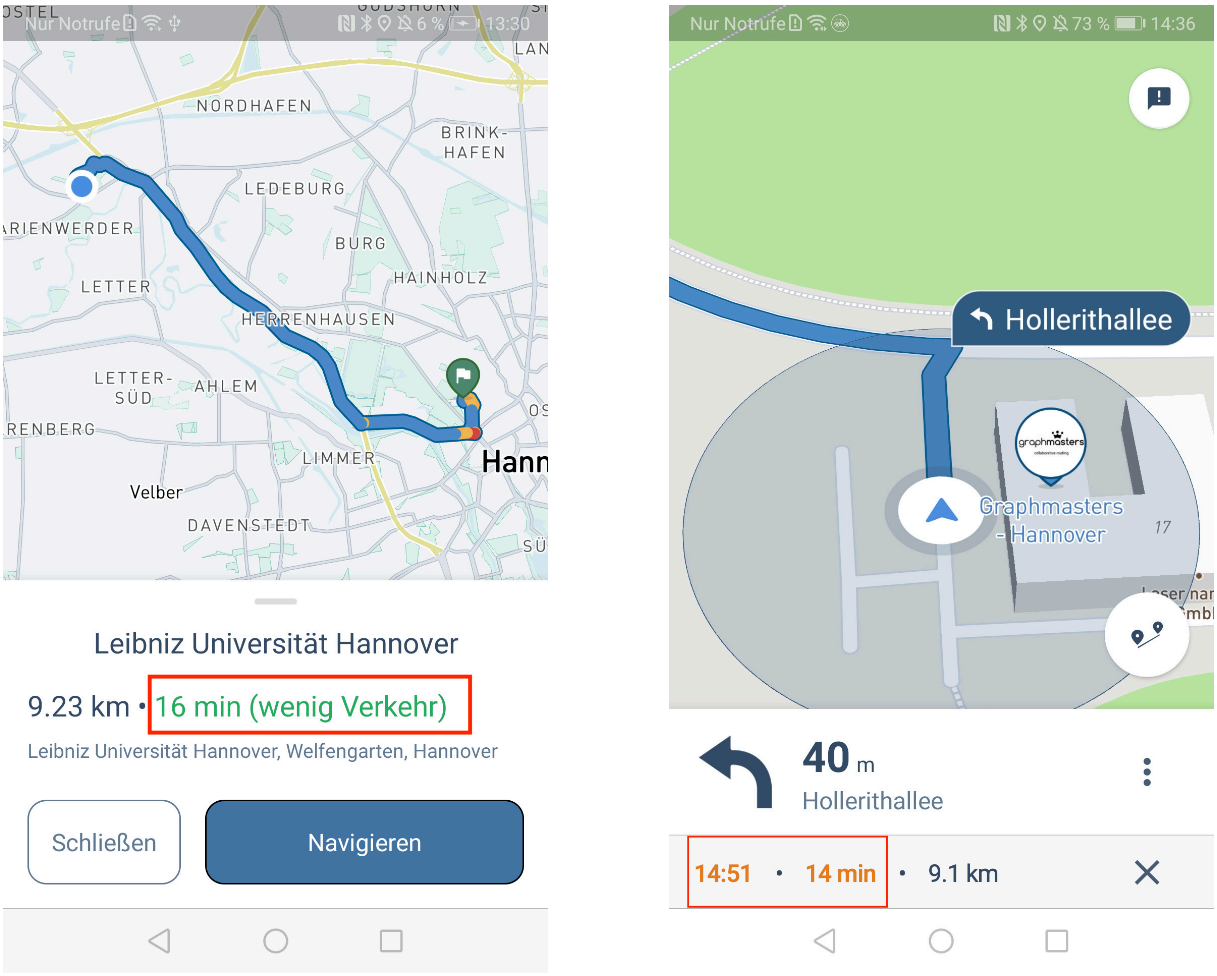
9.24 km • 16 min

Leibniz Universität Hannover, Welfengarten, Hannover

Figure 9.3: Final design for ER2

ER3 – Traffic Volume

This explanation should inform about traffic volume to achieve more route transparency. Before navigation start, brief explanations in a factual tone were presented automatically (system initiative) via visual color communication (e.g., red for roads with heavy traffic) grouped with short texts (e.g., "heavy traffic"). During navigation, information about traffic volume was provided via synthesized speech. Three presentation medium approaches were grouped: visual, textual, and auditory.



(a) Screen 1

(b) Screen 2

Figure 9.4: Final design for ER3

Explanations were context-sensitive, i.e., they varied according to the current traffic situation. The goal of this explanation was to explain possible reasons for what, from an end-user' perspective, could be considered as a "weird" route. The chosen approach to explain this was to display traffic events on the current route. This context information should help to understand why the application recommends unusual routes. However, this is not trivial, since the calculation of the traffic volume and the subjective perception of the end user about it is not clear. Another problem is that it was a technical challenge to define which route would be subjectively perceived as “normal” to each end-user on the same route. Therefore, the chosen solution was to calculate the ratio of average route duration and use it as baseline for comparison. The data is then mapped to “low traffic”, “moderate traffic”, and “high traffic” levels. There is also a “normal” traffic situation where no explanation is shown to the users.

Internal test drives were used to determine the threshold levels. The information might be obtained by users in three ways. The information was first provided in the route preview, along with information on the journey duration. The route’s total travel time was represented in green (low traffic), standard text color (regular traffic), orange (moderate traffic), and red (heavy traffic) (heavy traffic). Since it was discovered in the initial prototype that the colors alone were not relevant, a brief explanation was provided. Furthermore, the arrival time and journey time are presented in the appropriate color during the navigation and are updated as the traffic situation changes. During navigation, information about traffic volume was provided via synthesized speech as shown in Table D.4 (Appendix D). The tone of these voice prompts was somewhat emotional to create sympathy.

ER4 – GPS Inaccuracy

The purpose of this explanation was to inform users about GPS inaccuracy and alert them to potential route errors. During navigation, the system took the initiative to inform the user that the GPS was inaccurate by providing a brief explanation in a factual tone along with a circle on the map representing the GPS range (context information). The three presentation medium approaches (visual, textual, and auditory) were grouped. The third explanation is related to scrutability. The purpose was to alert users when they could not rely on the route’s present position. This is especially important in tunnels, when GPS is not as reliable. To determine whether the GPS is inaccurate, a new algorithm has been added into the navigation software application. In this scenario, not only the accuracy (as determined by the GPS module) but also the time of the latest GPS update were taken into account. The first design consisted only of an icon containing the text “inaccurate position” and a voice-over repeating the same information. Since voice overs are often deactivated by the end-user, the visibility of the explanation during test drives turned out to be insufficient. Therefore, in the final design, we changed the position of the explanation and added a circle on the map to indicate the accuracy range.

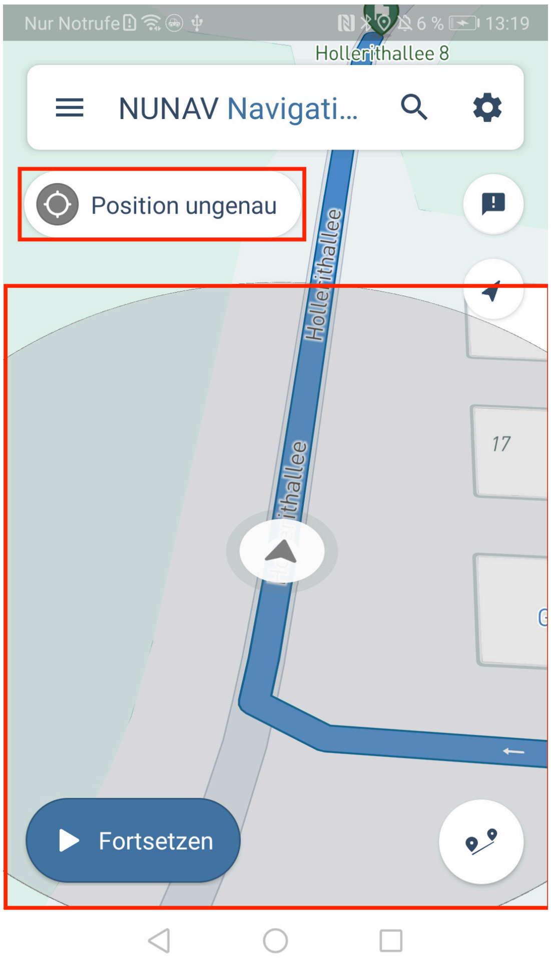


Figure 9.5: Final design for ER4

9.5 Evaluation of the Explanation Quality

Evaluation

The explanations were implemented into the navigation software and a quantitative evaluation based on a two-week experiment was conducted to evaluate the influence of the explanations on the discovered NFRs. The results were complemented with feedback obtained through a qualitative evaluation from a quasi-experiment with four users.

To test whether the system meets the defined quality goals, the metrics and indicators defined in the quality model (Figure 9.1) were assessed: 1) the number of routes per week and per user, to determine the usage increase; 2) the deviations from the recommended route per km, to determine the adherence of users to the recommended route and the system acceptance; 3) the user feedback (star-rating of the route at the end of the navigation), to assess the reported satisfaction with the route.

We expected that integrating explanations would be beneficial, or at least not negative. Since navigation is a context-sensitive task, we also expected that the context-sensitive explanations would have a more positive impact. To assess whether the explanations helped to achieve system acceptance, we analyzed the deviation from the recommended route.

In summary, we investigated the following hypotheses:

H1 Giving an explanation is always better than giving no explanation, or equally good.

H2 Context-sensitive explanations (ER3 and ER4) increase the system usage (NFR1) and user satisfaction (NFR3).

H3 To increase the system acceptance (NFR2), additional explanations concerning the routing algorithm (ER1 and ER2) are needed.

9.5.1 Quantitative Evaluation

We performed a two-week experiment with 9,745 participants. To compare the effect of different explanations, we performed a between-subject analysis, i.e., we compared the results between groups of participants that received different explanations as shown in Table 9.3.

Table 9.3: Overview of the data of the subject groups

Group	Explanation	Users	Routes	With Star Rating
Group 1	none	1778	4807	133
Group 2	static	1397	3413	135
Group 3	dynamic	468	4571	184
Group 4	all	369	3740	173
Total		4012	16531	625

Table 9.4: Mean values (and standard deviation) for collected indicators in the case study

Group	Frequency (routes/week)	Deviation (N/km)	Satisfaction (avg. stars)
Group 1	3.223 (3.169)	0.075 (0.129)	4.55 (0.86)
Group 2	3.198 (3.113)	0.079 (0.127)	4.58 (1.04)
Group 3	4.423 (4.251)	0.079 (0.136)	4.73 (0.85)
Group 4	4.438 (3.937)	0.072 (0.128)	4.60 (0.83)

The study results are depicted in Table 9.4 and described below. For all results, a Kruskal-Wallis-Test and a subsequent Dunn-Test with Bonferroni adjustment was used to analyze the statistical significance of detected differences between groups. Only significant results are reported.

Frequency of Use

We counted the number of routes driven each week per user during the testing period. For further analysis, all outliers were removed from the data set, resulting in 3,937 remaining users. The results show that the frequency of use increases when users receive context-sensitive explanations (around 4.4 routes in groups 3 and 4) compared to users that receive no explanations or only static explanations (around 3.2 routes in groups 1 and 2).

Adherence to the Recommended Route

After filtering out navigation data with frequent inaccurate positioning (as it significantly influences deviations from the route) and eliminating outliers, 16,314 routes remained in the dataset. When comparing the average number of route deviations per km, it is possible to see that when users receive both static and context-sensitive explanations (0.072 deviations/km), they deviate less from the route compared to those receiving no explanations (0.075 deviations/km) or only one kind of explanation (0.079 deviations/km).

Satisfaction with the Recommended Route

The results show that users give a significantly more positive rating for the route when they are shown context-sensitive explanations (average 4.73 stars vs. 4.55 stars without explanations). However, when users additionally received the static explanations, this effect is eliminated (average 4.6 stars). The individual ratings show that ratings in the "static explanations-only" group were more polarized between one and five stars, while contextual explanations mainly resulted in a shift toward more five-star ratings (cf. [274]). When both sorts of explanations are combined, it is reasonable to expect that these effects will interact. However, no definitive response can be offered based just on the statistics.

Demand for Explanations

To assess the demand, we recorded how many users requested the interactive static explanations and spent more than 1.5 seconds in the dialogue (to exclude accidental requests). In general, functions are considered irrelevant, if they are used from less than 5% of the users. Thus, an explanation is considered relevant, if at least 5% of the users demand it. The results in Figure 9.6 show that this demand exists for both static explanations. The detailed explanation

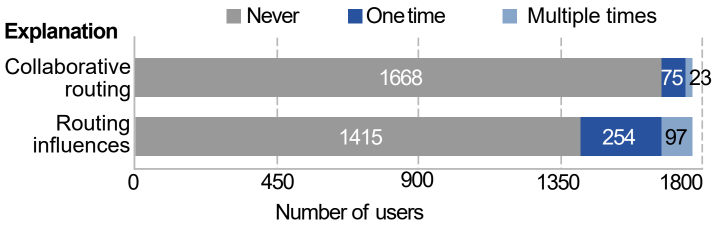


Figure 9.6: Number of users and accesses for ER1 and ER2

about the routing algorithm (ER1) was requested by 6% of the users in the study, while the explanation about the influence of traffic events (ER2) was requested by 20% of the participants.

Perceived Usefulness

To analyze the usefulness of the static explanations, we provided the possibility to rate the explanation as "useful" or "not useful". The results are depicted in Table 9.5. Both explanations were mostly rated as "useful" (90 % for ER1 and 87 % for ER2).

Table 9.5: *Usefulness* of help center articles

Article	Useful	Not Useful
Collaborative Routing	76	9
Influences on Route Calculation	209	32

In summary, the results suggest that providing an explanation has either a positive or no effect on the quality metrics, confirming hypothesis H1. Context-sensitive explanations, in particular, increase system usage (NFR1) and user satisfaction (NFR3), confirming hypothesis H2. However, there is no substantial favorable benefit for the static explanations alone, once again evidencing the importance of the context when considering explanations.

When combined with context-sensitive explanations, they had a positive effect on NFR2 (less route deviations), confirming hypothesis H3. As this combination of static and context-sensitive explanations eliminated the positive effect of context-sensitive explanations on route satisfaction, no definitive conclusions can be drawn about static explanations.

9.5.2 Qualitative Evaluation

In a subsequent quasi-experiment, we investigated the influences between the explanations and other quality attributes. The goal was to better understand the experiment results and to detect concrete problems and potential improvements.

All participants in the quasi-experiment were familiar with navigation applications, and three of the four participants (three men, one woman, 23 - 55 years old) used navigation applications regularly. All personas were represented: two participants had considerable experience with *NUNAV Navigation* (used >10 times), one participant was familiar with the software application (used two to three times), and one had never used the software application before.

During the study, participants were asked to interact with all explanations (ER1 - ER4) and to evaluate them by ranking statements on a Likert scale. The scales were used to assess the usefulness, perceived transparency and completeness of the explanations, as well as the demand for the respective explanation.

In addition, the participants were asked to share any thoughts they had while interacting with or reading the explanations (think-aloud protocol). Following each explanation, participants could suggest changes or point out missing information. The feedback was mostly positive and provided valuable insights for further improvements on the explanations. The feedback is summarized below.

Demand for Explanations

For all explanations, three of four participants stated that they needed those explanations. Two participants would have wanted to be able to hide the static explanations because they were only needed at the beginning. Nonetheless, they did not obstruct navigation. The demand for the explanation connected to ER2 was lower than for ER1. Some enhancements were proposed for the static explanations, including making the explanation connected to ER1 more clear and, in general, opening explanations within the software application rather than requiring an external browser. Some users mentioned a desire to hide the static explanations when they are no longer needed.

Explanation Content

To assess the quality of the content of the explanations, we assessed the satisfaction with and usefulness of the explanations, as well as their perceived transparency and completeness.

With one exception, participants scored the utility and understandability assertions as positive or neutral for all explanations. The same is true for claims about the perceived transparency and completeness of static explanations. The other participant would have liked to know when the collaborative routing has a direct impact on navigation. This would call for an additional context-sensitive explanation, which might be challenging to provide in terms of required data and implementation.

For the context-sensitive explanations, the feedback differ significantly. The content of the explanation tied to ER3 was only evaluated positively. Furthermore, two participants had previously favorable experiences with similar functionalities of competing products and appreciated the explanation since it was familiar.

Two participants rated the explanation connected to ER4 badly in terms of satisfaction and perceived transparency after failing to access an explanation by clicking on an icon. This occurred because the participants were used to this form of engagement from previous interactions in the software application and expected to receive another explanation on demand. This is a clear consistency problem (as defined by Nielsen [275]).

Presentation of Explanations

During the experiment, participants provided input on some presentation aspects. Two participants suggested the use of more visual elements for the explanations tied to ER1 and ER2. Three participants complimented the audio prompts in the explanations for ER3 and ER4, especially considering the driving context.

9.6 Limitations and Threats to Validity

The applicability and usefulness of the quality framework was evaluated in a corporate context in a live system to achieve realistic conditions. However, this evaluation was performed with only one software company and in an agile development context. Thus the results can only be seen as a positive indication but cannot be generalized to other software development settings. Since the organization employs an agile strategy with frequent releases, changes are rather gradual. Because of this, the time window for implementing the explanations and evaluating them – one release assessed by a two-week experiment – was perhaps too small. Another potential drawback is that one of the authors led the workshop when the framework was utilized as the workshop facilitator. The participants' impression of the framework's effectiveness may have been impacted because its use did not occur independently. The workshop facilitator made an effort to explain each category objectively while encouraging the participants to provide constructive criticism in order to impose as little interference as possible.

Additionally, the design choices for the explanations were constrained by trade-offs between explainability and potential detrimental consequences on, for example, consumer loyalty. This calls into question whether a higher (or lower) favorable effect might have been obtained with a more thorough exploration of other design options (at the same time, it also illustrates a typical RE challenge). The generalization of results is also limited due to the population size in the quasi-experiment (four participants). However, I am confident that we could provide important insights on the subject, paving the way for new evaluations. In this sense, replications or a further case study with a different application type are valuable.

9.7 Summary

Considering NFRs in a system is commonly linked to potential conflicts and trade-offs. To resolve these conflicts, requirements must be negotiated and decisions must be made to either adapt or prioritize them. In the case of explainability, its positive or negative effect depends on how explainability is refined to more fine-grained requirements, how these requirements are operationalized, and how it affects other NFRs [12]. As a result, appropriate design decisions must be taken to ensure that explanations help to achieve the desired goals rather than hurting other quality aspects in the system.

The results indicate that it is difficult to assess explainability's effects since different design decisions interact and have an impact on quality. For instance, we could see that an explanation cannot fully accomplish its goal if it is not readily available (like the explanation for collaborative routing, which required an active click on an icon). Similarly, an inconsistent interaction design (e.g., if only some icons provide additional explanations after a click) can have a negative impact on user satisfaction. To avoid these design issues, heuristics or guidelines for explainability design, similar to the ones proposed for user interface design [275], can help.

Additionally, as suggested by one participant, more context-sensitive information may be useful in linking the general understanding of algorithm internals to specific contexts. However, the quantitative analysis revealed a conflict between static and context-sensitive explanations of algorithm internals (with regard to route satisfaction). The reason for this observation can be manifold: explanations about algorithm internals might present general information that confuses a user who is experiencing an exception to the rule; or the tone of the explanations may be too positive, building false expectations (users might, for example, expect that they will not experience traffic jams anymore); or users may disagree with the logic implemented by the algorithm and, thus, dislike the result. More research is needed on how specific explainability design choices affect quality.

The most significant trade-off in requirements engineering is the relationship between cost and design effort for software quality since software is primarily assessed from an economic standpoint [19]. Projects are usually subjected to financial and schedule pressure, which frequently constrains whatever quality goals are set. "Traditional" constraints had a significant influence in limiting the solution space of explanations and the use of methodologies to create and assess them: staff and time limits, concurrent deadlines, and prudence in implementing changes that may result in customer loss.

While the beneficial benefits would have been stronger if these limits had not existed and more "elaborate" explanations had been developed, even the inclusion of "simpler" explanations had a favorable influence on the set quality goals. Indeed, the positive outcomes of explanations were so well embraced that several of the explanations (ER1, ER2, ER4) were developed further and are still in use as of the submission of this thesis. Because the algorithm that sent data for the explanation needed to be modified, ER3 was temporarily removed.

In this case study, we found that the quality framework assisted in the discussion of explainability, design choices, and evaluation. Practitioners' comments were also positive. When asked about their experiences with the framework, they reported that it helped them think about user profiles, use cases, constraints, and high-level requirements; it helped them think about more specific requirements on explanations (e.g., demand, content, presentation) and to meet design choices; and it helped them think about strategies to evaluate the impact of explanations on the established quality requirements.

Answering RQ4

> *I propose a quality framework to assist practitioners in the requirements engineering process for explainable systems. The quality framework consists of three abstraction levels that help practitioners transform abstract goals into concrete and measurable requirements that fulfill stakeholders' needs.*

Using frameworks and catalogues to discuss quality requirements is a typical RE activity [276, 17]. Because the example organization had an agile development culture, the concept of clearly specifying quality goals and defining requirements was first viewed with skepticism. However, participants (software engineers) reported that the framework facilitated dialogue about explainability goals, helped to establish a better vision, and aided in the development of a suitable design and evaluation strategy. As a result, it is possible to say that the framework provides a valuable basis for assisting with the RE process of explainable systems. The framework can be refined and tested in future research to keep up with state-of-the-art developments in explainability.

10
Conclusion

The first practical step to build quality systems is to perform good requirements engineering, which aids in comprehending the underlying needs, defining acceptable requirements, and ensuring that these requirements are satisfied to a reasonable extent within the available solution space. Therefore, the success of a software project relies strongly on RE [277], and the success of RE relies strongly on communication and on shared understanding [26].

Explainability is a quality aspect that has become a hot topic again in the recent years, gathering attention from different communities from different areas of knowledge*. Therefore, it started to be looked at and investigated as a quality aspect of software systems by the RE community [62, 278, 279, 280]. Quality aspects are classically a difficult topic during RE since they have a very abstract nature which is grounded on real-word abstract aspects. This abstract nature is exactly what contributes to the difficulty in specifying these concepts in software aspects, and the establishment of a shared understanding makes this discussion easier. Software engineering artifacts (e.g., conceptual models, knowledge catalogues, frameworks) exist to mitigate this complexity by offering a common vocabulary, guidelines, or taxonomies that can help to understand the nature and boundaries of a given quality aspect.

*e.g., machine learning, HCI, philosophy, psychology, cyber-physical and recommender systems

This thesis focused on facilitating the analysis of explainability as a software requirement, helping to understand this quality aspect and providing artifacts that support software engineers during RE. This final chapter summarizes the findings of this thesis, discusses some of the implications of those findings, and hints at major areas of future research related to explainability.

10.1 Summary of the Findings

The research presented in this thesis generated two theories that are supported and formed by artifacts, which help to make the theories understandable and actionable. The first theory is an *analytical-explanatory* theory that explores the role of explainability in the quality landscape and how it impacts different quality aspects throughout the landscape. I proposed three artifacts that are tied to and shape this theory: a definition, a conceptual model, and a knowledge catalogue. These artifacts should assist software engineers in better understanding the concept of explainability in the context of information systems, as well as demonstrating the importance and relevance of explainability as a quality requirement for future systems.

The second theory is a *design and action* theory that focuses on practitioners' needs to propose actions. The theory is shaped by two artifacts: a process reference model that should support software engineers in supplementing the software lifecycle with practices relevant to the development of explainable systems, as well as an explainability quality framework to aid in the definition of explainability requirements ranging from abstract concepts to concrete requirements.

I believe the following three findings are the most relevant in this thesis:

① ***Explainability has a fundamental role in the quality landscape.*** Explainability impacts all dimensions of the quality landscape, and can be a positive catalyst to achieve many quality goals, contributing to the overall system quality. This is the case because explainability impacts three fundamental quality aspects: informativeness, transparency, and understandability. These quality aspects are crucial to software quality and help to achieve many other quality goals.

② ***Existing techniques suffice for the development of explainable systems.*** User-centered techniques are most suited for developing explainable systems and there is no need to "reinvent the wheel" with unknown techniques. I propose a set of lightweight practices that can be incorporated into the development process of explainable systems, and recommend the appropriate techniques for determining the system's general goals, requirements, and constraints, as well as explanation design and evaluation.

③ ***The proposed theories and artifacts can guide the RE process for explainable systems.*** The proposed theories and the artifacts that compose them help to break down the current initial barrier and guide the RE process to explainable systems. Furthermore, the artifacts contribute to the development of a shared understanding of the concepts and aspects of explainability.

10.2 Discussion

Each of the aforementioned findings will be examined in more detail in the next subsections.

10.2.1 ① The Relevance of Explainability

Explainability is a new NFR that echoes the demand for more human oversight of systems [281]. It can bring positive or negative consequences across all quality dimensions: from users' needs to system aspects. Explainability's impact on so many crucial dimensions illustrates the growing need to take explainability into account while designing a system. To this end, the first contribution of this thesis was a definition of explainability in the context of software systems for software and requirements engineers (section 6.2).

This definition points out what should be considered when dealing with requirements and the appropriate functionality for explainable systems: aspects that should be explained, contexts, explainers, and addressees. Knowing these factors enables the elicitation and specification of explainability requirements, which supports the software development process. In this regard, the possible values for these variables (for instance, reasoning process as an element that has to be explicated) might work as a conceptual starting point for requirements analysis [†]. The proposed definition should also help to build a shared understanding of the semantics and structure of explainable systems. Overall, the supplied definition may be used as a guide to assist in the design of explainable systems and to make good design choices toward explainability requirements.

System Acceptability through Explainability

One of the key indicators determining information systems success is the level of system usage which is reflected in the technology acceptance by users [282]. System acceptability is often one of the ultimate goals of stakeholders when developing systems (also reflected in the case study's quality goals established by the stakeholders). According to Nielsen [5], system acceptability (also known as *system acceptance*) is "the question of whether the system is good enough

[†] As also suggested by Penzenstadler [64] with the definition for sustainable systems.

to satisfy all the needs and requirements of the users and other potential stakeholders, such as the users' clients and managers". As a result, stakeholders often strive (and also should) to achieve system acceptability.

Nielsen goes further and divides this concept of overall acceptability into social acceptability and practical acceptability. The practical acceptability involves the practical quality aspects of the system: its usefulness, cost, compatibility, reliability, security, etc. The social acceptability considers other aspects such as ethics and fitness to social standards. For instance, a profiling system can be useful and easy to use, being practically acceptable, but can be perceived by some as socially undesirable.

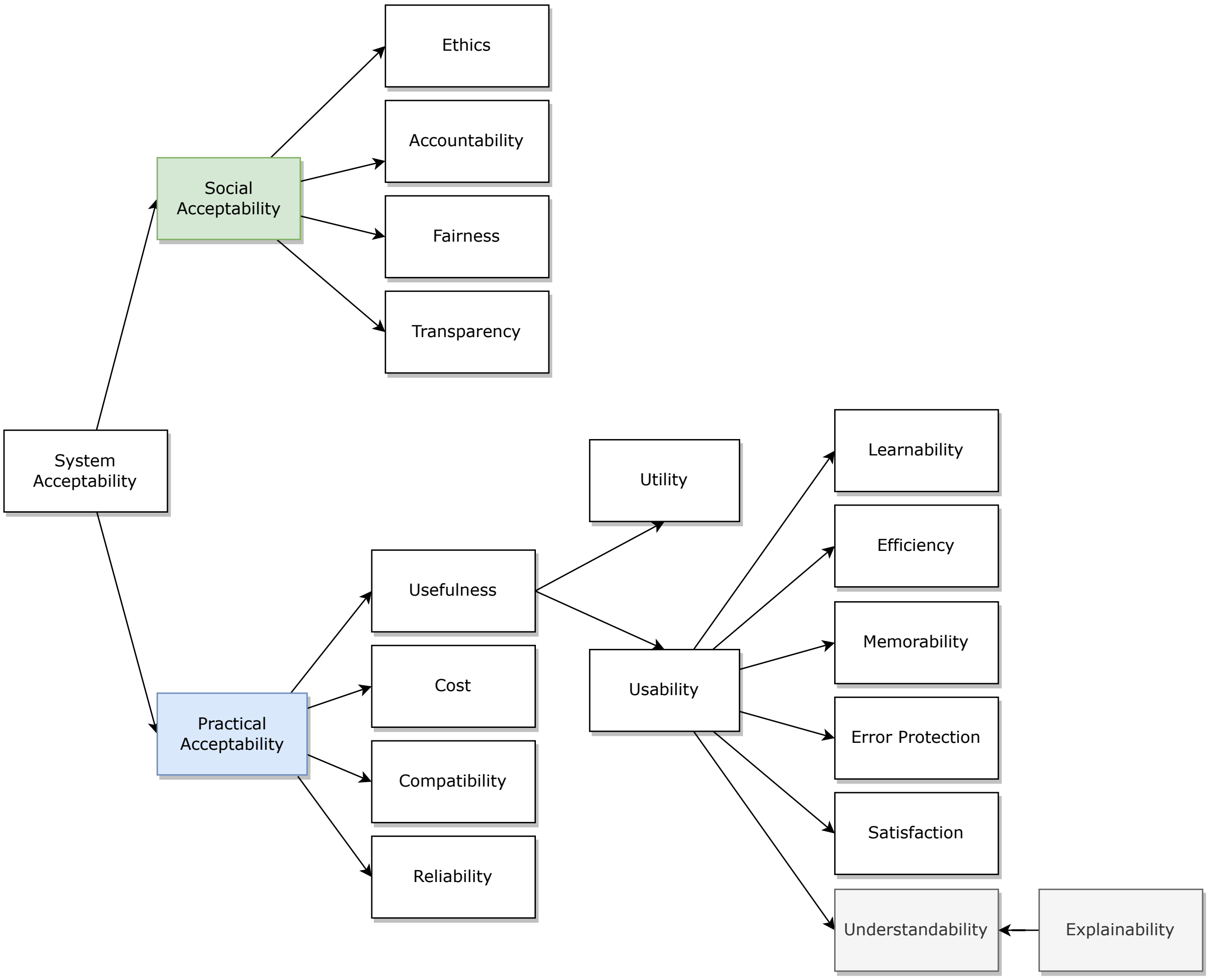


Figure 10.1: A model of the attributes of system acceptability, adapted from Nielsen [5]

Nielsen breaks down these quality aspects even further (extracted from [5]): "Given that the system is socially acceptable, it is possible to analyze the practical acceptability of the system relative to many different practical quality aspects, such as *usefulness*. Usefulness is the issue of whether the system can be used to achieve some desired goal. It can be further broken down into two other quality aspects: *utility* and *usability*. Utility is the question of whether the functionality of the system in principle can do what is needed, and usability is the question of how well users can use that functionality. [...] Usability applies to all aspects of a system with which a human might interact".

A model that illustrates these interrelationships between acceptability and other quality aspects is shown in Figure 10.1, adapted from Nielsen [5]. Note that usability influences usefulness, which in turn influences the acceptability of the system. Learnability, efficiency, memorability, error protection and satisfaction are among the quality requirements that help achieve usability (or, again, the usability *attributes*, cf. section 6.5). Since understanding a system is necessary for good usability and since usability should apply to all aspects of a system that a human might interact with, I added understandability as an additional component to this model. Furthermore, there are two ways to see the role of explainability. The first way is to consider its practical acceptability. First, in chapter 6, we could see that explainability is linked to understandability, usability, and system acceptance. Using Nielsen's model as a foundation, it is possible to draw the following conclusion: Explainability may assist in enhancing the system's understandability, which favorably affects usability and, in turn, favorably affects the system's practical acceptability (Figure 10.1). Besides, in chapter 3, we also saw that explainability can have an impact on many quality aspects that are relevant for social acceptability such as ethics, accountability, fairness, and transparency.

Another way to analyze explainability's role in software quality is to consider its social acceptability, with an emphasis on its ethical role. Vakkuri et al. [6] proposes a conceptualization of the relationships between key principles (which are basically quality aspects) in AI ethics. This conceptualization served as the foundation for my view of explainability's role in relation to social acceptability aspects such as ethics, accountability, transparency, and fairness, as well as more practical acceptability aspects such as understandability and usability. Figure 10.2 shows the adapted conceptualization.

In this adapted conceptualization, laws and norms can establish the need for accountability, which motivates a sense of responsibility within a development organization. The organization's ethical values can also motivate this sense of responsibility, which motivates establishing explainability as a quality goal. Explainability supports achieving quality goals related both to practical acceptability and to social acceptability. These quality goals either have a direct impact on system acceptability or an indirect impact on system trustworthiness, which ultimately has an impact on user trust and results in system acceptability.

Through these two perspectives, it is possible to see how explainability plays an important role in a system's practical and social acceptability, and how explainability can be a positive catalyst to support achieving various quality aspects. However, because of the double-edged sword effect of explainability (cf. section 6.4), explanations may have the exact opposite effect,

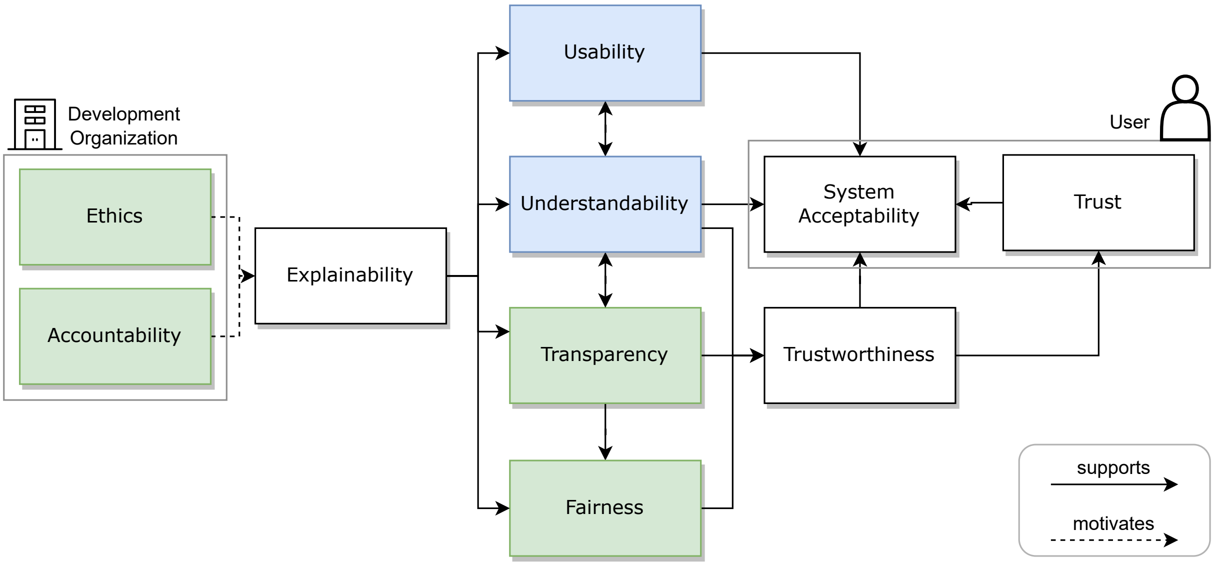


Figure 10.2: The ethical role of explainability, based on Vakkuri et al.[6]

and end up negatively impacting system acceptability. Software engineers should therefore focus on avoiding the negative effects of explanations. Poor explainability design choices can have a negative impact on the user's relationship (e.g., user experience issues), interfere with important quality aspects for a corporation (e.g., damaging brand image and customer loyalty), and bring disadvantages to the project or system (e.g., increasing development costs or hindering system performance). Explainability's role in human-system communication can be used to explain this sort of impact. Communication, depending on how it occurs in practice, can either strengthen or harm relationships.

Explainability as a Pillar of Human-Machine Communication

The importance of communication between humans and machines has been seen as so important that a new field of research has emerged in recent years: human-machine communication (HCM) [283]. NASA's [284] definition of "human-machine communication" also is focused on the point of contact between people and technology: "the characteristics of the interface through which a human user instructs or programs a machine, interacts with it during execution, and accepts information from it". This perspective on the purpose of people's interactions with machines is also essential in HCI research. The creation of the human-computer interface – the point at which people and computers exchange information – and the procedures that support it has been and remains one of HCI's main objectives. Jacob [285] states: "We can view the fundamental task of HCI as moving information between the brain of the user and the computer. Progress in this area attempts to increase the useful bandwidth across that interface by seeking faster, more natural, and more convenient communication means".

In 1985, Cathcart et al. [286] differentiate between three possible roles that can be fulfilled by a computer during human-computer interaction: 1) an unobtrusive role, where the user does not necessarily perceives the involvement of a computer, 2) a communication medium role, when communication happens *through* a computer rather than *with* a computer; 3) an interpersonal role, where the human interacts with the system, which in turn "responds appropriately in graphic, alphanumeric, or vocal modes establishing an ongoing sender/receiver relationship".

As previously stated in this thesis, we are living in the dawn of the algorithmic age. Advances in robotics are happening very fast and communication with computer systems will be more important than ever. In this way, systems should have the ability to be interlocutors, to interact with humans and to explain their decisions and rationale in an appropriate way. So, in my opinion, this emphasizes once more how important explainability is – not just as "one more" aspect of software quality or as a "nice to have", but as a fundamental pillar in the communication and interaction between people and information systems.

10.2.2 ② User-Centered Explainability

This thesis also lays the groundwork for the practices and techniques that should be incorporated into the software lifecycle to help practitioners navigate the steps required for the development of meaningful, explainable systems. The proposed approaches and techniques are user-centered, implying that user-centered processes are better suited to understanding explainability needs and developing explainable systems. They can be integrated into various development processes (e.g., traditional and agile), providing greater flexibility to the recommendations. Even though incorporating them may still represent a significant overhead for the industry, they can still be tailored to the needs of each individual project.

According to Cooper et al. [287], user-centered design roughly consist of: 1) understanding users' desires, needs, motivations, and contexts; 2) understanding business, technical, and domain opportunities, requirements, and constraints; and 3) using this knowledge as a foundation for plans to create products whose form, content, and behavior are useful, usable, and desirable, as well as economically viable and technically feasible. Table 10.1 summarizes the main differences between the traditional and user-centered practices. Traditional practices focus on the system itself and its features, while user-centered practices focus on understanding how the humans using the system perform their tasks and how the system can support them.

In section 6.5, I also discussed how the relationship between usability and explainability is important. Usability has a significant influence on software quality and should always be considered [5]. Although one should support the other, poor design and implementation of

Traditional Practices	User-Centered Practices
Technology/Developer-driven	User-driven
System Component Focus	User Solution Focus
Individual Contribution	Multidisciplinary Teamwork
Focus on Internal Architecture	Focus on External Attributes
Product Quality	Quality in Use
Implementation prior to Human Validation	Implementation based on User-validated Feedback
Establishing the Functional Requirements	Understanding the Context of Use

Table 10.1: Traditional practices in comparison with human-centered practices, adapted and extracted from Seffah and Metzker [8]

explanations could do more harm than good. Furthermore, the strong interaction between usability and explainability suggests that usability and explainability are so interrelated that it is difficult to separate one from the other. HCI principles are also the greatest option for achieving usability.

In fact, HCI also appears to be the best method for developing explainable systems because meeting stakeholders' needs and advancing addressee understanding are essential to explainability. Moreover, putting the human at the center of design decisions helps to develop ways of communicating and providing information that best suit their needs. Weigand et al. [288] specifically propose the user-centered development process defined in the ISO 9241-210 [289] to develop explainable systems, which takes into account user-centered principles and an iterative approach that incorporates feedback loops. The techniques recommended in chapter 7 help to gather as much information as possible about stakeholders, users, their environment, their expectations, the domain, and the project itself.

10.2.3 ③ Usefulness of the Proposed Theories and Artifacts

To evaluate the usefulness of the proposed theories, I use the list of criteria to evaluate theories proposed in the framework for SE theories [3], shown in Table 10.2.

First, I consider both theories testable since they can be applied in other contexts, domains and projects for empirical analyses that can refute and modify them. Empirical support is also present in the theories since they are all built on empirical studies. The theories have explanatory power since the concepts in the theories are explained based on the results of the studies and on the existing knowledge in the literature. The theories are also built with parsimony and their concepts are not so complex to make their application difficult. The theories can also be generalized to various contexts, domains, and projects and do not depend on specific settings. Finally, the utility of the theories was demonstrated through feedback from software engineers, through empirical studies, and through the case study.

Table 10.2: Criteria for evaluating theories, extracted from [3]

Testability	The degree to which a theory is constructed such that empirical refutation is possible
Empirical support	The degree to which a theory is supported by empirical studies that confirm its validity
Explanatory power	The degree to which a theory accounts for and predicts all known observations within its scope, is simple in that it has few ad-hoc assumption, and relates to that which is already well understood
Parsimony	The degree to which a theory is economically constructed with a minimum of concepts and propositions
Generality	The breadth of the scope of a theory and the degree to which the theory is independent of specific settings
Utility	The degree to which a theory supports the relevant areas of the software industry

Although the case study is only directly related to *T*2, it verifies at least partially the ideas associated with *T*1 since it employs the notion of explainable systems again and takes into account the concepts addressed in the conceptual model. However, a more extensive validation of *T*1 and the associated artifacts in practice is missing and should be pursued.

The artifacts that compose the theories also are useful for RE and can help to achieve shared understanding among stakeholders. First, the definition sets a common terminology and helps to delineate the elements and actors of explainable systems (e.g., aspects to be explained, addressee, context). The conceptual model and knowledge catalogue help both practitioners and researchers. For practitioners, they help to visualize the role and impact of explainability on the quality landscape, helping to discuss possible challenges and strategies, facilitating trade-off analysis and the negotiation with stakeholders. For researchers, they provide an overview of the interrelationships and interactions that can and should be explored.

The process reference model can guide practitioners in selecting the appropriate practices and techniques to develop explainable systems, as discussed in subsection 10.2.2. Software practitioners can use well-known user-centered practices to elicit, implement, and test requirements that are aligned with users' needs, demands, and context. Despite efforts to analyze software processes, there have been no proposals that explicitly focus on the development of explainable systems so far. As a result, I believe that the reference model provides both a starting point and a foundation that can be updated and improved in the future.

The quality framework links the dependencies, characteristics, and evaluation methods for explainability requirements. The case study's findings demonstrate three key points: 1) the quality framework was a useful tool for eliciting explainability requirements, selecting explanation designs, and assessing the success of the explanations; 2) the necessity of a thorough explanation design process to prevent negative effects brought on by trade-offs or design flaws;

and 3) the possibility that even straightforward and easily integrated explanations can have a positive effect on quality goals.

To additionally check whether the proposed artifacts can be useful for dealing with explainability as an NFR, I use the framework proposed by Paech et al. [16]. They proposed a list with 20 features that should be covered by NFR methods (see [16], Tab. 1). The artifacts can support all the features in this list. Some examples are: the identification of explainability goals (quality aspects that should be achieved through explainability), the visualization of dependencies between quality aspects and functional aspects, the identification of means of implementation, the comparison of different design strategies, as well as the identification of suitable metrics for evaluation.

10.3 Next Steps in Explainability Research

This thesis represents a step forward in the development of explainable systems. However, I recognize that more research is required. More specifically, the proposed artifacts in this thesis require more extensive validation. I propose expanding the validation of the artifacts through more extensive reviews and discussions involving more explainability experts as well as experts from other disciplines. I also encourage additional research that leads to any necessary updates and improvements.

When considering the next steps in explainability research, it is important to emphasize the importance of interdisciplinary collaboration when developing explainable systems. The fact that the RE community is so rich in methods and techniques that support software development is exactly why RE is so important for the success of a software project. Explainability research benefits greatly from taking requirements into account when studying explainable systems, but we also need to make sure that the techniques we develop address the needs of other communities.

The process of building *T*1 and the three related artifacts shows that research in RE can profit from insights of other disciplines when it comes to explainability. The fields of philosophy, psychology, and HCI, for example, have long researched aspects such as explanations or human interaction with systems (see [160] for research concerning explanations in several disciplines). At the same time, software and requirements engineers can contribute to the field of explainability by studying how to include such aspects in systems and adapt development processes. This knowledge, scattered among different areas of knowledge, must be made available and integrated into the development of systems.

Although the artifacts that compose *T*2 are a good start to guide the RE for explainable systems, more research is needed. Future research, for example, could look into specific aspects of XAI to see if the specific needs associated with these types of systems are not covered by the artifacts and should be included. Furthermore, more research on the interrelationships between explanation aspects is required to understand how the quality framework's levels and individual aspects interact with one another. Hands-on heuristics that aid in the design of explanations must also be investigated and developed. Although I believe that existing knowledge regarding how to organize the remaining stages of the development process is equally applicable to the development of explainable systems, the fact that the practices can be incorporated into an existing process does not eliminate the need for further study that addresses the entire lifecycle. Finally, the study in this thesis demonstrated that there is still much to learn about NFRs and that it is critical to investigate other new NFRs, such as ethics and fairness, as they become increasingly relevant. As software and requirements engineers, we must be aware of new quality aspects associated with modern systems and devise methods to translate these abstract aspects into requirements and functionality that meet stakeholder needs and the specified quality goals, thereby contributing to the development of high-quality systems for our society.

A

Survey Supplementary Material

A.1 Navigation Scenario

The navigation scenario was presented to the participants in two variations, depending on whether they use a built-in navigation system in their vehicle or just a navigation app on a smart device.

A.1.1 Built-In Navigation System

Hypothetical Situation: You are using the Navigation System in your car to drive to a certain destination. You are somewhat familiar with the way because you traveled before to this destination, but you notice that today the system is showing another route. This can be due to different reasons: maybe there is an accident on the road ahead or high congestion. You are not sure about the reason, as it does not give you any information about it. How interested are you in receiving an explanation about it?

A.1.2 Navigation App on a Smart Device

Hypothetical Situation: You are using the Navigation App in your smartphone to go from A to B, either by walking or by using public transportation. You are somewhat familiar with the way, but you notice that today the App is showing another route, lines or connections. You are not sure about the reason, as it does not give you any information about it. How interested would you be in receiving an explanation about it?

A.2 Survey Instrument

Assessing the Importance of Explanations in Software for End-Users

Welcome to our survey!

Thanks for taking the time to participate in our survey. This research seeks to understand the current perception of end-users regarding the impact of explanations on the transparency of software applications. The goal is to understand what is your perception about the current transparency of the software applications that you use in your daily life, as well as your interest in receiving explanations while using a software system. Please answer all the questions honestly, because authentic answers are essential for the success of this research. There is no right or wrong answer, so you may feel free to express your true opinion. By starting the survey, you confirm that you are of legal age (at least 18 years old). There are 18 questions in this survey

Demographics

[A1] What is your country of residence?
*Choose **one** of the following answers*
<List of countries as answer option...>

[A2] What year were you born?
Please enter a date: __________

[A3] What is your occupation? (In what area do you work?)
*Choose **one** of the following answers*

- Sales and Related
- Student
- Arts, Design, Entertainment, Sports, and Media
- Education, Training, and Library

- Research
- Protective Service
- Community and Social Service
- Legal
- Management
- Installation, Maintenance, and Repair
- Food Preparation and Serving Related
- Architecture and Engineering
- Computer and Mathematical
- Healthcare
- Office and Administrative Support
- Construction and Extraction
- Life, Physical and Social Science
- Farming, Fishing, Forestry
- Business and Financial
- Cleaning and Maintenance
- Transportation
- Other

Software Use

[B1] Please mark all items that apply to you:

Check all that apply

- I'm proficient with Word Processor Software (Microsoft Office suite including Word, Excel, and Power Point or similar). I'm very comfortable using these programs and have a lot of experience doing so.
- I know how to distinguish terms like hardware and software.
- I'm very comfortable using computers and I am confident in my ability to learn how to use new programs quickly.
- I can create fully functional spreadsheets and I am familiar with organizing and analyzing large sets of data (Eg. Macros and Functions)
- I understand basic conventions of a web page: URL, header, search box, links, footer.
- I understand networks concepts like IP address, VPN, routers, LANs.
- I understand Operating Systems indicators like use of CPU, memory, disk space.
- I've already have contact with programming and I consider I have basic programming skills.
- I'm a software/hardware developer.

[B2] Which of the following devices do you use in your everyday life?

Check all that apply

- Laptop

- Smartphone
- Tablet
- Desktop Computer
- On-Board Navigation System (integrated or integratable on vehicle) (e.g.,: Integrated on the vehicle / GPS device.
- Other:

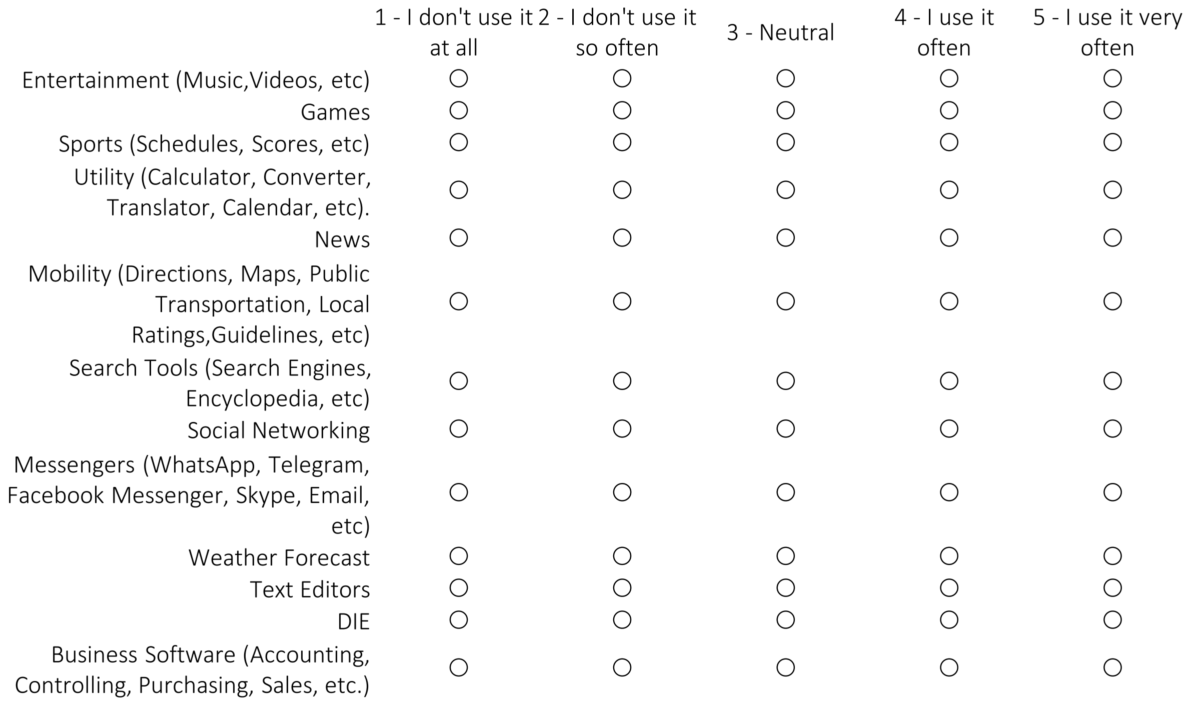
[B3] In a typical day, do you use software applications more for work or personal reasons?

*Choose **one** of the following answers*

- More often for work
- Quite a bit more often for work
- About an equal amount for work and personal reasons
- Quite a bit more often for personal reasons
- More often for personal reasons

[B4] In a typical day, which category of software/apps do you use on your digital devices most often? (More than one allowed)

Please choose the appropriate response for each item

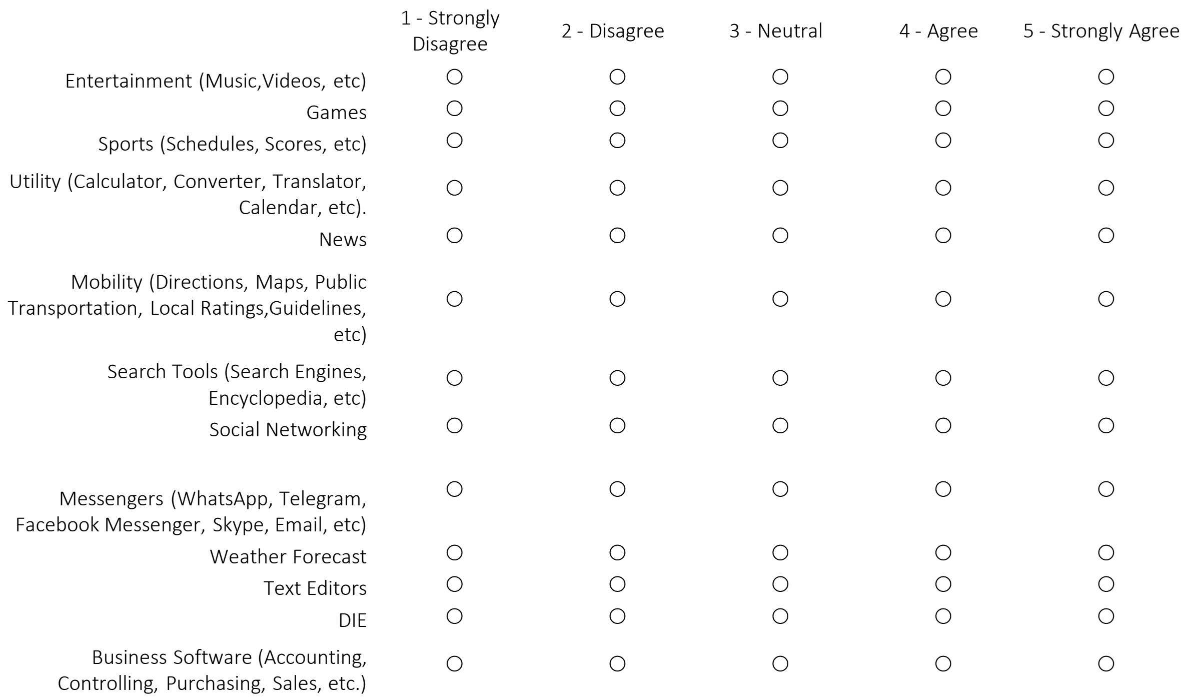


	1 - I don't use it at all	2 - I don't use it so often	3 - Neutral	4 - I use it often	5 - I use it very often
Entertainment (Music,Videos, etc)	○	○	○	○	○
Games	○	○	○	○	○
Sports (Schedules, Scores, etc)	○	○	○	○	○
Utility (Calculator, Converter, Translator, Calendar, etc).	○	○	○	○	○
News	○	○	○	○	○
Mobility (Directions, Maps, Public Transportation, Local Ratings,Guidelines, etc)	○	○	○	○	○
Search Tools (Search Engines, Encyclopedia, etc)	○	○	○	○	○
Social Networking	○	○	○	○	○
Messengers (WhatsApp, Telegram, Facebook Messenger, Skype, Email, etc)	○	○	○	○	○
Weather Forecast	○	○	○	○	○
Text Editors	○	○	○	○	○
DIE	○	○	○	○	○
Business Software (Accounting, Controlling, Purchasing, Sales, etc.)	○	○	○	○	○

[B5] I consider these categories of apps an essential part of my everyday life:

(The options below would appear depending on the chosen options in question B4)

Please choose the appropriate response for each item

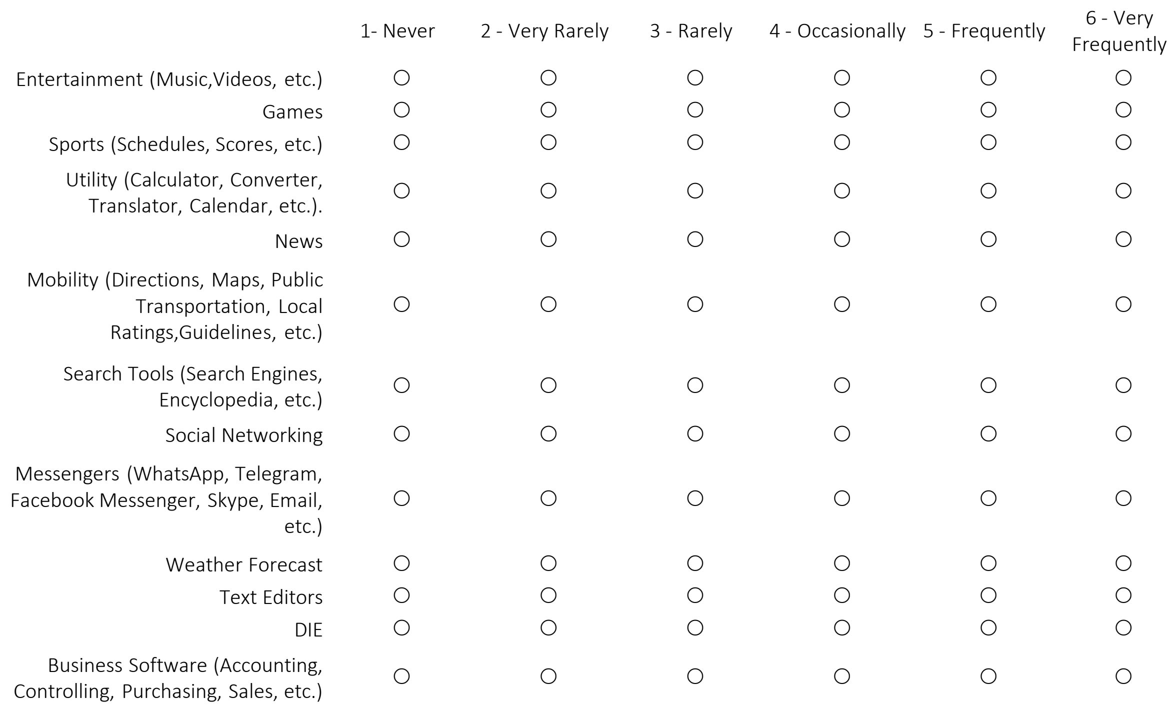


	1 - Strongly Disagree	2 - Disagree	3 - Neutral	4 - Agree	5 - Strongly Agree
Entertainment (Music,Videos, etc)	○	○	○	○	○
Games	○	○	○	○	○
Sports (Schedules, Scores, etc)	○	○	○	○	○
Utility (Calculator, Converter, Translator, Calendar, etc).	○	○	○	○	○
News	○	○	○	○	○
Mobility (Directions, Maps, Public Transportation, Local Ratings,Guidelines, etc)	○	○	○	○	○
Search Tools (Search Engines, Encyclopedia, etc)	○	○	○	○	○
Social Networking	○	○	○	○	○
Messengers (WhatsApp, Telegram, Facebook Messenger, Skype, Email, etc)	○	○	○	○	○
Weather Forecast	○	○	○	○	○
Text Editors	○	○	○	○	○
DIE	○	○	○	○	○
Business Software (Accounting, Controlling, Purchasing, Sales, etc.)	○	○	○	○	○

Problems with Software Use

[C1] How often do you have problems understanding the information presented to you in software belonging to these categories you regularly use?

(The options below would appear depending on the chosen options in question B4)



	1- Never	2 - Very Rarely	3 - Rarely	4 - Occasionally	5 - Frequently	6 - Very Frequently
Entertainment (Music,Videos, etc.)	○	○	○	○	○	○
Games	○	○	○	○	○	○
Sports (Schedules, Scores, etc.)	○	○	○	○	○	○
Utility (Calculator, Converter, Translator, Calendar, etc.).	○	○	○	○	○	○
News	○	○	○	○	○	○
Mobility (Directions, Maps, Public Transportation, Local Ratings,Guidelines, etc.)	○	○	○	○	○	○
Search Tools (Search Engines, Encyclopedia, etc.)	○	○	○	○	○	○
Social Networking	○	○	○	○	○	○
Messengers (WhatsApp, Telegram, Facebook Messenger, Skype, Email, etc.)	○	○	○	○	○	○
Weather Forecast	○	○	○	○	○	○
Text Editors	○	○	○	○	○	○
DIE	○	○	○	○	○	○
Business Software (Accounting, Controlling, Purchasing, Sales, etc.)	○	○	○	○	○	○

[C2] Can you share with us which was (were) the software(s)?
(The options below would appear depending on the chosen options in question C1)

- Entertainment (Music, Videos, etc)
- Games
- Sports (Schedules, Scores, etc)
- Utility (Calculator, Converter, Translator, Calendar, etc).(5)News
- Mobility (Directions, Maps, Public Transportation, Local Ratings, Guidelines, etc)
- Search Tools (Search Engines, Encyclopedia, etc)
- Social Networking
- Messengers (WhatsApp, Telegram, Facebook Messenger, Skype, etc)
- Weather Forecast
- Text Editors
- IDEs
- Business Software (Accounting, Controlling, Purchasing, Sales, etc.)

[C3] Can you share with us ONE situation where you could not understand the software behavior? Please, start by saying which of the software in the list below was the software in question.
(The options below would appear depending on the chosen options in question C2)
Please write your answer here: ____

Explanation Needs

[D1] Hypothetical Situation: You are using the Navigation System in your car to drive to a certain destination. You are somewhat familiar with the way because you traveled before to this destination, but you notice that today the system is showing another route. This can be due to different reasons: maybe there is an accident on the road ahead or high congestion. You are not sure about the reason, as it does not give you any information about it. How interested are you in receiving an explanation about it?
(This question would only appear if if the following conditions were met: Participant chose B2(5))
*Choose **one** of the following answers*

- 1 - Not at all interested
- 2 - Slightly Interested
- 3 - Indifferent
- 4 - Interested
- 5 - Extremely Interested

[D2] Hypothetical Situation: You are using the Navigation App in your smartphone to go from A to B, either by walking or by using public transportation. You are somewhat familiar with the way, but you notice that today the App is showing another route, lines or connections. You are not sure about the reason, as it does not give you any information about it. How interested would you be in receiving an explanation about it?

(This question would only appear if if the following conditions were met: Participant chose B2(2) AND B4(6) - from 3 to 5)

*Choose **one** of the following answers*

- 1 - Not at all interested
- 2 - Slightly Interested
- 3 - Indifferent
- 4 - Interested
- 5 - Extremely Interested

[D3] Consider the previously shown hypothetical situation(s). What kind of explanation would be helpful for you?

(This question would only appear if D1 or D2 would have been answered)

*Choose **one** of the following answers*

Please write your answer here: ______________________________

Opinion about Explanations in Software

[E1] What do you think are the 3 most important *advantages* of receiving explanations while using software?

Please write your answer here:

1. ______________________________
2. ______________________________
3. ______________________________

[E2] What do you think are the 3 most important *disadvantages* of receiving explanations while using software?

Please write your answer here:

1. ______________________________
2. ______________________________
3. ______________________________

Presentation of the Explanations

[F1] When should an explanation be presented?

Please choose only ***one*** *of the following:*

- Everytime I request it
- Automatically, just when something exceptional or unexpected happens
- Both: everytime I request an explanation and automatically when something exceptional or unexpected happens
- Never
- Other

[F2] First, let's present some definitions:

Explanations about:

Algorithms	Convey information and knowledge about the logic adopted in the software to generate the output and how is its reasoning process.
Dataset	Specify what kind of data the software is using to support its reasoning process and their sources (GPS, public databases, etc.)
History	Determine why the output/behavior is different from previous times (from what you are used to).
User Profile	Disclose the purposes and how your collected data is used and processed to make an internal profile about you and offer you tailored services/information.

Based on these definitions, about which aspects would you consider relevant to receive an explanation about

Check all that apply

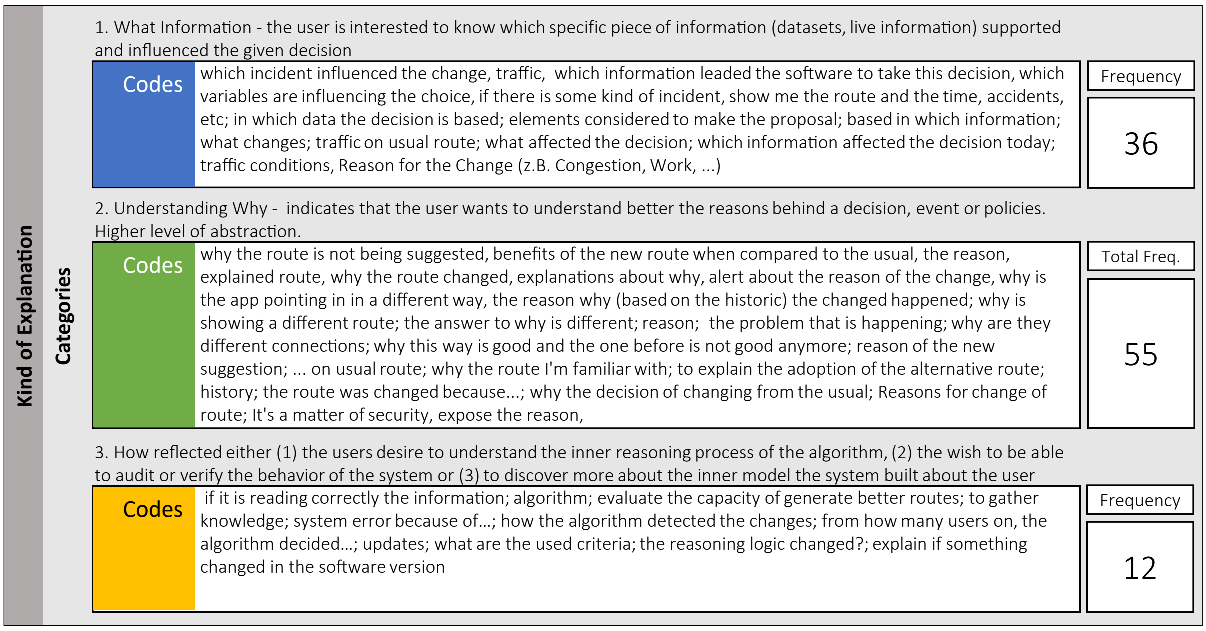
- Algorithms
- Dataset
- History
- User Profile
- I'm not interested in receiving any kind of explanation
- Other: ______________

We thank you for your efforts and time to support our research.

A.3 Codes

For the sake of clarity, some of the codes (first category) are shown below as examples. Each resulting code is assigned a color. An answer can be assigned to several codes. A detailed description of the coding process as well as the complete list of all codes can be found at [290].

A.3.1 Kinds of Explanations



Kind of Explanation

Categories

1. What Information - the user is interested to know which specific piece of information (datasets, live information) supported and influenced the given decision

Codes	which incident influenced the change, traffic, which information leaded the software to take this decision, which variables are influencing the choice, if there is some kind of incident, show me the route and the time, accidents, etc; in which data the decision is based; elements considered to make the proposal; based in which information; what changes; traffic on usual route; what affected the decision; which information affected the decision today; traffic conditions, Reason for the Change (z.B. Congestion, Work, ...)	Frequency 36

2. Understanding Why - indicates that the user wants to understand better the reasons behind a decision, event or policies. Higher level of abstraction.

Codes	why the route is not being suggested, benefits of the new route when compared to the usual, the reason, explained route, why the route changed, explanations about why, alert about the reason of the change, why is the app pointing in in a different way, the reason why (based on the historic) the changed happened; why is showing a different route; the answer to why is different; reason; the problem that is happening; why are they different connections; why this way is good and the one before is not good anymore; reason of the new suggestion; ... on usual route; why the route I'm familiar with; to explain the adoption of the alternative route; history; the route was changed because...; why the decision of changing from the usual; Reasons for change of route; It's a matter of security, expose the reason,	Total Freq. 55

3. How reflected either (1) the users desire to understand the inner reasoning process of the algorithm, (2) the wish to be able to audit or verify the behavior of the system or (3) to discover more about the inner model the system built about the user

Codes	if it is reading correctly the information; algorithm; evaluate the capacity of generate better routes; to gather knowledge; system error because of...; how the algorithm detected the changes; from how many users on, the algorithm decided...; updates; what are the used criteria; the reasoning logic changed?; explain if something changed in the software version	Frequency 12

Figure A.1: Kinds of explanations

A.3.2 Codes Excerpt: Advantages of Explanations

An excerpt of the codes is shown on the next page.

[QOE1] [1] What do you think are the 3 most important advantages of receiving explanations while using software?	Category 1	
Avoid losing time while trying to figure out the functions alone	Usability	improves use of time
Knowledge	Informativeness	understanding
Greater ease of use	Usability	user-friendliness
Understand the reason for an error that occurred	Informativeness	understanding
Understand the situation	Informativeness	understanding
Understand the software patterns while making decisions	Auditability / Verifiability	of technical aspects
Tutorial	Usability	guides the use
Certainty	Relationship	positive impact
Comprehensibility	Informativeness	understanding
Help users to use the software	Usability	guides the use
You better understand how the software works	Informativeness	understanding
Information	Informativeness	understanding
Inform yourself	Informativeness	understanding
Sich sicher fühlen	Relationship	builds trust
no loss of control: information helps to decide if the decision of the computer can be accepted	Informativeness	supports decision
Understand the purpose of an atypical operation	Informativeness	understanding
Information	Informativeness	understanding
If I'm using the software for the first time, information of how to use are extremly useful	Usability	guides the use
Understand the purpose	Informativeness	understanding
Understanding	Informativeness	understanding
Better understand the software	Informativeness	understanding
Ensure understanding of the benefits of using the software	Informativeness	understanding
Understanding	Informativeness	understanding
Adds value to software	Relationship	positive impact
Understand the context of the situation	Informativeness	understanding
feedback loop	?	#
Better understanding	Informativeness	understanding
Understand how the software works	Informativeness	understanding
Transparency	**Transparency**	
Eigene Entscheidung	#	#
Feeling secure while using it	Relationship	builds trust
Eigene Entscheidung	#	#
not to be confused by decision made by software	Informativeness	clears obscurity
Transparency	**Transparency**	
clarity in what is happening to my data or information.	Auditability / Verifiability	data transparency
Reliability	Relationship	builds trust

A.3.3 Codes Excerpt: Disadvantages of Explanations

An excerpt of the codes is shown on the next page.

[QOE2] [1] What do you think are the 3 most important disadvantages of receiving explanations while using software?	Category 1	
Upon receiving the explanations before using it, it becomes more difficult to verify that the software is intuitive.	Usability	impairs the use
Receive explanations about somethig you already know	Informativeness	unnecessary information
Time to understand the explanations	Usability	time consuming
Repetitive explanation everytime I use the software	Informativeness	unnecessary information
If too much, it can harm the experience of the user	Usability	impairs the use
Loss of time when not requested	Usability	time consuming
Time	Usability	time consuming
Confusion	Informativeness	hinders understanding
None, without explanations there are more disadvantages	**Other**	**none**
There are no disadvantages	**Other**	**none**
Can be inconvenient	Relationship	negative impact
Can use more resources like CPU, memory, etc	Usability	use of computational resources
Time consuming	Usability	time consuming
can be annoying	Relationship	negative impact
Insecurity when knowing the accumulation of personal data that a software can generate	Relationship	negative impact
None	**Other**	**none**
If the explanations are repetitive (it's bad)	Informativeness	unnecessary information
If it's long, it gets tiresome and not didactic	Relationship	negative impact
Not understand the supposed improvement	Informativeness	adds obscurity
When the explanation was not requested, makes it inconvenient	Relationship	negative impact
If little objective, it can become tiring	Relationship	negative impact
delay in receiving the main information	Usability	time consuming
Confusion	Informativeness	hinders understanding
Less practicality	Relationship	negative impact
The explanation is not always clear enough	Informativeness	adds obscurity
too much information	Informativeness	unnecessary information
For some, can be waste of time	Usability	time consuming
cause doubts	Informativeness	hinders understanding
Sometimes it can be a lot but really necessary information.	Informativeness	unnecessary information
More text	Informativeness	unnecessary information
Distractive	Usability	impairs the use
having more informations	Informativeness	unnecessary information
excess of information, which can cause confusion.	Informativeness	unnecessary information
you may be distracted	Usability	impairs the use

B

SLR and Coding Process Supplementary Material

B.1 Manual Search Sources and Paper Selection

Table B.1: Manual search sources and paper selection

Source	**Total Retrieved**	**Initial Selection**	**Final Selection**
International Requirements Engineering Conference (RE)	1312	15	4
Symposium on the Foundations of Software Engineering (FSE)	667	8	2
Information and Software Technology (IST)*	2668	23	0
Intelligent User Interfaces (IUI)	1158	52	18
Journal of Systems and Software (JSS)†	4121	8	0
Transaction on Software Engineering (TSE)	2910	23	0

*20 publications out of 2668 publications were not accessible.

†Seven publications out of 4121 publications were not accessible.

Conference on Information and Knowledge Management (CIKM)	2789	8	2
International Working Conference on Requirement Engineering: Foundation for Software Quality (REFSQ)	328	4	0
Transactions on Software Engineering and Methodology (TOSEM)	615	4	1
Conference on Recommender Systems (RecSys)	521	15	6
Requirements Engineering Journal (REJ)	455	9	2
RE Workshops	21	4	1
REFSQ Workshops[‡]	162	5	1
Minds and Machines	724	30	17
Big Data & Society	284	16	6
International Joint Conference on Artificial Intelligence - Workshop on eXplainable Artificial Intelligence[§]	63	41	34
Philosophy and Technology[¶]	259	10	9
Ethics and Information Technology	538	1	1

B.2 SLR References

B.2.1 Manual Search

The following 104 papers are the result of our manual search: [291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 182, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 116, 314, 315, 62, 316, 177, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 154, 327, 328, 329, 330, 331, 332, 333, 334, 185, 335, 336, 337, 49, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 278, 51, 358, 359, 360, 361, 362, 168, 363, 169, 364, 365, 366, 367, 157, 161, 368, 369, 370, 186, 15, 371, 372, 373, 374, 375, 376, 377, 378, 379]

B.2.2 Snowballing

The following 125 papers are the result of the forward and backward snowballing procedure: [380, 381, 382, 383, 384, 385, 386, 187, 69, 387, 388, 389, 390, 391, 176, 392, 393, 394, 395, 155, 396, 397, 398, 399, 400, 401, 402, 403, 404, 239, 405, 225, 406, 407, 408, 409, 410, 175, 411, 61, 412, 413, 414, 189, 70, 415, 416, 46, 196, 417, 418, 419, 181, 266, 420, 421, 422, 423, 160, 424, 425, 426, 427,

[‡] Only the proceedings of the years 2000, 2001, 2006-2008, 2010, 2011 and 2015-2019 were accessible.

[§] Proceedings from 2017-2019 were considered

[¶] Only the issues beginning from 2011 were accessible.

428, 429, 430, 431, 188, 432, 433, 434, 171, 435, 156, 166, 436, 437, 183, 190, 153, 438, 80, 50, 439, 440, 441, 442, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 165, 454, 172, 455, 456, 457, 458, 459, 13, 184, 460, 119, 461, 462, 463, 464, 465, 466, 467, 162, 255, 468, 469, 470, 471, 472, 35]

B.2.3 Aspects to Explain

Table B.2: Aspects to explain

Aspect to explain	References
System in General	[175, 153, 239, 116, 50, 366, 157, 464, 413, 70, 458, 420, 379, 429, 459]
System's Reasoning Processes	[353, 375, 465, 416, 424, 428, 171, 464, 61, 461, 154, 429]
System's Inner Logic	[116, 62, 324, 338, 346, 367, 371, 376, 405, 162, 472, 61, 428]
System's Model's Internals	[298, 116, 371, 372, 155, 419, 181, 428, 188, 351, 363]
System's Intention	[464, 156, 383]
System's Behavior	[157, 455, 255, 366, 461]
System's Decision	[116, 358, 388, 360, 419, 426, 50, 441, 418]
System's Performance	[375, 445, 464]
Knowledge about User or World	[157, 465, 360, 441, 366]

B.2.4 Quality Aspects in Literature

Table B.3: Quality aspects impacted by explainability

Quality Aspect	References	
Accountability	Pos:	[335, 340, 358, 225, 407, 419, 425, 428, 471, 181, 411, 80, 187, 49, 183, 388, 182, 294, 457, 400, 442, 461, 160]
Accuracy	Pos:	[320, 162, 176, 377, 436, 432, 348]
	Neg:	[295, 416, 344, 50, 302, 49, 62, 403, 407, 368, 61, 189, 155, 411, 465]
Adaptability	Workshop Exclusive	
Auditability	Pos:	[424, 308, 388, 153, 176, 168, 13, 50, 436, 323, 309, 456, 332, 471, 361, 156, 116, 358, 363, 165, 452, 319, 339, 392]
Complexity	Workshop Exclusive	
Compliance	Pos:	[181, 61, 182, 410, 370, 336, 457, 339]

Confidence in the System	Pos:	[154, 382, 446, 155, 70, 326, 15, 440, 171, 69, 183, 424, 456, 175, 447, 412, 304, 404, 425, 176, 46, 387, 428, 417, 398, 420, 433, 470]
	Neg:	[325, 183, 13]
Correctness	Pos:	[182, 388, 441]
Customer Loyalty	Pos:	[409, 166, 314, 239, 333, 69, 374, 357, 446, 304]
Debugging	Pos:	[182, 119, 80, 404, 176, 469, 440, 457, 471, 153, 392, 472, 62, 425, 412, 61, 69, 299, 430, 225, 423, 50, 407, 337, 332]
Decision Justification	Pos:	[382, 465, 182, 62, 183, 50, 184, 360]
Development Cost	Neg:	[379, 62, 403, 176, 50, 372, 156]
Effectiveness	Pos:	[336, 225, 370]
Efficiency	Neg:	[50, 403, 407]
Ethics	Pos:	[470, 294, 439, 155, 61, 181, 156]
Extensibility	Workshop Exclusive	
Fairness	Pos:	[346, 338, 411, 363, 410, 323, 388, 183, 153, 61, 300, 439, 155, 70, 309, 428, 418, 80, 445, 225, 349, 420, 50]
Guidance	Pos:	[162, 336, 116, 388, 297, 324]
Human-Machine Cooperation	Pos:	[380, 35, 417, 340, 153, 463, 323, 70]
Knowledge Discovery	Pos:	[153, 439, 70, 156, 445, 50, 297, 457, 61]
Learnability	Pos:	[429, 452, 412, 182, 458, 46, 160, 419, 428, 417, 356, 176, 459, 471]
Maintenability	Workshop Exclusive	
Mental Model Accuracy	Pos:	[344, 455, 445, 376, 168, 160, 13, 325, 356, 412, 428, 392, 166, 434, 424, 420]
Model Optimization	Pos:	[165, 343, 225, 184, 189, 418, 153, 70, 457]
Perceived Usefulness	Pos:	[384, 266, 362, 361, 162, 169, 69, 239, 428, 429, 399]
Perceived Value	Pos:	[443, 50, 409, 69, 455, 176, 172, 168, 428, 171, 460, 437, 442, 162, 359, 374, 420, 429, 370]
Performance	Pos:	[35, 445, 428, 172, 369]
	Neg:	[177, 402, 62, 298, 403, 70, 116]
Persuasiveness	Pos:	[409, 471, 415, 69, 266, 175, 176, 361, 360, 239, 428, 456, 443, 399, 172, 465, 162, 372, 324, 361, 357, 344, 333, 439, 160, 183, 387, 162]
Portability	Workshop Exclusive	
Predictability	Pos:	[344, 389, 326]

Privacy	Pos:	[51, 188, 153, 155, 181]
	Neg:	[339, 51, 449, 70, 425, 80, 411, 440, 439]
Privacy Awareness	Pos:	[460, 70, 153]
Real-Time Capability	Workshop Exclusive	
Reliability	Pos:	[466, 153, 181]
Robustness	Pos:	[70, 189, 186]
Safety	Pos:	[61, 187, 389, 153, 300, 155]
Scrutability	Pos:	[458, 373, 336, 437, 465, 183, 69, 239, 225]
Security	Pos:	[154, 189, 155]
	Neg:	[49, 339, 303, 154, 363, 411, 62, 80, 390]
Stakeholder Trust	Pos:	[69, 293, 344, 471, 15, 303, 35, 440, 460, 467, 61, 353, 157, 154, 371, 380, 435, 314, 301, 153, 463, 386, 186, 161, 390, 62, 185, 352, 365, 341, 51, 362, 438, 377, 310, 375, 400, 429, 339, 359, 360, 468, 382, 162, 420, 225, 323, 372, 357, 326, 404, 156, 304, 317, 340, 369, 370, 373, 187, 389, 392, 408, 415, 416, 183, 439, 466, 176, 300, 363, 318, 177, 321, 465, 387, 395, 155, 309, 239, 411, 413, 189, 46, 417, 419, 181, 160, 425, 430, 333, 171, 166, 50, 441, 443, 453, 361, 455, 470, 297]
	Neg:	[154, 372, 183, 225, 13, 440, 171, 162, 363, 344, 15, 362]
Support Decision Making	Pos:	[439, 415, 266, 399, 392, 428, 162, 116, 456, 420, 436, 384, 297, 360, 69, 324, 333, 225, 176, 458, 445, 379, 70]
System Acceptance	Pos:	[176, 297, 157, 361, 389, 445, 295, 352, 446, 433, 443, 326, 46, 162, 156, 324, 374, 392, 429, 441, 239, 420, 457, 51, 428, 398, 467]
Testability	Pos:	[153, 61, 176]
Trade Secrets	Neg:	[49, 116, 339, 70, 411]
Transferability	Pos:	[190, 156, 439, 70]
Transparency	Pos:	[69, 239, 154, 297, 370, 412, 183, 463, 413, 162, 437, 171, 376, 324, 361, 266, 424, 156, 46, 374, 225, 467, 317, 382, 365, 408, 176, 172, 465, 116]
Trustworthiness	Pos:	[470, 187, 295, 62, 412, 294, 70, 419, 35, 387, 334, 176]

Understandability	Pos:	[432, 51, 447, 161, 425, 440, 393, 188, 355, 320, 62, 172, 463, 15, 50, 385, 394, 386, 400, 160, 116, 35, 399, 304, 446, 334, 189, 403, 387, 184, 295, 278, 13, 375, 362, 225, 196, 417, 420, 329, 370, 428, 413, 336, 69, 384, 357, 392, 182, 381, 301, 342, 323, 466, 395, 171, 457, 358, 377, 305]
	Neg:	[372, 116, 401]
Usability	Pos:	[445, 320, 266, 239, 153, 116, 62, 428, 293, 403, 440, 314, 341, 333, 434, 374, 162, 176, 161, 460, 466, 69]
	Neg:	[116, 325, 413]
User Awareness	Pos:	[428, 171, 434, 440]
User Control	Pos:	[439, 460, 46, 294, 61, 69]
User Effectiveness	Pos:	[69, 176, 360, 162, 329, 175, 465, 387, 463]
	Neg:	[69, 13, 347]
User Efficiency	Pos:	[362, 162, 183, 69, 465, 239, 412, 176, 175, 344]
	Neg:	[177, 196, 362]
User Experience	Pos:	[359, 166, 333, 334, 172, 153, 373, 440]
	Neg:	[168, 116, 160]
User Performance	Pos:	[176, 429, 420, 426, 329, 171, 409]
User Satisfaction	Pos:	[69, 438, 176, 408, 415, 183, 166, 428, 362, 239, 429, 443, 372, 324, 162, 341, 392, 314, 437, 175, 333, 360, 426, 326, 446, 325, 433, 168, 403]
Validation	Pos:	[384, 365, 176, 69]
Verifiability	Pos:	[176, 424, 363, 420, 447, 326, 430, 184, 428, 418, 153, 15]

B.3 SLR Codes

In the material below, “P” indicates that explainability has a positive impact on the corresponding quality aspect, and “N” stands for a negative impact. During the coding procedure, a threshold was defined for producing a code. This means that only impacts were considered that were mentioned by at least *three* different sources. This approach was taken to ensure more reliable results. The material below is already cleaned up in this respect, such that codes that are not supported by at least three sources are not listed.

Accountability_P

- a critical piece in achieving accountability
- accountability
- allows to be held accountable
- assess accountability
- assignment of blame
- better manage liability
- enforce accountability under the law
- facilitate accountability
- fulfill the goal of accountability
- hold entities accountable
- important for responsibility of algorithms
- important for liability
- important to achieve responsible AI
- know where the blame lies when things go wrong
- legal accountability
- liability
- liability for machines
- means to promote accountability
- render decision-making more accountable
- render decisions more accountable
- shift responsibility from algorithm to participant
- used to determine legal culpability

Accuracy_N

- can negatively affect predictive accuracy
- cost of classification accuracy
- decrease in predictive performance
- inverse relationship between accuracy and explainability
- may conflict with precision
- requires a sacrifice for accuracy
- tension between accuracy and explainability
- the higher the accuracy, the lower the explainability
- the more explainable, the less accurate
- there is a well-known trade-off between accuracy and explainability
- trade off explainability against accuracy
- trade-off between accuracy and explanatory power
- yield lower accuracy

Accuracy_P

- improve classifiers
- improve prediction accuracy
- improve recommendation accuracy
- improved accuracy of decisions
- improving the accuracy
- improving the accuracy of VQA systems
- increase accuracy measures

Auditability_P

- allow the system to demonstrate partial capability
- allows for question-answering about the system's reasoning steps
- allows to look why the system has reached an anomalous result
- assess the reliability of systems
- auditability
- benefit the auditability
- describe the methods employed and the reasons for exploying them
- detect errors in input data that led to adverse decisions
- enables models to be audited
- estimate likelihood of errors in decision making
- examine the knowledge of the system
- help to identify errors and biases
- help users understand what contributed to the large error
- identify that the system has erred
- indicate prior reliability
- is an auditable and provable way to defend algorithmic decisions
- judge whether a prediction is correct
- satisfying the desire to know whethe an algorithm caused an error
- scrutinize individual cases
- support evaluation of system conclusions
- understand the scope of an error and solve discrepancies
- understand the underlying technicalities and models

Compliance_P

- ensure legal decisions are made
- implement a right to explanation
- make model performance and guideline compliance compatible
- necessary for compliance
- used to comply with regulatory and policy changes
- useful for increased compliance

Confidence in System_N

- decrease reliance if level of detail insufficient
- had a negative impact on user confidence
- question the system and lead to self-reliance

Confidence in System_P

- aim at acquiring confidence
- allow to develop appropriate reliance
- can lead to a higher level of confidence
- confidence
- confidence in decision making
- decides how much confidence to place in recommendation
- develop appropriate reliance
- higher confidence in system recommendations
- higher degrees of confidence
- improve confidence of the decision quality
- increase confidence in system's abilities
- increase reliance
- increase the end user’s confidence in the system’s conclusion
- increase user belief in system
- increase users' confidence in problem-solving competence
- increase user's confidence in the system
- increases their confidence
- instill confidence
- it can help build confidence
- might give us more confidence
- more user confidence
- significantly increase users’ confidence
- strenghten the confidence of users

Correctness_P

- help correct errors in input data
- necessary for correcting an error
- optimality and correctness

Customer Loyalty_P

- affect perceptions of the organization
- could help inspire loyalty
- decrease the likelihood of abandoning the system
- higher degrees of rapport
- higher intention to use the interface again
- increase the propability of using the system
- it can help build rapport
- make users more willing to continue the use of a system
- promote rapport
- rapport

Debugging_P

- better debug the system
- debug predictive models
- debug the procedure
- debugging
- debugging purposes
- determine and fix errors and faults
- diagnosis and refinement
- ease of debugging
- enable users to debug the intelligent agent
- enables models to be debugged
- end-user debugging
- for debugging
- for system debugging
- help in locating sources of error
- help to identify and correct errors
- likely to be used for debugging

- model debugging
- notifying users about the defects in the system
- play an important role in debugging systems
- potential for aiding program analysis
- support developers for system debugging
- support system debugging
- used in system diagnosis

Decision Justification_P

- justification (e.g., offering reasons for an action).
- justification (of individual recommendations)
- justification (why recommendation was made)
- justify a given decision
- justify actions and decisions
- justify the decision made
- justify the outcome of a particular classification
- provides the required information to justify results

Development Cost_N

- at the expense of development effort
- can be costly on multiple dimensions
- cost expensive
- could be costly
- development cost
- development time
- is expensive

Effectiveness_P

- increase overall effectiveness
- system effectiveness limited by inability to explain
- useful for increasing an intelligent system's overall effectiveness

Efficiency_N

- could result in less efficient systems
- trade off explainability against efficiency

Ethics_P

- achieve an evaluation of the ethical standards of a machine
- achieve ethical AI
- ensure ethical decisions are made
- ethical decision making
- ethics
- evaluate if a decision is not in conflict with ethical norms
- promote ethics
- support ethical reasons

Fairness_P

- a way to check fairness
- address concerns about lack of fairness
- assurance that fairness issues have been mitigated
- demonstrate a model's fairness
- detect discrimination
- develop solutions against algorithmic discrimination
- enable people to identify and address fairness
- enable the identification of discrimination
- enables to assess whether algorithm is just
- ensure algorithmic fairness
- ensure fair decisions are made
- ensure fairness
- ensure that algorithms do not encode discrimination
- fair decision making
- fairness
- fulfill the goal of fairness
- identify algorithmic biases
- important to ensure algorithmic fairness
- inspect fairness
- mitigate decision biases
- moderate assessments of fairness
- non-discrimination
- render decisions more fair
- support fairness in decision-making
- verify fairness
- way to defend algorithmic decisions as being fair

Guidance_P

- educate about product knowledge
- guide in solving problems
- guide users during the use of the system
- increase product knowledge
- provide guidance on how to reverse adverse decisions
- to better guide the user
- to educate users about product knowledge

Human-Machine Cooperation_P

- allow for a better human AI cooperation
- crucial for effective human machine interaction
- improve cooperation
- improve the human-machine interaction performance
- key to effective human-AI interaction
- necessary for human in the loop
- provide a more effective interface for the human in-the-loop
- the ability of a model to be interactive with the user

Knowledge Discovery_P

- causality
- gain further insights or evidence
- helpful for knowledge discovery
- helpful tool to gain knowledge
- knowledge generation
- knowledge/scientific discovery

Learnability_P

- aid learning
- allow developers to not make the same mistake again
- allow users to learn something from the system
- allows the humans to learn
- can facilitate learning
- educate the user about the process of generating a recommendation
- facilitate learning
- help developers and humans working with a system learn from it
- help users to learn about how the system works
- improve user learning
- learn how the system operates
- learning how the system works
- teach something and learn
- teach users how to achieve a task

Mental Model Accuracy_P

- a tool to build and refine inner knowledge models
- align and adapt the mental models of the participating parties
- assist participants in the conception of an accurate mental model
- be aware of the system's limitations
- can help users to develop better mental models of systems
- create a shared understanding of decisions
- discover when a system is pushed over the bound of its expertise
- form accurate picture about the system's reliability
- gradually improve the mental model
- help build useful mental models
- help the user to build a theory of mind
- help users to understand when they can rely on the system
- helps users determine whether their needs are known by the system
- helps users to revise their theory of mind
- keep agents on the same page with respect to their beliefs
- know what the limitations of the system are
- learn the methods and knowledge used in the problem solving process
- users can judge whether their goal match those of the system

Model Optimization_P

- detect bias in training data
- detecting model bias
- determine what should be learned by a model
- identify bias in training data
- identify problems in training data
- improve and develop ML models

- lead to the design of better models

Perceived Usefulness_P
- assess whether a recommendations is useful
- crucial aspects for system utility
- effect on perceived usefulness of recommendations
- enhance perceived usefulness of advice
- increase perceived information usefulness
- increase the perceived usefulness of a recommender system
- perceived as a useful system
- perceived system usefulness
- usefulness

Perceived Value_P
- a sense of control for the user
- adds to business value
- affect perceptions of fairness
- be perceived as easy to use
- being perceived as a trustworthy system
- benefit perceived reliability
- better user perception
- convince users of of a system's competence
- help improve user perceptions during system errors
- help increase perceived benevolence
- increase perceived control
- increase perceived efficiency
- increase perceived fairness
- increase perceived recommendation quality
- increase perceived safety
- increase users' perception of system competence
- increases users’ perception of the interface’s competence
- influence the extent to which the decision is viewed as fair
- more positive perceptions of a system
- perceive as more helpful
- perceive the interface as more competent
- perceived to be fair
- perception of system ease-of-use
- positively affect perceptions of system effectiveness
- promote more positive user perception of the system
- raise perception of integrity
- result in more positive user perceptions
- system perceived as more competent
- useful for increasing perceived value

Performance_N
- drawbacks in terms of performance
- loading memory issue
- loading time issue
- may conflict with performance
- require high computational costs
- take time to compute
- trade-off at the expense of performance
- trade-off with the performance of a model
- use of CPU, memory, storage, and battery

Performance_P
- help to decide between models
- improve AI performance
- improve overall system performance
- increase system performance
- learning time can be reduced

Persuasiveness_P
- affect task-taking motivation
- convince the user that the agent is correct
- convince the user to accept the result
- convince users that the suggested alternative is appropriate
- convince users to adopt recommentation
- convince users to try or buy
- critical to acceptance of systems' decisions
- greater adherence to system recommendation
- help users to understand and accept a recommendation
- improve persuasiveness
- improve user acceptance of recommendations
- improve users’ acceptance of recommendations
- increase acceptance of system conclusions
- increase persuasiveness of the recommended messages
- increase probability of accepting a suggestion
- increase the acceptance of recommendations
- increase the system's persuasiveness
- increased the acceptance of the recommendations
- increasing system persuasivness
- influence applicant reactions
- invoke user's interest in recommendation
- make recommendations more persuasive
- may accept the system’s choices
- more compliance with a request
- motivate users to consume items
- nudge the user in a certain direction
- persuade users to make a certain choice
- persuade users to try or purchase a recommended item
- persuasion
- persuasiveness
- positively affect users' purchasing beghavior
- suggestions being accepted by users

Predictability_P
- ability to predict the system’s performance correctly
- determine when the system will make a mistake
- increase the user’s ability to predict the classifier decision
- makes users more likely to correctly predict the model’s success

Privacy Awareness_P
- lower privacy concerns
- privacy awareness - ability to assess privacy
- provides a way to check privacy

Privacy_N
- can demolish stakeholder's anonymity
- can hurt privacy
- can threaten privacy
- could jeopardize privacy
- data privacy
- negative consequences regarding privacy
- pose a privacy risk
- privacy implications
- privacy issues regarding training data
- privacy might be violated

Privacy_P
- can help privacy
- ensure privacy is uphold
- ensure that sensitive information is protected
- privacy
- promote privacy

Reliability_P
- improve reliability
- reliability

Robustness_P
- facilitate robustness
- improve model robustness
- instrumental in developing more robust systems

Safety_P
- guarantee safety concerns
- guarantee safety concerns are met
- help create safer products
- help increase safety
- improve safety
- increase safety
- safety

Scrutability_P

- alter which features of an item are deemed most important
- customize recommendations to fit users' preferences
- explain corrections back to the system
- increase scrutability
- provide scrutability
- scrutability

Security_N

- allow for gaming
- can threaten security
- enable individuals to game the system
- enhance capabilities of attackers
- facilitate manipuations of the system
- introduce security vulnerabilities
- may conflict with security
- pose a security risk
- security implications
- trade off with security

Security_P

- bridge the gap between ‘actual security’ and ‘perceived security’
- identify illegitimate intrusion, malware, and other attacks
- protection mechanism can be improved by public scrutiny
- security

Stakeholder Trust_N

- bad explanation-for-trust may fail to create trust
- bad explanations may lead to distrust
- can undermine trust
- decrease trust
- effects on trust
- erode trust
- may lead to mistrust
- may lose trust
- minimum explanations can potentially harm user trust
- too much explanatio eroded user trust
- too much explanation can degrade trust

Stakeholder Trust_P

- achieve trust
- achieve user trust
- affect user trust
- agents tend to trust
- appropriately trust ML-based solutions
- assessing trust
- boost user trust
- breed trust
- build more trust in the interface
- build trust in the agent's choices
- build trust towards a machine's performance
- can address trust concerns
- can aim at acquiring trust
- can engender trust
- can promote trust in the system
- correctly calibrate trust in systems
- could help inspire user trust
- critical tool to build public trust in AI
- easier to trust
- enable domain experts and users to trust the model
- enable them to trust results
- enable to have appropriate level of trust
- engender trust in AI
- enhance trust in the classifier
- essential to assure trust
- foster trust
- foster user trust
- fundamental mechanism to increase user trust
- gain user trust
- generating trust in the model
- have a positive effect on user trust
- have the potential to build a trust relationship
- help gain user's trust
- help inspire trust in a recommender
- help promote trust
- help to establish a more appropriate level of trust
- impact perceptions of trust in the company
- important in assessing trust
- important in order to build human trust in systems
- improve user trust
- improving trust in decision by AI system
- improving trust in the advice
- increase trust
- increase trust in the recommender system
- increase trust in the system
- increases user trust
- increasing user trust
- instill trust
- means to establishing trust
- moderate trust to an appropriate level
- more likely to trust the underlying ML systems
- positive effects on organizational trust
- provide a way to check trust
- significant improvement in user trust
- significantly higher trust
- significantly increase users' trust
- students trust the system more
- support trust
- trust
- will enhance trust at the user side
- will trust the agent in the future

Support Decision Making_P

- aid the explainee in performing a task
- allow to make more accurate decsiosn
- allow to make more informed and accurate decisions
- allow users to make more informed decisions
- better decision whether to choose the recommended item
- employ for decision-making
- empower users to make informed choices
- enable users to make more accurate judgment on a decision
- enable users to make more informed and confident decisions
- enable users to make more informed decisions
- evaluate possible alternatives
- give enough information to decide
- help decide whether to engage with item further
- help facilitate improved user decisions
- help users make accurate decisions
- help users make to better decisions
- help users to evaluate items correctly
- help users to make more accurate decisions
- help users to resolve preference conflict
- helping users to make better decisions
- helping users to make good decisions
- improve the quality of user decisions
- improve users' decision making
- increased problem solving competence
- influence decision making
- make informed decisions
- might be critical to aid a person in making final decisions
- needed to make rational choices
- provide users with knowledge to make decisions
- support to make more accurate decisions
- support user decision-making

System Acceptance_P

- affect acceptance
- beneficial to the system's acceptance
- bring closer to acceptance by end users
- can enhance acceptability of systems
- can improve acceptance of recommender systems
- convince to use system
- determine the ovewrall adoption
- essential for gaining acceptance

- gain user acceptance
- greater user acceptance
- higher degree of willingness to use systems
- higher level of willingness to use autonomous systems
- important for acceptance of recommender systems
- important for user acceptance
- increase acceptance
- increase user acceptance
- increasing user acceptance
- influential for acceptance of intelligent systems
- key towards acceptance of AI systems
- may increase system acceptance
- means of influencing user acceptance
- open the chance of adoption
- pivotal to acceptance by society
- potential to increase acceptance
- promote acceptance of the system
- requirement for model acceptance
- user acceptance

Testability_P

- enables models to be tested
- help system designers to evaluate or test a system
- helpful for testing

Trade Secrets_N

- cannot protect proprietary information
- disclose proprietary knowledge
- model confidentiality
- trade secret

Transferability_P

- improve transfer learning
- support transferability
- transferability

Transparency_P

- be perceived as transparent
- can provide transparency
- contribute to transparency
- could provide transparency
- foster transparency
- fulfill the goal of transparency
- improve transparency
- improve transparency of the system
- increase the system's transparency
- increase transparency
- increased transparency
- increasing perceived transparency
- increasing system transparency
- increasing transparency of system reasoning
- make more transparent
- offer transparency
- perceived to be transparent
- provide transparency
- provide transparency of how system works
- providing system transparency
- render decisions more transparent
- support transparency
- system transparency
- transparency

Trustworthiness_P

- achieve complete trustworthiness
- create more trustable products
- essential for becoming trustworthy
- foster trustworthiness
- important to achieve trustworthy AI
- improve untrustworthy models
- increase trustworthiness
- make it more trustworthy
- trustworthiness
- trustworthy AI

Understandability_N

- create false binaries
- hinder understanding if not appropriate
- may cause information overload

Understandability_P

- ad in understanding network mistakes
- aid in understanding an automated vehicle’s function
- aid in understanding the decision's consequences
- allow its users to better understand its outputs
- allow to understand the behavior of a DNN
- allow users to understand why prediction is made
- assist understanding how the system processes data
- better understand what a model has learned
- can be easily comprehended
- can increase our understanding of the models
- can strongly infuence users’ understanding of complex data
- concerned with enabling human understanding
- effects on user understanding
- enable domain experts and users to understand the model
- enable human users to understand
- enable human users to understand AI systems
- enable understanding
- enhance understanding of training situations
- enrich people's understanding
- equip users to understand the system
- facilitate the understanding of the system
- get insights into predictions
- help a user to understand why an item is recommended
- help the human to understand the behavior
- help the human to understand why an agent failed
- help the user better to understand the rationale
- help user to understand agent behavior
- help user to understand why item is recommended
- help users to understand system's output
- help users understand the suggestions made
- help users understand the system's operation domain
- helpful for understanding intelligent system's reasoning process
- hint to better understanding of the job recommendations
- important for program comprehension
- important to understand how the system operates
- improve understandability of ML
- improve user understanding of the system
- improve users' understanding
- improving user understanding
- interpretability
- it can help build understanding
- key to understanding models better
- know the system's reasoning
- make decisions understandable
- make opaque algorithms more understandable
- positive effect on understandability
- provide insights into the model
- show a significant improvement in user understanding
- support intelligibility
- tell us something about the decison-making process
- tend to cause understanding
- tool to build understanding of the technology
- understand and contextualize the system strategy
- understand and evaluate the model
- understand system behavior
- understand system's reasoning
- understand the relevance feedback algorithm
- understanding
- understanding a system
- understanding how the application works
- understanding of why the outcome arose
- understanding system behavior
- way to better understand decisions
- way to interpret the decisions

Usability_N

- not every software should provide explanations
- reducing the sense of ease of use
- the focus is not on usability

Usability_P

- better utilize the AI
- can help reduce complex cognitive efforts
- crucial aspect for usability
- ease of use of the recommender system
- ensure effective interactions with ML systems
- help to improve the usability of the system
- improve usability
- increase ease of use
- leads to more effective use
- leads to more efficient use
- make better use of the system
- make effective use of system
- make it quicker and easier for users to find what they want
- reduce cognitive burden
- save user's cognitive effort
- save user's interaction efforts
- substantially affect usability
- to make a better use of a system's outputs
- usability

User Awareness_P

- can promote awareness
- increase situational awareness
- serve awareness

User Control_P

- control
- enable users to control their interaction more actively
- give greater control to the user
- human control over the decision
- meaningful human control over algorithms
- provide control
- starting point for better user control

User Effectiveness_N

- can be conflicting with effectiveness
- lead to agreement with incorrect system suggestions
- lead to automation bias when familiar with problem

User Effectiveness_P

- allow to assess whether the alternative is appropriate
- can lead to greater accuracy in decision making
- effectiveness
- improve users' decision effectiveness
- necessary for the user to carry out his or her task
- user effectiveness

User Efficiency_N

- require significant human effort
- taking more time
- users need to take time to analyze explanations

User Efficiency_P

- decreasing the time needed to reach a judgement
- help users make decisions faster
- help users make faster decisions
- reduce time for decision making
- user efficiency
- user's decision efficiency
- users need less time understanding the interface

User Experience_N

- become distracting over time
- can pollute user interface
- may have contributed to confusion or surprise
- too much explanation can create confusion

User Experience_P

- change user attitude towards the system
- improve users' experience
- influences actions, emotions and beliefs of the recipient
- lead to more positive user attitudes
- make the system more engaging
- play an important role in improving user experience
- promote feelings of familiarity
- user experience

User Performance_P

- affect test performance
- better user performance
- can increase performance on information retrieval tasks
- improve task performance
- improve user performance
- improved decision quality
- improved problem solving performance
- user performance

User Satisfaction_P

- be more comfortable
- brings enjoyment to users
- can increase user satisfaction
- can lead to higher level of satisfaction
- contributes to user satisfaction
- decrease frustration
- feel more comfortable
- has a positive effect on satisfaction
- higher degrees of satisfaction
- important for user satisfaction
- improved user satisfaction
- improving user satisfaction
- increase enjoyment
- increase satisfaction
- increase user satisfaction
- increase users' satisfaction
- increased participants' satisfaction with the recommendation
- lead to better user satisfaction
- lead to greater satisfaction
- lead to greater user satisfaction
- positive effect on satisfaction
- positive relationship with user satisfaction
- promote feelings of comfort
- satisfaction
- significantly impact satisfaction
- user satisfaction

Validation_P

- help users assess if the recommended alternative is truly adequate for them
- validate system knowledge
- way of validating the decision process of an AI
- way to verify the validity of a decision

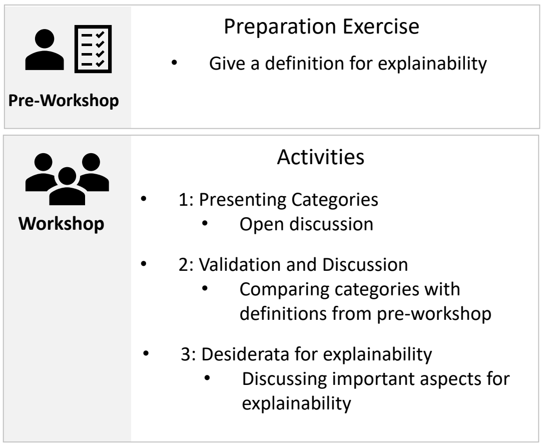
Verifiability_P

- assess whether a prediction is accurate
- check the system by examining the way it reasons
- enable users to verify that an algorithm is accurate
- ensure that algorithms are performing as expected
- ensure that algorithms perform as expected
- evaluate correctness of system ouputs
- evaluate the goodness of a match
- evaluate the proposed solution
- help users evaluate the accuracy of the system's prediction
- increase the ability to assess whether a prediction is accurate
- verification
- verify accuracy
- verify correct functioning
- verify that the system works as it should
- verify the correctness of the knowledge base or structure

B.4 Workshop Exercises

B.4.1 Workshop with Philosophers and Psychologists

The workshop with philosophers and psychologists consisted of one pre-workshop exercise and three workshop activities. The structure of the workshop is presented in Figure B.1.



Pre-Workshop

Preparation Exercise

- Give a definition for explainability

Workshop

Activities

- 1: Presenting Categories
  - Open discussion
- 2: Validation and Discussion
  - Comparing categories with definitions from pre-workshop
- 3: Desiderata for explainability
  - Discussing important aspects for explainability

Figure B.1: Workshop with philosophers and psychologists

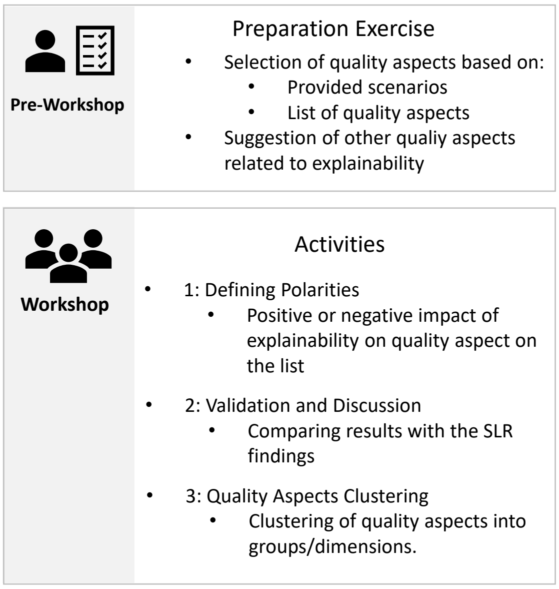
Pre-workshop exercise We asked participants to prepare a definition of explainability according to the participant's background before our workshop. The idea was to collect these definitions before the discussion to allow for comparison and to avoid bias from our preliminary results and the debate.

During the workshop In the first activity, we presented the categories regarding RQ1 found in the literature for discussion. We discussed whether these categories are accurate with what the participants consider explainability to be. In the second activity, the idea was to understand if the definitions found in the literature match participants' own definition of explainability, submitted before the workshop. We compared these categories and discussed the differences in order to reach a consensus. During the last activity, we discussed important *desiderata* for explainability. In this case, desiderata is the philosophical equivalent for quality aspects.

B.4.2 Workshop with Requirements Engineers

The workshop with requirements engineers consisted of one pre-workshop exercise and three workshop activities. The structure of the workshop is presented in Figure B.2.

Pre-workshop exercise For the preparation exercise, we asked participants to name quality aspects that are impacted by explainability. We sent a list of quality aspects resulting from our coding process without the detected polarities to avoid bias. To support participants, we also developed four hypothetical scenarios (subsection B.4.3) where explainable systems had to be designed. The scenarios consisted of short stories describing a domain and a business problem related to the need for explainability. The goal was to help participants in better understanding contexts where explainable systems may be needed. Based on the scenarios, we asked participants to specify desirable quality aspects for each system based on their expertise, and how explainability would interact with each of these aspects. Next, we asked participants to elaborate a list of the quality aspects and to specify if explainability would impact positively or negatively on each aspect. We also welcomed participants' suggestions about further quality aspects that could be connected to explainability but were not present in our list.



Pre-Workshop

Preparation Exercise

- Selection of quality aspects based on:
  - Provided scenarios
  - List of quality aspects
- Suggestion of other qualiy aspects related to explainability

Workshop

Activities

- 1: Defining Polarities
  - Positive or negative impact of explainability on quality aspect on the list
- 2: Validation and Discussion
  - Comparing results with the SLR findings
- 3: Quality Aspects Clustering
  - Clustering of quality aspects into groups/dimensions.

Figure B.2: Workshop with requirements engineers

During the workshop In the first activity, we presented the list of quality aspect without polarities to the participants and asked them to set the polarities. We established a round structure for this activity, where each participant would first define the polarity, justify the decision and at the end of the round all participants could discuss each others' choices. The idea was to provoke debate and reach consensus. In the second activity, we compared the polarities given by the participants with the findings from our coding process. We compared the results and had an open discussion to discuss differences and reach consensus. Experts could relate to all polarities that they did not mention initially, but were found in the literature.

In the third activity, we clustered the quality aspects collaboratively based on their relationship and discussed their impacts on the system.

B.4.3 Homework

For the workshop with requirements engineers, we designed the following hypothetical scenarios. Note that the workshops were originally in German, so the following scenario descriptions are translations.

Loan Issuing Software Bank Tresor Holdings Ltd. wants to change its loan issuing software. In the past, the bank has often come under suspicion of including gender or ethnicity when determining creditworthiness. For this reason, the new system is to be equipped with a feature that will allow the employee to find out why a particular person gets the calculated credit score. The system should be able to explain to the employee what the most important factor was that contributed to that exact classification, and what the customer would have to change to get a different (better) credit score. Tresor Holdings employees are experts in their field and can work through even complicated issues in a way that customers can understand. The customers themselves do not get to see anything from the system.

Medical Diagnosis System Saint Health hospital not only wants to help doctors make better judgments, but also to upgrade the doctor-patient interaction. To this end, it plans to introduce a new interactive disease prognosis system that not only reliably explains to doctors how a particular prognosis came about, but can also provide this information tailored to patients. With the help of the system, doctors will be able to better educate patients about their diagnoses so they can accept or reject a particular treatment in an informed and justified manner. Beyond being understandable to patients, of course, the system should also be able to provide doctors with additional specialist information that patients might not be able to relate to.

Applicant Selection Software The company GoodJobs wants to automate most of its hiring process. Before a select few applicants get a real interview, an initial screening is to take place completely online. The final result should not only be presented to the applicants, but also made comprehensible for them. Especially the rejected applicants should be able to understand what led to their rejection and how they can improve for the next time. The recipients of the explanations should not only be the applicants, but also the legal department, which must determine whether certain rejections were justified.

Autonomous Driving The car manufacturer AutoNomous GmbH is the first car manufacturer in the world to offer fully autonomous vehicles. As with normal vehicles, however, accidents do occur from time to time with autonomous vehicles, but much less frequently than with conventional automobiles driven by humans. In order to be able to resolve these rare accidents without the help of experts, an explanatory component is to be added to the software. The company can no longer afford to provide an expert to analyze and assess each accident. The software extension should make it easy for accident investigators to find out for themselves what led to the accident and who is responsible.

C
Interview Study Supplementary Material

This is the supplementary material related to the interview study explored in chapter 7. In particular, I present methodological information about the literature search as well as additional information about the interview study.

C.1 Literature Search

C.1.1 Definition of the search string

The search string was built using a combination of the keywords that were thought to be crucial for this study, namely *explainability, software,* and *system.* Words like *practice, activity, technique,* or *process* were omitted from the search string in order to broaden the solution space and find more information on activities, practices, and techniques that may not be specifically labeled as such. We also chose to concentrate specifically on requirements engineering and human-computer interaction since, as concluded in chapter 6, developing explainable systems is highly connected to aspects of the communication interface, which highlights the importance of focusing on the needs of the addressees of explanations. This line of thought lead to the final search string, which was extended by synonyms and abbreviations (e.g., *SE* for *software engineering*):

(explain* *OR* XAI)
AND
(system *OR* software *OR* design *OR* interface
OR HCI *OR* ”human-computer interaction”
OR RE *OR* ”requirements engineering”
OR SE *OR* ”software engineering”)

C.1.2 Inclusion and Exclusion criteria

Studies and publications that were not relevant for answering our research questions were eliminated. For a publication to be included, all inclusion criteria had to be met. A publication was rejected if at least one of the exclusion criteria was met. To increase the objectivity of the decision on the inclusion or exclusion of a paper, inclusion and exclusion criteria were formulated (Table C.1). Since explainability is an interdisciplinary research topic and since we wanted to focus on the development process of such systems, publications that are not related to computer science were removed (EC_5).

Table C.1: Inclusion (IC) and Exclusion (EC) Criteria

	Criterion	Description
Inclusion	IC_1	Publications that present methodologies, processes, activities, practices, or techniques to develop explainable systems.
	IC_2	Publications that are a peer-reviewed journal, conference, or workshop paper
Exclusion	EC_1	Publications that do not mention activities, practices, or techniques.
	EC_2	Tutorials, Proposals, and other non-peer reviewed publications
	EC_3	Publications that are not accessible (via university licenses)
	EC_4	Publication is neither written in German nor in English.
	EC_5	Publication is not related to computer science.

C.1.3 Database selection

Due to time constraints, sources are reduced in rapid reviews. Since selecting studies for a literature review can be a time-consuming and laborious process, we decided to conduct our search using Google Scholar (GS). GS retrieves results from all major databases, such as ACM Digital Libraries, IEEExplore, and Web of Science. Yasin et al. [473] investigated the suitability of GS for literature reviews and found that GS was able to retrieve 96% of primary studies in comparison to traditional database searches. The use of just one database introduces a threat to validity (presented in subsection 7.3.5). However, GS was sufficient since the goal was to conduct a rapid review of the literature rather than a thorough, systematic one.

C.1.4 Termination criterion

We applied again the concept of Wolfswinkel et al. [151] (cf. subsubsection 6.1.1) to define the termination criterion *. We followed this approach to decide when to conclude our search process. It is important to note that we only assume saturation with respect to the subset of publications we looked at during our literature search, not saturation in the traditional sense based on all available data. More specifically, we made the decision to stop looking after finding 50 papers one after another without gaining any new knowledge, that is, without at least one paper extending our solution space of practices and techniques. Therefore, I believe that the information gathered offers enough insights to serve as a solid foundation for addressing the research questions.

C.2 Interview with Practitioners

C.2.1 Interview Structure

1. Welcome
2. Presentation of the topic and the structure of the interview
3. Task 1: Draw a diagram of the company's existing software development process
4. Definition of explainability and presentation of a scenario
5. Example: Scenario on the planned development of a system that should be explainable. The participant should develop a process allowing to address explainability in the process
6. Task 2: Draw a diagram of a development process for explainable software and systems
7. Follow-up question: Applicability in industry
8. Closure

*"A literature review is never complete but at most saturated. This saturation is achieved when no new concepts or categories arise from the inspected data."

C.2.2 Practices found in the literature, organized by phase

Table C.2: Practices found in the literature, organized by phase

Practice	Sources
Requirements Engineering	
Focus Groups/Workshops	[224, 474]
Interviews	[207, 475, 250, 476, 208, 477, 298], [478, 76, 479, 219, 474]
Mental Models	[208, 219]
Personas	[480, 481, 12, 482, 483, 484], [485, 39]
Questionnaires	[208, 76, 486, 484, 487]
Scenarios	[480, 250, 246, 488], [94, 486, 224, 489, 490], [233, 485, 491, 39, 482]
Design/Implementation	
Low-Fidelity Prototypes	[208, 492, 485, 493, 494]
High-Fidelity Prototypes	[207, 13, 196, 208, 373, 495], [478, 76, 224, 77, 493, 494]
Validation/Testing	
Usability Tests	[196, 208, 496, 486, 224], [487, 494]
A/B Tests	[486, 491]
Interviews	[13, 208, 76, 224, 490, 207, 219, 494]
Mental Models	[492, 487, 208, 76, 219]
Questionnaires	[196, 486, 487, 489, 208]

C.2.3 Interview Script

Type: Semi-structured interview

Tools: Audacity (Voice Recording), Miro Board (Workflow design)

Interview Part One

1. Data about the interviewee
2. Data about the company
3. Introduction to the definition of explainable systems (c.f. section 6.2)
4. Translate definition into a concrete example for the interview
5. Questions about the explainability of the product the interviewee is working on

    (a) In your opinion, is the product you are working on explainable to the end user?
    (b) Can you explain the process of software development in your company?

6. Back-end analysis questions

    (a) Are your responsibilities divided into back-end and front-end?
    (b) Do you work on the back-end or front-end?
    (c) Are there meetings where back-end and front-end developers participate?

Interview Part Two

In this part of the interview, the interviewee should put himself in the role of a product owner of the fictitious company "*ExplainWare*". The task consisted of setting up the process flow of software development in this company.

Case Study ExplainWare
ExplainWare would like to develop a navigation software *X*, which suggests drivers a route with which they reach their destination quickly, safely and free of traffic jams. Each driver is actively distributed on the road network so that the roads are not roads are not overloaded in order to prevent traffic jams.

The challenge here is that end users need to understand that *X* not only suggests the fastest route without congestion, but actively prevents congestion by by distributing drivers appropriately across the road network. network. If the end user does not understand this, he will no longer want to use *X* because he thinks that the routes he already knows are faster.

Phase 1 - Questions

1. How would you analyze the stakeholders?
2. How would you elicit the requirements?
3. How would you document the requirements?
4. How would you validate the requirements?
5. What roles would you assign to your employees?
6. Do you maintain it is necessary for different roles to communicate with each other?

7. How would you ensure that the product is explainable?
8. How would you evaluate the product for explainability?

Phase 2 - Show guidelines

First, the interviewee was asked what he understood by the term "*requirement*"? Afterwards, the interviewee should explain each phase of Figure C.1 and correct if necessary if something is ”wrong” from the interviewee's point of view.

Requirements Analysis	Requirements Management	Design	Coding	Evaluation

Figure C.1: Phases in the development lifecycle

Furthermore, the following questions were asked:

1. Do you apply these phases in your company?

    (a) Do you have similar phases in your process?

        i. Yes: Which phases are similar?
        ii. No: *– go to the next question –*

2. Describe the process in your company

    (a) How are requirements for the software elicited?
    (b) How are the stakeholders identified/analyzed?
    (c) How are the requirements prioritized?
    (d) How are the requirements documented?
    (e) How are the requirements validated?

Phase 3 - Show components (all HCI & RE methods)

1. Do you apply these activities in your company? Which ones and in which phase?

Phase 4 - Show and compare developed workflow

1. What do you think about this workflow?
2. Do you think this workflow would be suitable for developing explainable software?
3. Which activities do you find suitable, which not and why?
4. Could you imagine implementing this in your company?
5. Do you think that this workflow can be integrated well into practice?

6. Do you think that this workflow is too costly in practice?
7. Would it be problematic to implement this workflow in your company? Why?

Finally

1. Is it worthwhile for the company to develop explainable software?
2. Do you consider explainability an important requirement that is worth the company's time and money?
3. What are the challenges to developing explainable software?
4. Can explainability be addressed during your company's development process?
5. Do you think that changes in your company's workflow would be easy to implement?

C.2.4 Interview Screenshot

Figure C.2 shows the methods that the literature review revealed. These could be used by the interviewees on the Miro Board and added to the individual activities.

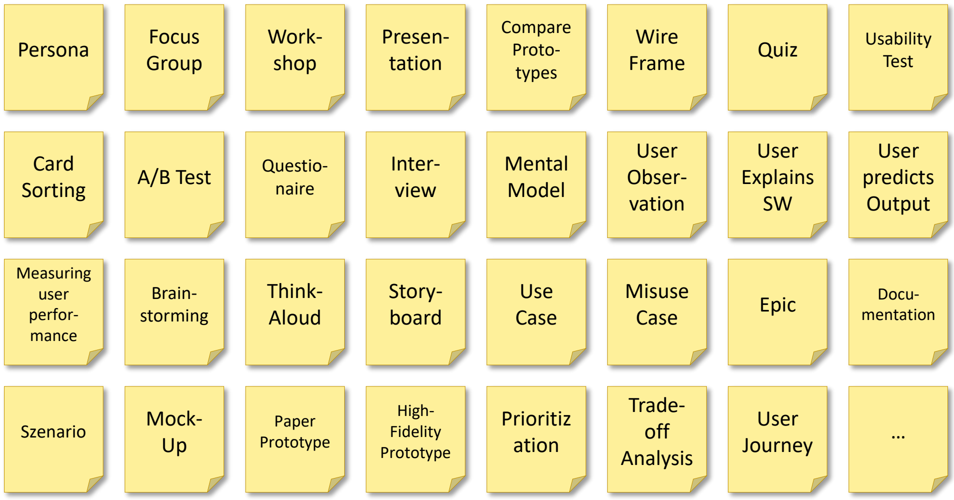


Figure C.2: Miro Board - Methods

Figure C.3 shows a section of the Miro board with the different phases in the development cycle. Here, the interviewees should assign the methods used in the company to the corresponding phase and, if necessary, also add further methods.

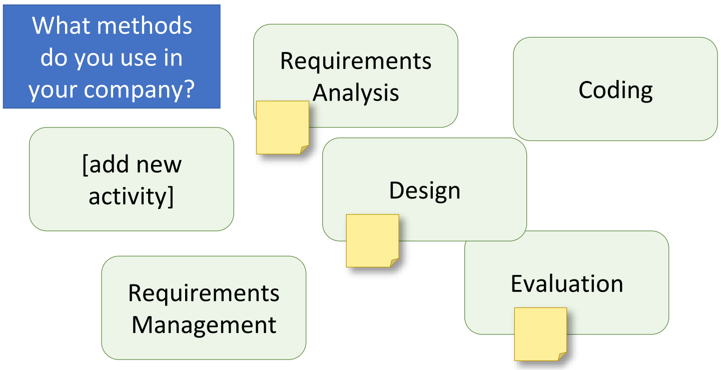


Figure C.3: Miro Board - Phase

D

Framework and Case Study Supplementary Material

D.1 Literature Search

D.1.1 Definition of the search string

The final search string is constructed in such a way that, in addition to the first condition, either a term from the second or third block of terms must match a search result:

> ((explainability *OR* explanation *OR* explanations *OR* explainable)
> *AND*
> (evaluation *OR* assessment *OR* analysis *OR* impact *OR* HCI *OR* “human-computer interaction” *OR* “human-computer interfaces” *OR* interaction *OR* “user interface” *OR* usability))

Since a search string is always subjectively defined from the researcher’s point of view, database searching and snowballing were again complemented by a manual search.

Table D.1: Inclusion (IC) and Exclusion (EC) Criteria

	Criterion	Description
Inclusion	IC_1	Publications that include either an evaluation of a specific statement or an overview of various evaluation options.
	IC_2	Publications that are a peer-reviewed journal, conference, or workshop papers.
	IC_2	Publications that consider end-users of explainable systems as stakeholders of explanations.
Exclusion	EC_1	Publications that focus exclusively on how explanations can be generated automatically (algorithm evaluation).
	EC_2	Publications that are limited exclusively to understanding underlying algorithms (ML-Interpretability).
	EC_3	Publications that are not accessible (via university licenses).
	EC_4	Publication that are neither written in German nor in English.
	EC_5	Publication that are not related to computer science.

D.1.2 Inclusion and Exclusion criteria

D.1.3 Database selection

The databases used for the search were *ACM Digital Library*[*], *IEEE Xplore*[†], *Science Direct*[‡], and *Springer Link*[§]. The aforementioned databases were chosen because they have already been used for literature searches in the area of explainability [69, 497]. When searching *Science Direct* and *Springer Link*, results were limited to *Computer Science*.

D.1.4 Termination criterion

In the literature search, we also followed the saturation principle based on Wolfswinkel [151], as described earlier in subsection C.1.4:

D.1.5 SLR References

Table D.2: References for aspects of the quality framework

Category	**Aspect in the Framework**	**References**
Objectives / Constraints	Objectives	[69]

[*] https://dl.acm.org

[†] https://ieeexplore.ieee.org

[‡] https://www.sciencedirect.com

[§] https://link.springer.com

	Construct	[236]
	Purpose	[69, 120]
	Goals	[250, 498, 233]
	Main Drive	[412]
	Intended Effect	[238]
Business Goals	Stakeholder Goals	[69, 70]
	Higher-level Goals	[69]
	Application Level Goals	[411]
Users' Perception	User Perceived Quality Factors	[69]
	(Consumer) Needs	[246, 116]
	User Goals	[246]
	Intermediate Requirements	[236]
	Human Level	[411]
Explanation Purpose	(Explanation) Purpose	[69]
	Explanatory Goal	[239, 238, 70]
	Function Level	[411]
Context	(Experimental) Context	[87, 116, 399, 236, 62, 255, 498, 53, 70]
	(Explanation) Scope	[120, 262, 53]
	Use Case	[236]
	Stakeholder	[61, 70]
User	(Target / End) User	[12, 258, 80, 197, 70]
	Stakeholder	[87]
	Consumer	[246]
	Explainee	[87, 62]
	Explanation Audience	[411, 70]
Task	Task	[87, 249]
	Activity	[120]
Environment	Environment	[87, 251]
	Application Area / Domain	[411, 251, 197, 70]
Demand	Demand	[87]
Initiative	Manual	[116, 239, 197]
	Automatic	[116, 262, 197, 347, 248]
Timing	Before	[61, 197, 263, 499, 500]
	During	[61, 197, 263]

	After	[61, 197, 263, 499, 500, 251]
Content	User Interface Component(s)	[69, 61]
	Content	[233]
	Granularity	[87, 62]
Information Type	Context	[12, 253, 259, 253, 240, 234, 69, 233]
	Inner Logic	[12, 243, 257, 234, 255, 233, 259]
	Causality	[12, 242, 248, 243] [253, 253, 240, 259, 257, 234, 255, 69, 254, 233, 392, 258]
Information Density	Amount	[233, 235, 256, 501]
	Abstraction Level	[257, 256]
Adaptivity	Context-Sensitivity	[258, 259]
	Controllability	[242, 196]
	Personalization	[258, 259, 80, 239, 411]
Presentation	Presentation	[61, 235]
	(Explanation) Type	[233, 61]
Medium	Textual	[411, 238, 239, 243, 262, 262, 242, 259, 70, 69]
	Visual	[411, 243, 265, 242, 69, 502, 70]
	Auditory	[251, 69, 268]
Tone	Factual	[262, 242, 263, 255]
	Emotional	[242, 263, 503, 253, 255]
Grouping	Single	[69, 238, 243, 262, 242]
	Grouped	[69, 238, 239]
Method	Perception Questionnaires	[504, 487, 268, 243, 503, 256, 399, 238]
Qualitative Indicators		
Explanations	Completeness	[504, 267]
	Persuasiveness	[243, 399]
	Usefulness	[243, 399, 238]
	for a comprehensive list	[267]

System	Satisfaction	[238]
	Transparency	[238, 268]
	Trust	[238, 503, 256]
	Scrutability	[238]
Quantitative Metrics	Number of explanation requests	[197, 116, 238]
	User Focus Duration	[238]
	Explanation Exposure Delta	[69]
	Frequency of system use	[505]
	Task performance	[505, 69, 236, 265, 239, 242, 240, 501, 221, 249, 263, 505]
	Learning Rate	[239, 249]
	Task Completion Time	[505]

D.2 Framework Versions

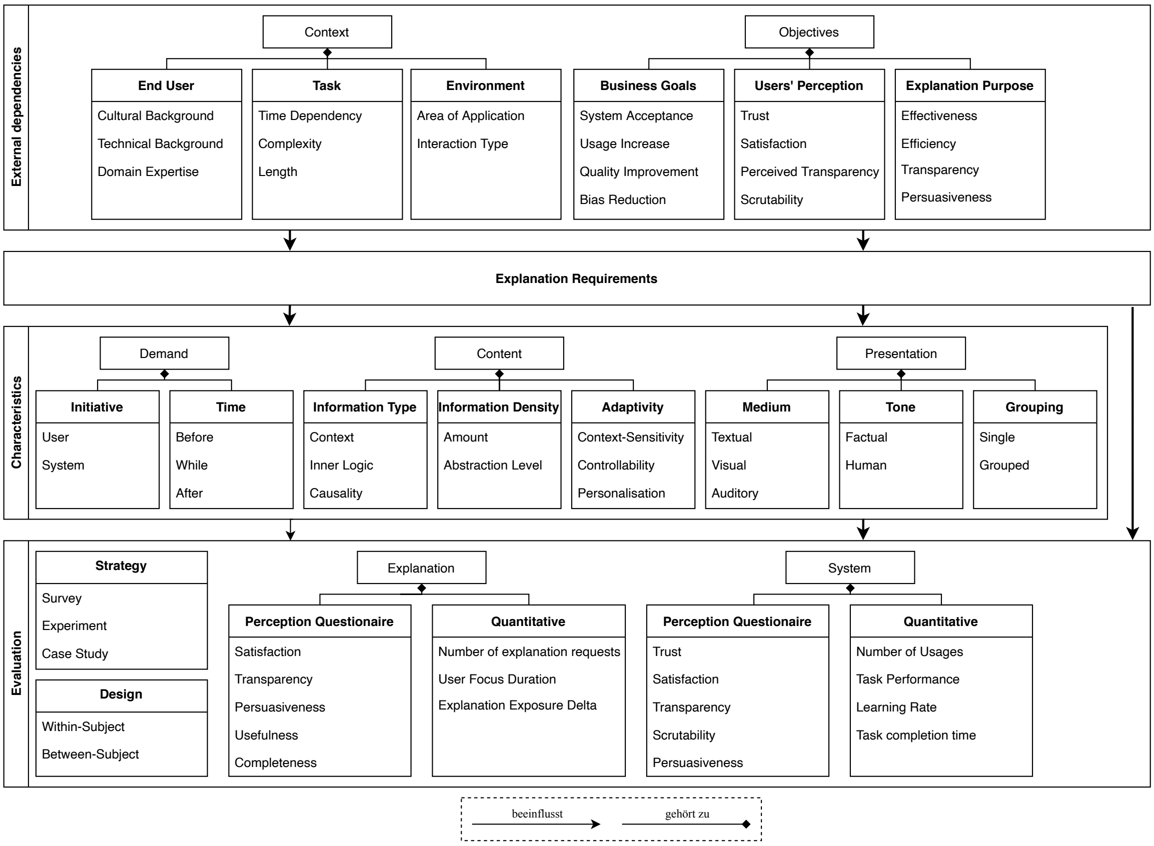


Figure D.1: The first version of the explainability framework, taken to the workshop.

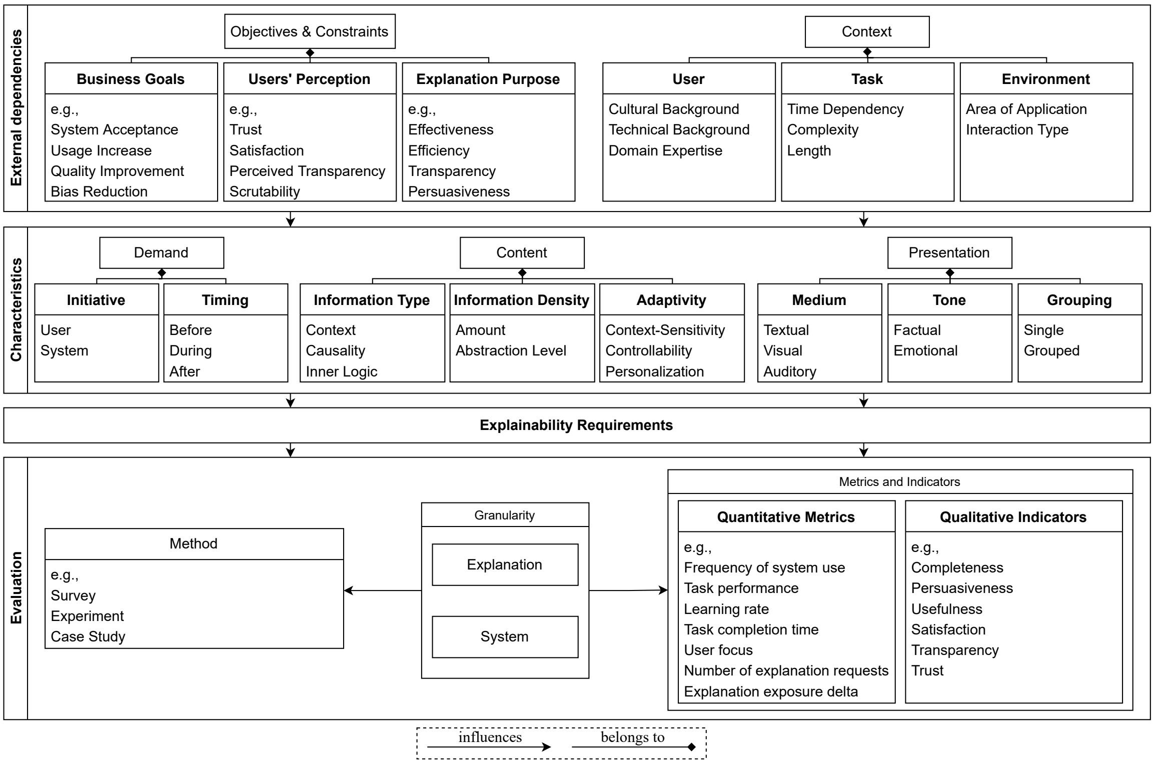


Figure D.2: The second version of the explainability framework, taken to the focus group.

D.3 Workshop with Software Engineers

In order to define the goals and requirements for the explanations and to collect initial first implementation ideas, a workshop was held with software engineers, and a customer support expert. By directly applying the developed guideline in the workshop, it was also possible to collect observations on how the framework can be useful. After the initial introduction on explainability, the guide for integrating explanations, and the results from the user feedback analysis, the workshop followed the structure of the framework for explanations. Consequently, information about potential users (Context), then goals (Objectives) and their implementation possibilities (Characteristics), and finally evaluation possibilities were discussed (Evaluation). The overview of the workshop is shown below:

Activity	Description
1. Introduction to Explainability	- Explainability as a NFR - Benefits of explainability in a mobility context - Presentation of the framework - Examples of explanations
2. Presentation of Common Problems	- Common problems found in user feedback - Prioritization of problems to be solved
3. Discussion about Personas	- User profiles, contexts
4. Definition of Goals	- Definition of main quality objectives - Concretization of objectives based on the framework
5. Brainstorming about Design Options	- Discussion about aspects of explanations - Aspects of interface design
6. Definition of Metrics	- Discussion about metrics and their feasibility - Definition of evaluation strategy

Table D.3: Workshop structure

D.4 Voice Prompts for ER3

The goal of explainability requirement (ER) 3 was to explain possible reasons for what, from an end-user' perspective, could be considered as a "weird" route. The chosen approach to explain this was to display traffic events on the current route. This context information should help to understand why the application recommends unusual routes. However, this is not trivial, since the calculation of the traffic volume and its subjective perception of the end user is not clear. Another problem is that it was a technical challenge to define which route would be subjectively perceived as "normal" to each end-user on the same route. Therefore, the chosen solution was to calculate the ratio of average route duration and use it as baseline for comparison. The data is then mapped to "low traffic", "moderate traffic", and "high traffic" levels. There is also a "normal" traffic situation where no explanation is shown to the end users. The threshold values were determined internally through test drives. End-users could obtain the information in three ways. First, the information was displayed in the route preview, together with the information about the route duration. The total travel time for the route was displayed in green (low traffic), standard text color (normal traffic), orange (moderate traffic) and red (heavy traffic). Since it was noticed in the first prototype that the colors alone were not meaningful, a short explanatory text was added. In addition, throughout the navigation, the arrival time and the travel time are displayed in the appropriate color and also updated as the traffic situation changes. During navigation, information about traffic volume was provided

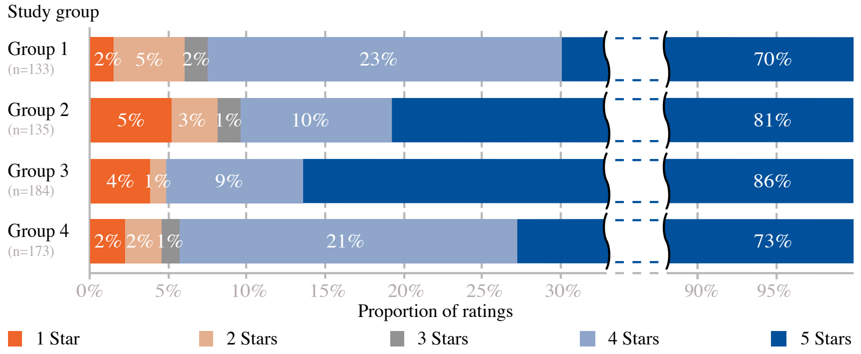


Figure D.3: Route satisfaction represented by star ratings

via synthesized speech as shown in Table D.4. The tone of these voice prompts was somewhat emotional to create sympathy.

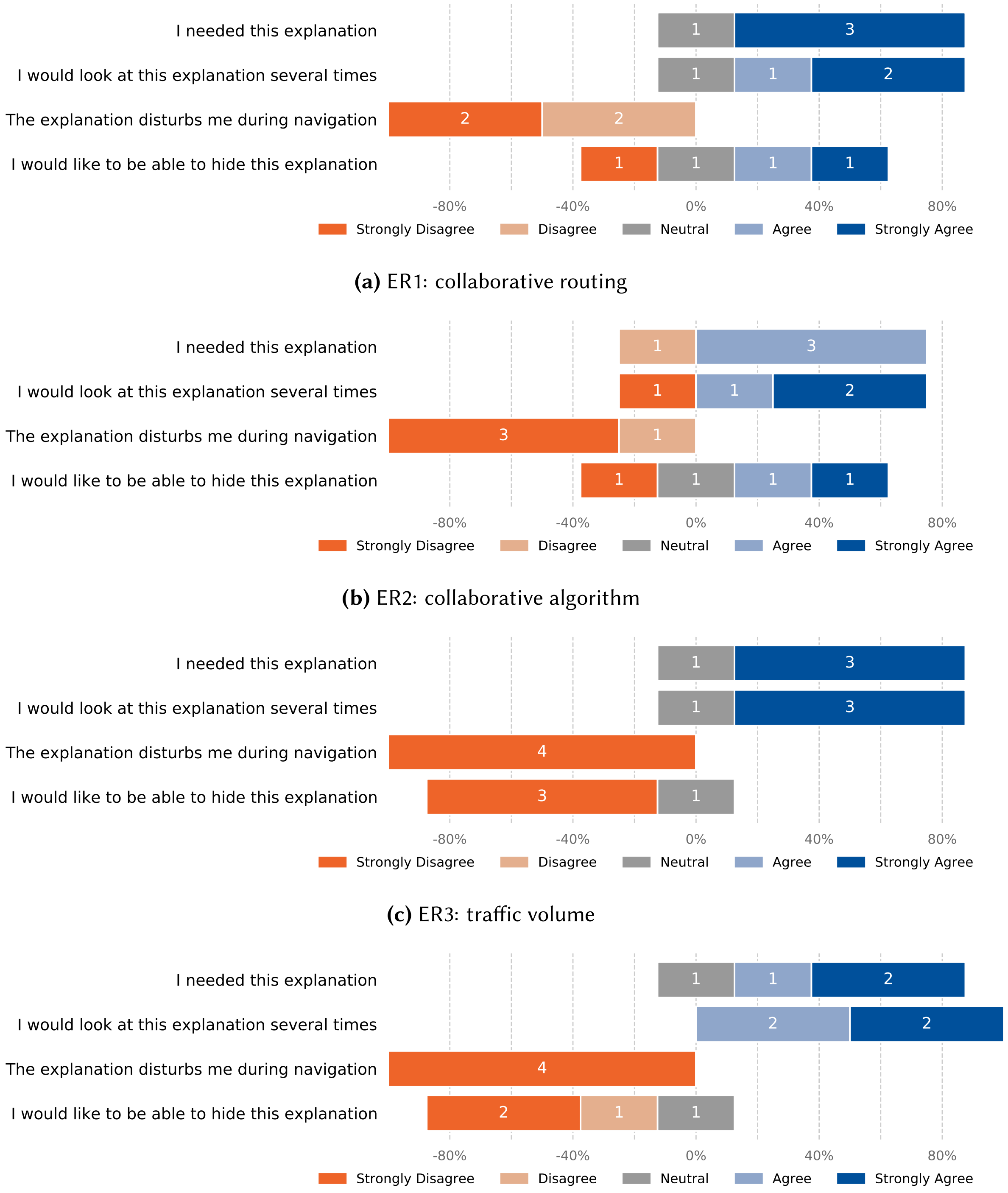
Traffic Volume	Voice Prompt
low	Today the roads are clear. You will reach your destination <destination name> at <time>.
normal	You will reach your destination <destination name> at <time>.
moderate	Since today is a bit busier, you will reach your destination <destination name> at <time>.
high	Since today is very busy, you will reach your destination <destination name> at <time>.

Table D.4: Voice prompts at the beginning of navigation dependent on the traffic volume

D.5 Additional Charts

In the quantitative analysis, we used the interactive format of the static explanations (ER1 and ER2) to get additional data as already described in chapter 9. Figure D.3 shows how the star ratings for the calculated routes differed between groups.

The results of the qualitative evaluation are depicted in Figure D.4 and Figure D.5. Figure D.4 shows the ratings on statements concerning the demand for explanations and Figure D.5 the ratings concerning the content of explanations.



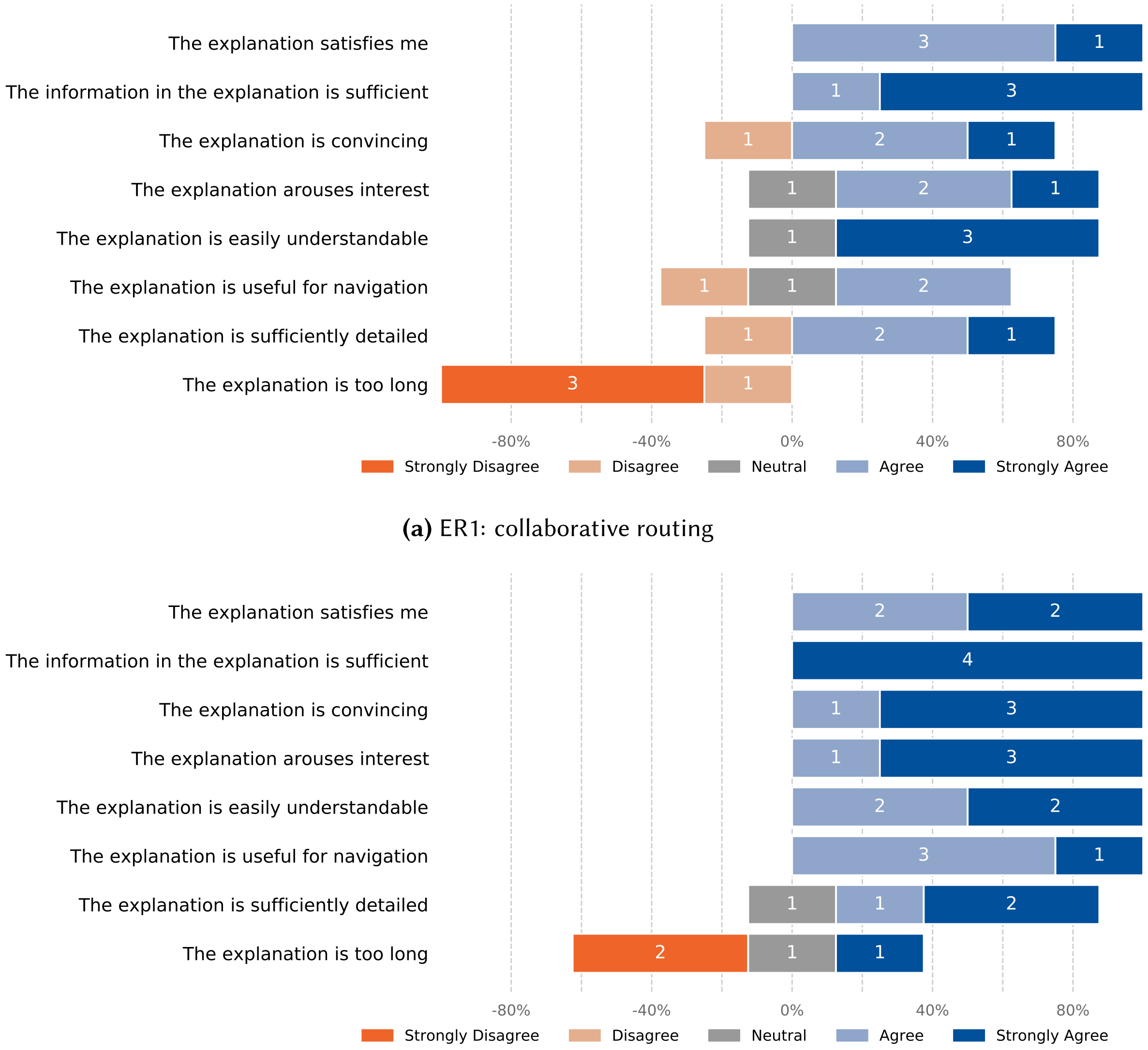
(a) ER1: collaborative routing

(b) ER2: collaborative algorithm

(c) ER3: traffic volume

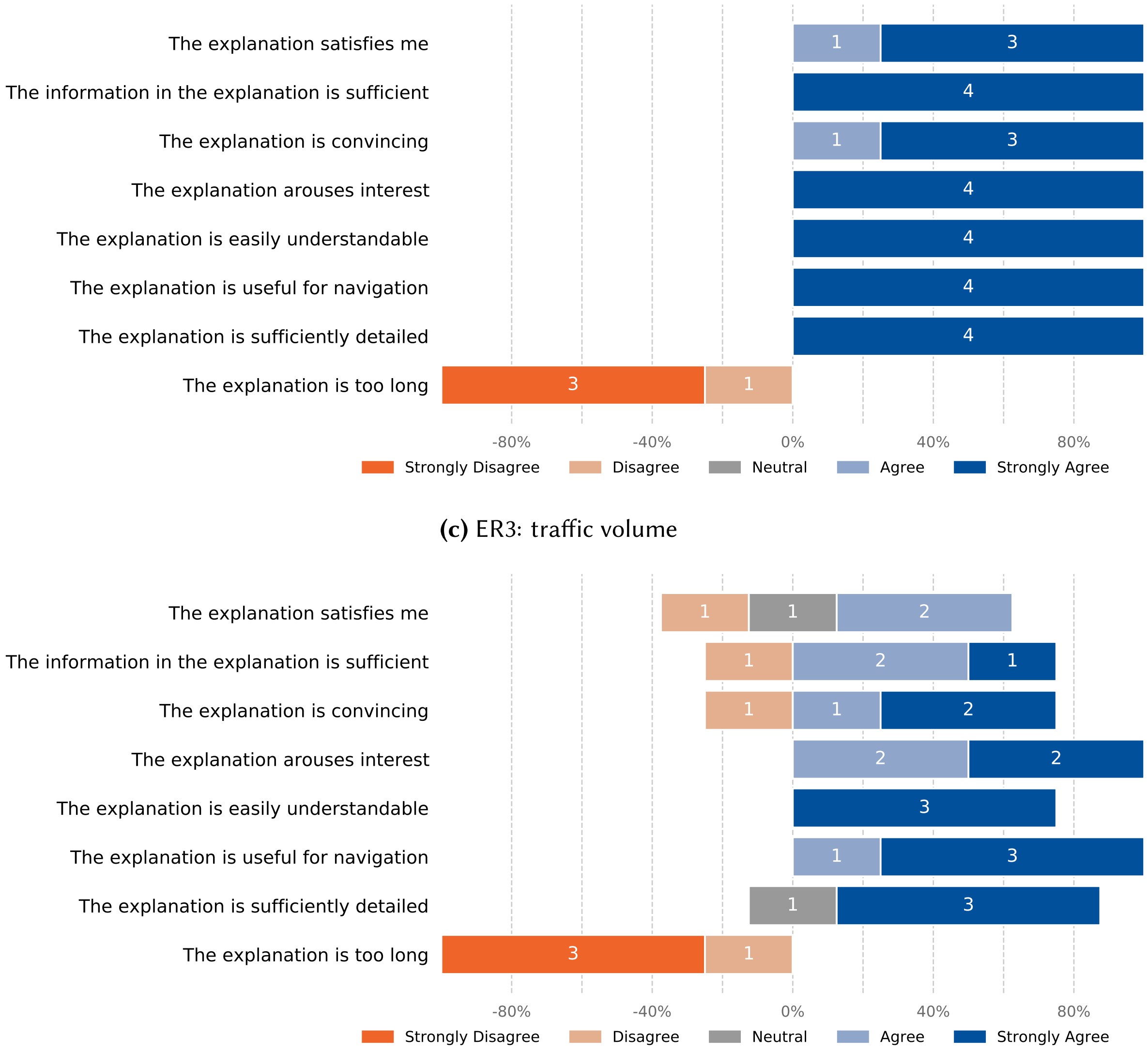
(d) ER4: position accuracy

Figure D.4: Results of the qualitative study for the explanation demand



(a) ER1: collaborative routing

(b) ER2: collaborative algorithm



(c) ER3: traffic volume

(d) ER4: position accuracy

Figure D.5: Results of the qualitative study for the explanation content

References

[1] E. Börger, B. Hörger, D. Parnas, and D. Rombach, "Requirements capture, documentation, and validation," in *Dagstuhl Seminar*, no. 99241. Dagstuhl Seminar, 1999.

[2] J. Droste, W. Brunotte, L. Chazette, and K. Schneider, "softXplain: Requirements for Self-Explaining Systems," https://doi.org/10.13140/RG.2.2.10669.77280, August 2022, online; accessed 31 August 2022.

[3] D. I. Sjøberg, T. Dybå, B. C. Anda, and J. E. Hannay, "Building theories in software engineering," in *Guide to advanced empirical software engineering*. Springer, 2008, pp. 312–336.

[4] Hevner, March, Park, and Ram, "Design Science in Information Systems Research," *MIS Quarterly*, vol. 28, no. 1, p. 75, 2004. doi: 10.2307/25148625. [Online]. Available: http://www.jstor.org/stable/25148625

[5] J. Nielsen, *Usability Engineering*, 1st ed. San Francisco, CA, USA: Morgan Kaufmann, November 1994. ISBN 978-0125184069

[6] V. Vakkuri, K.-K. Kemell, J. Kultanen, and P. Abrahamsson, "The current state of industrial practice in artificial intelligence ethics," *IEEE Software*, vol. 37, no. 4, pp. 50–57, 2020.

[7] B. K. Kahn, D. M. Strong, and R. Y. Wang, "Information Quality Benchmarks: Product and Service Performance," *Commun. ACM*, vol. 45, no. 4, p. 184–192, apr 2002. doi: 10.1145/505248.506007

[8] A. Seffah and E. Metzker, "The obstacles and myths of usability and software engineering," *Communications of the ACM*, vol. 47, no. 12, pp. 71–76, 2004.

[9] M. Weiser, "The computer for the 21st century," *ACM SIGMOBILE mobile computing and communications review*, vol. 3, no. 3, pp. 3–11, 1999.

[10] F. J. Furrer, "Software everywhere," in *Future-Proof Software-Systems.* Springer, 2019, pp. 3–10.

[11] I. Ozkaya, "Can We Really Achieve Software Quality?" *IEEE Software*, vol. 38, no. 3, pp. 3–6, 2021. doi: 10.1109/MS.2021.3060552

[12] L. Chazette and K. Schneider, "Explainability as a non-functional requirement: challenges and recommendations," *Requirements Engineering*, vol. 25, no. 4, pp. 493–514, 2020. doi: 10.1007/s00766-020-00333-1

[13] A. Bussone, S. Stumpf, and D. O'Sullivan, "The Role of Explanations on Trust and Reliance in Clinical Decision Support Systems," in *2015 International Conference on Healthcare Informatics.* New York, NY, USA: IEEE, 2015. doi: 10.1109/ICHI.2015.26 pp. 160–169.

[14] J. P. Winkler and A. Vogelsang, ""What Does My Classifier Learn?" A Visual Approach to Understanding Natural Language Text Classifiers," in *Natural Language and Information Systems*, F. Frasincar, A. Ittoo, L. M. Nguyen, and E. Métais, Eds. Springer, 2017. doi: 10.1007/978-3-319-59569-6_55. ISBN 978-3-319-59569-6 pp. 468–479.

[15] J. Zhou and F. Chen, "Towards Trustworthy Human-AI Teaming under Uncertainty," in *Proceedings of the IJCAI Workshop on Explainable Artificial Intelligence (XAI)*, 2019, pp. 143–147.

[16] B. Paech and D. Kerkow, "Non-functional Requirements Engineering - Quality is Essential," in *10th International Workshop on Requirements Engineering Foundation for Software Quality*, 2004.

[17] D. Mairiza and D. Zowghi, "Constructing a Catalogue of Conflicts among Non-functional Requirements," in *Evaluation of Novel Approaches to Software Engineering*, L. A. Maciaszek and P. Loucopoulos, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. ISBN 978-3-642-23391-3 pp. 31–44.

[18] M. Glinz, "A glossary of requirements engineering terminology," *Standard Glossary of the Certified Professional for Requirements Engineering (CPRE) Studies and Exam, Version*, vol. 1, p. 56, 2011.

[19] K. Schneider, *Abenteuer Softwarequalität: Grundlagen und Verfahren für Qualitätssicherung und Qualitätsmanagement.* Heidelberg, DE: dpunkt.verlag, 2012. ISBN 3898647846

[20] Flavien Huynh, “Software quality is not a myth: Real examples,” 2020. [Online]. Available: https://www.coderskitchen.com/software-quality-real-examples/

[21] “37 Epic Software Failures that Mandate the Need for Adequate Software Testing,” 2016. [Online]. Available: https://www.cigniti.com/blog/37-software-failures-inadequate-software-testing/

[22] ISO Central Secretary, “ISO/IEC 25010:2011 Systems and Software Engineering-Systems and Software Quality Requirements and Evaluation (SQuaRE) - System and Software Quality Models,” International Organization for Standardization, Standard ISO/IEC 25010:2011, 2011. [Online]. Available: https://www.iso.org/standard/35733.html

[23] B. Nuseibeh and S. Easterbrook, “Requirements Engineering: A Roadmap,” in *Proceedings of the Conference on The Future of Software Engineering*, ser. ICSE ’00. New York, NY, USA: Association for Computing Machinery, 2000. doi: 10.1145/336512.336523. ISBN 1581132530 p. 35–46.

[24] J. Valacich and C. Schneider, *Information Systems Today: Managing the Digital World: Global Edition*, 4th ed. Pearson Education, 2009. ISBN 9780138157623

[25] I. Sommerville, *Software Engineering*, 8th ed. Harlow, England: Addison-Wesley, 2007. ISBN 9780321313799

[26] M. Glinz and S. A. Fricker, “On shared understanding in software engineering: an essay,” *Computer Science - Research and Development*, vol. 30, no. 3, pp. 363–376, Aug 2015. doi: 10.1007/s00450-014-0256-x

[27] K. Pohl and C. Rupp, *Basiswissen Requirements Engineering: Aus- und Weiterbildung nach IREB-Standard zum Certified Professional for Requirements Engineering Foundation Level*, 5th ed. Heidelberg: dpunkt.verlag, 2021. ISBN 9783969102473. [Online]. Available: https://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&db=nlabk&AN=2913478

[28] C. Rupp, M. Simon, and F. Hocker, “Requirements Engineering und Management,” *HMD Praxis der Wirtschaftsinformatik*, vol. 46, no. 3, pp. 94–103, Jun 2009. doi: 10.1007/BF03340367

[29] I. C. S. S. E. T. Committee, *ANSI/ IEEE Std 729-1983: IEEE Standard Glossary of Software Engineering Terminology.* Institute of Electrical and Electronics Engineers (IEEE), 1983. ISBN 1-5044-0425-4. [Online]. Available: http://lib.ugent.be/catalog/ebk01:3710000000614477

[30] V. Salehi and L. Burseg, "System Driven Product Development (SDPD) by Means of Development of a Mechatronic Systems in an Industrial Context," in *Product Lifecycle Management in the Era of Internet of Things*, A. Bouras, B. Eynard, S. Foufou, and K.-D. Thoben, Eds. Cham: Springer International Publishing, 2016. ISBN 978-3-319-33111-9 pp. 729–737.

[31] M. Glinz, "On Non-Functional Requirements," in *15th IEEE International Requirements Engineering Conference (RE 2007)*, 2007. doi: 10.1109/RE.2007.45 pp. 21–26.

[32] M. Daneva, L. Buglione, and A. Herrmann, "Software Architects' Experiences of Quality Requirements: What We Know and What We Do Not Know?" in *Requirements Engineering: Foundation for Software Quality*, J. Doerr and A. L. Opdahl, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. ISBN 978-3-642-37422-7 pp. 1–17.

[33] E. A. C. Bittner and J. M. Leimeister, "Why Shared Understanding Matters – Engineering a Collaboration Process for Shared Understanding to Improve Collaboration Effectiveness in Heterogeneous Teams," in *2013 46th Hawaii international conference on system sciences*. Piscataway: IEEE, 2013. doi: 10.1109/HICSS.2013.608. ISBN 978-1-4673-5933-7 pp. 106–114.

[34] S. Easterbrook, "Coordination breakdowns: why groupware is so difficult to design," in *Proceedings of the Twenty-Eighth Annual Hawaii International Conference on System Sciences*, vol. 4. IEEE, 1995, pp. 191–199.

[35] Ribeiro, Marco Túlio, Sameer Singh, and Carlos Guestrin, ""Why Should I Trust You?": Explaining the Predictions of Any Classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM, 2016. doi: 10.1145/2939672.2939778 pp. 1135–1144.

[36] P. W. Koh and P. Liang, "Understanding black-box predictions via influence functions," *arXiv preprint arXiv:1703.04730*, 2017.

[37] J. A. Konstan and J. Riedl, "Recommender systems: from algorithms to user experience," *User modeling and user-adapted interaction*, vol. 22, no. 1-2, pp. 101–123, 2012.

[38] D. V. Pynadath, M. J. Barnes, N. Wang, and J. Y. C. Chen, *Transparency Communication for Machine Learning in Human-Automation Interaction*. Cham: Springer International Publishing, 2018, pp. 75–90. ISBN 978-3-319-90403-0

[39] W. Xu, "Toward Human-Centered AI: A Perspective from Human-Computer Interaction," *Interactions*, vol. 26, no. 4, p. 42–46, jun 2019. doi: 10.1145/3328485

[40] A. C. Scott, W. J. Clancey, R. Davis, and E. H. Shortliffe, "Explanation Capabilities of Production-Based Consultation Systems," *American Journal of Computational Linguistics*, pp. 1–50, 1977.

[41] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual Explanations From Deep Networks via Gradient-Based Localization," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017, pp. 618–626.

[42] B. Kim, E. Glassman, B. Johnson, and J. Shah, "iBCM: Interactive Bayesian case model empowering humans via intuitive interaction," Massachusetts Institute of Technology - Computer Science and Artificial Intelligence Laboratory, Cambridge, MA, USA, Technical Report MIT-CSAIL-TR-2015-010, April 2015, http://hdl.handle.net/1721.1/96315.

[43] M. Možina, J. Demšar, M. Kattan, and B. Zupan, "Nomograms for Visualization of Naive Bayesian Classifier," in *Knowledge Discovery in Databases: PKDD 2004*, J.-F. Boulicaut, F. Esposito, F. Giannotti, and D. Pedreschi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. ISBN 978-3-540-30116-5 pp. 337–348.

[44] A. Jakulin, M. Možina, J. Demšar, I. Bratko, and B. Zupan, "Nomograms for Visualizing Support Vector Machines," in *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, ser. KDD '05. New York, NY, USA: ACM, 2005. ISBN 1-59593-135-X pp. 108–117.

[45] L. Hamel, "Visualization of Support Vector Machines with Unsupervised Learning," in *2006 IEEE Symposium on Computational Intelligence and Bioinformatics and Computational Biology*, 2006. doi: 10.1109/CIBCB.2006.330984 pp. 1–8.

[46] J. L. Herlocker, J. A. Konstan, and J. Riedl, "Explaining Collaborative Filtering Recommendations," in *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work (CSCW).* New York, NY, USA: ACM, 2000. doi: 10.1145/358916.358995. ISBN 978-1-58113-222-9 pp. 241–250.

[47] A. Papadimitriou, P. Symeonidis, and Y. Manolopoulos, "A generalized taxonomy of explanations styles for traditional and social recommender systems," *Data Mining and Knowledge Discovery*, vol. 24, no. 3, pp. 555–583, 2012.

[48] Z. C. Lipton, "The Mythos of Model Interpretability," *Commun. ACM*, vol. 61, no. 10, pp. 36–43, 2018.

[49] B. Lepri, N. Oliver, E. Letouzé, A. Pentland, and P. Vinck, “Fair, transparent, and accountable algorithmic decision-making processes,” *Philosophy & Technology*, vol. 31, no. 4, pp. 611–627, 2018. doi: 10.1007/s13347-017-0279-x

[50] A. Adadi and M. Berrada, “Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI),” *IEEE Access*, vol. 6, pp. 52 138–52 160, 2018. doi: 10.1109/ACCESS.2018.2870052

[51] L. M. Cysneiros, M. Raffi, and J. C. S. do Prado Leite, “Software Transparency as a Key Requirement for Self-Driving Cars,” in *2018 IEEE 26th International Requirements Engineering Conference (RE).* IEEE, 2018, pp. 382–387.

[52] Richard Tomsett, Dave Braines, Dan Harborne, Alun D. Preece, and Supriyo Chakraborty, “Interpretable to Whom? A Role-based Model for Analyzing Interpretable Machine Learning Systems,” *CoRR*, vol. abs/1806.07552, 2018.

[53] F. Doshi-Velez and B. Kim, “Towards A Rigorous Science of Interpretable Machine Learning,” 2017.

[54] B. C. Van Fraassen, “The pragmatics of explanation,” *American philosophical quarterly*, vol. 14, no. 2, pp. 143–150, 1977.

[55] D. Sandborg, “Mathematical explanation and the theory of why-questions,” *The British journal for the philosophy of science*, vol. 49, no. 4, pp. 603–624, 1998.

[56] C. G. Hempel and P. Oppenheim, “Studies in the Logic of Explanation,” *Philosophy of science*, vol. 15, no. 2, pp. 135–175, 1948.

[57] W. C. Salmon, *Scientific explanation and the causal structure of the world.* Princeton University Press, 1984.

[58] J. Woodward, *Making things happen: A theory of causal explanation.* Oxford university press, 2005.

[59] R. C. Schank, “Explanation: A first pass,” *Experience, memory, and reasoning*, pp. 139–165, 1986.

[60] P. Achinstein, *Evidence, explanation, and realism: Essays in philosophy of science.* Oxford University Press, 2010.

[61] A. Rosenfeld and A. Richardson, “Explainability in human–agent systems,” *Autonomous Agents and Multi-Agent Systems*, vol. 33, no. 6, pp. 673–705, 2019. doi: 10.1007/s10458-019-09408-y

[62] M. A. Köhl, K. Baum, M. Langer, D. Oster, T. Speith, and D. Bohlender, "Explainability as a Non-Functional Requirement," in *2019 IEEE 27th International Requirements Engineering Conference (RE)*, 2019. doi: 10.1109/RE.2019.00046 pp. 363–368.

[63] P. Ghazi and M. Glinz, "Challenges of working with artifacts in requirements engineering and software engineering," *Requirements Engineering*, vol. 22, no. 3, pp. 359–385, 2017. doi: 10.1007/s00766-017-0272-z. [Online]. Available: https://link.springer.com/article/10.1007/s00766-017-0272-z#citeas

[64] B. Penzenstadler, "Towards a Definition of Sustainability in and for Software Engineering," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, ser. SAC '13. New York, NY, USA: Association for Computing Machinery, 2013. doi: 10.1145/2480362.2480585. ISBN 9781450316569 p. 1183–1185.

[65] D. Wixon and C. Wilson, "Chapter 27 - The Usability Engineering Framework for Product Design and Evaluation," in *Handbook of Human-Computer Interaction (Second Edition)*, 2nd ed., M. G. Helander, T. K. Landauer, and P. V. Prabhu, Eds. Amsterdam: North-Holland, 1997, pp. 653–688. ISBN 978-0-444-81862-1

[66] P. Weichbroth, "Usability of Mobile Applications: A Systematic Literature Study," *IEEE Access*, vol. 8, pp. 55 563–55 577, 2020. doi: 10.1109/ACCESS.2020.2981892

[67] H. Stachowiak, *Allgemeine Modelltheorie*. Springer, 1973.

[68] E. Hull, K. Jackson, and J. Dick, *Introduction*. London: Springer London, 2011, pp. 1–23.

[69] I. Nunes and D. Jannach, "A systematic review and taxonomy of explanations in decision support and recommender systems," *User Modeling and User-Adapted Interaction*, vol. 27, no. 3-5, pp. 393–444, 2017. doi: 10.1007/s11257-017-9195-0

[70] A. Barredo Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, and F. Herrera, "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI," *Information Fusion*, vol. 58, pp. 82–115, 2020. doi: 10.1016/j.inffus.2019.12.012

[71] O. for the Advancement of Structured Information Standards, "Reference Model for Service Oriented Architecture 1.0," OASIS Open, Woburn, MA, USA, Standard, October 2006.

[72] Y. Cherdantseva and J. Hilton, "A Reference Model of Information Assurance amp; Security," in *2013 International Conference on Availability, Reliability and Security*, 2013. doi: 10.1109/ARES.2013.72 pp. 546–555.

[73] M. M. Alani, *OSI Model*. Cham: Springer International Publishing, 2014, pp. 5–17. ISBN 978-3-319-05152-9

[74] K. C. Weber, E. E. R. Araújo, A. R. C. da Rocha, C. A. F. Machado, D. Scalet, and C. F. Salviano, "Brazilian Software Process Reference Model and Assessment Method," in *Computer and Information Sciences - ISCIS 2005*, p. Yolum, T. Güngör, F. Gürgen, and C. Özturan, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. ISBN 978-3-540-32085-2 pp. 402–411.

[75] E. Geisberger, M. B. T. U. M. B. Berenbach, J. Kazmeier, D. Paulish, and A. R. S. C. R. Princeton, "Requirements Engineering Reference Model (REM)," Technische Universität München - Institut für Informatik, Tech. Rep., 2006.

[76] S. Mohseni, N. Zarei, and E. D. Ragan, "A Multidisciplinary Survey and Framework for Design and Evaluation of Explainable AI Systems," *ACM Trans. Interact. Intell. Syst.*, vol. 11, no. 3–4, aug 2021. doi: 10.1145/3387166

[77] T. A. Schoonderwoerd, W. Jorritsma, M. A. Neerincx, and K. van den Bosch, "Human-centered XAI: Developing design patterns for explanations of clinical decision support systems," *International Journal of Human-Computer Studies*, vol. 154, p. 102684, 2021. doi: 10.1016/j.ijhcs.2021.102684

[78] E. Mnkandla, "About software engineering frameworks and methodologies," in *AFRICON 2009*, 2009. doi: 10.1109/AFRCON.2009.5308117 pp. 1–5.

[79] A. Simkute, E. Luger, B. Jones, M. Evans, and R. Jones, "Explainability for experts: A design framework for making algorithms supporting expert decisions more explainable," *Journal of Responsible Technology*, vol. 7-8, p. 100017, 2021. doi: 10.1016/j.jrt.2021.100017

[80] K. Sokol and P. Flach, "One Explanation Does Not Fit All," *KI - Künstliche Intelligenz*, vol. 34, no. 2, pp. 235–250, Jun 2020. doi: 10.1007/s13218-020-00637-y

[81] M. Serrano and M. Serrano, "Ubiquitous, Pervasive and Mobile Computing: A Reusable-Models-based Non-Functional Catalogue," in *Proceedings of Requirements Engineering-Brazil 2013, Rio de Janeiro, Brazil, July 16, 2013*, ser. CEUR Workshop Proceedings,

J. Castro, F. M. R. Alencar, M. Lucena, and G. A. C. Filho, Eds., vol. 1005. CEUR-WS.org, 2013.

[82] R. C. Torres and L. E. G. Martins, "NFR catalogues for RFID middleware," *Journal of Computer Science and Technology*, vol. 14, no. 02, pp. 102–108, 2018.

[83] R. M. Carvalho, R. M. C. Andrade, and K. M. Oliveira, "How developers believe Invisibility impacts NFRs related to User Interaction," in *28th IEEE International Requirements Engineering Conference (RE).* New York, NY, USA: IEEE, 2020. doi: 10.1109/RE48521.2020.00022 pp. 102–112.

[84] J. C. S. do Prado Leite and C. Cappelli, "Software Transparency," *Business & Information Systems Engineering*, vol. 2, no. 3, pp. 127–139, 2010. doi: 10.1007/s12599-010-0102-z

[85] R. M. Carvalho, R. M. C. Andrade, V. Lelli, E. G. Silva, and K. M. de Oliveira, "What About Catalogs of Non-Functional Requirements?" in *Proceedings of REFSQ-2020 Workshops*, vol. 2584. Aachen, DE: CEUR, 2020.

[86] D. Santos, A. Resende, P. A. Junior, and H. Costa, "Attributes and metrics of internal quality that impact the external quality of object-oriented software: A systematic literature review," in *2016 XLII Latin American Computing Conference (CLEI)*, 2016. doi: 10.1109/CLEI.2016.7833322 pp. 1–12.

[87] L. Chazette, W. Brunotte, and T. Speith, "Exploring Explainability: A Definition, a Model, and a Knowledge Catalogue," in *2021 IEEE 29th International Requirements Engineering Conference (RE)*, 2021. doi: 10.1109/RE51729.2021.00025 pp. 197–208.

[88] O. Boruszewski, "Unterstützung der Koetxistenz von agilen und traditionellen Anforderungsartefakten," Ph.D. dissertation, Leibniz Universität Hannover, Fachgebiet Software Engineering, 2016.

[89] "Dimension." [Online]. Available: https://dictionary.cambridge.org/dictionary/english/dimension

[90] S. Freeman and N. Pryce, *Growing Object-Oriented Software, Guided by Tests.* Boston, MA, USA: Addison-Wesley, 2009.

[91] "Spectrum." [Online]. Available: https://dictionary.cambridge.org/dictionary/english/spectrum

[92] S. McConnell, *Code Complete.* Redmond, WA, USA: Microsoft Press, 2004.

[93] M. Glinz and R. J. Wieringa, "Guest Editors' Introduction: Stakeholders in Requirements Engineering," *IEEE Software*, vol. 24, no. 2, pp. 18–20, 2007. doi: 10.1109/MS.2007.42

[94] M. Langer, D. Oster, T. Speith, H. Hermanns, L. Kästner, E. Schmidt, A. Sesing, and K. Baum, "What do we want from Explainable Artificial Intelligence (XAI)? – A stakeholder perspective on XAI and a conceptual model guiding interdisciplinary XAI research," *Artificial Intelligence*, vol. 296, p. 103473, 2021. doi: 10.1016/j.artint.2021.103473

[95] D. Landes and R. Studer, "The treatment of non-functional requirements in MIKE," in *Software Engineering — ESEC '95*, W. Schäfer and P. Botella, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995. ISBN 978-3-540-45552-3 pp. 294–306.

[96] L. M. Cysneiros, J. C. S. do Prado Leite, and J. d. M. S. Neto, "A framework for integrating non-functional requirements into conceptual models," *Requirements Engineering*, vol. 6, no. 2, pp. 97–115, 2001.

[97] L. Chung and do Prado Leite, Julio Cesar Sampaio, *On Non-Functional Requirements in Software Engineering.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 363–379. ISBN 978-3-642-02463-4

[98] D. E. Damian and D. Zowghi, "An insight into the interplay between culture, conflict and distance in globally distributed requirements negotiations," in *36th Annual Hawaii International Conference on System Sciences, 2003. Proceedings of the.* IEEE, 2003, pp. 10–pp.

[99] P. N. Otto and A. I. Antón, "Addressing legal requirements in requirements engineering," in *15th IEEE International Requirements Engineering Conference (RE 2007).* IEEE, 2007, pp. 5–14.

[100] J. min Choe, "The consideration of cultural differences in the design of information systems," *Information & Management*, vol. 41, no. 5, pp. 669–684, 2004.

[101] T.-F. Kummer, J. M. Leimeister, and M. Bick, "On the Importance of National Culture for the Design of Information Systems," *Business & Information Systems Engineering*, vol. 4, no. 6, pp. 317–330, 2012. doi: 10.1007/s12599-012-0236-2

[102] A. Pacey, *The Culture of Technology.* Cambridge, MA, USA: MIT Press, 1983.

[103] "Ethics guidelines for trustworthy AI," https://ec.europa.eu/digital-single-market/en/news/ethics-guidelines-trustworthy-ai, Apr 2019, Accessed 30 November 2019.

[104] "2019 CIGI-Ipsos Global Survey on Internet Security and Trust," https://www.cigionline.org/internet-survey-2019, 2019, Accessed 30 November 2019.

[105] S. Thomsen, "Corporate Values and Corporate Governance," *Corporate Governance: The international journal of business in society*, vol. 4, no. 4, pp. 29–46, Jan 2004. doi: 10.1108/14720700410558862

[106] J. Whittle, "Is Your Software Valueless?" *IEEE Software*, vol. 36, no. 3, pp. 112–115, 2019. doi: 10.1109/MS.2019.2897397

[107] "REGULATION (EU) 2016/679 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 27 April 2016 (General Data Protection Regulation)," Apr 2016. [Online]. Available: https://eur-lex.europa.eu/eli/reg/2016/679/oj

[108] A. Siena, J. Mylopoulos, A. Perini, and A. Susi, "Designing Law-Compliant Software Requirements," in *Conceptual Modeling - ER 2009*, A. H. F. Laender, S. Castano, U. Dayal, F. Casati, and J. P. M. de Oliveira, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. ISBN 978-3-642-04840-1 pp. 472–486.

[109] T. D. Breaux, M. W. Vail, and A. I. Anton, "Towards Regulatory Compliance: Extracting Rights and Obligations to Align Requirements with Regulations," in *14th IEEE International Requirements Engineering Conference (RE'06)*, Sep. 2006. doi: 10.1109/RE.2006.68. ISSN 2332-6441 pp. 49–58.

[110] L. Chung and B. A. Nixon, "Dealing with non-functional requirements: three experimental studies of a process-oriented approach," in *1995 17th International Conference on Software Engineering.* IEEE, 1995, pp. 25–25.

[111] J. Vanwelkenhuysen, "Quality requirements analysis in customer-centered software development," in *Proceedings of the Second International Conference on Requirements Engineering.* IEEE, 1996, pp. 117–124.

[112] C. O'Neil, *Weapons of Math Destruction: How Big Data Produces Inequality and Threatens Democracy.* New York: Penguin Randomhouse Ltd, 2016.

[113] J. P. Carvallo, X. Franch, and C. Quer, "Managing Non-Technical Requirements in COTS Components Selection," in *14th IEEE International Requirements Engineering Conference (RE).* New York, NY, USA: IEEE, 2006. doi: 10.1109/RE.2006.40 pp. 323–326.

[114] D. Ameller, C. Ayala, J. Cabot, and X. Franch, "How do software architects consider non-functional requirements: An exploratory study," in *2012 20th IEEE International*

Requirements Engineering Conference (RE), 2012. doi: 10.1109/RE.2012.6345838 pp. 41–50.

[115] F. Fotrousi, S. A. Fricker, and M. Fiedler, "Quality requirements elicitation based on inquiry of quality-impact relationships," in *2014 IEEE 22nd International Requirements Engineering Conference (RE)*, 2014. doi: 10.1109/RE.2014.6912272 pp. 303–312.

[116] L. Chazette, O. Karras, and K. Schneider, "Do End-Users Want Explanations? Analyzing the Role of Explainability as an Emerging Aspect of Non-Functional Requirements," in *2019 IEEE 27th International Requirements Engineering Conference (RE)*, 2019. doi: 10.1109/RE.2019.00032 pp. 223–233.

[117] B. Abdollahi and O. Nasraoui, *Transparency in Fair Machine Learning: the Case of Explainable Recommender Systems.* Cham: Springer International Publishing, 2018, pp. 21–35. ISBN 978-3-319-90403-0

[118] A. Bunt, M. Lount, and C. Lauzon, "Are explanations always important?: a study of deployed, low-cost intelligent interactive systems," in *Proceedings of the 2012 ACM international conference on Intelligent User Interfaces.* ACM, 2012, pp. 169–178.

[119] T. Kulesza, S. Stumpf, M. Burnett, S. Yang, I. Kwan, and W.-K. Wong, "Too much, too little, or just right? Ways explanations impact end users' mental models," in *2013 IEEE Symposium on Visual Languages and Human Centric Computing.* IEEE, 2013, pp. 3–10.

[120] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering*, 1st ed. Springer Berlin, Heidelberg, 2012. ISBN 978-3-642-29043-5

[121] H. Leßmann, "Durchführung einer Umfrage-Studie zur Nutzung von Code-Reviews in der Praxis," Master's thesis, Leibniz Universität Hannover, Fachgebiet Software Engineering, 2017.

[122] J. Saldaña, *The Coding Manual for Qualitative Researchers*, 2nd ed. Thousand Oaks, CA, USA: SAGE Publications Inc., 2013. ISBN 978-1-44624-737-2

[123] K. Charmaz, *Constructing Grounded Theory: A Practical Guide Through Qualitative Analysis*, 2nd ed. Thousand Oaks, CA, USA: SAGE Publications Inc., 2014. ISBN 978-0-85702-913-3

[124] M. B. Miles and A. M. Huberman, *Qualitative Data Analysis: An Expanded Sourcebook.* Thousand Oaks, CA, USA: SAGE Publications, 1994.

[125] J. Cohen, “Weighted Kappa: Nominal Scale Agreement Provision for Scaled Disagreement or Partial Credit.” *Psychological bulletin*, vol. 70, no. 4, 1968.

[126] J. R. Landis and G. G. Koch, “The Measurement of Observer Agreement for Categorical Data,” *Biometrics*, vol. 33, no. 1, pp. 159–174, 1977. doi: 10.2307/2529310

[127] M. Kuhnke, “Identifzierung von Erklärungsbedarf via User-Feedback-Analyse,” Master’s thesis, Leibniz Universität Hannover, Fachgebiet Software Engineering, Hannover, Germany, July 2020.

[128] R. Hartson and P. Pyla, *The UX Book: Process and Guidelines for Ensuring a Quality User Experience*, 1st ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2012. ISBN 0123852412

[129] M. Frese, “A theory of control and complexity: Implications for software design and integration of computer systems into the work place,” *Psychological issues of human computer interaction in the work place*, pp. 313–337, 1987.

[130] K. L. Norman and B. Shneiderman, *The Psychology of Menu Selection: Designing Cognitive Control at the Human/Computer Interface.* USA: Greenwood Publishing Group Inc., 1991. ISBN 089391553X

[131] E. Aronson, T. D. Wilson, and M. B. Brewer, “Experimentation in social psychology,” *The handbook of social psychology*, vol. 1, pp. 99–142, 1998.

[132] Jakob Nielsen, “Usability 101: Introduction to Usability,” 2012. [Online]. Available: https://www.nngroup.com/articles/usability-101-introduction-to-usability/

[133] J. L. De La Vara, K. Wnuk, R. Berntsson-Svensson, J. Sánchez, and B. Regnell, “An Empirical Study on the Importance of Quality Requirements in Industry,” in *SEKE*, 2011, pp. 438–443.

[134] E. C. Groen, S. Kopczyńska, M. P. Hauer, T. D. Krafft, and J. Doerr, “Users — The Hidden Software Product Quality Experts?: A Study on How App Users Report Quality Aspects in Online Reviews,” in *2017 IEEE 25th International Requirements Engineering Conference (RE)*, 2017. doi: 10.1109/RE.2017.73 pp. 80–89.

[135] C. Flavián, M. Guinalíu, and R. Gurrea, “The role played by perceived usability, satisfaction and consumer trust on website loyalty,” *Information & management*, vol. 43, no. 1, pp. 1–14, 2006.

[136] T. D. Cook, D. T. Campbell, and W. Shadish, *Experimental and quasi-experimental designs for generalized causal inference.* Houghton Mifflin Boston, 2002.

[137] S. Gregor, "The Nature of Theory in Information Systems," *MIS Quarterly*, vol. 30, no. 3, pp. 611–642, 2006. [Online]. Available: http://www.jstor.org/stable/25148742

[138] V. Wilson, "Research methods: triangulation," *Evidence based library and information practice*, vol. 9, no. 1, pp. 74–75, 2014.

[139] L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos, *Non-functional requirements in software engineering*, 1st ed. Boston, MA, USA: Springer New York, NY, 2012. ISBN 978-0-7923-8666-7

[140] B. W. Boehm, J. R. Brown, and M. Lipow, "Quantitative Evaluation of Software Quality," in *Proceedings of the 2nd International Conference on Software Engineering*, ser. ICSE '76. Washington, DC, USA: IEEE Computer Society Press, 1976, p. 592–605.

[141] F. Deissenböck, E. Jürgens, K. Lochmann, and S. Wagner, "Software quality models: Purposes, usage scenarios and requirements," in *2009 ICSE Workshop on Software Quality (WoSQ@ICSE).* New Yor, NY, USA: IEEE, 2009. doi: 10.1109/WOSQ.2009.5071551 pp. 9–14.

[142] R. Gacitúa, L. Ma, B. Nuseibeh, P. Piwek, A. N. D. Roeck, M. Rouncefield, P. Sawyer, A. Willis, and H. Yang, "Making Tacit Requirements Explicit," in *Second International Workshop on Managing Requirements Knowledge (MARK@RE).* New York, NY, USA: IEEE, 2009. doi: 10.1109/MARK.2009.7 pp. 40–44.

[143] J. E. Hannay, D. I. Sjoberg, and T. Dyba, "A Systematic Review of Theory Use in Software Engineering Experiments," *IEEE Transactions on Software Engineering*, vol. 33, no. 2, pp. 87–107, 2007. doi: 10.1109/TSE.2007.12

[144] G. J. De Vreede, R. O. Briggs, and A. P. Massey, "Collaboration engineering: foundations and opportunities: editorial to the special issue on the journal of the association of information systems," *Journal of the Association for Information Systems*, vol. 10, no. 3, p. 7, 2009.

[145] R. L. Q. Portugal, T. Li, L. Silva, E. Almentero, and do Prado Leite, Julio Cesar S., "NFRfinder: A Knowledge Based Strategy for Mining Non-Functional Requirements," in *Proceedings of the XXXII Brazilian Symposium on Software Engineering.* New York, NY, USA: ACM, 2018. doi: 10.1145/3266237.3266269. ISBN 9781450365031 pp. 102–111.

[146] B. Kitchenham and S. Charters, "Guidelines for performing Systematic Literature Reviews in Software Engineering."

[147] 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*, New York, NY, USA, 2014. doi: 10.1145/2601248.2601268 pp. 1–10.

[148] B. G. Buchanan and E. H. Shortliffe, *Rule-based expert systems: the MYCIN experiments of the Stanford Heuristic Programming Project.* Boston, MA, USA: Addison-Wesley, 1984.

[149] D. C. Brock, "Learning from Artificial Intelligence's Previous Awakenings: The History of Expert Systems," *AI Magazine*, vol. 39, no. 3, pp. 3–15, Sep. 2018. doi: 10.1609/aimag.v39i3.2809

[150] J. L. Fleiss, "Measuring nominal scale agreement among many raters," *Psychological Bulletin*, vol. 76, no. 5, pp. 378–382, 1971. doi: 10.1037/h0031619

[151] J. F. Wolfswinkel, E. Furtmueller, and C. P. M. Wilderom, "Using grounded theory as a method for rigorously reviewing literature," *European journal of Information Systems*, vol. 22, no. 1, pp. 45–55, 2013. doi: 10.1057/ejis.2011.51

[152] R. E. Boyatzis, *Transforming Qualitative Information: Thematic Analysis and Code Development.* Thousand Oaks, CA, USA: SAGE Publications, 1998.

[153] D. V. Carvalho, E. M. Pereira, and J. S. Cardoso, "Machine learning interpretability: A survey on methods and metrics," *Electronics*, vol. 8, no. 8, 2019. doi: 10.3390/electronics8080832

[154] W. Pieters, "Explanation and trust: what to tell the user in security and AI?" *Ethics and Information Technology*, vol. 13, no. 1, pp. 53–64, 2011. doi: 10.1007/s10676-010-9253-3

[155] A. Holzinger, G. Langs, H. Denk, K. Zatloukal, and H. Müller, "Causability and explainability of artificial intelligence in medicine," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 9, no. 4, pp. 1–13, 2019. doi: 10.1002/widm.1312

[156] J. Hois, D. Theofanou-Fuelbier, and A. J. Junk, "How to Achieve Explainability and Transparency in Human AI Interaction," in *International Conference on Human-Computer Interaction (HCI).* Cham, CH: Springer International Publishing, 2019. doi: 10.1007/978-3-030-23528-4_25. ISBN 978-3-030-23528-4 pp. 177–183.

[157] A. Glass, D. L. McGuinness, and M. Wolverton, "Toward establishing trust in adaptive agents," in *Proceedings of the 13th International Conference on Intelligent User Interfaces (IUI).* New York, NY, USA: ACM, 2008. doi: 10.1145/1378773.1378804 pp. 227–236.

[158] Q. Vera Liao, Daniel M. Gruen, and Sarah Miller, "Questioning the AI: Informing Design Practices for Explainable AI User Experiences," in *Proceedings of the 2020 Conference on Human Factors in Computing Systems (CHI).* New York, NY, USA: ACM, 2020. doi: 10.1145/3313831.3376590 pp. 1–15.

[159] P. Dourish, "What we talk about when we talk about context," *Personal and Ubiquitous Computing*, vol. 8, no. 1, pp. 19–30, 2004. doi: 10.1007/s00779-003-0253-8

[160] T. Miller, "Explanation in Artificial Intelligence: Insights from the Social Sciences," *Artificial Intelligence*, vol. 267, pp. 1–38, 2019. doi: 10.1016/j.artint.2018.07.007

[161] C. Henin and M. Le Daniel, "Towards a Generic Framework for Black-Box Explanation Methods," in *Proceedings of the IJCAI Workshop on Explainable Artificial Intelligence (XAI)*, 2019, pp. 28–34.

[162] L. Chen, D. Yan, and F. Wang, "User Evaluations on Sentiment-Based Recommendation Explanations," *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 9, no. 4, pp. 1–38, 2019. doi: 10.1145/3282878

[163] A. D. Preece, D. Harborne, D. Braines, R. Tomsett, and S. Chakraborty, "Stakeholders in Explainable AI," *CoRR*, vol. abs/1810.00184, 2018.

[164] A. Weller, "Transparency: Motivations and Challenges," in *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, W. Samek, G. Montavon, A. Vedaldi, L. K. Hansen, and K.-R. Müller, Eds. Cham, CH: Springer International Publishing, 2019, ch. 2, pp. 23–40. ISBN 978-3-030-28954-6

[165] M. Hind, D. Wei, M. Campbell, N. C. F. Codella, A. Dhurandhar, A. Mojsilović, K. Natesan Ramamurthy, and K. R. Varshney, "TED: Teaching AI to Explain Its Decisions," in *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society.* New York, NY, USA: ACM, 2019. doi: 10.1145/3306618.3314273. ISBN 9781450363242 pp. 123–129.

[166] M. O. Riedl, "Human-centered artificial intelligence and machine learning," *Human Behavior and Emerging Technologies*, vol. 1, no. 1, pp. 33–36, 2019. doi: 10.1002/hbe2.117

[167] J. McInerney, B. Lacker, S. Hansen, K. Higley, H. Bouchard, A. Gruson, and R. Mehrotra, "Explore, exploit, and explain: personalizing explainable recommendations with

bandits," in *Proceedings of the 12th ACM Conference on Recommender Systems (RecSys)*. New York, NY, USA: ACM, 2018. doi: 10.1145/3240323.3240354 pp. 31–39.

[168] C. J. Cai, J. Jongejan, and J. Holbrook, "The effects of example-based explanations in a machine learning interface," in *Proceedings of the 24th International Conference on Intelligent User Interfaces (IUI)*. New York, NY, USA: ACM, 2019. doi: 10.1145/3301275.3302289 pp. 258–262.

[169] Zanker, Markus, "The influence of knowledgeable explanations on users' perception of a recommender system," in *Proceedings of the sixth ACM conference on Recommender systems (RecSys)*. New York, NY, USA: ACM, 2012. doi: 10.1145/2365952.2366011 pp. 269–272.

[170] P. Pu and L. Chen, "Trust building with explanation interfaces," in *Proceedings of the 11th International Conference on Intelligent User Interfaces (IUI)*. New York, NY, USA: ACM, 2006. doi: 10.1145/1111449.1111475 pp. 93–100.

[171] R. F. Kizilcec, "How Much Information? Effects of Transparency on Trust in an Algorithmic Interface," in *Proceedings of the 2016 Conference on Human Factors in Computing Systems (CHI)*. New York, NY, USA: ACM, 2016. doi: 10.1145/2858036.2858402. ISBN 978-1-4503-3362-7 pp. 2390–2395.

[172] H. Cramer, V. Evers, S. Ramlal, V. S. Maarten, L. Rutledge, N. Stash, L. Aroyo, and B. Wielinga, "The effects of transparency on trust in and acceptance of a content-based art recommender," *User Modeling and User-adapted interaction*, vol. 18, no. 5, p. 455, 2008. doi: 10.1007/s11257-008-9051-3

[173] N. Tintarev and J. Masthoff, "Effective explanations of recommendations: user-centered design," in *Proceedings of the 2007 ACM Conference on Recommender Systems (RecSys)*. New York, NY, USA: ACM, 2007. doi: 10.1145/1297231.1297259 pp. 153–156.

[174] C. Tsai and P. Brusilovsky, "Explaining recommendations in an interactive hybrid social recommender," in *Proceedings of the 24th International Conference on Intelligent User Interfaces (IUI)*. New York, NY, USA: ACM, 2019. doi: 10.1145/3301275.3302318 pp. 391–396.

[175] N. Tintarev and J. Masthoff, "Evaluating the effectiveness of explanations for recommender systems," *User Modeling and User-Adapted Interaction*, vol. 22, no. 4-5, pp. 399–439, 2012. doi: 10.1007/s11257-011-9117-5

[176] K. Darlington, "Aspects of Intelligent Systems Explanation," *Universal Journal of Control and Automation*, vol. 1, no. 2, pp. 40–51, 2013. doi: 10.13189/ujca.2013.010204

[177] P. S. Kumar, M. Saravanan, and S. Suresh, "Explainable Classification Using Clustering in Deep Learning Models," in *Proceedings of the IJCAI Workshop on Explainable Artificial Intelligence (XAI)*, 2019, pp. 115–121.

[178] J. Dodge, Q. V. Liao, Y. Zhang, R. K. E. Bellamy, and C. Dugan, "Explaining models: an empirical study of how explanations impact fairness judgment," in *Proceedings of the 24th International Conference on Intelligent User Interfaces (IUI).* New York, NY, USA: ACM, 2019. doi: 10.1145/3301275.3302310 pp. 275–285.

[179] V. Putnam and C. Conati, "Exploring the Need for Explainable Artificial Intelligence (XAI) in Intelligent Tutoring Systems (ITS)," in *Joint Proceedings of the ACM IUI 2019 Workshops.* Aachen, DE: CEUR, 2019.

[180] J. Schneider and J. Handali, "Personalized explanation in machine learning: A conceptualization," *Proceedings of the 27th European Conference on Information Systems (ECIS)*, 2019.

[181] L. H. Gilpin, C. Testart, N. Fruchter, and J. Adebayo, "Explaining Explanations to Society," in *NIPS Workshop on Ethical, Social and Governance Issues in AI*, 2018, pp. 1–6.

[182] I. Monteath and R. Sheh, "Assisted and incremental medical diagnosis using explainable artificial intelligence," in *Proceedings of the IJCAI/ECAI Workshop on Explainable Artificial Intelligence (XAI)*, 2018, pp. 104–108.

[183] R. Binns, M. Van Kleek, M. Veale, U. Lyngs, J. Zhao, and N. Shadbolt, "'It's Reducing a Human Being to a Percentage': Perceptions of Justice in Algorithmic Decisions," in *Proceedings of the 2018 Conference on Human Factors in Computing Systems (CHI).* New York, NY, USA: ACM, 2018. doi: 10.1145/3173574.3173951 pp. 1–14.

[184] K. McCarthy, J. Reilly, L. McGinty, and B. Smyth, "Thinking positively-explanatory feedback for conversational recommender systems," in *Proceedings of the European Conference on Case-Based Reasoning (ECCBR) Explanation Workshop*, 2004, pp. 115–124.

[185] I. Lage, D. Lifschitz, F. Doshi-Velez, and O. Amir, "Exploring Computational User Models for Agent Policy Summarization," in *Proceedings of the IJCAI Workshop on Explainable Artificial Intelligence (XAI)*, 2019, pp. 59–65.

[186] R. Borgo, M. Cashmore, and D. Magazzeni, “Towards providing explanations for AI planner decisions,” *Proceedings of the IJCAI/ECAI Workshop on Explainable Artificial Intelligence (XAI)*, pp. 11–17, 2018.

[187] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, “A Survey of Methods for Explaining Black Box Models,” *ACM Computing Survey*, vol. 51, no. 5, pp. 1–42, 2019. doi: 10.1145/3236009

[188] F. Hohman, A. Head, R. Caruana, R. DeLine, and S. M. Drucker, “Gamut: A Design Probe to Understand How Data Scientists Understand Machine Learning Models,” in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems.* New York, NY, USA: ACM, 2019. doi: 10.1145/3290605.3300809. ISBN 9781450359702 pp. 1–13.

[189] S. M. Mathews, “Explainable Artificial Intelligence Applications in NLP, Biomedical, and Malware Classification: A Literature Review,” in *Intelligent Computing – Proceedings of the Computing Conference.* Cham, CH: Springer International Publishing, 2019. doi: 10.1007/978-3-030-22868-2. ISBN 978-3-030-22868-2 pp. 1269–1292.

[190] J. Chen, F. Lécué, J. Z. Pan, I. Horrocks, and H. Chen, “Knowledge-Based Transfer Learning Explanation,” in *Proceedings of the Sixteenth International Conference for Principles of Knowledge Representation and Reasoning (KR).* Palo Alto, CA, USA: AAAI, 2018, pp. 349–358.

[191] Ka-Ping Yee, “Aligning security and usability,” *IEEE Security Privacy*, vol. 2, no. 5, pp. 48–55, 2004.

[192] P. Gutmann and I. Grigg, “Security Usability,” *IEEE security & privacy*, vol. 3, no. 4, pp. 56–58, 2005. doi: 10.1109/MSP.2005.104

[193] L. F. Cranor and S. Garfinkel, “Guest Editors’ Introduction: Secure or Usable?” *IEEE Security Privacy*, vol. 2, no. 5, pp. 16–18, 2004.

[194] R. M. Carvalho, “Dealing with Conflicts Between Non-functional Requirements of UbiComp and IoT Applications,” in *2017 IEEE 25th International Requirements Engineering Conference (RE)*, 2017. doi: 10.1109/RE.2017.51 pp. 544–549.

[195] “Information System.” [Online]. Available: https://www.britannica.com/topic/information-system

[196] H.-F. Cheng, R. Wang, Z. Zhang, F. O’Connell, T. Gray, F. M. Harper, and H. Zhu, “Explaining decision-making algorithms through UI: Strategies to help non-expert

stakeholders," in *Proceedings of the 2019 chi conference on human factors in computing systems.* New York, NY, USA: Association for Computing Machinery, 2019. doi: 10.1145/3290605.3300789. ISBN 978-1-4503-5970-2 pp. 1–12.

[197] G. Wiegand, M. Eiband, M. Haubelt, and H. Hussmann, ""I'd like an Explanation for That!"Exploring Reactions to Unexpected Autonomous Driving," in *22nd International Conference on Human-Computer Interaction with Mobile Devices and Services*, ser. MobileHCI '20. New York, NY, USA: Association for Computing Machinery, 2020. doi: 10.1145/3379503.3403554. ISBN 9781450375160

[198] L. Chazette, J. Klünder, M. Balci, and K. Schneider, "How Can We Develop Explainable Systems? Insights from a Literature Review and an Interview Study," in *Proceedings of the International Conference on Software and System Processes and International Conference on Global Software Engineering*, ser. ICSSP'22. New York, NY, USA: Association for Computing Machinery, 2022. doi: 10.1145/3529320.3529321. ISBN 9781450396745 p. 1–12.

[199] M. Balci, "Entwurf eines Requirements Engineering Workflows für erklärbare Systeme," Master's thesis, Leibniz Universität Hannover, Fachgebiet Software Engineering, Hannover, Germany, November 2021.

[200] M. J. Grant and A. Booth, "A typology of reviews: an analysis of 14 review types and associated methodologies," *Health Information & Libraries Journal*, vol. 26, no. 2, pp. 91–108, 2009. doi: 10.1111/j.1471-1842.2009.00848.x

[201] A. C. Tricco, J. Antony, W. Zarin, L. Strifler, M. Ghassemi, J. Ivory, L. Perrier, B. Hutton, D. Moher, and S. E. Straus, "A scoping review of rapid review methods," *BMC Medicine*, vol. 13, no. 1, p. 224, 2015. doi: 10.1186/s12916-015-0465-6

[202] B. Kitchenham, O. Pearl Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic literature reviews in software engineering – A systematic literature review," *Information and Software Technology*, vol. 51, no. 1, pp. 7–15, 2009. doi: 10.1016/j.infsof.2008.09.009 Special Section - Most Cited Articles in 2002 and Regular Research Papers.

[203] P. Mayring, "Qualitative content analysis: Demarcation, varieties, developments," in *Forum: Qualitative Social Research*, vol. 20, no. 3. Freie Universität Berlin, 2019, pp. 1–26.

[204] C. Rolland, "A comprehensive view of process engineering," in *International Conference on Advanced Information Systems Engineering.* Springer, 1998, pp. 1–24.

[205] I. F. Alexander and L. Beus-Dukic, *Discovering Requirements: How to Specify Products and Services.* West Sussex, England: John Wiley and Sons, 2009. ISBN 978-0-470-71240-5

[206] J. Gulliksen, B. Göransson, I. Boivie, S. Blomkvist, J. Persson, and Å. Cajander, "Key principles for user-centred systems design," *Behaviour and Information Technology*, vol. 22, no. 6, pp. 397–409, 2003.

[207] P. Brusilovsky and C. H. Tsai, "Designing Explanation Interfaces for Transparency and Beyond," in *CEUR Workshop Proceedings*, vol. 2327. CEUR-WS, 2019.

[208] M. Eiband, H. Schneider, M. Bilandzic, J. Fazekas-Con, M. Haug, and H. Hussmann, "Bringing Transparency Design into Practice," in *23rd International Conference on Intelligent User Interfaces*, ser. IUI '18. New York, NY, USA: Association for Computing Machinery, 2018. doi: 10.1145/3172944.3172961. ISBN 9781450349451 p. 211–223.

[209] L. Longo, R. Goebel, F. Lecue, P. Kieseberg, and A. Holzinger, "Explainable artificial intelligence: Concepts, applications, research challenges and visions," in *International Cross-Domain Conference for Machine Learning and Knowledge Extraction.* Springer, 2020, pp. 1–16.

[210] D. A. Melis, H. Kaur, H. Daumé III, H. Wallach, and J. W. Vaughan, "From Human Explanation to Model Interpretability: A Framework Based on Weight of Evidence," in *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, vol. 9, 2021, pp. 35–47.

[211] L. Kopitar, L. Cilar, P. Kocbek, and G. Stiglic, "Local vs. Global Interpretability of Machine Learning Models in Type 2 Diabetes Mellitus Screening," in *Artificial Intelligence in Medicine: Knowledge Representation and Transparent and Explainable Systems*, M. Marcos, J. M. Juarez, R. Lenz, G. J. Nalepa, S. Nowaczyk, M. Peleg, J. Stefanowski, and G. Stiglic, Eds. Cham: Springer International Publishing, 2019. ISBN 978-3-030-37446-4 pp. 108–119.

[212] P. Berander, L.-O. Damm, J. Eriksson, T. Gorschek, K. Henningsson, P. Jönsson, S. Kågström, D. Milicic, F. Mårtensson, K. Rönkkö *et al.*, "Software quality attributes and trade-offs," *Blekinge Institute of Technology*, 2005.

[213] B. Boehm and H. In, "Identifying quality-requirement conflicts," *IEEE Software*, vol. 13, no. 2, pp. 25–35, 1996. doi: 10.1109/52.506460

[214] P. N. Johnson-Laird, *Mental models: Towards a cognitive science of language, inference, and consciousness.* Harvard University Press, 1983, no. 6.

[215] D. E. Kieras and S. Bovair, "The role of a mental model in learning to operate a device," *Cognitive Science*, vol. 8, no. 3, pp. 255–273, 1984. doi: 10.1016/S0364-0213(84)80003-8

[216] D. A. Norman, "Some observations on mental models," in *Mental models.* Psychology Press, 2014, pp. 15–22.

[217] I. Young, *Mental Models: Aligning Design Strategy with Human Behavior*, 1st ed. Brooklyn, New York: Rosenfeld Media, 2008. ISBN 978-1-933820-06-4

[218] D. Mishra, A. Mishra, and A. Yazici, "Successful requirement elicitation by combining requirement engineering techniques," in *2008 First International Conference on the Applications of Digital Information and Web Technologies (ICADIWT).* IEEE, 2008, pp. 258–263.

[219] C.-H. Tsai, Y. You, X. Gui, Y. Kou, and J. M. Carroll, "Exploring and Promoting Diagnostic Transparency and Explainability in Online Symptom Checkers," in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, ser. CHI '21. New York, NY, USA: Association for Computing Machinery, 2021. doi: 10.1145/3411764.3445101. ISBN 9781450380966

[220] D. J. Mayhew and D. Mayhew, *The usability engineering lifecycle: a practitioner's handbook for user interface design.* Morgan Kaufmann, 1999.

[221] K. Martin, A. Liret, N. Wiratunga, G. Owusu, and M. Kern, "Evaluating Explainability Methods Intended for Multiple Stakeholders," *KI - Künstliche Intelligenz*, vol. 35, no. 3, pp. 397–411, Nov 2021. doi: 10.1007/s13218-020-00702-6

[222] A. Seffah, J. Gulliksen, and M. C. Desmarais, "An introduction to human-centered software engineering," in *Human-Centered Software Engineering—Integrating Usability in the Software Development Lifecycle.* Springer, 2005, pp. 3–14.

[223] J. Hehn and F. Uebernickel, "The Use of Design Thinking for Requirements Engineering: An Ongoing Case Study in the Field of Innovative Software-Intensive Systems," in *2018 IEEE 26th International Requirements Engineering Conference (RE).* IEEE, 2018, pp. 400–405.

[224] M. Naiseh, D. Al-Thani, N. Jiang, and R. Ali, "Explainable recommendation: when design meets trust calibration," *World Wide Web*, vol. 24, no. 5, pp. 1857–1884, 2021.

[225] D. Wang, Q. Yang, A. Abdul, and B. Y. Lim, "Designing Theory-Driven User-Centric Explainable AI," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, ser. CHI '19. New York, NY, USA: Association for Computing Machinery, 2019. doi: 10.1145/3290605.3300831. ISBN 9781450359702 p. 1–15.

[226] S. T. Mueller, R. R. Hoffman, W. Clancey, A. Emrey, and G. Klein, "Explanation in Human-AI Systems: A Literature Meta-Review, Synopsis of Key Ideas and Publications, and Bibliography for Explainable AI," 2019.

[227] P. Tell, J. Klünder, S. Küpper, D. Raffo, S. G. MacDonell, J. Münch, D. Pfahl, O. Linssen, and M. Kuhrmann, "What are Hybrid Development Methods Made Of? An Evidence-Based Characterization," in *2019 IEEE/ACM International Conference on Software and System Processes (ICSSP)*, 2019. doi: 10.1109/ICSSP.2019.00022 pp. 105–114.

[228] J. Klünder, D. Karajic, P. Tell, O. Karras, C. Münkel, J. Münch, S. G. MacDonell, R. Hebig, and M. Kuhrmann, "Determining context factors for hybrid development methods with trained models," in *Proceedings of the International Conference on Software and System Processes*, 2020, pp. 61–70.

[229] J. Klünder, R. Hebig, P. Tell, M. Kuhrmann, J. Nakatumba-Nabende, R. Heldal, S. Krusche, M. Fazal-Baqaie, M. Felderer, M. F. G. Bocco *et al.*, "Catching up with method and process practice: An industry-informed baseline for researchers," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 2019, pp. 255–264.

[230] L. Chazette, V. Klös, F. Herzog, and K. Schneider, "Requirements on Explanations: A Quality Framework for Explainability," in *2022 IEEE 30th International Requirements Engineering Conference (RE)*, 2022.

[231] F. Herzog, "Konzeption und Evaluation eines Modells zur Unterstützung des Designs von Erklärungen in erklärbaren Systemen," Master's thesis, Leibniz Universität Hannover, Fachgebiet Software Engineering, Hannover, Germany, October 2021.

[232] Stefan Wagner, Andreas Goeb, Lars Heinemann, Michael Kläs, Constanza Lampasona, Klaus Lochmann, Alois Mayr, Reinhold Plösch, Andreas Seidl, Jonathan Streit, and Adam Trendowicz, "Operationalised product quality models and assessment: The Quamoco approach," *Information and Software Technology*, vol. 62, p. 101..123, 2015. doi: 10.1016/j.infsof.2015.02.009

[233] M. Ribera and A. Lapedriza, "Can we do better explanations? A proposal of user-centered explainable AI." in *Joint Proceedings of the ACM Workshops on Intelligent User Interfaces (IUI)*, 2019.

[234] S. Chari, O. Seneviratne, D. M. Gruen, M. A. Foreman, A. K. Das, and D. L. McGuinness, "Explanation Ontology: A Model of Explanations for User-Centered AI," in *The Semantic Web – ISWC 2020*, ser. Lecture Notes in Computer Science, J. Z. Pan, V. Tamma, C. d'Amato, K. Janowicz, B. Fu, A. Polleres, O. Seneviratne, and L. Kagal, Eds. Cham: Springer International Publishing, 2020. doi: 10.1007/978-3-030-62466-8_15. ISBN 978-3-030-62466-8 pp. 228–243.

[235] P. Kouki, J. Schaffer, J. Pujara, J. O'Donovan, and L. Getoor, "User Preferences for Hybrid Explanations," in *Proceedings of the Eleventh ACM Conference on Recommender Systems*, ser. RecSys '17. New York, NY, USA: Association for Computing Machinery, 2017. doi: 10.1145/3109859.3109915. ISBN 978-1-4503-4652-8 pp. 84–88, event-place: Como, Italy.

[236] J. van der Waa, E. Nieuwburg, A. Cremers, and M. Neerincx, "Evaluating XAI: A comparison of rule-based and example-based explanations," *Artificial Intelligence*, vol. 291, p. 103404, 2021. doi: 10.1016/j.artint.2020.103404

[237] D. Samadhiya, S.-H. Wang, and D. Chen, "Quality models: Role and value in software engineering," in *2010 2nd International Conference on Software Technology and Engineering*, vol. 1. IEEE, 2010, pp. V1–320.

[238] K. Balog and F. Radlinski, "Measuring Recommendation Explanation Quality: The Conflicting Goals of Explanations," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '20. New York, NY, USA: Association for Computing Machinery, 2020. doi: 10.1145/3397271.3401032. ISBN 978-1-4503-8016-4 pp. 329–338, event-place: Virtual Event, China.

[239] N. Tintarev and J. Masthoff, "Designing and evaluating explanations for recommender systems," in *Recommender systems handbook*. Springer, 2011, pp. 479–510.

[240] M. Zolotas and Y. Demiris, "Towards Explainable Shared Control using Augmented Reality," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019. doi: 10.1109/IROS40897.2019.8968117 pp. 3020–3026, iSSN: 2153-0866.

[241] K. Rjoob, R. Bond, D. Finlay, V. McGilligan, S. J. Leslie, A. Rababah, A. Iftikhar, D. Guldenring, C. Knoery, A. McShane, and A. Peace, "Towards Explainable Artificial Intelligence and Explanation User Interfaces to Open the 'Black Box' of Automated ECG

Interpretation," in *Advanced Visual Interfaces. Supporting Artificial Intelligence and Big Data Applications*, T. Reis, M. X. Bornschlegl, M. Angelini, and M. L. Hemmje, Eds. Cham: Springer International Publishing, 2021. ISBN 978-3-030-68007-7 pp. 96–108.

[242] A. Abdulrahman, D. Richards, H. Ranjbartabar, and S. Mascarenhas, "Belief-Based Agent Explanations to Encourage Behaviour Change," in *Proceedings of the 19th ACM International Conference on Intelligent Virtual Agents*, ser. IVA '19. New York, NY, USA: Association for Computing Machinery, 2019. doi: 10.1145/3308532.3329444. ISBN 978-1-4503-6672-4 pp. 176–178, event-place: Paris, France.

[243] M. Sato, S. Kawai, and H. Nobuhara, "Action-Triggering Recommenders: Uplift Optimization and Persuasive Explanation," in *2019 International Conference on Data Mining Workshops (ICDMW)*, 2019. doi: 10.1109/ICDMW.2019.00155 pp. 1060–1069, iSSN: 2375-9259.

[244] W. M. Gentleman, "If software quality is a perception, how do we measure it?" in *Quality of Numerical Software.* Springer, 1997, pp. 32–43.

[245] "Perception," accessed 17 February 2022. [Online]. Available: https://dictionary.cambridge.org/dictionary/english/perception

[246] U. Ehsan and M. O. Riedl, "Human-Centered Explainable AI: Towards a Reflective Sociotechnical Approach," in *HCI International 2020 - Late Breaking Papers: Multimodality and Intelligence*, ser. Lecture Notes in Computer Science, C. Stephanidis, M. Kurosu, H. Degen, and L. Reinerman-Jones, Eds. Cham: Springer International Publishing, 2020. doi: 10.1007/978-3-030-60117-1_33. ISBN 978-3-030-60117-1 pp. 449–466.

[247] Kim Salazar, "Contextual Inquiry: Inspire Design by Observing and Interviewing Users in Their Context," 2020, online; Accessed 17 February 2022. [Online]. Available: https://www.nngroup.com/articles/contextual-inquiry/

[248] T. Yamada, T. Sato, T. Tomiyama, and N. Ueki, "Evaluating Explanation Function in Railway Crew Rescheduling System by Think-Aloud Test," in *2016 5th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI).* Kumamoto, Japan: IEEE, 2016. doi: 10.1109/IIAI-AAI.2016.93. ISBN 978-1-4673-8985-3 pp. 991–994.

[249] D. Gunning and D. Aha, "DARPA's Explainable Artificial Intelligence (XAI) Program," *AI Magazine*, vol. 40, no. 2, pp. 44–58, Jun. 2019. doi: 10.1609/aimag.v40i2.2850

[250] D. Cirqueira, D. Nedbal, M. Helfert, and M. Bezbradica, "Scenario-Based Requirements Elicitation for User-Centric Explainable AI," in *Machine Learning and Knowledge*

Extraction, ser. Lecture Notes in Computer Science, A. Holzinger, P. Kieseberg, A. M. Tjoa, and E. Weippl, Eds. Cham: Springer International Publishing, 2020. doi: 10.1007/978-3-030-57321-8_18. ISBN 978-3-030-57321-8 pp. 321–341.

[251] G. Wiegand, M. Schmidmaier, T. Weber, Y. Liu, and H. Hussmann, "I drive-you trust: Explaining driving behavior of autonomous cars," in *Extended abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*, 2019. doi: 10.1145/3290607.3312817 pp. 1–6.

[252] R. Hari and L. Parkkonen, "The brain timewise: how timing shapes and supports brain function," *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, vol. 370, no. 1668, 2015. doi: 10.1098/rstb.2014.0170

[253] Z. Zahedi, A. Olmo, T. Chakraborti, S. Sreedharan, and S. Kambhampati, "Towards Understanding User Preferences for Explanation Types in Model Reconciliation," in *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI).* Daegu, Korea (South): IEEE, 2019. doi: 10.1109/HRI.2019.8673097. ISBN 978-1-5386-8555-6 pp. 648–649.

[254] L. Zhu and T. Williams, "Effects of Proactive Explanations by Robots on Human-Robot Trust," in *Social Robotics*, A. R. Wagner, D. Feil-Seifer, K. S. Haring, S. Rossi, T. Williams, H. He, and S. Sam Ge, Eds. Cham: Springer International Publishing, 2020. ISBN 978-3-030-62056-1 pp. 85–95.

[255] M. A. Neerincx, J. van der Waa, F. Kaptein, and J. van Diggelen, "Using Perceptual and Cognitive Explanations for Enhanced Human-Agent Team Performance," in *Engineering Psychology and Cognitive Ergonomics*, D. Harris, Ed. Cham: Springer International Publishing, 2018. ISBN 978-3-319-91122-9 pp. 204–214.

[256] D. C. Hernandez-Bocanegra, T. Donkers, and J. Ziegler, "Effects of Argumentative Explanation Types on the Perception of Review-Based Recommendations," in *Adjunct Publication of the 28th ACM Conference on User Modeling, Adaptation and Personalization*, ser. UMAP '20 Adjunct. New York, NY, USA: Association for Computing Machinery, 2020. doi: 10.1145/3386392.3399302. ISBN 978-1-4503-7950-2 pp. 219–225, event-place: Genoa, Italy.

[257] R. Thomson and J. R. Schoenherr, "Knowledge-to-Information Translation Training (KITT): An Adaptive Approach to Explainable Artificial Intelligence," in *Adaptive Instructional Systems*, ser. Lecture Notes in Computer Science, R. A. Sottilare and

J. Schwarz, Eds. Cham: Springer International Publishing, 2020. doi: 10.1007/978-3-030-50788-6_14. ISBN 978-3-030-50788-6 pp. 187–204.

[258] F. Kaptein, J. Broekens, K. Hindriks, and M. Neerincx, "Personalised self-explanation by robots: The role of goals versus beliefs in robot-action explanation for children and adults," in *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2017. doi: 10.1109/ROMAN.2017.8172376 pp. 676–682, iSSN: 1944-9437.

[259] J. Cassens and R. Wegener, "Ambient Explanations: Ambient Intelligence and Explainable AI," in *Ambient Intelligence*, ser. Lecture Notes in Computer Science, I. Chatzigiannakis, B. De Ruyter, and I. Mavrommati, Eds. Cham: Springer International Publishing, 2019. doi: 10.1007/978-3-030-34255-5_30. ISBN 978-3-030-34255-5 pp. 370–376.

[260] "Medium," accessed 17 February 2022. [Online]. Available: https://dictionary.cambridge.org/dictionary/english/medium

[261] "Tone," accessed 17 February 2022. [Online]. Available: https://dictionary.cambridge.org/dictionary/english/tone

[262] M. Eiband, D. Buschek, A. Kremer, and H. Hussmann, "The Impact of Placebic Explanations on Trust in Intelligent Systems," in *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*, ser. CHI EA '19. New York, NY, USA: Association for Computing Machinery, 2019. doi: 10.1145/3290607.3312787. ISBN 978-1-4503-5971-9 pp. 1–6, event-place: Glasgow, Scotland Uk.

[263] J. Kunkel, T. Donkers, L. Michael, C.-M. Barbu, and J. Ziegler, "Let Me Explain: Impact of Personal and Impersonal Explanations on Trust in Recommender Systems," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, ser. CHI '19. New York, NY, USA: Association for Computing Machinery, 2019. doi: 10.1145/3290605.3300717. ISBN 978-1-4503-5970-2 pp. 1–12, event-place: Glasgow, Scotland Uk.

[264] A. Fink, *The Survey Handbook*, 2nd ed. Thousand Oaks, California: SAGE Publications, Inc., Sep 2003. ISBN 0-7619-2510-4

[265] H. Mucha, S. Robert, R. Breitschwerdt, and M. Fellmann, "Interfaces for Explanations in Human-AI Interaction: Proposing a Design Evaluation Approach," in *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, ser. CHI EA '21. New York, NY, USA: Association for Computing Machinery, 2021. doi: 10.1145/3411763.3451759. ISBN 978-1-4503-8095-9 Event-place: Yokohama, Japan.

[266] M. Bilgic and R. J. Mooney, “Explaining recommendations: Satisfaction vs. promotion,” in *Beyond personalization workshop, IUI*, vol. 5, 2005, p. 153.

[267] G. Vilone and L. Longo, “Notions of Explainability and Evaluation Approaches for Explainable Artificial Intelligence,” *Information Fusion*, vol. 76, pp. 89–106, 2021. doi: 10.1016/j.inffus.2021.05.009

[268] N. Wang, D. V. Pynadath, E. Rovira, M. J. Barnes, and S. G. Hill, “Is It My Looks? Or Something I Said? The Impact of Explanations, Embodiment, and Expectations on Trust and Performance in Human-Robot Teams,” in *Persuasive Technology*, J. Ham, E. Karapanos, P. P. Morita, and C. M. Burns, Eds. Cham: Springer International Publishing, 2018. ISBN 978-3-319-78978-1 pp. 56–69.

[269] S. Jantunen, “Exploring software engineering practices in small and medium-sized organizations,” in *Proceedings of the 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering*, 2010, pp. 96–101.

[270] P. Runeson and M. Höst, “Guidelines for conducting and reporting case study research in software engineering,” *Empirical Software Engineering*, vol. 14, no. 2, pp. 131–164, 2009. doi: 10.1007/s10664-008-9102-8

[271] C. Rupp *et al.*, *Requirements-Engineering und-Management: Aus der Praxis von klassisch bis agil.* Carl Hanser Verlag GmbH Co KG, 2014.

[272] N. Balasubramaniam, M. Kauppinen, K. Hiekkanen, and S. Kujala, “Transparency and Explainability of AI Systems: Ethical Guidelines in Practice,” in *Requirements Engineering: Foundation for Software Quality*, V. Gervasi and A. Vogelsang, Eds. Cham: Springer International Publishing, 2022. ISBN 978-3-030-98464-9 pp. 3–18.

[273] J. Robertson and S. Robertson, “Volere Requirements Specification Template,” *Requirements Specification Templates*, 2000.

[274] L. Chazette, W. Brunotte, and T. Speith, “Supplementary Material for Journal Article ”Explainability for Software Systems - From Abstract to Actionable”,” Jan. 2022. [Online]. Available: https://figshare.com/s/f73d41c41345dd08cf39

[275] Jakob Nielsen, “10 Usability Heuristics for User Interface Design,” 1994, online; Accessed 17 February 2022. [Online]. Available: https://www.nngroup.com/articles/ten-usability-heuristics/

[276] D. Firesmith, “Using quality models to engineer quality requirements,” *Journal of Object Technology*, vol. 2, no. 5, pp. 67–75, 2003.

[277] J. Verner, K. Cox, S. Bleistein, and N. Cerpa, “Requirements Engineering and Software Project Success: an industrial survey in Australia and the U.S,” *Australasian Journal of Information Systems*, vol. 13, no. 1, Nov. 2005. doi: 10.3127/ajis.v13i1.73. [Online]. Available: https://journal.acs.org.au/index.php/ajis/article/view/73

[278] A. Vogelsang and M. Borg, “Requirements engineering for machine learning: Perspectives from data scientists,” in *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW).* IEEE, 2019, pp. 245–251.

[279] M. Sadeghi, V. Klös, and A. Vogelsang, “Cases for explainable software systems: characteristics and examples,” in *2021 IEEE 29th International Requirements Engineering Conference Workshops (REW).* IEEE, 2021, pp. 181–187.

[280] W. Brunotte, L. Chazette, V. Klös, and T. Speith, “Quo Vadis, Explainability?–A Research Roadmap for Explainability Engineering,” in *International Working Conference on Requirements Engineering: Foundation for Software Quality.* Springer, 2022, pp. 26–32.

[281] S. Thiebes, S. Lins, and A. Sunyaev, “Trustworthy Artificial Intelligence,” *Electronic Markets*, pp. 1–18, 2020. doi: 10.1007/s12525-020-00441-4

[282] K. Amoako-Gyampah, “ERP implementation factors: a comparison of managerial and end-user perspectives,” *Business Process Management Journal*, 2004.

[283] A. L. Guzman, “What is human-machine communication, anyway?” in *Human-Machine Communication*, Andrea L. Guzman, Ed. New York, USA: Peter Lang Verlag, 2018, pp. 1–28.

[284] J. James and R. McDonald, “A Forecast of Space Technology 1980-2000,” National Aeronautics and Space Adminstration, Tech. Rep., 1976.

[285] R. J. Jacob, “Computers in human-computer interaction,” in *The human-computer interaction handbook: fundamentals, evolving technologies and emerging applications*, 2002, pp. 147–149.

[286] R. Cathcart and G. Gumpert, “The person-computer interaction: A unique source,” *Information and behavior*, vol. 1, pp. 113–124, 1985.

[287] A. Cooper, R. Reimann, and D. Cronin, *About Face 3: The Essentials of Interaction Design.* USA: John Wiley & Sons, Inc., 2007. ISBN 9780470084113

[288] A. C. Weigand, D. Lange, and M. Rauschenberger, “How can Small Data Sets be Clustered?” *Mensch und Computer 2021-Workshopband*, 2021.

[289] ISO Central Secretary, “ISO/IEC 9241-210:2019 Ergonomics of human-system interaction — Part 210: Human-centred design for interactive systems,” International Organization for Standardization, Standard ISO 9241-210:2019, 2019. [Online]. Available: https://www.iso.org/standard/77520.html

[290] L. Chazette, “Need for Explanations - Survey - Online Material,” https://doi.org/10.5281/zenodo.3366062, Jun. 2019, accessed 31 August 2022.

[291] A. N. Venturelli, “A cautionary contribution to the philosophy of explanation in the cognitive neurosciences,” *Minds and Machines*, vol. 26, no. 3, pp. 259–285, 2016.

[292] J. Sliwinski, M. Strobel, and Y. Zick, “A characterization of monotone influence measures for data classification,” in *Proceedings of the IJCAI Workshop on Explainable Artificial Intelligence (XAI 2017)*, 2017, pp. 48–52.

[293] C. Brinton, “A Framework for Explanation of Machine Learning Decisions,” in *Proceedings of the IJCAI Workshop on Explainable Artificial Intelligence (XAI)*, 2017, pp. 14–18.

[294] S. Robbins, “A Misdirected Principle with a Catch: Explicability for AI,” *Minds and Machines*, vol. 29, no. 4, pp. 495–514, 2019. doi: 10.1007/s11023-019-09509-3

[295] R. Pierrard, J.-P. Poli, and C. Hudelot, “A New Approach for Explainable Multiple Organ Annotation with Few Data,” in *Proceedings of the IJCAI Workshop on Explainable Artificial Intelligence (XAI 2019)*, 2019, pp. 101–107.

[296] A. I. Antón and J. B. Earp, “A requirements taxonomy for reducing web site privacy vulnerabilities,” *Requirements engineering*, vol. 9, no. 3, pp. 169–185, 2004.

[297] A. Richardson and A. Rosenfeld, “A survey of interpretability and explainability in human-agent systems,” in *Proceedings of the IJCAI/ECAI Workshop on Explainable Artificial Intelligence (XAI 2018)*, 2018, pp. 137–143.

[298] M. Hall, D. Harborne, R. Tomsett, V. Galetic, S. Quintana-Amate, A. Nottle, and A. Preece, “A Systematic Method to Understand Requirements for Explainable AI (XAI) Systems,” in *Proceedings of the IJCAI 2019 Workshop on Explainable Artificial Intelligence (XAI)*, 2019, pp. 21–27.

[299] T. Eiter, Z. G. Saribatur, and P. Schüller, “Abstraction for Zooming-In to Unsolvability Reasons of Grid-Cell Problems,” in *Proceedings of the IJCAI Workshop on Explainable Artificial Intelligence (XAI 2019)*, 2019, pp. 7–13.

[300] M. Krishnan, “Against Interpretability: A Critical Examination of the Interpretability Problem in Machine Learning,” *Philosophy & Technology*, pp. 1–16, 2019. doi: 10.1007/s13347-019-00372-9

[301] L. Floridi, J. Cowls, M. Beltrametti, R. Chatila, P. Chazerand, V. Dignum, C. Luetge, R. Madelin, U. Pagallo, F. Rossi, B. Schafer, P. Valcke, and E. Vayena, “AI4People — An Ethical Framework for a Good AI Society: Opportunities, Risks, Principles, and Recommendations,” *Minds and Machines*, vol. 28, no. 4, pp. 689–707, 2018. doi: 10.1007/s11023-018-9482-5

[302] P. B. De Laat, “Algorithmic decision-making based on machine learning from Big Data: Can transparency restore accountability?” *Philosophy & Technology*, vol. 31, no. 4, pp. 525–541, 2018. doi: 10.1007/s13347-017-0293-z

[303] E. S. Dahl, “Appraising black-boxed technology: the positive prospects,” *Philosophy & Technology*, vol. 31, no. 4, pp. 571–591, 2018. doi: 10.1007/s13347-017-0275-1

[304] U. Ehsan, P. Tambwekar, L. Chan, B. Harrison, and M. O. Riedl, “Automated rationale generation: a technique for explainable AI and its effects on human perceptions,” in *Proceedings of the 24th International Conference on Intelligent User Interfaces*, 2019, pp. 263–274.

[305] A. Aggarwal, P. Lohia, S. Nagar, K. Dey, and D. Saha, “Black Box Fairness Testing of Machine Learning Models,” in *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. New York, NY, USA: Association for Computing Machinery, 2019. doi: 10.1145/3338906.3338937. ISBN 9781450355728 p. 625–635. [Online]. Available: https://doi.org/10.1145/3338906.3338937

[306] K. L. Theurer, “Compositional explanatory relations and mechanistic reduction,” *Minds and Machines*, vol. 23, no. 3, pp. 287–307, 2013.

[307] D. J. Buller, “Confirmation and the computational paradigm (or: Why do you think they call it artificial intelligence?),” *Minds and Machines*, vol. 3, no. 2, pp. 155–181, 1993.

[308] A. Lucic, H. Haned, and d. R. Maarten, "Contrastive Explanations for Large Errors in Retail Forecasting Predictions through Monte Carlo Simulations," in *Proceedings of the IJCAI Workshop on Explainable Artificial Intelligence (XAI 2019)*, 2019, pp. 66–72.

[309] K. Sokol and P. A. Flach, "Conversational Explanations of Machine Learning Predictions Through Class-contrastive Counterfactual Statements." in *Proceedings of the IJCAI/ECAI Workshop on Explainable Artificial Intelligence (XAI 2018)*, 2018, pp. 5785–5786.

[310] M. L. Olson, L. Neal, F. Li, and W.-K. Wong, "Counterfactual States for Atari Agents via Generative Deep Learning," in *Proceedings of the IJCAI Workshop on Explainable Artificial Intelligence (XAI 2019)*, 2019, pp. 87–93.

[311] H. A. Simon, "Discovering Explanations," *Minds and Machines*, vol. 8, no. 1, pp. 7–37, Feb 1998.

[312] L. Hiley, A. Preece, Y. Hicks, D. Marshall, and H. Taylor, "Discriminating spatial and temporal relevance in deep Taylor decompositions for explainable activity recognition," in *Proceedings of the IJCAI Workshop on Explainable Artificial Intelligence (XAI 2019)*, 2019, pp. 35–39.

[313] Y. Coppens, K. Efthymiadis, T. Lenaerts, A. Nowé, T. Miller, R. Weber, and D. Magazzeni, "Distilling deep reinforcement learning policies in soft decision trees," in *Proceedings of the IJCAI Workshop on Explainable Artificial Intelligence (XAI 2019)*, 2019, pp. 1–6.

[314] N. Tintarev and J. Masthoff, "Effective explanations of recommendations: user-centered design," in *Proceedings of the 2007 ACM conference on Recommender systems*, 2007, pp. 153–156.

[315] S. T. Mckinlay, "Evidence, explanation and predictive data modelling," *Philosophy & Technology*, vol. 30, no. 4, pp. 461–473, 2017.

[316] T. Miller, P. Howe, and L. Sonenberg, "Explainable AI: Beware of Inmates Running the Asylum. Or: How I Learnt to Stop Worrying and Love the Social and Behavioural Sciences," in *Proceedings of the IJCAI 2017 Workshop on Explainable Artificial Intelligence (XAI)*, D. W. Aha, T. Darrell, M. Pazzani, D. Reid, C. Sammut, and P. Stone, Eds. Santa Clara County, CA, USA: IJCAI, 2017, pp. 36–42.

[317] M. Fox, D. Long, and D. Magazzeni, "Explainable Planning," in *Proceedings of the IJCAI Workshop on Explainable Artificial Intelligence (XAI 2017)*, 2017, pp. 24–30.

[318] P. Madumal, T. Miller, L. Sonenberg, and F. Vetere, “Explainable Reinforcement Learning Through a Causal Lens,” in *Proceedings of the IJCAI Workshop on Explainable Artificial Intelligence (XAI 2019)*, 2019, pp. 73–79.

[319] Z. Juozapaitis, A. Koul, A. Fern, M. Erwig, and F. Doshi-Velez, “Explainable Reinforcement Learning via Reward Decomposition,” in *Proceedings of the IJCAI Workshop on Explainable Artificial Intelligence (XAI 2019)*, 2019, pp. 47–53.

[320] F. Gutiérrez, S. Charleer, R. De Croon, N. N. Htun, G. Goetschalckx, and K. Verbert, “Explaining and exploring job recommendations: a user-driven approach for interacting with knowledge-based job recommender systems,” in *Proceedings of the 13th ACM Conference on Recommender Systems*, 2019, pp. 60–68.

[321] R. O. Weber, H. Hong, and P. Goel, “Explaining Citation Recommendations: Abstracts or Full Texts?” in *Proceedings of the IJCAI Workshop on Explainable Artificial Intelligence (XAI 2019)*, 2019, pp. 136–142.

[322] N. Angius and G. Tamburrini, “Explaining engineered computing systems’ behaviour: the role of abstraction and idealization,” *Philosophy & Technology*, vol. 30, no. 2, pp. 239–258, 2017.

[323] J. Dodge, Q. V. Liao, Y. Zhang, R. K. Bellamy, and C. Dugan, “Explaining models: an empirical study of how explanations impact fairness judgment,” in *Proceedings of the 24th International Conference on Intelligent User Interfaces*, 2019, pp. 275–285.

[324] L. Chen and F. Wang, “Explaining recommendations based on feature sentiments in product reviews,” in *Proceedings of the 22nd international conference on intelligent user interfaces*, 2017, pp. 17–28.

[325] C.-H. Tsai and P. Brusilovsky, “Explaining recommendations in an interactive hybrid social recommender,” in *Proceedings of the 24th International Conference on Intelligent User Interfaces*, 2019, pp. 391–396.

[326] O. Biran and C. Cotton, “Explanation and justification in machine learning: A survey,” in *IJCAI-17 workshop on explainable AI (XAI)*, vol. 8, no. 1, 2017, pp. 8–13.

[327] A. Gopnik, “Explanation as orgasm,” *Minds and machines*, vol. 8, no. 1, pp. 101–118, 1998.

[328] J. Fernández, “Explanation by computer simulation in cognitive science,” *Minds and Machines*, vol. 13, no. 2, pp. 269–284, 2003.

[329] H. Johnson and P. Johnson, "Explanation facilities and interactive systems," in *Proceedings of the 1st International Conference on Intelligent User Interfaces (IUI).* New York, NY, USA: Association for Computing Machinery, 1993. doi: 10.1145/169891.169951. ISBN 0897915569 pp. 159–166.

[330] W. F. Brewer, C. A. Chinn, and A. Samarapungavan, "Explanation in scientists and children," *Minds and Machines*, vol. 8, no. 1, pp. 119–136, 1998.

[331] A. Rueger, "Explanations at Multiple Levels," *Minds and Machines*, vol. 11, no. 4, pp. 503–520, 2001.

[332] J. De Winter, "Explanations in software engineering: The pragmatic point of view," *Minds and Machines*, vol. 20, no. 2, pp. 277–289, 2010. doi: 10.1007/s11023-010-9190-2

[333] N. Tintarev, "Explanations of Recommendations," in *Proceedings of the 2007 ACM Conference on Recommender Systems.* New York, NY, USA: Association for Computing Machinery, 2007. doi: 10.1145/1297231.1297275. ISBN 978-1-59593-730-8 pp. 203–206.

[334] J. McInerney, B. Lacker, S. Hansen, K. Higley, H. Bouchard, A. Gruson, and R. Mehrotra, "Explore, exploit, and explain: personalizing explainable recommendations with bandits," in *Proceedings of the 12th ACM conference on recommender systems*, 2018, pp. 31–39.

[335] C. Lucero, B. Coronado, O. Hui, and D. S. Lange, "Exploring explainable artificial intelligence and autonomy through provenance," in *Proceedings of the IJCAI/ECAI Workshop on Explainable Artificial Intelligence (XAI 2018)*, 2018, p. 85.

[336] V. Putnam and C. Conati, "Exploring the Need for Explainable Artificial Intelligence (XAI) in Intelligent Tutoring Systems (ITS)." in *IUI Workshops*, vol. 19, 2019.

[337] A. J. Ko and B. A. Myers, "Extracting and answering why and why not questions about Java program output," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 20, no. 2, pp. 1–36, 2010.

[338] M. Veale and R. Binns, "Fairer machine learning in the real world: Mitigating discrimination without collecting sensitive data," *Big Data & Society*, vol. 4, no. 2, pp. 1–17, 2017. doi: 10.1177/2053951717743530

[339] M. Hosseini, A. Shahri, K. Phalp, and R. Ali, "Four reference models for transparency requirements in information systems," *Requirements Engineering*, vol. 23, no. 2, pp. 251–275, 2018.

[340] G. J. Nalepa, M. van Otterlo, S. Bobek, and M. Atzmueller, "From context mediation to declarative values and explainability," in *Proceedings of the IJCAI/ECAI Workshop on Explainable Artificial Intelligence (XAI 2018)*, 2018.

[341] S. J. Green, P. Lamere, J. Alexander, F. Maillet, S. Kirk, J. Holt, J. Bourque, and X.-W. Mak, "Generating transparent, steerable recommendations from textual descriptions of items," in *Proceedings of the third ACM conference on Recommender systems*, 2009, pp. 281–284.

[342] J. Schaffer, P. Giridhar, D. Jones, T. Höllerer, T. Abdelzaher, and J. O'donovan, "Getting the message? A study of explanation interfaces for microblog data analysis," in *Proceedings of the 20th international conference on intelligent user interfaces*, 2015, pp. 345–356.

[343] J. C.-T. Ho, "How biased is the sample? Reverse engineering the ranking algorithm of Facebook's Graph application programming interface," *Big Data & Society*, vol. 7, no. 1, pp. 1–15, 2020. doi: 10.1177/2053951720905874

[344] A. Papenmeier, G. Englebienne, and C. Seifert, "How Model Accuracy and Explanation Fidelity Influence User Trust in AI," in *Proceedings of the IJCAI 2019 Workshop on Explainable Artificial Intelligence (XAI)*, 2019, pp. 94–100.

[345] T. Barik, D. Ford, E. Murphy-Hill, and C. Parnin, "How should compilers explain problems to developers?" in *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2018, pp. 633–643.

[346] J. Burrell, "How the machine 'thinks': Understanding opacity in machine learning algorithms," *Big Data & Society*, vol. 3, no. 1, pp. 1–12, 2016. doi: 10.1177/2053951715622512

[347] J. Schaffer, J. O'Donovan, J. Michaelis, A. Raglin, and T. Höllerer, "I Can Do Better than Your AI: Expertise and Explanations," in *Proceedings of the 24th International Conference on Intelligent User Interfaces*, ser. IUI '19. New York, NY, USA: Association for Computing Machinery, 2019. doi: 10.1145/3301275.3302308. ISBN 978-1-4503-6272-6 pp. 240–251, event-place: Marina del Ray, California.

[348] H. Liu, J. Wen, L. Jing, J. Yu, X. Zhang, and M. Zhang, "In2Rec: Influence-based interpretable recommendation," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 1803–1812.

[349] B. Ghosh, D. Malioutov, and K. S. Meel, "Interpretable Classification Rules in Relaxed Logical Form," in *Proceedings of the IJCAI Workshop on Explainable Artificial Intelligence (XAI 2019)*, 2019, pp. 14–20.

[350] T. Huber, D. Schiller, and E. André, "Introducing Selective Layer Wise Relevance Propagation to Dueling Deep Q learning," in *Proceedings of the IJCAI Workshop on Explainable Artificial Intelligence (XAI 2019))*, 2019.

[351] W. Bechtel, "Levels of Description and Explanation in Cognitive Science," *Minds and Machines*, vol. 4, no. 1, pp. 1–25, Feb 1994. doi: 10.1007/BF00974201

[352] X. Watts and F. Lécué, "Local Score Dependent Model Explanation for Time Dependent Covariates," in *Proceedings of the IJCAI Workshop on Explainable Artificial Intelligence (XAI 2019)*, 2019, pp. 129–135.

[353] L. Michael, "Machine Coaching," in *Proceedings of the IJCAI Workshop on Explainable Artificial Intelligence (XAI 2019)*, 2019, pp. 80–86.

[354] R. McClamrock, "Marr's three levels: A re-evaluation," *Minds and Machines*, vol. 1, no. 2, pp. 185–196, 1991.

[355] M. T. Stuart and N. J. Nersessian, "Peeking inside the black box: a new kind of scientific visualization," *Minds and Machines*, vol. 29, no. 1, pp. 87–107, 2019. doi: 10.1007/s11023-018-9484-3

[356] T. Kulesza, M. Burnett, W.-K. Wong, and S. Stumpf, "Principles of explanatory debugging to personalize interactive machine learning," in *Proceedings of the 20th international conference on intelligent user interfaces*, 2015, pp. 126–137.

[357] P. Sawyer, N. Bencomo, J. Whittle, E. Letier, and A. Finkelstein, "Requirements-aware systems: A research agenda for re for self-adaptive systems," in *2010 18th IEEE International Requirements Engineering Conference.* IEEE, 2010, pp. 95–103.

[358] C. Zednik, "Solving the black box problem: A normative framework for explainable artificial intelligence," *Philosophy & Technology*, pp. 1–24, 2019. doi: 10.1007/s13347-019-00382-7

[359] R. Häuslschmid, M. von Buelow, B. Pfleging, and A. Butz, "Supporting Trust in autonomous driving," in *Proceedings of the 22nd international conference on intelligent user interfaces*, 2017, pp. 319–329.

[360] J. Vig, S. Sen, and J. Riedl, "Tagsplanations: Explaining Recommendations Using Tags," in *Proceedings of the 14th International Conference on Intelligent User Interfaces (IUI.* New York, NY, USA: Association for Computing Machinery, 2009. doi: 10.1145/1502650.1502661. ISBN 978-1-60558-168-2 pp. 47–56.

[361] K. Ehrlich, S. E. Kirk, J. Patterson, J. C. Rasmussen, S. I. Ross, and D. M. Gruen, "Taking Advice from Intelligent Systems: The Double-Edged Sword of Explanations," in *Proceedings of the 16th International Conference on Intelligent User Interfaces (IUI).* New York, NY, USA: Association for Computing Machinery, 2011. doi: 10.1145/1943403.1943424. ISBN 978-1-4503-0419-1 pp. 125–134.

[362] V. Dominguez, P. Messina, I. Donoso-Guzmán, and D. Parra, "The effect of explanations and algorithmic accuracy on visual recommender systems of artistic images," in *Proceedings of the 24th International Conference on Intelligent User Interfaces*, 2019, pp. 408–416.

[363] B. D. Mittelstadt, P. Allo, M. Taddeo, S. Wachter, and L. Floridi, "The ethics of algorithms: Mapping the debate," *Big Data & Society*, vol. 3, no. 2, pp. 1–21, 2016. doi: 10.1177/2053951716679679

[364] L. Floridi, "The method of levels of abstraction," *Minds and machines*, vol. 18, no. 3, pp. 303–329, 2008.

[365] A. Páez, "The Pragmatic Turn in Explainable Artificial Intelligence (XAI)," *Minds & Machines*, vol. 29, pp. 441–459, 2019. doi: 10.1007/s11023-019-09502-w

[366] R. A. Wilson and F. Keil, "The shadows and shallows of explanation," *Minds and machines*, vol. 8, no. 1, pp. 137–159, 1998.

[367] M. T. Keane and E. M. Kenny, "The twin-system approach as one generic solution for XAI: An overview of ANN-CBR twins for explaining deep learning," in *Proceedings of the IJCAI Workshop on Explainable Artificial Intelligence (XAI 2019)*, 2019, pp. 54–58.

[368] J. Clos, N. Wiratunga, and S. Massie, "Towards Explainable Text Classification by Jointly Learning Lexicon and Modifier Terms," in *Proceedings of the IJCAI Workshop on Explainable Artificial Intelligence (XAI 2017)*, 2017, pp. 19–23.

[369] H. Wicaksono, C. Sammut, and R. Sheh, "Towards Explainable Tool Creation by a Robot," in *Proceedings of the IJCAI Workshop on Explainable Artificial Intelligence (XAI 2017)*, 2017, pp. 63–67.

[370] V. Putnam, L. Riegel, and C. Conati, "Towards Personalized XAI: A Case Study in Intelligent Tutoring Systems," in *Proceedings of the IJCAI Workshop on Explainable Artificial Intelligence (XAI 2019)*, 2019, pp. 108–114.

[371] J. Zerilli, A. Knott, J. Maclaurin, and C. Gavaghan, "Transparency in algorithmic and human decision-making: is there a double standard?" *Philosophy & Technology*, vol. 32, no. 4, pp. 661–683, 2019. doi: 10.1007/s13347-018-0330-6

[372] H. Felzmann, E. F. Villaronga, C. Lutz, and A. Tamò-Larrieux, "Transparency you can trust: Transparency requirements for artificial intelligence between legal norms and contextual concerns," *Big Data & Society*, vol. 6, no. 1, pp. 1–14, 2019. doi: 10.1177/2053951719860542

[373] X. He, T. Chen, M.-Y. Kan, and X. Chen, "Trirank: Review-aware explainable recommendation by modeling aspects," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, 2015, pp. 1661–1670.

[374] P. Pu and L. Chen, "Trust building with explanation interfaces," in *Proceedings of the 11th international conference on Intelligent user interfaces*, 2006, pp. 93–100.

[375] J. Drozdal, J. Weisz, D. Wang, G. Dass, B. Yao, C. Zhao, M. Muller, L. Ju, and H. Su, "Trust in automl: Exploring information needs for establishing trust in automated machine learning systems," in *Proceedings of the 25th International Conference on Intelligent User Interfaces*, 2020, pp. 297–307.

[376] D. Holliday, S. Wilson, and S. Stumpf, "User trust in intelligent systems: A journey over time," in *Proceedings of the 21st International Conference on Intelligent User Interfaces*, 2016, pp. 164–168.

[377] N. F. Rajani and R. J. Mooney, "Using Explanations to Improve Ensembling of Visual Question Answering Systems," in *Proceedings of the IJCAI 2017 Workshop on Explainable Artificial Intelligence (XAI)*, 2017, pp. 43–47.

[378] D. van Eck, "Validating function-based design methods: An explanationist perspective," *Philosophy & Technology*, vol. 28, no. 4, pp. 511–531, 2015.

[379] S. Jasanoff, "Virtual, visible, and actionable: Data assemblages and the sightlines of justice," *Big Data & Society*, vol. 4, no. 2, pp. 1–15, 2017. doi: 10.1177/2053951717724477

[380] P. Madumal, T. Miller, L. Sonenberg, and F. Vetere, "A Grounded Interaction Protocol for Explainable Artificial Intelligence," in *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*. Richland, SC, USA:

International Foundation for Autonomous Agents and Multiagent Systems, 2019. doi: 10.5555/3306127.3331801. ISBN 978-1-4503-6309-9 pp. 1033–1041.

[381] D. A. Wilkenfeld, "Functional explaining: A new approach to the philosophy of explanation," *Synthese*, vol. 191, pp. 3367–3391, 2014. doi: 10.1007/s11229-014-0452-z

[382] L. Viganò and D. Magazzeni, "Explainable Security," in *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS PW)*, 2020. doi: 10.1109/EuroSPW51379.2020.00045 pp. 293–300.

[383] M. Milkowski, "A Mechanistic Account of Computational Explanation in Cognitive Science," in *Proceedings of the 35th Annual Meeting of the Cognitive Science Society*, M. Knauff, M. Pauen, N. Sebanz, and I. Wachsmuth, Eds. Cognitive Science Society, 2013, pp. 3050–3055. [Online]. Available: https://mindmodeling.org/cogsci2013/papers/0545/index.html

[384] D. Billsus and M. J. Pazzani, "A personal news agent that talks, learns and explains," in *Proceedings of the third annual conference on Autonomous Agents*, 1999, pp. 268–275.

[385] M. Harbers, K. van den Bosch, and J.-J. C. Meyer, "A study into preferred explanations of virtual agent behavior," in *International Workshop on Intelligent Virtual Agents.* Springer, 2009, pp. 132–145.

[386] M.-A. Clinciu and H. Hastie, "A Survey of Explainable AI Terminology," in *Proceedings of the 1st Workshop on Interactive Natural Language Technology for Explainable Artificial Intelligence (NL4XAI 2019).* Association for Computational Linguistics, 2019. doi: 10.18653/v1/W19-8403 pp. 8–13.

[387] G. Friedrich and M. Zanker, "A taxonomy for generating explanations in recommender systems," *AI Magazine*, vol. 32, no. 3, pp. 90–98, 2011.

[388] A. Datta, S. Sen, and Y. Zick, "Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems," in *2016 IEEE symposium on security and privacy (SP).* IEEE, 2016, pp. 598–617.

[389] *An Evaluation of the Human-Interpretability of Explanation*, 2018.

[390] L. M. Cysneiros and V. M. B. Werneck, "An Initial Analysis on How Software Transparency and Trust Influence each other." in *WER*, 2009.

[391] K. Čyras, D. Letsios, R. Misener, and F. Toni, "Argumentation for Explainable Scheduling," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01,

pp. 2752–2759, Jul. 2019. doi: 10.1609/aaai.v33i01.33012752. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/4126

[392] B. Y. Lim and A. K. Dey, "Assessing demand for intelligibility in context-aware applications," in *Proceedings of the 11th International Conference on Ubiquitous Computing*, 2009. doi: 10.1145/1620545.1620576 pp. 195–204.

[393] E. S. Vorm, "Assessing Demand for Transparency in Intelligent Systems Using Machine Learning," in *2018 Innovations in Intelligent Systems and Applications (INISTA)*. IEEE, 2018. doi: 10.1109/INISTA.2018.8466328 pp. 1–7.

[394] Y. Fukuchi, M. Osawa, H. Yamakawa, and M. Imai, "Autonomous self-explanation of behavior for interactive reinforcement learning agents," in *Proceedings of the 5th International Conference on Human Agent Interaction*, 2017, pp. 97–101.

[395] Z. Zeng, C. Miao, C. Leung, and J. J. Chin, "Building more explainable artificial intelligence with argumentation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[396] J. Y. Halpern and J. Pearl, "Causes and explanations: A structural-model approach. Part II: Explanations," *The British journal for the philosophy of science*, vol. 56, no. 4, pp. 889–911, 2005.

[397] T. Laugel, M.-J. Lesot, C. Marsala, X. Renard, and M. Detyniecki, "Comparison-based inverse classification for interpretability in machine learning," in *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*. Springer, 2018, pp. 100–111.

[398] A. A. Freitas, "Comprehensible Classification Models: A Position Paper," *SIGKDD Explorations Newsletter*, vol. 15, no. 1, pp. 1–10, 2014. doi: 10.1145/2594473.2594475

[399] M. Sato, K. Nagatani, T. Sonoda, Q. Zhang, and T. Ohkuma, "Context Style Explanation for Recommender Systems," *Journal of Information Processing*, vol. 27, pp. 720–729, 2019. doi: 10.2197/ipsjjip.27.720

[400] R. M. J. Byrne, "Counterfactuals in Explainable Artificial Intelligence (XAI): Evidence from Human Reasoning," in *IJCAI*, 2019, pp. 6276–6282.

[401] M. Hosseini, A. Shahri, K. Phalp, and R. Ali, "Crowdsourcing transparency requirements through structured feedback and social adaptation," in *2016 IEEE Tenth International Conference on Research Challenges in Information Science (RCIS)*. IEEE, 2016, pp. 1–6.

[402] A. J. Ko and B. A. Myers, "Debugging Reinvented: Asking and Answering Why and Why Not Questions about Program Behavior," in *Proceedings of the 30th International Conference on Software Engineering*, ser. ICSE '08. New York, NY, USA: Association for Computing Machinery, 2008. doi: 10.1145/1368088.1368130. ISBN 9781605580791 p. 301–310. [Online]. Available: https://doi.org/10.1145/1368088.1368130

[403] R. Sheh and I. Monteath, "Defining Explainable AI for Requirements Analysis," *KI-Künstliche Intelligenz*, vol. 32, no. 4, pp. 261–266, 2018. doi: 10.1007/s13218-018-0559-3

[404] U. Chajewska and J. Y. Halpern, "Defining Explanation in Probabilistic Systems," in *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997. ISBN 1558604855 p. 62–71.

[405] T. Hirsch, K. Merced, S. Narayanan, Z. E. Imel, and D. C. Atkins, "Designing Contestability: Interaction Design, Machine Learning, and Mental Health," in *Proceedings of the 2017 Conference on Designing Interactive Systems*, ser. DIS '17. New York, NY, USA: Association for Computing Machinery, 2017. doi: 10.1145/3064663.3064703. ISBN 9781450349222 p. 95–99.

[406] D. Walton, "Dialogical Models of Explanation." *ExaCt*, vol. 2007, pp. 1–9, 2007.

[407] R. Sheh, "Different XAI for different HRI," in *AAAI Fall Symposium*. AAAI Press, 2017, pp. 114–117.

[408] M. Ter Hoeve, M. Heruer, D. Odijk, A. Schuth, and M. de Rijke, "Do news consumers want explanations for personalized news rankings," in *FATREC Workshop on Responsible Recommendation Proceedings*, 2017.

[409] D. M. Truxillo, T. E. Bodner, M. Bertolino, T. N. Bauer, and C. A. Yonce, "Effects of explanations on applicant reactions: A meta-analytic review," *International Journal of Selection and Assessment*, vol. 17, no. 4, pp. 346–361, 2009.

[410] B. Goodman and S. Flaxman, "European Union regulations on algorithmic decision-making and a "right to explanation"," *AI Magazine*, vol. 38, no. 3, pp. 50–57, 2017. doi: 10.1609/aimag.v38i3.2741

[411] K. Sokol and P. Flach, "Explainability Fact Sheets: A Framework for Systematic Assessment of Explainable Approaches," in *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, ser. FAT* '20. New York, NY, USA: Association for Computing Machinery, 2020. doi: 10.1145/3351095.3372870. ISBN 978-1-4503-6936-7 pp. 56–67, event-place: Barcelona, Spain.

[412] S. Anjomshoae, A. Najjar, D. Calvaresi, and K. Främling, "Explainable Agents and Robots: Results from a Systematic Literature Review," in *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS).* Richland County, SC, USA: International Foundation for Autonomous Agents and Multiagent Systems, 2019. doi: 10.5555/3306127.3331806. ISBN 978-1-4503-6309-9 pp. 1078–1088.

[413] J. Zhu, A. Liapis, S. Risi, R. Bidarra, and G. M. Youngblood, "Explainable AI for designers: A human-centered perspective on mixed-initiative co-creation," in *2018 IEEE Conference on Computational Intelligence and Games (CIG).* IEEE, 2018. doi: 10.1109/CIG.2018.8490433 pp. 1–8.

[414] F. Kamiran and I. Žliobaitė, "Explainable and non-explainable discrimination in classification," in *Discrimination and Privacy in the Information Society.* Springer, 2013, pp. 155–170.

[415] N. Wang, H. Wang, Y. Jia, and Y. Yin, "Explainable recommendation via multi-task learning in opinionated text data," in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, 2018, pp. 165–174.

[416] H. K. Dam, T. Tran, and A. Ghose, "Explainable Software Analytics," in *Proceedings of the 40th International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER).* New York, NY, USA: Association for Computing Machinery, 2018. doi: 10.1145/3183399.3183424. ISBN 978-1-4503-5662-6 pp. 53–56.

[417] R. R. Hoffman, G. Klein, and S. T. Mueller, "Explaining Explanation For "Explainable AI"," *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 62, no. 1, pp. 197–201, 2018. doi: 10.1177/1541931218621047

[418] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal, "Explaining Explanations: An Overview of Interpretability of Machine Learning," in *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, IEEE. IEEE, 2018. doi: 10.1109/DSAA.2018.00018 pp. 80–89.

[419] B. D. Mittelstadt, C. Russell, and S. Wachter, "Explaining Explanations in AI," in *Proceedings of the 2019 Conference on Fairness, Accountability, and Transparency.* New York, NY, USA: Association for Computing Machinery, 2019. doi: 10.1145/3287560.3287574 pp. 279–288.

[420] K. Cotter, J. Cho, and E. Rader, "Explaining the news feed algorithm: An analysis of the "News Feed FYI" blog," in *Proceedings of the 2017 CHI Conference Extended Abstracts on*

Human Factors in Computing Systems (CHI EA). New York, NY, USA: Association for Computing Machinery, 2017. doi: 10.1145/3027063.3053114. ISBN 978-1-4503-4656-6 pp. 1553–1560.

[421] T. Lombrozo and N. Z. Gwynne, "Explanation and inference: Mechanistic and functional explanations guide property generalization," *Frontiers in Human Neuroscience*, vol. 8, p. 700, 2014.

[422] K. McCain, "Explanation and the nature of scientific knowledge," *Science & Education*, vol. 24, no. 7, pp. 827–854, 2015.

[423] F. C. Keil, "Explanation and Understanding," *Annual Review of Psychology*, vol. 57, no. 1, pp. 227–254, 2006. doi: 10.1146/annurev.psych.57.102904.190100

[424] F. Sørmo, J. Cassens, and A. Aamodt, "Explanation in case-based reasoning–perspectives and goals," *Artificial Intelligence Review*, vol. 24, no. 2, pp. 109–143, 2005. doi: 10.1007/s10462-005-4607-7

[425] G. Ras, v. G. Marcel, and P. Haselager, "Explanation methods in deep learning: Users, values, concerns and challenges," in *Explainable and Interpretable Models in Computer Vision and Machine Learning.* Springer, 2018, pp. 19–36. ISBN 978-3-319-98131-4

[426] E. I. Sklar and M. Q. Azhar, "Explanation through Argumentation," in *Proceedings of the 6th International Conference on Human-Agent Interaction*, ser. HAI '18. New York, NY, USA: Association for Computing Machinery, 2018. doi: 10.1145/3284432.3284470. ISBN 9781450359535 p. 277–285.

[427] D. Walton, "Explanations and Arguments Based on Practical Reasoning." in *ExaCt*, 2009, pp. 72–83.

[428] E. Rader, K. Cotter, and J. Cho, "Explanations as Mechanisms for Supporting Algorithmic Transparency," in *Proceedings of the 2018 Conference on Human Factors in Computing Systems (CHI).* New York, NY, USA: Association for Computing Machinery, 2018. doi: 10.1145/3173574.3173677. ISBN 978-1-4503-5620-6 pp. 1–13.

[429] S. Gregor and I. Benbasat, "Explanations from Intelligent Systems: Theoretical Foundations and Implications for Practice," *MIS Quarterly*, vol. 23, no. 4, pp. 497–530, 1999. doi: 10.2307/249487

[430] S. Anjomshoae, K. Främling, and A. Najjar, "Explanations of Black-Box Model Predictions by Contextual Importance and Utility," in *Explainable, Transparent Autonomous*

Agents and Multi-Agent Systems. Springer, 2019. doi: 10.1007/978-3-030-30391-4_6. ISBN 978-3-030-30391-4 pp. 95–109.

[431] E. Weber, J. Van Bouwel, and R. Vanderbeeken, "Forms of causal explanation," *Foundations of Science*, vol. 10, no. 4, pp. 437–454, 2005.

[432] L. A. Hendricks, Z. Akata, M. Rohrbach, J. Donahue, B. Schiele, and T. Darrell, "Generating visual explanations," in *European Conference on Computer Vision.* Springer, 2016, pp. 3–19.

[433] R. Sevastjanova, F. Beck, B. Ell, C. Turkay, R. Henkin, M. Butt, D. A. Keim, and M. El-Assady, "Going beyond Visualization: Verbalization as Complementary Medium to Explain Machine Learning Models," in *VIS Workshop on Visualization for AI Explainability (VISxAI)*, 2018, pp. 1–6.

[434] S. Sreedharan, T. Chakraborti, and S. Kambhampati, "Handling Model Uncertainty and Multiplicity in Explanations via Model Reconciliation," *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 28, no. 1, 2018.

[435] M. M. De Graaf and B. F. Malle, "How people explain action (and autonomous intelligent systems should too)," in *2017 AAAI Fall Symposium Series*, 2017.

[436] O. Biran and K. McKeown, "Human-Centric Justification of Machine Learning Predictions," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence.* AAAI Press, 2017. ISBN 9780999241103 p. 1461–1467.

[437] B. Lamche, U. Adıgüzel, and W. Wörndl, "Interactive explanations in mobile shopping recommender systems," in *Joint Workshop on Interfaces and Human Decision Making in Recommender Systems*, vol. 14, 2014.

[438] B. P. Knijnenburg and A. Kobsa, "Making decisions about privacy: information disclosure in context-aware recommender systems," *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 3, no. 3, pp. 1–23, 2013.

[439] J. Schneider and J. P. Handali, "PERSONALIZED EXPLANATION FOR MACHINE LEARNING: A CONCEPTUALIZATION," in *27th European Conference on Information Systems (ECIS)*, 2019.

[440] J. Zhou, H. Hu, Z. Li, K. Yu, and F. Chen, "Physiological Indicators for User Trust in Machine Learning with Influence Enhanced Fact-Checking," in *International Cross-Domain Conference for Machine Learning and Knowledge Extraction.* Cham, CH:

Springer International Publishing, 2019. doi: 10.1007/978-3-030-29726-8_7. ISBN 978-3-030-29726-8 pp. 94–113.

[441] T. Chakraborti, S. Sreedharan, S. Grover, and S. Kambhampati, "Plan explanations as model reconciliation," in *14th ACM/IEEE International Conference on Human-Robot Interaction (HRI).* IEEE, 2019. doi: 10.1109/HRI.2019.8673193 pp. 258–266.

[442] M. K. Lee, A. Jain, H. J. Cha, S. Ojha, and D. Kusbit, "Procedural Justice in Algorithmic Fairness," *Proceedings of the 2019 ACM on Human-Computer Interaction*, vol. 3, pp. 1–26, 2019. doi: 10.1145/3359284

[443] S. Nagulendra and J. Vassileva, "Providing awareness, explanation and control of personalized filtering in a social networking site," *Information Systems Frontiers*, vol. 18, no. 1, pp. 145–158, 2016. doi: 10.1007/s10796-015-9577-y

[444] M. J. Druzdzel, "Qualitative verbal explanations in bayesian belief networks," *AISB QUARTERLY*, pp. 43–54, 1996.

[445] Q. V. Liao, D. Gruen, and S. Miller, "Questioning the AI: informing design practices for explainable AI user experiences," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 2020, pp. 1–15.

[446] U. Ehsan, B. Harrison, L. Chan, and M. O. Riedl, "Rationalization: A neural machine translation approach to generating natural language explanations," in *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, 2018, pp. 81–87.

[447] M. R. Wick and W. B. Thompson, "Reconstructive expert system explanation," *Artificial Intelligence*, vol. 54, no. 1-2, pp. 33–70, 1992.

[448] J. Van Bouwel and E. Weber, "Remote causes, bad explanations?" *Journal for the Theory of Social Behaviour*, vol. 32, no. 4, pp. 437–449, 2002.

[449] O. Zinovatna and L. M. Cysneiros, "Reusing knowledge on delivering privacy and transparency together," in *2015 IEEE Fifth International Workshop on Requirements Patterns (RePa).* IEEE, 2015, pp. 17–24.

[450] J. F. Osborne and A. Patterson, "Scientific argument and explanation: A necessary distinction?" *Science Education*, vol. 95, no. 4, pp. 627–638, 2011.

[451] J. D. Trout, "Scientific explanation and the sense of understanding," *Philosophy of Science*, vol. 69, no. 2, pp. 212–233, 2002.

[452] L. Edwards and M. Veale, “Slave to the algorithm: Why a right to an explanation is probably not the remedy you are looking for,” *Duke L. & Tech. Rev.*, vol. 16, p. 18, 2017.

[453] I. Baaj, J.-P. Poli, and W. Ouerdane, “Some Insights Towards a Unified Semantic Representation of Explanation for eXplainable Artificial Intelligence,” in *Proceedings of the 2019 Workshop on Interactive Natural Language Technology for Explainable Artificial Intelligence (NL4XAI).* Association for Computational Linguistics, 2019. doi: 10.18653/v1/W19-8404 pp. 14–19.

[454] M. Strevens, “The causal and unification approaches to explanation unified—causally,” *Noûs*, vol. 38, no. 1, pp. 154–176, 2004.

[455] F. Nothdurft, T. Heinroth, and W. Minker, “The Impact of Explanation Dialogues on Human-Computer Trust,” in *International Conference on Human-Computer Interaction (HCI).* Springer, 2013. doi: 10.1007/978-3-642-39265-8_7. ISBN 978-3-642-39265-8 pp. 59–67.

[456] L. R. Ye and P. E. Johnson, “The impact of explanation facilities on user acceptance of expert systems advice,” *Mis Quarterly*, pp. 157–172, 1995.

[457] A. Vellido, “The importance of interpretability and visualization in machine learning for applications in medicine and health care,” *Neural computing and applications*, pp. 1–15, 2019.

[458] T. Lombrozo, “The Instrumental Value of Explanations,” *Philosophy Compass*, vol. 6, no. 8, pp. 539–551, 2011. doi: 10.1111/j.1747-9991.2011.00413.x

[459] J. Trout, “The psychology of scientific explanation,” *Philosophy Compass*, vol. 2, no. 3, pp. 564–591, 2007.

[460] T.-W. Chen and S. S. Sundar, “This App Would Like to Use Your Current Location to Better Serve You: Importance of User Assent and System Transparency in Personalized Mobile Services,” in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, ser. CHI ’18. New York, NY, USA: Association for Computing Machinery, 2018. doi: 10.1145/3173574.3174111. ISBN 9781450356206 p. 1–13.

[461] M. Sridharan and B. Meadows, “Towards a Theory of Explanations for Human–Robot Collaboration,” *KI-Künstliche Intelligenz*, vol. 33, no. 4, pp. 331–342, 2019. doi: 10.1007/s13218-019-00616-y

[462] V. Dominguez, P. Messina, C. Trattner, and D. Parra, "Towards Explanations for Visual Recommender Systems of Artistic Images," in *Joint Workshop on Interfaces and Human Decision Making for Recommender Systems* , 2018.

[463] M. Atzmueller, "Towards Socio-Technical Design of Explicative Systems: Transparent, Interpretable and Explainable Analytics and Its Perspectives in Social Interaction Contexts information," in *Proceedings of the 2019 Workshop on Affective Computing and Context Awareness in Ambient Intelligence (AfCAI)*, 2019, pp. 1–8.

[464] R. Iyer, Y. Li, H. Li, M. Lewis, R. Sundar, and K. Sycara, "Transparency and Explanation in Deep Reinforcement Learning Neural Networks," in *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, ser. AIES '18. New York, NY, USA: Association for Computing Machinery, 2018. doi: 10.1145/3278721.3278776. ISBN 9781450360128 p. 144–150.

[465] K. Balog, F. Radlinski, and S. Arakelyan, "Transparent, Scrutable and Explainable User Models for Personalized Recommendation," in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval.* New York, NY, USA: Association for Computing Machinery, 2019. doi: 10.1145/3331184.3331211. ISBN 978-1-4503-6172-9 pp. 265–274.

[466] A. Abdul, J. Vermeulen, D. Wang, B. Y. Lim, and M. Kankanhalli, "Trends and Trajectories for Explainable, Accountable and Intelligible Systems: An HCI Research Agenda," in *Proceedings of the 2018 Conference on Human Factors in Computing Systems (CHI).* New York, NY, USA: ACM, 2018. doi: 10.1145/3173574.3174156 pp. 1–18.

[467] P. Pu and L. Chen, "Trust-inspiring explanation interfaces for recommender systems," *Knowledge-Based Systems*, vol. 20, no. 6, pp. 542–556, 2007.

[468] J. Bidot, S. Biundo-Stephan, T. Heinroth, W. Minker, F. Nothdurft, and B. Schattenberg, "Verbal Plan Explanations for Hybrid Planning," in *MKWI*, 2010.

[469] S. Rosenthal, S. P. Selvaraj, and M. Veloso, "Verbalization: Narration of Autonomous Robot Experience," in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, ser. IJCAI'16. AAAI Press, 2016. ISBN 9781577357704 p. 862–868.

[470] Derek Doran, Sarah Schulz, and Tarek R. Besold, "What Does Explainable AI Really Mean? A New Conceptualization of Perspectives," in *Proceedings of the First International Workshop on Comprehensibility and Explanation in AI and ML*, vol. 2071. Aachen, DE: CEUR, 2017.

[471] R. Sheh, "" Why did you do that?" Explainable intelligent robots," in *AAAI Workshop-Technical Report*, 2017, pp. 628–634.

[472] T. Kulesza, S. Stumpf, W.-K. Wong, M. M. Burnett, S. Perona, A. Ko, and I. Oberst, "Why-oriented end-user debugging of naive Bayes text classification," *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 1, no. 1, pp. 1–31, 2011. doi: 10.1145/2030365.2030367

[473] A. Yasin, R. Fatima, L. Wen, W. Afzal, M. Azhar, and R. Torkar, "On Using Grey Literature and Google Scholar in Systematic Literature Reviews in Software Engineering," *IEEE Access*, vol. 8, pp. 36 226–36 243, 2020. doi: 10.1109/ACCESS.2020.2971712

[474] L. Yang, H. Wang, and L. A. Deleris, "What Does It Mean to Explain? A User-Centered Study on AI Explainability," in *Artificial Intelligence in HCI*, H. Degen and S. Ntoa, Eds. Cham: Springer International Publishing, 2021. ISBN 978-3-030-77772-2 pp. 107–121.

[475] F. Cech and M. Wagner, "eRollin' On Green: A Case Study on Eco-Feedback Tools for eMobility," in *Proceedings of the 9th International Conference on Communities & Technologies - Transforming Communities*, ser. C&T '19. New York, NY, USA: Association for Computing Machinery, 2019. doi: 10.1145/3328320.3328402. ISBN 9781450371629 p. 121–125.

[476] F. Du, C. Plaisant, N. Spring, K. Crowley, and B. Shneiderman, "EventAction: A visual analytics approach to explainable recommendation for event sequences," *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 9, no. 4, pp. 1–31, 2019.

[477] A. I. Grimaldo and J. Novak, "User-centered visual analytics approach for interactive and explainable energy demand analysis in prosumer scenarios," in *International Conference on Computer Vision Systems*. Springer, 2019, pp. 700–710.

[478] E. Holder and N. Wang, "Explainable artificial intelligence (XAI) interactively working with humans as a junior cyber analyst," *Human-Intelligent Systems Integration*, pp. 1–15, 2021.

[479] T. Schneider, J. Hois, A. Rosenstein, S. Ghellal, D. Theofanou-Fülbier, and A. R. S. Gerlicher, "ExplAIn Yourself! Transparency for Positive UX in Autonomous Driving," in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, 2021, pp. 1–12.

[480] J. Andres, C. T. Wolf, S. Cabrero Barros, E. Oduor, R. Nair, A. Kjærum, A. B. Tharsgaard, and B. S. Madsen, "Scenario-based XAI for Humanitarian Aid Forecasting," in *Extended*

Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems, 2020, pp. 1–8.

[481] O. Barkan, Y. Fuchs, A. Caciularu, and N. Koenigstein, "Explainable recommendations via attentive multi-persona collaborative filtering," in *Fourteenth ACM Conference on Recommender Systems*, 2020, pp. 468–473.

[482] J. Sun, Q. V. Liao, M. Muller, M. Agarwal, S. Houde, K. Talamadupula, and J. D. Weisz, "Investigating Explainability of Generative AI for Code through Scenario-Based Design," in *27th International Conference on Intelligent User Interfaces*, ser. IUI '22. New York, NY, USA: Association for Computing Machinery, 2022. doi: 10.1145/3490099.3511119. ISBN 9781450391443 p. 212–228. [Online]. Available: https://doi.org/10.1145/3490099.3511119

[483] S. Jesus, C. Belém, V. Balayan, J. Bento, P. Saleiro, P. Bizarro, and J. Gama, "How can I choose an explainer? An Application-grounded Evaluation of Post-hoc Explanations," in *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, 2021, pp. 805–815.

[484] H. Ramos, M. Fonseca, and L. Ponciano, "Modeling and Evaluating Personas with Software Explainability Requirements," in *Iberoamerican Workshop on Human-Computer Interaction*. Springer, 2021, pp. 136–149.

[485] H. Subramonyam, C. Seifert, and E. Adar, "Towards A Process Model for Co-Creating AI Experiences," in *Designing Interactive Systems Conference 2021*, ser. DIS '21. New York, NY, USA: Association for Computing Machinery, 2021. doi: 10.1145/3461778.3462012. ISBN 9781450384766 p. 1529–1543.

[486] Y. Mualla, I. H. Tchappi, A. Najjar, T. Kampik, S. Galland, and C. Nicolle, "Human-agent Explainability: An Experimental Case Study on the Filtering of Explanations," in *ICAART (1)*, 2020, pp. 378–385.

[487] C.-H. Tsai and P. Brusilovsky, "The effects of controllability and explainability in a social recommender system," *User Modeling and User-Adapted Interaction*, vol. 31, no. 3, pp. 591–627, Jul 2021. doi: 10.1007/s11257-020-09281-5

[488] U. Ehsan, P. Wintersberger, Q. V. Liao, M. Mara, M. Streit, S. Wachter, A. Riener, and M. O. Riedl, "Operationalizing Human-Centered Perspectives in Explainable AI," in *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, ser. CHI EA '21. New York, NY, USA: Association for Computing Machinery,

2021. doi: 10.1145/3411763.3441342. ISBN 978-1-4503-8095-9 Event-place: Yokohama, Japan.

[489] D. Omeiza, K. Kollnig, H. Web, M. Jirotka, and L. Kunze, "Why not explain? effects of explanations on human perceptions of autonomous driving," in *2021 IEEE International Conference on Advanced Robotics and Its Social Impacts (ARSO)*. IEEE, 2021, pp. 194–199.

[490] J. Rebanal, J. Combitsis, Y. Tang, and X. Chen, "XAlgo: a Design Probe of Explaining Algorithms' Internal States via Question-Answering," in *26th International Conference on Intelligent User Interfaces*, 2021, pp. 329–339.

[491] C. T. Wolf, "Explainability scenarios: towards scenario-based XAI design," in *Proceedings of the 24th International Conference on Intelligent User Interfaces*, 2019, pp. 252–257.

[492] T. Kulesza, S. Stumpf, M. Burnett, W.-K. Wong, Y. Riche, T. Moore, I. Oberst, A. Shinsel, and K. McIntosh, "Explanatory debugging: Supporting end-user debugging of machine-learned programs," in *2010 IEEE Symposium on Visual Languages and Human-Centric Computing*. IEEE, 2010, pp. 41–48.

[493] Y. Xie, M. Chen, D. Kao, G. Gao, and X. A. Chen, "CheXplain: Enabling Physicians to Explore and Understand Data-Driven, AI-Enabled Medical Imaging Analysis," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, ser. CHI '20. New York, NY, USA: Association for Computing Machinery, 2020. doi: 10.1145/3313831.3376807. ISBN 9781450367080 p. 1–13.

[494] T. Zhou, H. Sheng, and I. Howley, "Assessing Post-hoc Explainability of the BKT Algorithm," in *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, 2020, pp. 407–413.

[495] J. Heier, "Design Intelligence-Taking Further Steps Towards New Methods and Tools for Designing in the Age of AI," in *International Conference on Human-Computer Interaction*. Springer, 2021, pp. 202–215.

[496] A. Holzinger, A. Carrington, and H. Müller, "Measuring the quality of explanations: the system causability scale (SCS)," *KI-Künstliche Intelligenz*, pp. 1–6, 2020.

[497] R. M. Carvalho, R. M. de Castro Andrade, K. M. de Oliveira, I. de Sousa Santos, and C. I. M. Bezerra, "Quality characteristics and measures for human–computer interaction evaluation in ubiquitous systems," *Software Quality Journal*, vol. 25, no. 3, pp. 743–795, 2017. doi: 0.1145/1864349.1864395

[498] F. Sovrano, F. Vitali, and M. Palmirani, “Modelling GDPR-Compliant Explanations for Trustworthy AI,” in *Electronic Government and the Information Systems Perspective*, ser. Lecture Notes in Computer Science, A. Kő, E. Francesconi, G. Kotsis, A. M. Tjoa, and I. Khalil, Eds. Cham: Springer International Publishing, 2020. doi: 10.1007/978-3-030-58957-8_16. ISBN 978-3-030-58957-8 pp. 219–233.

[499] J. Koo, J. Kwac, W. Ju, M. Steinert, L. Leifer, and C. Nass, “Why did my car just do that? Explaining semi-autonomous driving actions to improve driver understanding, trust, and performance,” *International Journal on Interactive Design and Manufacturing (IJIDeM)*, vol. 9, no. 4, pp. 269–275, 2015. doi: 10.1007/s12008-014-0227-2

[500] J. Haspiel, N. Du, J. Meyerson, L. P. Robert Jr., D. Tilbury, X. J. Yang, and A. K. Pradhan, “Explanations and Expectations: Trust Building in Automated Vehicles,” in *Companion of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, ser. HRI ’18. New York, NY, USA: Association for Computing Machinery, 2018. doi: 10.1145/3173386.3177057. ISBN 978-1-4503-5615-2 pp. 119–120, event-place: Chicago, IL, USA.

[501] K. Martin, A. Liret, N. Wiratunga, G. Owusu, and M. Kern, “Developing a Catalogue of Explainability Methods to Support Expert and Non-expert Users,” in *Artificial Intelligence XXXVI*, ser. Lecture Notes in Computer Science, M. Bramer and M. Petridis, Eds. Cham: Springer International Publishing, 2019. doi: 10.1007/978-3-030-34885-4_24. ISBN 978-3-030-34885-4 pp. 309–324.

[502] T. Schrills and T. Franke, “Color for Characters - Effects of Visual Explanations of AI on Trust and Observability,” in *Artificial Intelligence in HCI*, ser. Lecture Notes in Computer Science, H. Degen and L. Reinerman-Jones, Eds. Cham: Springer International Publishing, 2020. doi: 10.1007/978-3-030-50334-5_8. ISBN 978-3-030-50334-5 pp. 121–135.

[503] K. Weitz, D. Schiller, R. Schlagowski, T. Huber, and E. André, “Do You Trust Me?: Increasing User-Trust by Integrating Virtual Agents in Explainable AI Interaction Design,” in *Proceedings of the 19th ACM International Conference on Intelligent Virtual Agents*, ser. IVA ’19. New York, NY, USA: Association for Computing Machinery, 2019. doi: 10.1145/3308532.3329441. ISBN 978-1-4503-6672-4 pp. 7–9, event-place: Paris, France.

[504] M. Riveiro and S. Thill, ““That’s (not) the output I expected!” On the role of end user expectations in creating explanations of AI systems,” *Artificial Intelligence*, vol. 298, p. 103507, 2021. doi: 10.1016/j.artint.2021.103507

[505] N. Tintarev and J. Masthoff, “A survey of explanations in recommender systems,” in *2007 IEEE 23rd International Conference on Data Engineering Workshop*. IEEE, 2007. doi: 10.1109/ICDEW.2007.4401070 pp. 801–810.

List of Scientific Publications

[1] A. d. S. O. Filho, F. A. M. Abdelnor, and L. Pimentel, “Arquitetura de Software para um Ambiente de Redes Orientadas a Plano de Deslocamento Contínuo como forma de Comunicação entre Comunidades Remotas e Grandes Centros Urbanos,” Bachelor’s thesis, Universidade da Amazônia, Belém, Pará, Brasil, October 2015.

[2] L. Pimentel, D. Rosário, E. Cerqueira, T. Braun, A. Abelém, and M. Gerla, “Context-aware adaptation mechanism for video dissemination over Flying Ad-Hoc Networks,” in 2014 IFIP Wireless Days (WD), 2014, pp. 1–3.

[3] L. Pimentel, D. Rosário, M. Seruffo, Z. Zhao, and T. Braun, “Adaptive Beaconless Opportunistic Routing for Multimedia Distribution,” in Wired/Wireless Internet Communications, M. C. Aguayo-Torres, G. Gómez, and J. Poncela, Eds. Cham: Springer International Publishing, 2015, pp. 122–135.

[4] L. M. Pimentel, “CABR: Protocolo de Roteamento Beaconless, Oportunístico e Adaptativo Para Distribuição Multimídia Em Flying AD-HOC Networks,” Master’s thesis, Universidade Federal Do Pará, Belém, Pará, Brasil, October 2015.

[5] L. Chazette, M. Becker, and H. Szczerbicka, “Basic algorithms for bee hive monitoring and laser-based mite control,” in 2016 IEEE Symposium Series on Computational Intelligence (SSCI), 2016, pp. 1–8.

[6] J. Greenyer, L. Chazette, D. Gritzner, and E. Wete, “A Scenario-Based MDE Process for Dynamic Topology Collaborative Reactive Systems-Early Virtual Prototyping of Car-to-X System Specifications.” in Modellierung (Workshops), 2018, pp. 111–120.

[7] L. Chazette, O. Karras, and K. Schneider, “Do End-Users Want Explanations? Analyzing the Role of Explainability as an Emerging Aspect of Non-Functional Requirements,” in 2019 IEEE 27th International Requirements Engineering Conference (RE), 2019, pp. 223–233.

[8] L. Chazette, “Mitigating Challenges in the Elicitation and Analysis of Transparency Requirements,” in 2019 IEEE 27th International Requirements Engineering Conference (RE), 2019, pp. 470–475.

[9] L. Chazette and K. Schneider, “Explainability as a non-functional requirement: challenges and recommendations,” Requirements Engineering, vol. 25, no. 4, pp. 493–514, 2020.

[10] L. Chazette, M. Busch, K. Korte, and K. Schneider, “Designing Software Transparency: A Multidisciplinary Endeavor,” in Softwaretechnik-Trends, vol. 40, no. 1, 2020, pp. 5–6.

[11] L. Chazette, W. Brunotte, and T. Speith, “Exploring Explainability: A Definition, a Model, and a Knowledge Catalogue,” in 2021 IEEE 29th International Requirements Engineering Conference (RE), 2021, pp. 197–208.

[12] W. Brunotte, L. Chazette, and K. Korte, “Can Explanations Support Privacy Awareness? A Research Roadmap,” in 2021 IEEE 29th International Requirements Engineering Conference Workshops (REW), 2021, pp. 176–180.

[13] W. Brunotte, L. Chazette, V. Klös, E. Knauss, T. Speith, and A. Vogelsang, “Welcome to the First International Workshop on Requirements Engineering for Explainable Systems (RE4ES),” in 2021 IEEE 29th International Requirements Engineering Conference Workshops (REW), 2021, pp. 157–158.

[14] L. Chazette, J. Klünder, M. Balci, and K. Schneider, “How Can We Develop Explainable Systems? Insights from a Literature Review and an Interview Study,” in Proceedings of the International Conference on Software and System Processes and International Conference on Global Software Engineering, ser. ICSSP’22. New York, NY, USA: Association for Computing Machinery, 2022, p. 112.

[15] M. Herrmann, M. Obaidi, L. Chazette, and J. Klünder, “On the subjectivity of emotions in software projects: How reliable are pre-labeled data sets for sentiment analysis?” Journal of Systems and Software, vol. 193, p. 111448, 2022.

[16] W. Brunotte, L. Chazette, L. Kohler, J. Klunder, and K. Schneider, “What About My Privacy?Helping Users Understand Online Privacy Policies,” in Proceedings of the International Conference on Software and System Processes and International Conference on Global Software Engineering, ser. ICSSP’22. New York, NY, USA: Association for Computing Machinery, 2022, p. 5665.

[17] L. Chazette, V. Klös, F. Herzog, and K. Schneider, “Requirements on Explanations: A Quality Framework for Explainability,” in 2022 IEEE 30th International Requirements Engineering Conference (RE), 2022.

[18] W. Brunotte, L. Chazette, V. Klös, and T. Speith, “Quo Vadis, Explainability?–A Research Roadmap for Explainability Engineering,” in International Working Conference on Requirements Engineering: Foundation for Software Quality. Springer, 2022, pp. 26–32.

Persönliche Daten

Name, Vorname	Chazette, Larissa
E-Mail:	larissa.chazette@inf.uni-hannover.de

Studium

08.2018 – 10.2022

Promotion Informatik

Leibniz Universität Hannover – Deutschland

Dissertationsthema: Requirements Engineering für Erklärbare Systeme

- Forschung in den Bereichen erklärbare Systeme, Software Engineering und Requirements Engineering.
- Titel der Dissertation: „Requirements Engineering for Explainable Systems."

01.2013 – 05.2015

M.Sc. in Informatik

Bundesuniversität von Pará (UFPA) – Belém – Brasilien

Masterarbeitsthema: Computernetzwerk

- Konzeption und Entwicklung eines Routing-Protokolls für „Unmanned Aerial Vehicles"
- Titel der Masterarbeit: „CABR: Adaptive Beaconless Opportunistic Routing Protocol for Multimedia Distribution over Flying Ad-Hoc Networks"

02.2008 -12.2011

B.Sc. in Informatik

Universität Amazoniens (UNAMA) – Belém – Brasilien

Bachelorarbeitsthema: Computernetzwerk

- Konzeption und Entwicklung eines Routing-Protokolls und eines Systems für ferngesteuerte medizinische Termine.
- Titel der Bachelorarbeit: „ Software für den fernmedizinischen Termin über Delay Tolerant Networks " (übersetzt)

Praktische Erfahrung

10.2017 – 10.2022	**Wissenschaftliche Mitarbeiterin - Fachgebiet Software Engineering** Leibniz Universität Hannover – Hannover – Deutschland
02.2016 – 10.2017	**Studentische Hilfskraft - Fachgebiet Simulation und Modellierung** Leibniz Universität Hannover – Hannover – Deutschland
01.2016 – 02.2016	**Softwareentwicklerin** VoxData Telecommunications – Belém – Brasilien
06.2015-12.2015	**Robotik Lehrerin** LEGO Education – Belém – Brasilien
07.2012-05.2015	**Wissenschaftliche Mitarbeiterin (Forschungsstipendiatin)** Bundesuniversität von Pará (UFPA) – Belém – Brasilien