Requirements Engineering (2022) 27:457–487
https://doi.org/10.1007/s00766-022-00393-5

ORIGINAL ARTICLE

Explainable software systems: from requirements analysis to system evaluation

Larissa Chazette[1] · **Wasja Brunotte**[1,2] · **Timo Speith**[3,4]

Received: 31 January 2022 / Accepted: 1 November 2022 / Published online: 14 November 2022

Abstract
The growing complexity of software systems and the influence of software-supported decisions in our society sparked the need for software that is transparent, accountable, and trustworthy. Explainability has been identified as a means to achieve these qualities. It is recognized as an emerging non-functional requirement (NFR) that has a significant impact on system quality. Accordingly, software engineers need means to assist them in incorporating this NFR into systems. This requires an early analysis of the benefits and possible design issues that arise from interrelationships between different quality aspects. However, explainability is currently under-researched in the domain of requirements engineering, and there is a lack of artifacts that support the requirements engineering process and system design. In this work, we remedy this deficit by proposing four artifacts: a definition of explainability, a conceptual model, a knowledge catalogue, and a reference model for explainable systems. These artifacts should support software and requirements engineers in understanding the definition of explainability and how it interacts with other quality aspects. Besides that, they may be considered a starting point to provide practical value in the refinement of explainability from high-level requirements to concrete design choices, as well as on the identification of methods and metrics for the evaluation of the implemented requirements.

Keywords Explainability · Explainable artificial intelligence · Non-functional requirements · Quality aspects · Conceptual model · Reference model · Knowledge catalogue

1 Introduction

We are living in the algorithmic age [1]. Software decision-making has spread from simple daily decisions, such as the choice of a navigation route, to more critical ones, such as the diagnosis of cancer patients [2]. Systems have been strongly influencing various aspects of our lives with their outputs, but they can be as mysterious as black boxes to us [3].

The ubiquitous influence of such "black-box systems" has induced discussions about the transparency and ethics of modern systems [4]. Responsible collection and use of data, privacy, safety, and security are just a few among many concerns. In light of this, it is becoming increasingly crucial to understand how to incorporate these concerns into systems and, thus, how to deal with them during software engineering (SE) and requirements engineering (RE).

1.1 Explainability as the solution

In this regard, explainability is increasingly seen as the preferred solution to mitigate a system's lack of transparency [5] and as a fruitful way to address ethical concerns about modern systems [6]. The concept of explainability has received a lot of attention recently, and it is slowly

✉ Larissa Chazette
larissa.chazette@inf.uni-hannover.de

Wasja Brunotte
wasja.brunotte@inf.uni-hannover.de

Timo Speith
timo.speith@uni-bayreuth.de

1 Software Engineering Group, Leibniz University Hannover, Hannover, Germany

2 Cluster of Excellence PhoenixD, Leibniz University Hannover, Hannover, Germany

3 Chair for Philosophy, Computer Science and Artificial Intelligence, University of Bayreuth, Bayreuth, Germany

4 Center for Perspicuous Computing, Saarland University, Saarbrücken, Germany

establishing itself as an important non-functional requirement (NFR)[1] for high system quality [8, 9].

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 also has an impact on the relationship of trust in and reliance on a system [10], it may avoid feelings of frustration [11], and can, thus, lead to greater acceptance by end users [12]. In general, previous studies have shown that explainability is not only a means of achieving transparency and calibrating trust, but that it is also linked to other important NFRs, such as usability, auditability, and privacy [5, 13–15].

However, although explainability has been identified as such an essential NFR for software-supported decisions [16] and even as one of the pillars for trustworthy artificial intelligence (AI) [17], there is still a lack of a comprehensive overview that investigates the impact of incorporating explainability in a system.[2] For example, it is often overlooked that these impacts are not only positive, but can also be negative.

To address this gap in knowledge, we investigate the concept of explainability and its interaction with other quality aspects[3,4] in this article. Accordingly, we want to provide clarity on what constitutes explainability as an NFR and how it can be integrated into the RE process. Just like other NFRs, however, explainability is difficult to elicit, negotiate, and validate.

1.2 Challenges in RE and how artifacts can help solve them

Due to the subjective, interactive, and relative nature of NFRs, eliciting and modeling them presents many challenges for software engineers[5] [20]. First, knowledge about NFRs is mostly tacit, disseminated, and based on experience [21, 22], which makes it difficult to grasp the existing knowledge. Here, explainability is no exception.

Furthermore, software quality is a multidimensional concept based on real-world needs that comprise different layers of abstraction [5, 23]. Therefore, another challenge for software engineers is to translate abstract notions, such as quality goals (e.g., a smart home system that improves user *satisfaction* and *comfort*) into agreed-upon tangible functionality that help achieve the quality goals (e.g., the smart home system dims the lights automatically when the user is tired). This translation process is followed by the need to evaluate how the derived functionality (also called "operationalizations") influence software quality [20, 24]. This process from abstract to concrete often leads to trade-offs between different NFRs in a system that must be identified and resolved during requirements analysis [20, 25].

RE is not simply a process of identifying and describing requirements; it is also a process of supporting efficient communication of these requirements among various stakeholders [26]. For this reason, proper communication is another difficulty for RE. External stakeholders and internal team members may unintentionally use different words for the same concepts or the same words for different concepts due to a lack of shared understanding, which can make communication challenging and lead to problems because of misunderstandings [27]. Therefore, a shared understanding is crucial for efficient communication and for reducing the risk of stakeholder dissatisfaction and rework [19].

Software engineers can create, use, and reuse artifacts to achieve a shared understanding in software projects [19]. An artifact is "any kind of textual or graphical document with the exception of source code" [28]. Artifacts can take on a variety of shapes, including textual requirement papers, visual models, glossaries, charts, frameworks, or quality models. Artifacts that are often used to support the RE process are conceptual models [21], knowledge catalogues [20], and reference models [29]. Such artifacts may describe the taxonomy of a given kind of system or process, or compile knowledge about specific NFRs and their interactions with other quality aspects.

To illustrate the importance of both a shared understanding and artifacts in software projects, RE itself may be taken as an example: Communication problems may arise from different interpretations of what RE is and how the RE process is structured. Börger et al. [30] proposed a reference model (i.e., a type of artifact) for RE to achieve a shared understanding of it. To achieve a shared understanding about the meaning of RE, the proposed model describes the concept and divides the RE process into two main areas (*requirements analysis* and *requirements management*) and their related activities.[6]

[1] We follow Glinz and see NFRs as *attributes of* or *constraints on* a system [7].

[2] We deal with a very broad conception of explainability that applies to software systems in general. Accordingly, our focus is not specifically on AI systems, and we are, therefore, not only concerned with so-called *explainable AI* (XAI).

[3] We use the notion of *quality aspects* to refer both to *NFRs* and to *aspects* that relate to or compose NFRs.

[4] The terms *NFR*, *quality aspects*, and *quality goals* are used throughout this article. We consider quality goals as the quality aspects that are agreed upon for system quality within a project, which can be stated as or refined into NFRs [18].

[5] 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 [19].

[6] In this article, we stick to this reference model for our conception of RE. Furthermore, because the majority of system development errors and risks occur primarily during the requirements analysis phase and result in significant financial expenditure [31], we focus on the area

Chung et al. [21] explain the importance of models and knowledge catalogues as resources for the use and reuse of knowledge during system development. Models and catalogues can compile either abstract or concrete knowledge. At a more abstract level, such artifacts can compile knowledge about different NFRs and their interrelationships with other quality aspects (e.g., positive or negative influence of an NFR on another quality aspect). Likewise, models and catalogues can also compile more concrete knowledge, such as about methods and techniques in the field that can be used to operationalize a given NFR.

Existing works propose to build artifacts to capture and structure knowledge that is scattered among several sources [20, 21, 32, 33]. Software engineers can use such artifacts during different activities in RE such as during elicitation, interpretation and trade-off analysis, negotiations with stakeholders, as well as to support the documentation of requirements and decisions. In summary, 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) achieve a shared understanding that leads to better communication and to the definition of the "right" system's requirements.

1.3 Goal and structure of this article

As for explainability, there is a scarcity of artifacts that compile structured knowledge about this quality aspect and assist software engineers in understanding the factors that should be considered during the development of explainable systems, helping to achieve a shared understanding of the topic.

Therefore, *we propose four artifacts that should aid in achieving a shared understanding of explainability, supporting the creation of explainable systems*: a definition, a conceptual model, a knowledge catalogue, and a reference model. Overall, our goal is to advance the knowledge towards a common terminology and semantics of explainability, facilitating the discussion and analysis of this important NFR during the RE process. To this end, we used an interdisciplinary systematic literature review (SLR) and workshops as part of a multi-method research strategy.

In particular, we distill definitions of explainability into an own suggestion that is appropriate for SE and RE. We use this definition as a starting point to create a conceptual model that represents the impacts of explainability across different quality dimensions. Subsequently, we compile a knowledge catalogue of explainability and its impacts on the various quality aspects along these dimensions. Finally, we conceive a reference model for explainability that describes key aspects to consider when developing explainable systems during requirements analysis, design, and evaluation. The goal of these artifacts is to support the identification, communication, and evaluation of key elements of explainable systems, their attributes, and relationships.

This article is an extension of a paper originally published in the *29th IEEE International Requirements Engineering Conference*: [34]. In this extension, we (1) include more details about our research method, and (2) propose a reference model that may be used to identify relevant components for the RE process and the development of explainable systems.

This article is structured as follows: in the following Sect. 2, we present the background and related work. In Sect. 3, we lay the foundation for the more substantive chapters of this article by introducing our research questions (RQs) and outlining the chosen research design. This is followed by a section for each artifact we present. Accordingly, we suggest our definition of explainability in Sect. 4, introduce our conceptual model in Sect. 5, and present the explainability catalogue in Sect. 6. Building on the previous artifacts, we conceive the reference model in Sect. 7. We discuss our results in Sect. 8, and we debate threats to validity in Sect. 9. Finally, we conclude our article in Sect. 10.

2 Background and related work

Artifacts are commonly used in RE and SE to support software professionals during their tasks. For instance, software engineers typically use or reuse artifacts as guidance during SE (or RE) activities. They may also create artifacts to gather knowledge (e.g., catalogues), or they create artifacts as a form of documentation (e.g., requirements specification or story cards). This section provides background information on the types of artifacts that we propose as well as on explainability.

2.1 Definitions

Definitions are the first and most crucial step in facilitating communication for a given topic or concept. Definitions aid in defining the scope of a particular idea, for instance, by indicating its constituents. Definitions in SE and RE give a rough guidance for software engineers on the scope and elements of nearly everything. The definition of a quality aspect, for example, assists software engineers in understanding it during RE and especially quality assurance.

Definitions support a common terminology that facilitates communication. A lack of consensus may result in the

Footnote 6 (continued)

of requirements analysis and its corresponding activities: *elicitation*, *interpretation*, *negotiation*, *documentation*, and *validation/verification*.

specification and integration of the wrong requirements. Explicitly shared vocabulary decreases the likelihood of misunderstandings when ideas employing this terminology are not stated or are only loosely specified. Following this idea, Wixon [35] emphasizes the significance of defining usability and how crucial it is for development teams to reach consensus on this concept. For instance, usability may signify long-term efficiency to some developers while it may represent simplicity of use to others.

2.2 Models

A model is an abstraction of a system that deliberately focuses on some of its aspects while excluding others [36]. Models provide an overview of a field by partitioning it into broad categories. According to Hull et al. [36], a single model never says everything about a system. For this reason, different, possibly interrelated, models of systems are often used to cover a variety of different aspects. Models can be used for a number of different purposes. On the one hand, they can be utilized for more technical tasks like software design or configuration. On the other hand, models make it easier to describe and optimize organizational concerns including business processes and domains.

Conceptual models can be used to define and describe a concept, helping to understand the taxonomy or characteristics of a particular quality aspect during requirements analysis. Conceptual models document, for example, knowledge about a given domain, concept, or NFR. Taxonomies are well-known examples of conceptual models. The knowledge required to develop conceptual models is typically derived from literature, previous experiences, and domain expertise.

Quality models are another example of models. A quality model can be defined as "the set of characteristics, and the relationships between them that provides the basis for specifying quality requirements and evaluation" [37]. Quality models help to specify and illustrate how quality aspects translate to functional requirements.

A special category of models are the so-called *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" [38]. Reference models may be used as a blueprint for software system construction and are sometimes called universal models, generic models, or model patterns [39]. A reference model can serve as a template for creating and deriving other models (e.g., quality models) or understanding the high-level structure of a process or domain. To this end, *reference models* can be used to help identify important factors for the analysis, operationalization, and evaluation of a given quality aspect.

To give an example, the Open Systems Interconnection model (OSI) [40] 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 used as (1) abstract frameworks or templates for understanding significant relationships among the entities of some environment or domain (e.g., computer networks or, in our case, explainable systems) or (2) to standardize or describe processes [41]. This abstract nature gives them flexibility, making them easily adaptable.

Cherdantseva et al. [42] propose a reference model for the information assurance and security (IAS) domain. The model highlights the important aspects of IAS systems, and serves as a conceptual framework for researchers. The authors say that reference models foster a better understanding of IAS and, as a result, help software engineers to do their job more efficiently, serving as "a blueprint for the design of a secure information system" and providing "a basis for the elicitation of security requirements".

To clearly distinguish the different types of models, we consider a conceptual model to be an artifact that describes a concept or captures the taxonomy of a certain concept or domain (our conceptual illustrates the impact of explainability on system quality); and we consider a reference model as a template or framework that may be used by software engineers to build other models or to guide them as they design explainable systems.

2.3 Catalogues

Catalogues document knowledge about a given topic (e.g., a specific domain or about quality aspects, in the case of software systems). They can, for example, document relationships between different quality aspects. Some researchers developed catalogues for specific domains based on the premise of the NFR framework [21]. As a result, they can help with trade-off analysis, where it is critical to understand how two or more NFRs will interact in a system and how they can coexist [43].

Serrano and Serrano [32] developed a catalogue specifically for the ubiquitous, pervasive, and mobile computing domain. Torres and Martins [44] propose the use of NFR catalogues in the construction of 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.

Finally, Carvalho et al. [45] 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.

Mairiza et al. [20] conducted a literature review to identify conflicts among existing NFRs. They constructed a catalogue to synthesize the results and suggest that it can assist software developers in identifying, analyzing, and resolving conflicts between NFRs. Carvalho et al. [33] 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 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 them addresses explainability.

2.4 Explainability

Since explainability has rapidly expanded as a research field in the last years, publications about this topic have become quite numerous, and it is hard to keep track of the terms, methods, and results that came up [46]. For this reason, there have been numerous literature reviews presenting overviews concerning certain aspects (e.g., methods or definitions) of explainability research.

Many of these reviews focus on a specific community or application domain. For instance, [47] focuses on explainability of recommender systems, [48] on explainability of robots and human-robot interaction, [49] on the human-computer interaction (HCI) domain, and [50] on biomedical and malware classification. Another focus of these reviews is to demarcate different, but related terms often used in explainability research (see, e.g., [4, 46, 51]). For instance, the terms "explainablilty" and "interpretability" are sometimes used as synonyms and sometimes not [52, 53].

Our review differs from others in the following ways. To the best of our knowledge, our SLR is the first overview specifically targeting software and software engineers, to support them in dealing with explainability as a very new and complex NFR. For this reason, quality aspects are the pivotal focus of our work. As far as we are aware, only a few reviews explicitly include the interaction between explainability and quality aspects (most notably [47, 54]). In contrast to preceding reviews, however, we do not only consider positive impacts of explainability on other quality aspects, but we also take negative ones into account.

Furthermore, many other reviews do not have an interdisciplinary focus. Even if they do not focus on a specific community (e.g., HCI), reviews rarely incorporate views on explainability outside of computer science. From our point of view, however, it is crucial to include fields such as philosophy and psychology in an investigation of explainability, since these fields have much experience in explanation research. Psychology is concerned with aspects of human cognition, while philosophy is interested in the definition of concepts as well as the nature of knowledge and reality. These aspects are crucial to understanding the features, implications, and significance of explainability.

3 Research goal and design

We frame our study into four RQs:

RQ1: What is an appropriate definition of explainability to achieve shared understanding in SE and RE?

RQ2: What are the quality aspects impacted by explainability in a system context?

RQ3: How does explainability impact these quality aspects?

RQ4: How to support software professionals in identifying important factors for the analysis, operationalization, and evaluation of requirements for explainable systems?

Since other disciplines have a long history working on explainability, their insights should prove valuable for software engineering and enable us to refine the scope of the term explainability for this area. In particular, philosophy and psychology have a long history in making different conceptions of explanation explicit, for instance, in formalisms and operationalizations.

Accordingly, *RQ1* focuses on harnessing the work of other sciences in the field of explainability to compile a definition that is appropriate and useful for the disciplines of software and requirements engineering. Useful in this context means that the definition facilitates the discussion around the topic, contributing to a shared understanding among stakeholders and engineers as well as a clear vision of what "explainable system" means in the RE and SE contexts.

RQ2 focuses on providing an overview of the quality aspects that may be impacted by explainability. Similar to the work of Leite and Capelli [55], who investigated the interaction between transparency and other qualities, our goal is to offer an overview for explainability and its impact on other quality aspects within a system.

With *RQ3* we want to assess what impact explainability has on other quality aspects. More specifically, our goal is

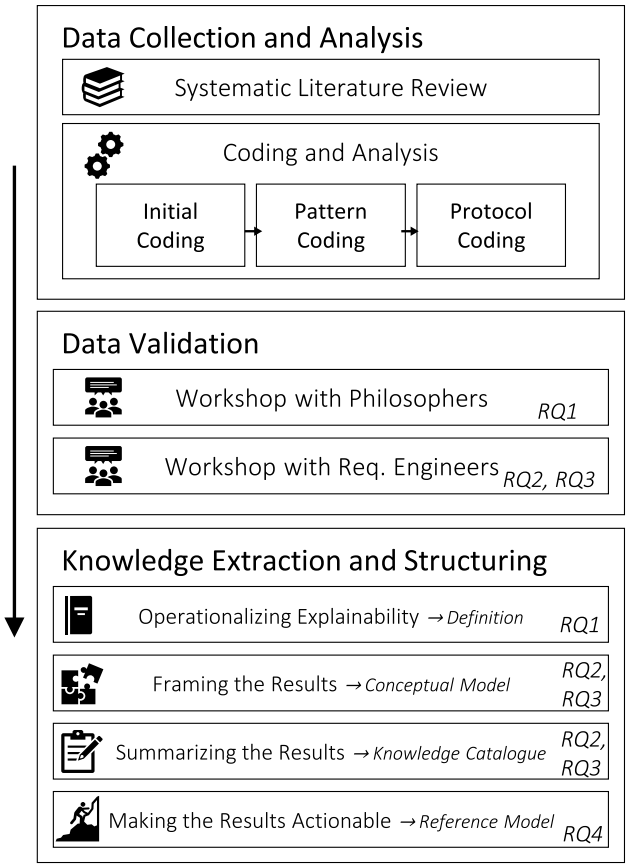

Fig. 1 Overview of the research design

to analyze the polarity of these impacts: whether they are positive or negative. To answer RQ2 and RQ3, we built a conceptual model and a catalogue that compile knowledge about the impacts of explainability on other quality aspects.

The goal of *RQ4* is to make our results more actionable. To this end, we provide a reference model for explainability that aims to build shared understanding around the factors to be considered in the development of explainable systems, assisting software engineers in identifying relevant factors for explainability in different phases of the software lifecycle: requirements analysis, design, and evaluation.

An overview of our research design is shown in Fig. 1. Our research consisted of a multi-method approach that combined two qualitative methods to achieve higher data reliability.

The first method focuses on systematic data collection and qualitative data analysis. For the data collection, we conducted an interdisciplinary *SLR* that resulted in a total of 229 papers. We *coded* the gathered data by using an open coding approach [56]. As a next step, we *analyzed* the resulting codes for definitions of explainability (RQ1), for relationships between explainability and other quality aspects (RQ2), and for information about the polarity of these relationships (RQ3).

To validate and complement our findings, we employed a second qualitative method: two *workshops* with experts. We framed the obtained knowledge in a *conceptual model* by structuring and grouping the quality aspects impacted by explainability along four dimensions and developed a *catalogue* based on it.

Finally, we used the responses to RQ1-RQ3 as a starting point and reviewed the data from our SLR to identify other principles and constituent parts of explainability to aid in the analysis of requirements, as well as in the design and evaluation of explainable systems. We combine this knowledge with works in the literature to build a *reference model* for explainability (RQ4), factoring in all our previous results.

3.1 Data collection and analysis

In what follows, we will describe our research design in more detail. To this end, we will start with a more detailed description of our SLR.

3.1.1 Systematic literature review

We followed guidelines from Kitchenham et al. [57], and Wohlin [58] when conducting our SLR. The search strategy for our SLR consisted of a manual search followed by a snowballing process.

3.1.1.1 Manual Search During a manual search, an investigator usually scans all the publications in specific sources such as proceedings or journals. First, we identified relevant sources from different domains such as computer science, philosophy, and psychology by consulting experts and relying on our own expertise. As a next step, the selected sources were independently reviewed for suitability by researchers of the specific domains.

Since we conducted an interdisciplinary SLR, sources from other disciplines were also considered during the manual search. In addition to computer science, the disciplines of philosophy and psychology were chosen because they have decades of experience in explanation research. Furthermore, during the snowballing process, other research areas were also taken into account. We believe that the choice of sources for our manual search is sufficiently representative to uncover research in the area of explainability.

The manual search was performed independently by the authors of this article and resulted in 104 papers. We used Fleiss' Kappa statistics [59] to assess the reliability of the selection process. The calculated value of $\kappa = 0.81$ showed an almost perfect agreement [60].

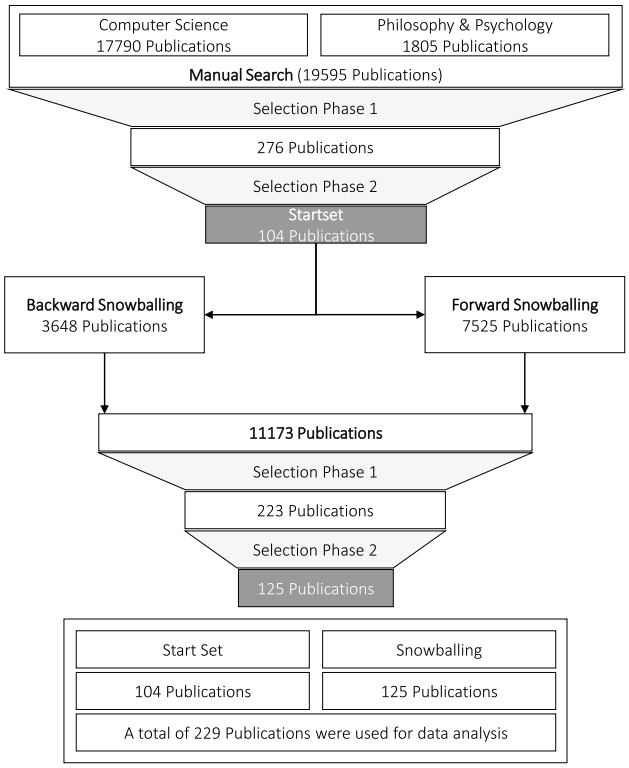

Fig. 2 Overview of the SLR

3.1.1.2 Snowballing After the manual search, we performed snowballing to complement the search results. The snowballing process includes backward and forward snowballing as described by Wohlin [58]. Our literature review process is partially based on a grounded theory (GT) approach for literature reviews proposed by Wolfswinkel et al. [61]. 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 GT.

According to [61], a literature review is never complete but at most saturated. This saturation is achieved when no new concepts or categories arise from the data (i.e., the publications that were inspected). We followed this approach to decide when to conclude our snowballing process. Therefore, we only performed one iteration, as we could not gain any new insights or concepts during a second iteration.

The snowballing was independently conducted by the authors, resulting in additional 125 papers. The calculated value of $\kappa = 0.87$ also shows an almost perfect agreement. Overall, our SLR yielded a total of 229 papers. A comprehensive overall summary of the number of papers inspected and selected in the different phases of the SLR is shown in Fig. 2.

3.1.1.3 Inclusion & Exclusion Criteria We included publications that met the following inclusion criteria (IC):

IC_1 Provide information that is relevant to answering (partially or completely) one of our research questions

IC_2 Were published between 01/1984 and 03/2020

IC_3 Are peer-reviewed journal, conference, and workshop publications

and we excluded publications that met the following exclusion criteria (EC):

EC_1 Are non English-language publications

EC_2 Are publications exploring or proposing rough algorithmic techniques without further discussion about the theoretical background of explainability

We chose 1984 as the starting date because that was the year in which the first major work on explainability was published (namely, [62]). Furthermore, we started the SLR on 03/2020.[7]

We are aware of the fact that 36 years is a long period of time. However, by choosing this time period, we wanted to get as broad an overview of the topic as possible. When it comes to explainability, it is useful to recognize that this topic was already important in the eighties and is not as new a research field as often believed [63].

To include a publication, all inclusion criteria must be met. If at least one of the exclusion criteria was met, the publication was rejected. Our selection process consisted of a two-phase selection procedure. In phase one, we have selected candidate papers based on title, abstract, and keywords. In cases where the aforementioned elements did not provide sufficient information, we have also analyzed the conclusion section. EC_2 did not apply in this phase. In phase two, we have selected papers based on full text and also applied EC_2.

3.1.2 Coding and analysis

We followed an open-coding approach [56] for the qualitative analysis of the papers we found during our search. This approach consists of up to three consecutive coding cycles. For our first coding cycle, we applied *Initial Coding* [64] to preserve the views and perspectives of the authors in the code. In the second coding cycle, we

[7] The fact that the SLR was not updated following the submission of the original conference paper constitutes a threat to validity, which will be addressed in Sect. 9.

Table 1 Structure of the two workshops to validate the data related to RQ1, RQ2, and RQ3

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.
Software Engineers	Select quality aspects based on provided scenarios and list of quality aspects.	1. Enter positive and negative impacts of explainability on other quality aspects.
	Suggest other quality aspects related to explainability.	2. Compare results with the findings from the SLR.
		3. Cluster the found quality aspects into groups.

clustered the initial codes based on similarities, using *Pattern Coding* [65]. This allowed us to group the data from the first coding cycle into categories. Next, we discussed these categories until we reached an agreement on whether they adequately reflected the meaning behind the codes. These categories allowed us to structure the data for better analysis and to identify similarities.

For RQ2 and RQ3, we conducted a third coding cycle to further classify the categories into quality aspects. We applied *Protocol Coding* [66] as a procedural coding method in this cycle. For this method, we used a pre-established list of NFRs from Chung et al. [21]. 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 [21] to the data, we discussed together and selected a quality aspect that would adequately describe the idea presented in the text fragment.

All coding and review processes were conducted independently by the authors of this article. In terms of review processes, this means that each of the authors independently read and analyzed the literature. The coding processes were also executed independently by each author. After each review and coding session, we discussed our results before proceeding to the next phase. We had regular consensus sessions to discuss discrepancies. A list of all codes is available in our supplementary material [67].

Finally, for RQ4, we conducted an additional round of data extraction to complement our previous insights. During this round, the focus was on the existing types of explanations, possible implementation strategies for explainability (including presentation forms), and methods used to measure the quality of explanations. The coding process followed the same procedure as mentioned above. However, this time, the protocol coding was supported by the taxonomy proposed by Speith [46] for the implementation strategies and the reviews by Vilone and Longo [68] as well as Zhou [69] for measuring.

3.2 Data validation

We held two workshops to validate and augment the knowledge gathered during data collection: one exclusively with philosophers and psychologists, and one exclusively with software engineers. The structure of the workshop is depicted in Table 1.

Each of the workshops lasted for four hours. To prepare for the discussions, we gave all participants of both workshops preparatory exercises to work on individually about a week before the workshop began.

In both workshops, we discussed the categories and other relevant information that were identified during our coding. For RQ1, the categories consisted of competing definitions of explainability that we extracted from the literature. For RQ2, the categories consisted in the identified quality aspects that have a relationship with explainability. Finally, for RQ3, we identified the kind of impact that explainability can have on each of the extracted quality aspects.

3.2.1 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 excepts for one doctoral candidate do research in the field of explanation, one professor and the postdoc even as a focus. Scholars in these disciplines have a long history in researching explanations and, thus, explainability. After consulting with experts from these disciplines on the workshop design, we decided on an open discussion.

The **preparatory exercise** of the philosophers and psychologists was to write down a definition of explainability, taking into account their own background knowledge. 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.

The workshop consisted of three activities. In the first activity, we presented the categories with respect to RQ1 found in the literature for discussion. We debated whether these categories accurately reflect the participants' perceptions on the meaning of explainability. In the second activity, the idea was to compare the definitions found in the literature with participants' own definition of explainability, submitted before the workshop. We compared the definitions, and also identified and discussed the differences in order to reach a consensus. During the last activity, we discussed interdependencies between explainability and other software quality aspects.

3.2.2 Workshop with software engineers

We validated the data related to RQ2 and RQ3 in a workshop with software engineers (three professors, two postdocs, one practitioner, one doctoral candidate). All three professors do research in the field of requirements engineering and two of them also in direct relation to explainability, as do the two postdoctoral researchers. The practitioner is a product owner in an international company, and the research field of the doctoral candidate is the interplay between requirements engineering and agile development. Two experts in the field of RE with experience in the topic of NFRs and software quality were consulted about the workshop design (depicted in Table 1).

For the **preparatory exercise**, we asked participants to list quality aspects that can be impacted by explainability. To support them in the task, we developed 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). We also welcomed participants' suggestions about further quality aspects that could be connected to explainability but were not present in our list.

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 better understand 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 to analyze how explainability would interact with each of these identified aspects (positively or negatively). The four hypothetical scenarios are described in our additional material [67].

This workshop also included three activities, each lasting for approximately an hour. 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 rigorous 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. Again, we compared the results and had an open discussion to discuss differences and reach consensus. Experts agreed on all polarities that they had not mentioned before but had been discovered in the literature. In the third activity, we clustered the quality aspects collaboratively based on their relationship and discussed their impacts on the system.

3.3 Knowledge structuring

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

3.3.1 Operationalizing explainability: definition

We operationalized the concept of explainability in a system context by distilling a definition of it. In the workshop with psychologists and philosophers, we integrated proposed definitions with ones that we found in the literature. The resulting definition contains several variables so as to be as flexible as possible to be adjusted to specific project contexts while at the same time providing a shared understanding of explainability among stakeholders.

3.3.2 Framing the results: conceptual model

We built a conceptual model to frame our knowledge catalogue. This model illustrates the impact of explainability on several quality dimensions (see Fig. 3; RQ2). During the workshop with software engineers, we discussed possible ways to classify the different quality aspects. Here, the participants offered useful ideas. To further supplement these ideas, we consulted the literature and found three promising ways to classify the results (more details in Sect. 5). These three ways are analogous to the suggestions made by the workshop participants and supported us in the development of our conceptual model.

3.3.3 Summarizing the results: knowledge catalogue

We summarized the results for RQ3 in a knowledge catalogue for explainability. Overall, we have extracted 57 quality aspects that might be influenced by explainability. We present these quality aspects and how they are influenced by explainability in Fig. 4. Additionally, we extracted a representative example from the literature for all positive and negative influences listed in our catalogue to show how this influence may come about. These examples also serve to illustrate our understanding of certain quality aspects.

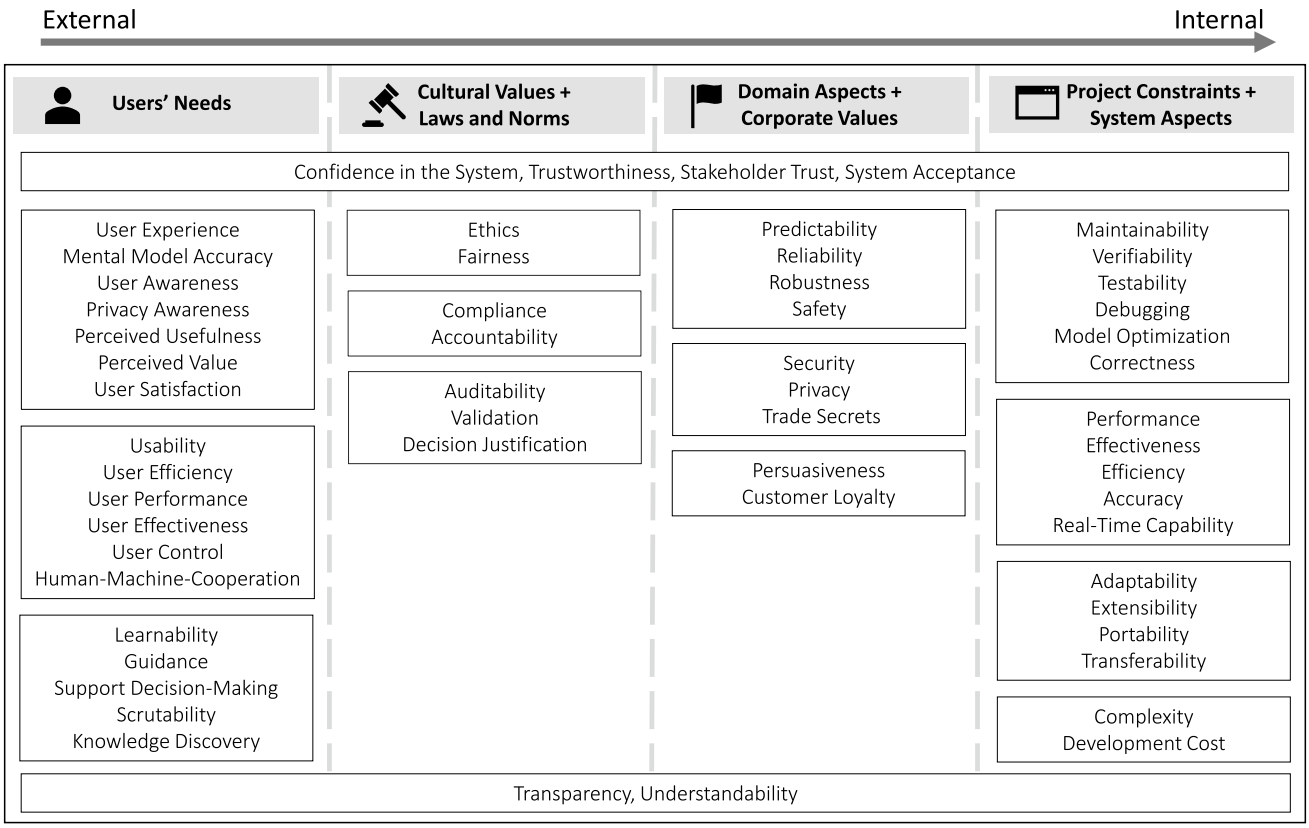

Fig. 3 A conceptual model illustrating the impact of explainability across different quality dimensions

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

Fig. 4 The knowledge catalogue for explainability: how explainability impacts other quality aspects

3.3.4 Making the results actionable: reference model

We have compiled and summarized the extracted information for RQ4 to conceive a reference model for explainability. The reference model encompasses three phases: requirements analysis, design, and evaluation. Each of these phases in the model include relevant aspects that should be considered in each phase in the software lifecycle.

We used our previous findings (RQ1–RQ3) and an additional round of data extraction to shape the phases in the reference model. In particular, the categories in the "requirements analysis" phase are based on variables of the definition we set out, and on the work of Chazette and Schneider [5]. The categories on the "design" phase that make up the implementation strategy are based on the work by Speith [46], and the categories on the "evaluation" phase are based on the findings of our SLR. Furthermore, we illustrate how our template can be applied by means of a running example.

We present the results for RQ1 in Sect. 4, the results for RQ2 and RQ3 in Sects. 5, 6, and the results for RQ4 in Sect. 7.

4 A definition of explainability

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

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 [9]. However, this leaves open what exactly is to be explained. In the literature, definitions of explainability vary considerably in this regard. Moreover, our review has revealed other aspects in which definitions of explainability differ. Consequently, there is not *one* definition of explainability, but several complementary ones.

Similarly, Köhl et al. also found that there is not just one type of explainability, but that a system may be explainable in one respect but not in another [9]. Based on their definition of explainability, the definitions we found in the literature, and results from our workshop with philosophers and psychologists, we were able to develop an abstract definition of explainability 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 important variables of an explainable system that are relevant for requirements and software engineers. These variables provide guidance on the elements that are important in an explainable system and, therefore, need to be considered during elicitation and design. In particular, an exemplary application of how our definition can support requirements engineering in practice can be found in Sect. 7.

There were differences in the literature concerning the values of the following variables presented in the above definition: *aspects* of a system that should be explained, *contexts* in which to explain, the *entity* that does the explaining (the *explainer*), and *addressees* that receive the explanation. Being aware of these differences is crucial for requirements engineers to elicit the right kind of explainability for a project as well as specifying the fitting requirements on explanations.

4.1 Aspects that should be explained

Concerning the aspects that should be explained, we found the following options in the literature and validated them during the workshop with philosophers and psychologists: the system in general (e.g., global aspects of a system) [70], and, more specifically, its reasoning processes (e.g., inference processes for certain problems) [71], its inner logic (e.g., relationships between the inputs and outputs) [9], its model's internals (e.g., parameters and data structures) [72], its intention (e.g., pursued outcome of actions) [73], its behavior (e.g., real-world actions) [74], its decision (e.g., underlying criteria) [4], its performance (e.g., predictive accuracy) [75], and its knowledge about the user or the world (e.g., user preferences) [74].

4.2 Contexts and explainers

A context is set by a situation consisting of the interaction between a person, a system, a task, and an environment [76]. Plausible influences on the context are time-pressure, the stakes involved, and the type of system [54].

Explainers refer to a system or specific parts of a system that supply its stakeholders with the needed information. Semantically speaking, our definition allows that these specific parts of the system do not necessarily have to be technical components (such as algorithms or even hardware elements) of the system itself. 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 [54]. Although the person applying the definition should be the one to decide where the boundaries of an explainable system should be set, in the context of our work, we focus on self-explainable systems: systems that explain themselves "directly" to an end user.

Consider the following example. A patient (addressee) is in a hospital and has been examined by a physician (context) using a medical diagnosis system. The medical findings are processed by the system (aspects) and presented directly to the patient in an electronic dashboard. However, these findings cannot be interpreted and understood by patients directly because they do not have 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 the proposed 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 we focus on self-explainable systems, the system in the example above is only deemed explainable in our perspective if the physician is the intended addressee for explanations. If the patients are the intended addressees, the system would be considered explainable if the system explains itself to the patient directly (since the patient is the end user) and comprehensively 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, we 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 system must be explainable to physicians.

4.3 Addressee's understanding

A vast number of papers in the literature make reference to the addressee's evoked understanding as important factor for the success of explainability (e.g., [14, 51, 70, 77, 78]). 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 [5].

The variables in our definition will become important later on, when we discuss our reference model (Sect. 7). In this template, these variables essentially constitute different aspects that need to be elicited in a project context before concrete implementation strategies are devised. Accordingly, we will return to the variables later on, using an example to help us understand them better.

5 A conceptual model of explainability

Conceptual models and catalogues compile knowledge about quality aspects and help to better visualize their possible impact on a system. Based on the data extracted from the literature and on our qualitative data analysis and validation, we were able to build a conceptual model and a knowledge catalogue for explainability.

In this section, we will first discuss our conceptual model. Overall, this model serves as a kind of classification scheme and is divided into four so-called *quality dimensions*. A quality dimension is a conceptual layer that groups quality aspects that make up a system. It represents a perspective from which to consider the quality of the system.

Answering RQ2 We framed the quality aspects that are impacted by explainability in a conceptual model that spans different quality dimensions of a system (see Fig. 3).

We considered three existing concepts to shape and compose the conceptual model: stakeholder classes, elicitation dimensions, and quality spectrum. These concepts help us illustrate our vision of system quality and how it is impacted by explainability.

5.1 Stakeholder classes

Langer et al. categorize quality aspects that are influenced by explainability according to so-called *stakeholder classes* and distinguish the following ones: *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 different stakeholder classes may conflict [54].

This first concept is related to our definition and based on the insight that understanding is pivotal for explainability. Individuals differ in their background-knowledge, values, experiences, and many further respects. For this reason, they also differ in what is required for them to understand certain aspects of a system.

Furthermore, Langer and colleagues also hold that some persons are more likely to be interested in a certain quality aspect than others [54]. For instance, a developer might be more interested in the maintainability of a system than a user.

Against this background, using stakeholder classes to organize quality aspects seems promising. Since it is a

stakeholder who needs to understand a system for it to be explainable according to our definition, the stakeholder class provides a frame of reference for software engineers.

5.2 Elicitation dimensions

Chazette and Schneider identified six dimensions that affect the elicitation and analysis of explainability [5]: the *users' needs and expectations*, *cultural values*, *corporate values*, *laws and norms*, *domain aspects*, and *project constraints*. Their results indicate that different factors distributed across these dimensions influence the identification of explainability as being a necessary quality aspect within a system, as well as the design choices towards its operationalization. In other words, these dimensions influence the (explainability) requirements of a system.[8]

The *users' needs and expectations* for example, is a dimension that considers the user and will thus reflect directly on the end-user requirements. *Cultural values* refer to the *ethos* of a group or society [79], and how culture influence the system design [80]. *Corporate values* refer to the strategic vision and values of an organization [81], and how they shape software systems. *Laws and norms* concern the regulatory and legal influence on the requirements and design of a system. *Domain aspects* consider the subject area on which the system is intended to be applied, and will dictate the logic around the application [82]. The *project constraints* are more practical aspects (also known as *non-technical aspects* [83]), such as available resources (e.g., time, money, technologies, manpower).

5.3 Quality spectrum

The external/internal quality concept based on the ISO 25010 [37] and proposed by Freeman and Pryce [84] is the final concept that we use to shape our model. In particular, we use it to categorize the quality dimensions (and thus the quality aspects within them) themselves.

The external/internal quality concept is the basis for what we call the *quality spectrum*. Since a spectrum is "a range of different positions between two extreme points" [85], we consider the system quality spectrum as a range between the two extreme points: internal and external quality. In our model, the quality dimensions represent "positions" or "directions" in the quality spectrum.

In this sense, an *external* quality dimension is more related to the users or the *quality in use*, and an *internal* quality dimension is more related to the developers or the *system itself*. The same applies to the quality aspects inside these dimensions.

However, as pointed out by McConnel [86], the difference between internal and external quality is not completely clear-cut, meaning that a quality aspect can belong or affect several dimensions. Therefore, we do not assign the dimensions and quality aspects of our conceptual model as clear-cut internal or external, but rather acknowledge a continuous shift from external to internal.

5.4 Compiling the concepts

Based on these three concepts and input from our workshop with software engineers, we developed a conceptual model for the impact of explainability on other quality aspects. To this end, we combined the concepts of stakeholder classes and elicitation dimensions to form four new dimensions, into which we sort the quality aspects that are impacted by explainability: *user's needs*, *cultural values and laws and norms*, *domain aspects and corporate values*, and *project constraints and system aspects*. Furthermore, we arrange these quality dimensions in the quality spectrum.

We also identified quality aspects that are present in all dimensions. In particular, we identified quality aspects that form a foundation for the four dimensions (e.g., transparency). Without the influence of explainability on these foundational qualities, many other quality aspects would not be influenced. Furthermore, we identified superodinated qualities. These quality aspects are influenced by all other aspects, sometimes being described as the *goals* of explainability (e.g., trust).

More details on the individual dimensions will be given in the next section, when we discuss the catalogue, since the dimensions are closely linked to the quality aspects they frame. The dimensions and their respective quality aspects are illustrated in Fig. 3. In the figure, the quality aspects are grouped according to similarity, based on our workshops' results. Furthermore, the listing order corresponds to that of the descriptive text in Sect. 6. Overall, our conceptual model should support software engineers in understanding how explainability can affect a system, facilitating requirements analysis.

6 A catalogue of explainability's impacts

In this section, we present the catalogue and discuss the quality aspects in relation to our conceptual model. To this end, we analyze them, whenever possible, based on the three categorizations we have described above: the stakeholders involved, the dimensions that affect the elicitation and analysis of explainability, and the external/internal categorization.

[8] We adopt and extend this notion in that we consider these dimensions to be decisive not only for the RE process, for example, by influencing whether a requirements engineer considers a quality aspect as relevant because of existing laws, but also for a system in general, by influencing the quality of the system from a given perspective.

Answering RQ3 We built a catalogue that lists all quality aspects found in our study and the kind of impact that explainability has on each one of these aspects (Fig. 4).

6.1 Foundational qualities

Explainability can influence two quality aspects that have a crucial role: **transparency** and **understandability**. These quality aspects provide a foundation for all four dimensions, thereby having an influence on the other aspects inside these dimensions (and, in some cases, vice versa).

Receiving explanations about a system, its processes and outputs can facilitate understanding on many levels [87]. Furthermore, explanations contribute to a higher system transparency [88]. For instance, understandability and transparency are required on a more external dimension so that users understand the outputs of a system (e.g., an explanation about a route change), which may positively impact user experience. They are also important on a more internal dimension, where they can contribute to understanding aspects of the code, facilitating debugging and maintainability.

6.2 User's needs

Most papers concerning stakeholders in Explainable Artificial Intelligence (XAI) state *users* as a common class of stakeholders (e.g., [51, 89, 90]). This, in turn, also coincides with the view from requirements engineering, where (end) users also count as a common class of stakeholders [91]. Among others, users take into account recommendations of software systems to make decisions [13]. Members of this stakeholder class can be physicians, loan officers, judges, or hiring managers. Usually, users are not knowledgeable about the technical details and the functioning of the systems they use [54].

When explainability is "integrated" into a system, different groups of users will certainly have different expectations, experiences, personal values, preferences, and needs. Such aspects mean that individuals can perceive quality differently. At the same time, explainability influences aspects that are extremely important from a user perspective.

The quality aspects we have associated with users are mostly external. In other words, they are not qualities that depend solely on the system. To be more precise, they depend 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 [92] and make it more engaging [93]. In this case, user experience profits from explainability. On the other side, explanations can cause emotions such as confusion, surprise [94], and distraction [78], 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 [75], helping them to develop better mental models of it [94]. 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 [75]. This can be attributed to an improved **user awareness** about a situation or about the system [12]. 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** [15, 51]. Explainability can also positively impact the **perceived usefulness** of a system or a recommendation [95], which contributes to the **perceived value** of a system, increasing users' perception of a system's competence [96] and integrity [97] and leading to more positive attitudes towards the system [98]. Finally, all of this demonstrates that explainability can positively impact the **user satisfaction** with a system [94].

Explainability can also influence the **usability** of a system. On the positive side, explanations can increase the ease of use of a system [47], lead to more efficient use [12], and make it easier for users to find what they want [99]. On the negative side, explanations can overwhelm users with excessive information [100] and can also impair the user interface design [5]. Explanations can help to improve **user performance** on problem solving and other tasks [97]. Another plausible positive impact of explainability is on **user effectiveness** [101]. With explanations, users may experience greater accuracy in decision-making by understanding more about a recommended option or product [102]. However, user effectiveness can also suffer when explanations lead users to agree with incorrect system suggestions [10]. **User efficiency** is another quality aspect that can be positively and negatively influenced by explainability. Analyzing and understanding explanation takes time and effort [103], possibly reducing user efficiency. Overall, however, the time needed to make a judgment could also be reduced with complementary information [101], 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 [14]. Explainability can also have a positive influence on **human-machine cooperation** [77] since explanations may provide a more effective interface for humans [104], improving interactivity and cooperation [33], 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 [102]. It may also provide **guidance**, helping users in solving problems and educating them about product

knowledge [105]. As these examples illustrate, explanations can **support decision-making** processes for users [47]. 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 [47]. Finally, explainability can help **knowledge discovery** [14]. 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 [75].

6.3 Cultural values and laws and norms

Although [5] distinguished *Cultural Values* and *Laws and Norms* as two separate dimensions and [54] did the same for regulators and affected parties, we have combined them 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. We adopt the same approach for the dimensions discussed in Sects. 6.4, 6.5.

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 [54]. 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 [54]. Affected parties are (groups of) people in the scope of a system's impact. They are stakeholders, as for them much depends on the decision of a system. Patients, job or loan applicants, or defendants at court are typical examples of this stakeholder class [54].

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 [79, 80]. These values resonate in the conception of laws and norms, which enforce constraints that must be met and guaranteed in the design of systems. Explainability can influence key aspects on this dimension.

With regard to the internal/external distinction, a clear attribution is not possible. Rather, the quality aspects seem to occupy a hybrid position. Whether or not they are present does not only depend on the system itself, but it also does not depend on a person using them. Rather, it depends on general conventions (e.g., legal, societal) that are in place. For this reason, we take them to be more internal than the quality aspects from the last dimensions: general conventions are better implementable than individual preferences.

On the cultural side, explanations can contribute to the achievement of **ethical** decision-making [106] and, more specifically, ethical AI. On the one hand, explaining the agent's choice may support ensuring that ethical decisions are made [14]. On the other hand, providing explanations can be seen as an ethical aspect itself [6]. Furthermore, explainability may also contribute to **fairness**, enabling the identification of harms and decision biases to ensure fair decision-making [14], or helping to mitigate decision biases [75].

On the legal side, explainability can promote a system's **compliance** with regulatory and policy goals [107]. Explaining an agent's choice can ensure that legal decisions are made [14]. 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 [108]. In the literature, many authors refer to this as *liability* [108] or *legal accountability* [109].

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 NFR that can help in this regard is **auditability**. Explainability positively impacts this NFR, since explanations can help to identify whether a system made a mistake [10], can help to understand the underlying technicalities and models [73], and allow users to inspect a system's inner workings to judge whether it is acceptable or not [110]. In a similar manner, **validation** can be positively impacted, since explainability makes it possible for users to validate a system's knowledge [102] or assess if a recommended alternative is truly adequate for them [47]. 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 [108]. On the other hand, they can also help to uncover whether a decision is actually justified [4].

6.4 Domain aspects and 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 a domain, known as domain experts. People have to work with the deployed systems and, consequently, new people fall inside the range of affected people [54].

This dimension is shaped by two aspects: (1) the corporate values and vision of an organization [81], and (2) the domain aspects that shape a system's design since

explanations may be more urgent in some domains than in others.

We consider this dimension as more internal to the system, since it encompasses quality aspects that are more related to the domain or the values of the corporation or the team. Generally, the integration of such aspects affects the design of a system on an architectural level. However, there are some exceptions, as the organization's vision may aim at external factors like customer loyalty.

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 [111]. Furthermore, explainability can support the **reliability** of a system [70]. In general, explainability supports the development of more **robust** systems for critical domains [112]. All of this contributes to a positive impact on **safety**, helping to meet safety standards [14], or helping to create safer systems [113]. 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 [71], 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 [3]. 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 [15, 114]. By the same principle, however, privacy can be hurt since one may need to disclose sensitive information that could jeopardize privacy [12]. Explainability can also threaten model confidentiality and **trade secrets**, which companies are reluctant to reveal [51].

Explainability can contribute to **persuasiveness**, since explanations may increase the acceptance of a system's decisions and the probability that users adopt its recommendations [47]. Furthermore, explainability influences **customer loyalty** positively, since it supports the continuity of use [92] and may inspire feelings of loyalty towards the system [99].

6.5 Project constraints and system aspects

Individuals who design, build, and program systems are, among others, developers, quality engineers, and software architects. They count as stakeholders [115], 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 [83], 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 internal 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 [102] or to help users evaluate the accuracy of a system's prediction [116]. **Testability** falls in the same line, since explanations can help to evaluate or test a system or a model [14]. Explainability has a positive influence on **debugging**, as explanations can help developers to identify and fix bugs [4]. Specifically, in the case of machine learning (ML) applications, this could enable developers to identify and fix biases in the learned model and, thus, **model optimization** is positively affected [50]. 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 [108].

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 [77]. In this regard, explainability positively influences system **effectiveness**. On the other hand, however, explanations can also lead to drawbacks in terms of performance [103] by requiring loading time, memory, and computational cost [5]. Thus, as the additional explainability capacities are likely to require computational resources, the **efficiency** of the system might decrease [4]. 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 [50]. On the negative side, there exists a trade-off between the predictive accuracy of a model and explainability [4]. A system that is inherently explainable, for instance, may have to sacrifice predictive power in order to be so [72]. 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 preserve 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 system is to be ported to has no elements that allow for visual outputs. On the positive side, explainability helps **transferability** [117]. 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 [117].

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** [9].

6.6 Superordinated qualities

We were able to identify some aspects that hold regardless of dimension. These aspects 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 [51]. Overall, many of the quality aspects we could find 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, **confidence** and **trust** in a system originate solely from trustworthy systems. Although one could trust an untrustworthy system, this trust would be unjustified and inadequate [118]. For this reason, explainability can both contribute to and hurt trust or confidence in a system [12, 71]. Regardless of the system's actual trustworthiness, bad explanations can always degrade trust [71]. Finally, all of this can influence the system's **acceptance**. A system that is trustworthy can gain acceptance [74] and explainability is key to this.

7 A reference model for explainability

Building on the previous artifacts, we propose a reference model for explainability. Reference models can support the design and implementation of software systems, making it easier to understand primordial factors to the conception and design of these systems [119].

Building on this idea, we propose a reference model that provide a frame of reference of the main factors and relevant points that should be considered when defining explainability from requirements analysis (e.g., eliciting explainability requirements) to the design phase (i.e., operationalization of the elicited requirements) and evaluation (i.e., measuring if the requirements are achieved). In light of this, we answer RQ4 as follows:

> *Answering RQ4* We propose a *reference model* for explainable systems (Fig. 5) based on the findings from our SLR. This reference model includes relevant factors that should be considered for the development of explainable systems at various stages of the software lifecycle, assisting software engineers in the analysis, operationalization, and evaluation of requirements for explainable systems.

7.1 Constituents of the reference model

Our reference model draws on ideas from three sources. The first source is a stepwise approach from abstract quality notions to concretely measurable requirements proposed by Schneider [120]. This stepwise approach considers three levels of abstraction: abstract goals, concrete characteristics, and measures or indicators [120, 121]. The abstract goals[9] correspond to the objectives and constraints for a system; the concrete characteristics define the design decisions for the abstract goals; and the measures or indicators help evaluate whether the abstract goals have been achieved.

Based on this approach, we create the structure of a reference model that goes from the abstract to the concrete: from abstract aspects of the "real-world" that are relevant for requirements analysis and must be translated into requirements, to factors that influence concrete design decisions, to evaluation strategies. We have transformed these levels of abstraction into phases of the software lifecycle to make them more practical and to provide software engineers with guidance on what to consider in each phase. In our model, these levels correspond to *requirements analysis*, *design*, and *evaluation*, respectively.

During *requirements analysis*, software engineers, in conjunction with relevant stakeholders, elicit and define explainability requirements. Such requirements are not only the quality aspects that the system should have, but may also include an initial overview of the system, a vision of the system or a part of it, a list of key features, constraints, etc. During the *design* phase, the requirements should be refined into tangible system solutions or design choices, based on the

[9] Abstract goals can also be understood as so-called "high-level" or "raw" requirements [122], i.e., requirements that are not yet very specific and concrete, but express preliminary ideas, concepts, or visions that need to be further refined.

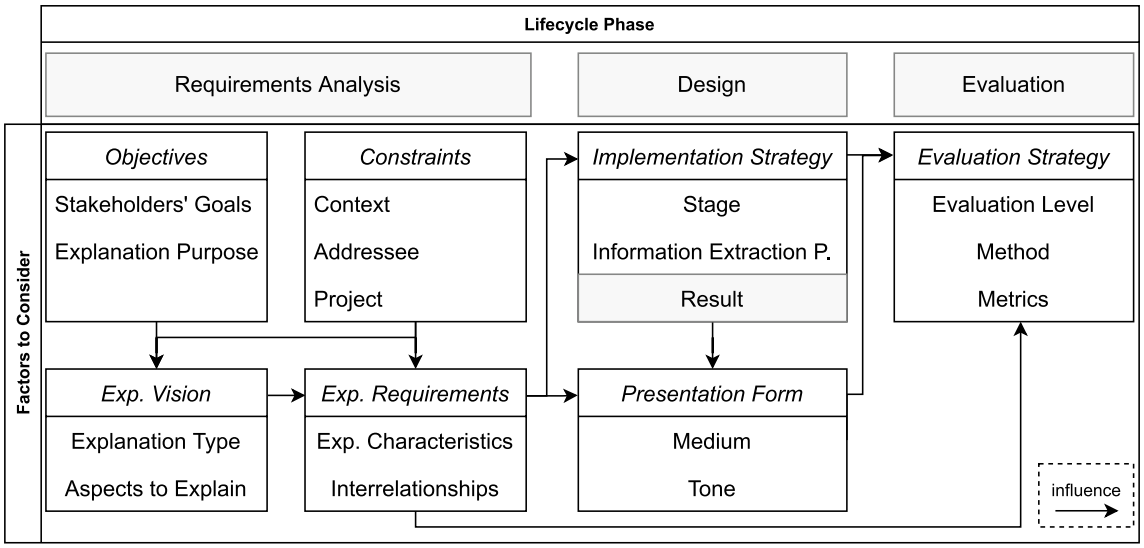

Fig. 5 A reference model to support the development of explainable systems

factors considering during requirements analysis. Finally, during *evaluation*, quantitative measurements or qualitative indicators should be specified for each concrete solution as a method to subsequently analyze whether a requirement was met.

The work of Chazette et al. [123] is the second source of ideas upon which our reference model is based. Chazette et al. propose six core activities and associated practices for the development of explainable systems. These activities include (1) establishing a system vision (also with respect to explainability), (2) assessing the needs of the involved stakeholders, deciding how to implement explainability, (3) thinking about the algorithm to be explained, (4) weighing the trade-offs between explainability and other quality aspects, (5) making design decisions that favor explainability, and (6) assessing the impact of explainability on the system.

We also consider ideas from the work of Speith [46] to put together our reference model. Speith presents an overview and taxonomy of various approaches that can be used to translate the more abstract considerations about explainability into concrete implementations. Although his work is primarily intended for AI systems, we will adopt and generalize it so that it is applicable to all kind of software systems.

We combine ideas found in these three works with the results of our study. In summary, our reference model proposal is based on (1) the three levels of abstraction adapted to phases of the software lifecycle [120], (2) the six core activities for the development of explainable systems [123], (3) the approaches used to translate abstract considerations into concrete implementations [46], and (4) our research findings. The reference model is illustrated in Fig. 5 and presents the critical elements to take into account when developing explainable systems, broken down into seven categories that cover different phases of the software lifecycle.

A simple hypothetical scenario will serve to illustrate each phase of the reference model and is intended to facilitate and guide the discussion:

Scenario: A car manufacturer produces fully autonomous vehicles. The manufacturer wants passengers to better understand the system's navigation decisions. In conjunction with stakeholders, it was decided that explanations before and during navigation should help reach this goal, also aiming to improve user experience.

Explanations might be shown on the screen of the board computer or provided acoustically. To keep the passenger informed and aware of the various navigation possibilities, it is critical to provide a full review of the available route options and the reasoning behind each recommendation prior to navigation start. During navigation, passengers should be informed of any route changes, accidents, threats, or delays in a timely manner.

7.2 Requirements analysis

Requirements analysis is "the analysis of elicited requirements in order to understand and document them" [122]. Understanding and documenting the needs of stakeholders is one of the primary goals of RE, and requirements analysis is essential in this process.

The definition proposed in Sect. 4 offers an initial overview of some of the essential factors that should be considered and elicited during requirements analysis. According

to the suggested definition, gathering knowledge about the context of use, the aspect of the system to be explained, and the addressee (entity or person who receives the explanation) is pivotal in order to provide meaningful explanations.

Other factors are related to the impact of explainability across different quality dimensions, as addressed in Sect. 5. These dimensions influence the requirements. In particular, these dimensions have an influence on whether explainability is identified as an important quality aspect in a system, which objectives are associated with it (e.g., more transparency or usability in a system), and what constraints there are (e.g., project deadlines, financial limitations).

The objectives and constraints serve as the basis for higher-level decisions, such as defining an *explainability vision*. The explainability vision is a high-level definition of explainability and includes the consideration of the type of explanation to be presented and the important aspects to explain. Furthermore, all of these considerations (objectives, constraints, explainability vision) culminate in the explainability requirements.

7.2.1 Objectives

The *objectives* of explainability are the first items to be defined. They depend on the *stakeholders' goals* and on the *reasons and purposes* that motivate their need for explainability in a system. These goals, reasons, and purposes, in turn, may be influenced by any of the quality dimensions mentioned earlier (e.g., cultural values, domain aspects, etc.).

As we have explored in the previous sections, explainability can be seen as an *enabler*, as a means of achieving other crucial quality aspects. Thus, one must describe which quality aspects are to be achieved through explainability. In principle, these quality aspects can come from any of the dimensions presented in Fig. 3.

Perhaps the stakeholders in a particular project need an explainable system because explainability contributes to a better user experience, or because they see the need of developing a system that is aligned with ethical values. Overall, all of the 57 quality aspects listed in our model (Fig. 3), in any combination, can function as an objective.

7.2.2 Constraints

The *constraints* are factors that influence or limit design decisions in some way. Constraints are usually associated with factors such as the *context*, the *addressee*, and the *project* circumstances. These factors have a major influence on the system design. As mentioned in Sect. 4, the context is an interaction between a *person*, a *system*, a *task*, and an *environment*.

For instance, the context of a physician (a *person*) interacting with an explainable cancer diagnosis system (the *system*) that supports the interpretation of image exams of cancer tumors (*task*) during a medical appointment in a hospital (*environment*) will directly influence the requirements on explanations.

In particular, the characteristics of the physician in their role as addressee play a significant role, along with the task they perform with the system and the environment in which the system is used. All of these influences have to be taken into account to define a vision of the system to be developed, refine it into requirements, and to prepare the field for subsequent concretizations, since constraints will significantly influence how concrete design decisions are shaped and realized.

7.2.3 Explainability vision

One of the high-level design decisions during requirements analysis is the *type of explanation* that is required. This decision is crucial when it comes to explainability, as choosing the appropriate kind of explanation is the precondition for conveying information in an appropriate way. In general, the decision on the type of explanation will depend on the objectives and constraints that were defined.

It is important to note that these abstracts thoughts about explanations do not necessarily define their implementation. When thinking about the design of a system (e.g., the medical system mentioned above), one is not directly concerned with *how* an explanation is implemented. First, one considers it at a higher abstraction level. This is because the type of explanation one wants is likely to draw on one's everyday behavior and explanation practices. Accordingly, such types of explanation are very abstract and distinct from concrete implementation and design ideas.

Miller et al. [124] found that implementations of explainability are often based on intuitions rather than on informed deliberations. For this reason, it is crucial to have an idea of explanation types, as well as plausible ways to implement them. As we discussed in previous sections, philosophy and psychology have long investigated explanations, offering significant knowledge about the many sorts of explanations and their usefulness for various contexts and situations. We can use this existing knowledge to gain valuable insights into how to design useful explanations in RE contexts.

With this in mind, Miller examined the philosophical and psychological literature to gain knowledge about explainability and found that the various forms of explanation each have their own advantages and disadvantages [78]. These advantages and disadvantages must be weighed in order to determine what type of explanation is suitable for each case in order to achieve the intended objectives. We analyzed the

data from our literature review to learn about the many types of explanations that exist and to provide important insights for software engineers.

Explanation Types and Aspects to Explain Explanations are commonly described as answers to "w"-questions (e.g., why, what, when, but also how) [125, 126]. When it comes to more fine-grained details, however, there is significant disagreement. There are many different types of explanations, each with its own advantages and drawbacks. While an extensive description of explanation types goes beyond the scope of this article, we will shortly discuss some prominent ones.

One prominent type of explanation is *causal explanation* [127, 128]. As the name implies, causal explanations commonly cite the causes of a phenomenon to explain it. Closely linked to causal explanations are *counterfactual explanations* [129]. This type of explanation uses counterfactual scenarios (i.e., what-if-things-had-been-different scenarios) to explain a phenomenon. For instance, if the window were opened, the letter would not have been swept away by a gust of air. The link to causal explanation can be easily seen: in the example above, the gust of air blowing is the cause for the letter being swept away.

There are many types of causal explanations, and in recent years, *mechanistic explanation* gained prominence. These explanations aim at explaining a phenomenon by explaining the mechanism constitutive of the phenomenon coming to be [130]. For instance, a mechanistic explanation of the heart pumping blood would involve the parts of heart (e.g., atria, ventricles, valves, etc.), the operations performed by these parts (e.g., contraction and relaxation by the chambers), and their organization (e.g., blood flows from each atrium to valve and ultimately into the circulatory system) [131].

Another type of explanation are *reason* explanations [71, 125]. Reason explanations explain the action of an agent by referring to the subjective reasons this agent has had for executing a said action. Note the difference to causal explanation: while causal explanations try to uncover the physical processes directly leading to a phenomenon, reason explanation make use of the reasons that motivated an agent to execute a specific action.

In addition to the type of explanation, the explainability vision also includes the *aspects to explain*. In order to properly integrate explainability, it is critical to understand which *aspects* or components of the particular system must be explained. For this reason, the aspect that should be explained is part of our definition of explainability (see Sect. 4.1). Some exemplary aspects are: a system's rationale, its predictive accuracy, or its intention.

Note that the concept of aspects to explain is a more fine-grained version of what Speith [46] calls "scope" in his taxonomy. Scope concerns whether the whole AI model should be explained (i.e., a *global* scope) or just a single prediction (i.e., a *local* scope). When it comes to the aspects we found, the aspect "system in general" corresponds to a global scope and the aspects "a system's decision" to a local one. Accordingly, the other aspects we found expand the work of Speith [46].

Explainability Requirements To define requirements, it is necessary to compare the goals of the various stakeholders and understand how these goals translate to what is expected from the system. Explainability requirements comprise all of the investigated components (i.e., objectives, constraints, explainability vision). To define requirements for explainability, it is, thus, important to operationalize these components in order to establish *explanation characteristics* (i.e., the requirements on explanations).

Requirements typically begin as high-level requirements that should be operationalized further until they reach the level of functional requirements, which express specific design choices. For example, the need for more security in a system (a high-level requirement) is refined into the need for more security in login, which is refined into the need for a two-factor authentication procedure. The need for a two-factor authentication procedure, in turn, translates into requirements that define the functional processes needed to achieve the high-level requirement of more security.

It is also essential to analyze what are the existing *interrelationships* between requirements in the system and how defined requirements affect and interact with other (existing or planned) requirements in the system. Therefore, a trade-off analysis should take place to help prioritize requirements according to factors such as the stakeholders' goals and observed constraints. During trade-off analysis, our proposed conceptual model and knowledge catalogue can assist in considering and identifying potential trade-offs.

Contextualizing: In our scenario, two quality aspects are pursued: *understandability* and *user experience*.[10] These quality aspects are seen as objectives during requirements analysis.

The objective of achieving better understandability can be achieved through built-in explanations that help understand route choices. This goal is motivated by an upcoming legal regulations in autonomous driving that require end users to be informed about system decisions. The goal of a better user experience, in contrast, is motivated by corporate interests.

The addressees are passengers in an autonomous car. In this scenario, timing is an important contextual factor, and,

[10] It should be noted that this is only an exemplary excerpt for didactic purposes and considers *understandability* and *user experience* as the only (quality) objectives.

therefore, a *constraint* tied to the explanations: before navigation, passengers may have more time to make a decision concerning a route recommendation, than during navigation. Consequently, timeliness is a crucial factor during navigation, as explanations need to be displayed at the right time and be understood quickly, so that the addressee can use them to make a decision.

After defining the objectives and constraints, the *explanation type* can be determined. In our autonomous vehicle scenario, causal explanations are likely the adequate explanation type. Causal explanations are useful for describing causes of occurrences like traffic jams or for clarifying the reasons for a delay (i.e., *aspects to explain*). Those help to define the *explanation characteristics* that should be described in the requirements.

7.3 Design

In the design phase, all goals, objectives, constraints, preliminary thoughts, and established requirements for explainability come together and influence the concrete design choices. In this phase, the factors defined in the previous phase must be linked to a concrete implementation strategy.

According to Speith's taxonomy, one must decide on the *scope* (the *aspects to explain* discussed earlier), *stage*, *information extraction procedure*[11], *result*, and *presentation form* when implementing explainability. We call the choice of these factors together the *implementation strategy*.

7.3.1 Implementation strategy

The *implementation strategy* refers to how explainability will be implemented into the system. This includes functions, modules (in terms of algorithmic solutions), and interface elements that should be implemented in the system to provide explanations. Functions can vary in complexity depending on what should be explained and on the complexity of the underlying model. Modules work as an additional part of the system or a separate entity that can act as an explainer (e.g., a virtual assistant). Finally, interface factors are more related to design choices concerning how the explanations are going to be presented in the interface.

Stage This factor concerns the question of *when* explainability is incorporated into a system. When working on a project where explainability is a quality goal, one can either try to "explain" an already existing system by implementing and adding appropriate mechanisms (this would be the stage *post-hoc*) [113, 125], or one could start from scratch, designing an inherently explainable system (this would be the stage *ante-hoc*) [50, 51].

Here, a significant distinction between systems based on "traditional" software [12] and those powered by AI must be made. In many cases, to make AI-based systems explainable, an entirely new system must be added to explain the old one. This illustrates why "black box" is a fitting metaphor for these systems: once trained, they cannot be adjusted easily. Making a traditional software system explainable, however, can usually be done by simply adapting its code.

Information extraction procedure This factor describes how the information needed for creating the explanations is derived. While the information extraction procedure is of particular interest for AI-based systems (because the information must often be derived by an additional module), it also has some applications for traditional systems, as we explore below.

One big distinction when it comes to the information extraction procedure is whether it is necessary to have access to the system code to obtain the information required to explain it. If one has access to the system, one could analyze the code and interpret the algorithmic rationale to obtain internal information that can be used to construct explanations [102, 113].

In fact, however, it is not necessary to have access to the system code; one can also gather the data required to create explanations through external analyses. An external analysis [48, 111] could be based on perturbing the input to see how the output changes. This information extraction procedure is called (local) perturbation [4, 77].

Result The extracted information (obtained through the information extraction procedure) commonly has a specific semantic, called the *result* [46]. Carefully selecting the semantics of an explanation may prove essential, depending on the addressee of the explanation and the context in which it is produced. This is the case because the semantics of an explanation often influence who can understand it.

There are different types of results. For instance, the extracted information could indicate the relevance of certain input features for the output. This result is called *feature relevance* [14, 50]. Another type of results are *examples* [49, 94]. Examples could be representative instances of similar decisions.

The result can be used as a link to the explanation type. For instance, highlighting a particular feature can be used to implement counterfactual explanations, and causal explanations could be conveyed by offering instances (i.e., examples).

[11] Originally, Speith calls this factor *functioning* [46]. However, since functioning is an overloaded term in the context of software systems, we will not use it here.

[12] We use the term "traditional" to refer to software or systems that are neither ML or AI-based.

7.3.2 Presentation form

Another factor to consider during the design phase is the presentation form of the explanation. Explanations can be represented in many forms, for instance, textually, numerically, or visually. The result and presentation form are intrinsically related: While the result dictates the semantic of the given explanation, the presentation form constitutes its representation and delivery. With the concept of a *representation format*, we generalize the notion of output format in Speith's taxonomy, enriching it by the concept of *tone*.

It is essential to determine the *medium* of the explanation (this is what Speith originally calls *output format* [46]). Primary mediums are textual [113, 125], visual [47, 132], and auditory [48, 133]. Each of these formats have further sub-classes. For instance, textual formats can be (instructions) in natural language [47, 102], but they can also be in the form of rules (e.g., if-else statements) [105, 134]. The latter format, which is especially helpful for those trying to debug systems, was primarily employed in the 1980s when it came to expert system explanations [133]. Visual formats include icons, (heat)maps, and videos. Finally, auditory formats might be alert tones, but also (synthesized) speech instructions.

In addition, each of these presentations forms also have their own particularities (or requirements), such as the **tone** that will be used in communication (formal vs. casual, technical vs. non-technical), length, volume, brightness, and so on. The choice of an appropriate presentation form can be crucial. For example, auditory explanations of decisions made by a navigation algorithm in a non-autonomous vehicle would be preferable to textual explanations for safety reasons.

Contextualizing: The goals, objectives, and constraints defined during the requirements analysis have to be refined into tangible technical strategies or design choices.

A list of possible aspects to be explained was given in Sect. 4. In our hypothetical scenario, the aspect that must be explained is the system's rationale for route calculation with a focus on the variables that were considered for route choice: arrival time and traffic condition.

Furthermore, an explanation module must be added to the already existing system to provide the explanations. Thus, the stage would be a post-hoc. The need for such a module arises because the system in our hypothetical scenario is supported by a deep neural network (DNN). Consequently, the module's information extraction procedure has be compatible with DNNs (e.g., TCAV [135]) or it needs to be applicable to all types of AI models (e.g., LIME [77]).

The explanations should be presented with interface-specific design strategies such as auditory explanations, icons, and animations (e.g., to highlight a new event such as an accident) on the board computer's screen and possibly with brief texts. Furthermore, the communication tone should be casual rather than formal.

7.4 Evaluation

Evaluation bridges the gap between a customer's overall goals on the one hand and the metrics agreed upon for measurement on the other [120]. Accordingly, to analyze the influence of explanations, evaluation methods and corresponding metrics[13] should be defined. In particular, metrics should help find out whether the chosen technical solutions contribute to meeting the defined requirements and help to assess whether the chosen implementation of explainability is adequate or needs to be improved [120].

7.4.1 Evaluation levels for explainability

When it comes to the evaluation of explainability, an important distinction must be made. We have emphasized that explainability is a kind of enabler, and can contribute to achieving other quality aspects. Accordingly, we can evaluate explainability by measuring how much it contributed to these other quality aspects. We call such an evaluation an evaluation on the *system level*. However, we can also evaluate explainability by assessing the generated explanations themselves. This would be an evaluation on the *explanation level*.

Evaluation on the system level The quality of an explanation may be assessed on the system level, by analyzing how it affects the system in terms of a specific quality aspect (e.g., the influence of an explanation on usability or performance). Many of these quality aspects have already been intensively investigated by software engineers, so that there are already established evaluation methods for them. As an example, the impact of explanations on the system's usability can be evaluated quantitatively through usability tests scores or qualitatively, by assessing users' perception on the system's usability [5].

Since there are so many quality aspects associated with explainability, it is not possible to discuss all of the evaluation methods and relevant metrics in this publication. For this reason, we concentrate on strategies for evaluating explanations and advise the interested reader to search for literature that addresses evaluation for the particular quality aspect of interest.

Evaluation on the explanation level So far, many different methods have been proposed for evaluating explanations of

[13] For simplicity, we generalize the term *metric* to refer to both metrics and indicators.

software systems on the explanation level [68, 69]. To the best of our knowledge, however, there is no consensus on what is a good evaluation method for such explanations [68, 136], which also means that there is no gold standard for evaluating explanations of software systems [68, 137]. For this reason, the choice of evaluation methods and metrics depends heavily on which objectives have been defined in advance.

7.4.2 Methods

Since the effectiveness and quality of an explanation is subjective in most cases, the focus of evaluations is often on end-user feedback. This was confirmed by Nunes and Jannach [47] in an SLR on explanations in recommender systems. They discovered that the most important methods used to evaluate explanations are user studies. Likewise, Vilone and Longo [68] found that subjective assessments are the most common way of evaluation. This was again confirmed in a recent study that ascertained that user-centered activities and methods are often recommended for the development of explainable systems [123].

Consistent with this research, we found in our SLR that the most important evaluation methods are focused on end-user feedback. In particular, the methods we identified include: user studies in general [138, 139], but also more specifically questionnaires [47, 140], A/B tests [93, 111], case studies [109, 141], and interviews [75, 98]. These user-centered methods may be used to determine whether or not a given explanation design (as defined during requirements analysis) helps to achieve an objective.

User studies can adopt different methodologies to evaluate explanations. For instance, in questionnaires, study participants might be asked whether they better understood a certain aspect of the system after receiving explanations (e.g., "From the explanation, I understand how the [software, algorithm, tool] works.", [142]).

The methodology *of A/B tests* is to compare several types of explanations to discover which style is best suited given the objectives that were defined (e.g., comparing "explanation variant A" against "explanation variant B" with respect to the stipulated objective of achieving more user satisfaction with the system) [143].

A *case study* is another empirical research strategy which focuses on studying a particular case or phenomenon (e.g., an individual person, a group, a setting, or an organization) in its own context or environment. Case studies typically combine a variety of data collection techniques, which helps to better understand the complexity of individual cases and to increase the accuracy of data acquired and conclusions drawn [144]. In this sense, a case study could consist in assessing explainable systems in a particular environment and deriving detailed information about their impact.

Finally, an *interview* is a qualitative research method in which the population of interest (i.e., the people being interviewed) are questioned by the researcher. These questions can be pre-established and fixed (structured interviews), or the interviewer can think about the questions during the interview, with (semi-structured interviews) or without (unstructured interviews) the help of a checklist of topics to be covered [144]. In general, conducting interviews allows for a more in-depth understanding of a population's viewpoints on certain topics.

7.4.3 Metrics

There is a wide variety of metrics for explanations that can become important depending on the context. Explanations can be evaluated with respect to metrics such as their soundness [113], plausibility [145], realism [146], and persuasiveness [147]. The comprehensibility of an explanation, its relevance, length, timeliness, completeness, and usefulness are also metrics that are frequently addressed in the literature [68, 113, 148]. In general, Vilone and Longo compiled a list of explanation properties[14] [68] that can be used to assess the quality of explanations. Furthermore, we also provide a list of all the properties we found in the literature in our supplementary material [67].

Contextualizing: In the case of our hypothetical scenario, it is crucial to evaluate whether the implemented explanations are suitable to achieve the stated objectives, taking into account the existing constraints.

Timeliness was identified as a constraint. An explanation is perceived as helpful or relevant not only if it is comprehensive, but also if it is given at an opportune moment so that it supports a decision at the right time. During navigation, for example, an explanation outlining the reasons for a route change should be understood in a timely manner so that the passenger can make a decision in time for the vehicle to still take a particular exit. In this scenario, the difference between the time the user takes from receiving the explanation to taking the action can serve as a metric.

When it comes to achieving the goal of system understandability, a comparison of mental models in a user study may be one way to ascertain whether the chosen explanation design is beneficial. Using such a comparison, it is possible to determine whether the user's mental model (i.e., the user's interpretation of the system) is sufficiently comparable to the actual model of the system, as developed by the engineers (i.e., what the system actually is and does).

For example, such a comparison can be used to assess whether the user understood the explanation of the route

[14] We also consider this properties as metrics.

change as it should have been understood, whether the explanation in the case contributed to the comprehensibility of the system, or whether there are still gaps in the communication of the information.

Finally, a user study can help assess participants' perceptions of the user experience.

8 Discussion

Explainability is a quality aspect that echoes the demand for more human oversight of systems [17]. 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. However, there is currently little guidance on how to do so. Developing proper elicitation methodologies, artifacts, and tools to capture explainability requirements and assist the RE process is critical for project success. Artifacts are important components of RE and software projects because they promote knowledge exchange and help develop a shared understanding, both of which are essential for the definition of good requirements. Artifacts contribute to the development of a common vocabulary, which enables communication and the calibration of expectations in relation to system and project requirements.

To this end, our first contribution is a definition of explainability for software engineers (Sect. 4). 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. A common definition facilitates communication during a project. Besides, being aware of these variables facilitates the software development process, supporting the elicitation and specification of explainability requirements. In this sense, the possible values for these variables (e.g., *reasoning process* as an aspect that should be explained) we found in the literature can serve as an abstract starting point during requirements analysis. Overall, our artifacts can help engineering explainable systems and to make good design choices towards explainability requirements.

In contrast, poor design choices regarding explainability (e.g., inadequate information, wrong presentation choices) can negatively affect the relationship with the user (e.g., user experience issues), interfere with important quality aspects for a corporation (e.g., damaging brand image and customer loyalty), and bring disadvantages for the project or the system (e.g., increasing development costs or hindering system performance). This kind of impact may stem from the fact that explainability might be seen as an aspect of communication between systems and humans. Depending on how it happens in practice, communication can either strengthen or harm relationships.

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 [78] for research concerning explanations in several disciplines). At the same time, 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.

To this end, two other additional contributions of this article are a conceptual model (Sect. 5) and a knowledge catalogue (Sect. 6). Conceptual models are useful to abstract, comprehend, and communicate information. Among others, our catalogue can serve as checklist during elicitation and also during trade-off analysis. It can help software engineers avoid conflicts between quality aspects and choose the best strategies for achieving the desired quality outcomes. Both artifacts contain information that may be used to turn explainability into a positive catalyst for other essential system qualities in modern systems.

Building up on this, our reference model (presented in Sect. 7) brings together the proposed definition, conceptual model, and knowledge catalogue into one artifact. This reference model is intended to help software engineers develop explainable systems by providing them with a guide of the main aspects to consider during three key phases of the software lifecycle: requirements analysis, design, and evaluation. To incorporate explainability into a system, many different aspects need to be considered in the software engineering process. For example, the envisioned quality goals, the different types of explanation a customer may want, and the appropriate form of presentation (audio, text, icon, etc.), all of which should be evaluated using appropriate methods and metrics. In this regard, the reference model serves as a guide to identify the relevant things to consider for developing explainable systems at each phase in the software engineering process.

Since reference models are usually minimal sets (see Sect. 2.2), our proposed reference model can be extended and adapted to fit different contexts or as a base for other models. In fact, during the process of publishing this article, a study was conducted and this reference model was expanded and used to build a quality framework for explainability. The results were published in a conference paper (see [18]). Although this is not a direct evaluation of our reference model, it may be seen as an indication that such an artifact is promising and can be used as a base model that can be extended or as a model template for creating other models. In addition, a minimal model

implies that these components are essential and should not be disregarded.

To check whether our artifacts can be useful for dealing with explainability as an NFR, we compare them with the guideline set out by Paech et al. [24]. They proposed a list with 20 features that should be covered by methods that focus on quality aspects (see [24], Table 1). Taking these features into account, our artifacts can support: the identification of quality 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.

Overall, building these artifacts has revealed that there is much to do in the field of NFRs. On the one hand, we believe that there may be other emerging quality aspects besides explainability. Aspects such as human-machine cooperation, privacy awareness, and mental-model accuracy show that there are specific needs that should be better understood when developing modern systems. Furthermore, ethics, fairness, and legal compliance are all good examples of quality aspects that are gaining in importance and should be better researched [149].

On the other hand, we have identified that explainability can exhibit an impact on nearly all traditional NFRs that can be found in the ISO 25010 [37]: performance, efficiency, usability, reliability, security, maintainability, and portability. As such, the importance of explainability has to be further acknowledged. In this line of thought, the impact of other NFRs on explainability should be better researched and existing catalogues could be updated to incorporate explainability. The RE community needs to explore what kind of activities, artifacts, methods, and tools need to be incorporated into the software development process in order to accommodate the necessary steps towards building explainable systems. Our work is an essential step in this direction.

9 Limitations and threats to validity

Our work is mostly based on qualitative data analysis. Consequently, there is the possibility that the results are affected by subjectivity during analysis. Therefore, we decided on a multi-method approach to produce results that are more robust and compelling than single method studies. In what follows, we discuss the main threats to validity in each part of our research.

9.1 SLR and coding

The review process assumed a common understanding among all researchers involved in this work with respect to the search and analysis methods used. Results could be subject to bias if the methods and concepts are misunderstood. We mitigated this threat by elaborating a review protocol and discussing it before starting the review to reach a good level of shared understanding.

We have formulated inclusion and exclusion criteria to reduce biases due to subjective decisions in the selection process. Some criteria, such as the publication period, are objective, while others, focusing on the content of the papers, are still subjective.

To decrease the amount of researcher bias, we conducted the analysis independently. For both the literature review and the coding process, in case of disagreement, the decision on inclusion or exclusion (for a paper) or the code assignment (for the extracted data) was taken by all researchers and validated by the Fleiss' Kappa statistic.

Another limiting factor of our SLR is that it only covers the period up to March 2020. Explainability is a rapidly evolving field of research, especially in recent years, so it is not guaranteed that our results will remain up-to-date. To counteract this problem at least somewhat, we have used the most recent research as a guide when constructing the reference model. In this regard, there were no problems embedding our results. Moreover, our SLR period already spans 36 years, during which we have found that similar concepts remain steadily in the literature.

9.2 Workshops

There are some threats to validity for the workshops. First, some of the homework given to the participants was based on the results of our literature review. Accordingly, it could be that they were implicitly biased by it.

To prevent such an implicit influence, we removed as much additional information from the homework as possible. Furthermore, we encouraged the experts to think of own experiences an be not restrained by the given material.

Furthermore, as the workshops took place online, this could be a limiting factor. Here, the homework was aimed at getting the experts attuned to the subject matter so that the workshops could take place in a focused manner despite the online format. Additionally, plenty of breaks and clear tasks helped the participants be productive during the workshop.

Finally, the time allotted for each workshop was short, at four hours each. However, since we only had three tasks for each workshop, this was an acceptable time frame in our eyes. This was also confirmed during the workshops, as we did not have to cancel any of the tasks due to lack of time.

9.3 Proposed artifacts

When it comes to our conceptual model and the knowledge catalogue, the clustering and categorization of the quality aspects into their different dimensions was prone to subjective judgment. As steps to mitigate this, we rooted this categorization on well-known concepts present in the literature and conducted workshops with experts. This allowed us to inspect our clustering through internal and external reviews.

During the internal reviews, the categorization was discussed among the authors to clarify ambiguities and reach agreements. During the external reviews, we compared the findings from the literature with expert knowledge. Due to these review processes, we are confident to have achieved a proper level of validity of the catalogue.

Moreover, as researchers in the field, we are confident that both our catalogue and conceptual model are reasonably accurate for the field studied, developed over debates that formed our shared knowledge on the subject.

The coding and clustering for the reference model was also prone to subjective judgment. To mitigate this threat, we decreased the amount of researcher bias by conducting the analysis (coding and clustering) independently. As we did not conduct workshops to validate our findings, we used established results from previous work to form our categories. Furthermore, we illustrated the use of the reference model with a running example, showcasing its applicability. However, its correctness and realism have not been evaluated and further studies are needed in order to validate the model in practice.

10 Conclusion and future work

Explainability is increasingly seen as an appropriate means of achieving essential quality aspects in a system, such as transparency, accountability, and trust. As building these values into our systems becomes more urgent, there is a need for tools and methods that help elicit, implement, and validate related requirements. For this reason, we should be concerned with understanding explainability as a whole: its meaning, its effects, its taxonomy. Furthermore, the interplay of explainability with increasingly important quality aspects such as ethics, greenability, privacy, and trust should also be researched.

In this sense, our proposed definition can help to facilitate communication and align expectations when referring to explainability. Our conceptual model can help professionals to understand its taxonomy, and our knowledge catalogue can help to identify conflicts between explainability and other important qualities. Finally, our suggested reference model for explainability can assist software engineers understand the relevant and influential aspects for the requirements analysis, design, and evaluation of explainable systems. With the support of these artifacts, it is possible to think of design strategies and implementation level solutions that result in positive effects for all stakeholders involved. Overall, we hope that our work lays a foundation for the RE community to better understand and investigate the topic of explainability.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s00766-022-00393-5.

Acknowledgements This work was supported by the research initiative Mobilise between the Technical University of Braunschweig and Leibniz University Hannover, funded by the Ministry for Science and Culture of Lower Saxony and by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy within the Cluster of Excellence PhoenixD (EXC 2122, Project ID 390833453). Work on this paper was also funded by the Volkswagen Foundation grants AZ 98509 and AZ 98514 "Explainable Intelligent Systems" (EIS) and by the DFG grant 389792660 as part of TRR 248. We thank Martin Glinz for his feedback on our research design and our colleague Nils Prenner for his feedback and the fruitful discussions. Furthermore, we thank all workshop participants, the anonymous reviewers, and the colleagues who gave feedback on our manuscript. Special gratitude goes to Dieter Belle, for his immeasurable help in staying organized.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

1. Mike walsh: welcome to the algorithmic age. Accessed 27 Jan 2022 (2018). https://www.mike-walsh.com/news/welcome-to-the-algorithmic-age
2. Panesar A (2019) Ethics of intelligence. Machine learning and AI for healthcare: big data for improved health outcomes. Apress, New York, NY, pp 207–254. https://doi.org/10.1007/978-1-4842-3799-1_6
3. Lepri B, Oliver N, Letouzé E, Pentland A, Vinck P (2018) Fair, transparent, and accountable algorithmic decision-making processes. Philos Technol 31(4):611–627. https://doi.org/10.1007/s13347-017-0279-x
4. Adadi A, Berrada M (2018) Peeking inside the black-box: a survey on explainable artificial intelligence (XAI). IEEE Access 6:52138–52160. https://doi.org/10.1109/ACCESS.2018.2870052

5. Chazette L, Schneider K (2020) Explainability as a non-functional requirement: challenges and recommendations. Requir Eng 25(4):493–514. https://doi.org/10.1007/s00766-020-00333-1
6. Langer M, Baum K, Hartmann K, Hessel S, Speith T, Wahl J (2021) Explainability auditing for intelligent systems: a rationale for multi-disciplinary perspectives. In: 2021 IEEE 29th international requirements engineering conference workshops (REW). pp 164–168. https://doi.org/10.1109/REW53955.2021.00030
7. Glinz M (2007) On non-functional requirements. In: 15th IEEE international requirements engineering conference (RE). pp 21–26. https://doi.org/10.1109/RE.2007.45
8. Chazette L, Karras O, Schneider K (2019) 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). pp 223–233. https://doi.org/10.1109/RE.2019.00032
9. Köhl MA, Baum K, Langer M, Oster D, Speith T, Bohlender D (2019) Explainability as a non-functional requirement. In: 27th IEEE international requirements engineering conference (RE). IEEE, New York, NY, pp 363–368. https://doi.org/10.1109/RE.2019.00046
10. Bussone A, Stumpf S, O'Sullivan D (2015) The role of explanations on trust and reliance in clinical decision support systems. In: 2015 international conference on healthcare informatics. IEEE, New York, NY, pp 160–169. https://doi.org/10.1109/ICHI.2015.26
11. Winkler JP, Vogelsang A (2017) "What does my classifier learn?" A visual approach to understanding natural language text classifiers. In: Frasincar F, Ittoo A, Nguyen LM, Métais E (eds) Natural language and information systems. pp 468–479. https://doi.org/10.1007/978-3-319-59569-6_55
12. Zhou J, Hu H, Li Z, Yu K, Chen F (2019) Physiological indicators for user trust in machine learning with influence enhanced fact-checking. In: International cross-domain conference for machine learning and knowledge extraction. Springer, Cham, pp 94–113. https://doi.org/10.1007/978-3-030-29726-8_7
13. Hind M, Wei D, Campbell M, Codella NCF, Dhurandhar A, Mojsilović A, Natesan RK, Varshney KR (2019) TED: teaching AI to explain its decisions. In: Proceedings of the 2019 AAAI/ACM conference on AI, ethics, and society. ACM, New York, NY, pp 123–129. https://doi.org/10.1145/3306618.3314273
14. Rosenfeld A, Richardson A (2019) Explainability in human-agent systems. Auton Agent Multi-Agent Syst 33(6):673–705. https://doi.org/10.1007/s10458-019-09408-y
15. Brunotte W, Specht A, Chazette L, Schneider K (2022) Privacy explanations – a means to end-user trust. arXiv. https://doi.org/10.48550/ARXIV.2210.09706
16. Abdollahi B, Nasraoui O (2018) Transparency in fair machine learning: the case of explainable recommender systems. Human and machine learning: visible, explainable, trustworthy and transparent. Springer, Cham, CH, pp 21–35. https://doi.org/10.1007/978-3-319-90403-0_2
17. Thiebes S, Lins S, Sunyaev A (2020) Trustworthy artificial intelligence. Electron Mark 1–18 . https://doi.org/10.1007/s12525-020-00441-4
18. Chazette L, Klös V, Herzog F, Schneider K (2022) Requirements on explanations: a quality framework for explainability. In: Proceedings of the 2022 IEEE 30th international requirements engineering conference (RE), pp 140–152. https://doi.org/10.1109/RE54965.2022.00019
19. Glinz M, Fricker SA (2015) On shared understanding in software engineering: an essay. Comput Sci Res Dev 30(3):363–376. https://doi.org/10.1007/s00450-014-0256-x
20. Mairiza D, Zowghi D (2011) Constructing a catalogue of conflicts among non-functional requirements. Evaluation of novel approaches to software engineering. Springer, Berlin, Heidelberg, pp 31–44
21. Chung L, Nixon BA, Yu E, Mylopoulos J (2012) Non-functional requirements in software engineering. Springer, Boston, MA. https://doi.org/10.1007/978-1-4615-5269-7
22. Gacitúa R, Ma L, Nuseibeh B, Piwek P, Roeck AND, Rouncefield M, Sawyer P, Willis A, Yang H (2009) Making tacit requirements explicit. In: Second international workshop on managing requirements knowledge (MARK@RE). IEEE, New York, NY, pp 40–44. https://doi.org/10.1109/MARK.2009.7
23. Santos D, Resende A, Junior PA, Costa H (2016) 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). pp 1–12. https://doi.org/10.1109/CLEI.2016.7833322
24. Paech B, Kerkow D (2004) Non-functional requirements engineering-quality is essential. In: 10th international workshop on requirments engineering foundation for software quality
25. Cysneiros LM (2007) Evaluating the effectiveness of using catalogues to elicit non-functional requirements. In: Workshop em engenharia de requisitos (WER 2007). pp 107–115
26. Nuseibeh B, Easterbrook S (2000) Requirements engineering: a roadmap. In: Proceedings of the conference on the future of software engineering. ICSE '00. Association for Computing Machinery, New York, NY, pp 35–46. https://doi.org/10.1145/336512.336523
27. Bittner EAC, Leimeister JM (2013) 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. IEEE, Piscataway, pp 106–114. https://doi.org/10.1109/HICSS.2013.608
28. Ghazi P, Glinz M (2017) Challenges of working with artifacts in requirements engineering and software engineering. Requir Eng 22(3):359–385. https://doi.org/10.1007/s00766-017-0272-z
29. Boehm BW, Brown JR, Lipow M (1976) Quantitative evaluation of software quality. In: Proceedings of the 2nd international conference on software engineering. ICSE '76. IEEE Computer Society Press, Washington, DC, pp 592–605
30. Börger E, Hörger B, Parnas D, Rombach D (1999) Requirements capture, documentation, and validation. In: Dagstuhl seminar. Dagstuhl Seminar
31. Rupp C, Simon M, Hocker F (2009) Requirements engineering und management. HMD Prax Wirtsch 46(3):94–103
32. Serrano M, Serrano M (2013) Ubiquitous, pervasive and mobile computing: a reusable-models-based non-functional catalogue. In: Proceedings of requirements engineering@Brazil, vol. 1005. CEUR, Aachen, DE
33. Carvalho RM, Andrade RMC, Lelli V, Silva EG, de Oliveira KM (2020) What about catalogs of non-functional requirements? In: Proceedings of REFSQ-2020 workshops, vol. 2584. CEUR, Aachen, DE
34. Chazette L, Brunotte W, Speith T (2021) Exploring explainability: a definition, a model, and a knowledge catalogue. In: 2021 IEEE 29th international requirements engineering conference (RE). IEEE, pp 197–208
35. Wixon D, Wilson C (1997) Chapter 27 - the usability engineering framework for product design and evaluation. In: Helander MG, Landauer TK, Prabhu PV (eds) Handbook of human-computer interaction, 2nd edn. North-Holland, Amsterdam, pp 653–688. https://doi.org/10.1016/B978-044481862-1.50093-5
36. Hull E, Jackson K, Dick J (2011) Introduction. Springer, London, pp 1–23. https://doi.org/10.1007/978-1-84996-405-0_1
37. ISO Central Secretary: ISO/IEC 25010:2011 systems and software engineering-systems and software quality requirements

and evaluation (SQuaRE) - system and software quality models. Standard ISO/IEC 25010:2011, International Organization for Standardization (2011). https://www.iso.org/standard/35733.html
38. MacKenzie CM, Laskey K, McCabe F, Brown PF, Metz R, Hamilton BA (2006) Reference model for service oriented architecture 1.0. OASIS Stand 12(S 18)
39. Fettke P, Loos P (2003) Classification of reference models: a methodology and its application. IseB 1(1):35–53. https://doi.org/10.1007/BF02683509
40. Alani MM (2014) OSI model. Springer, Cham, pp 5–17. https://doi.org/10.1007/978-3-319-05152-9_2
41. Weber KC, Araújo EER, da Rocha ARC, Machado CAF, Scalet D, Salviano CF (2005) Brazilian software process reference model and assessment method. In: Yolum P, Güngör T, Gürgen F, Özturan C (eds) Computer and information sciences - ISCIS 2005. Springer, Berlin, Heidelberg, pp 402–411
42. Cherdantseva Y, Hilton J (2013) A reference model of information assurance and security. In: 2013 international conference on availability, reliability and security. pp 546–555. https://doi.org/10.1109/ARES.2013.72
43. Gutmann P, Grigg I (2005) Security usability. IEEE Secur Priv 3(4):56–58. https://doi.org/10.1109/MSP.2005.104
44. Torres RC, Martins LEG (2018) NFR catalogues for RFID middleware. J Comput Sci Technol 14(02):102–108
45. Carvalho RM, Andrade RMC, Oliveira KM (2020) How developers believe invisibility impacts NFRs related to user interaction. 28th IEEE international requirements engineering conference (RE). IEEE, New York, NY, pp 102–112. https://doi.org/10.1109/RE48521.2020.00022
46. Speith T (2022) A review of taxonomies of explainable artificial intelligence (xai) methods. In: Proceedings of the 2022 conference on fairness, accountability, and transparency. FAccT '22. Association for Computing Machinery, New York, NY. https://doi.org/10.1145/3531146.3534639
47. Nunes I, Jannach D (2017) A systematic review and taxonomy of explanations in decision support and recommender systems. User Model User-Adap Inter 27(3–5):393–444. https://doi.org/10.1007/s11257-017-9195-0
48. Anjomshoae S, Najjar A, Calvaresi D, Främling K (2019) Explainable agents and robots: results from a systematic literature review. In: Proceedings of the 18th international conference on autonomous agents and multiagent systems (AAMAS). International Foundation for Autonomous Agents and Multiagent Systems, Richland County, SC, pp 1078–1088. https://doi.org/10.5555/3306127.3331806
49. Abdul A, Vermeulen J, Wang D, Lim BY, Kankanhalli M (2018) 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). ACM, New York, NY, pp 1–18. https://doi.org/10.1145/3173574.3174156
50. Mathews SM (2019) Explainable artificial intelligence applications in NLP, biomedical, and malware classification: a literature review. In: Intelligent computing – proceedings of the computing conference. Springer, Cham, pp. 1269–1292. https://doi.org/10.1007/978-3-030-22868-2_90
51. Arrieta AB, Díaz-Rodríguez N, Ser JD, Bennetot A, Tabik S, Barbado A, Garcia S, Gil-Lopez S, Molina D, Benjamins R, Chatila R, Herrera F (2020) Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. Inf Fusion 58:82–115. https://doi.org/10.1016/j.inffus.2019.12.012
52. Clinciu M-A, Hastie H (2019) A survey of explainable ai terminology. In: Alonso JM, Catala A (eds) Proceedings of the 1st workshop on interactive natural language technology for explainable artificial intelligence (NL4XAI). Association for Computational Linguistics, Stroudsburg, PA, pp 8–13. https://doi.org/10.18653/v1/W19-8403
53. Graziani M, Dutkiewicz L, Calvaresi D, Amorim JP, Yordanova K, Vered M, Nair R, Abreu PH, Blanke T, Pulignano V, Prior JO, Lauwaert L, Reijers W, Depeursinge A, Andrearcyk V, Müller H (2022) A global taxonomy of interpretable AI: unifying the terminology for the technical and social sciences. Artif Intell Rev 1–32
54. Langer M, Oster D, Speith T, Hermanns H, Kästner L, Schmidt E, Sesing A, Baum K (2021) What do we want from explainable artificial intelligence (XAI)? - a stakeholder perspective on XAI and a conceptual model guiding interdisciplinary XAI research. Articif Intell. https://doi.org/10.1016/j.artint.2021.103473
55. do Prado Leite JCS, Cappelli C (2010) Software transparency. Bus Inf. Syst Eng 2(3):127–139. https://doi.org/10.1007/s12599-010-0102-z
56. Saldaña J (2021) The coding manual for qualitative researchers. SAGE Publications, Thousand Oaks, CA
57. Kitchenham B, Charters S (2007) Guidelines for performing systematic literature reviews in software engineering. Technical report, Keele University
58. Wohlin C (2014) 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. pp 1–10. https://doi.org/10.1145/2601248.2601268
59. Fleiss JL (1971) Measuring nominal scale agreement among many raters. Psychol Bull 76(5):378–382. https://doi.org/10.1037/h0031619
60. Landis JR, Koch GG (1977) The measurement of observer agreement for categorical data. Biometrics 33(1):159–174. https://doi.org/10.2307/2529310
61. Wolfswinkel JF, Furtmueller E, Wilderom CPM (2013) Using grounded theory as a method for rigorously reviewing literature. Eur J Inf Syst 22(1):45–55. https://doi.org/10.1057/ejis.2011.51
62. Buchanan BG, Shortliffe EH (1984) Rule-based expert systems: the MYCIN experiments of the stanford heuristic programming project. Addison-Wesley, Boston, MA
63. Brock DC (2018) Learning from artificial intelligence's previous awakenings: the history of expert systems. AI Mag 39(3):3–15. https://doi.org/10.1609/aimag.v39i3.2809
64. Charmaz K (2006) Constructing grounded theory: a practical guide through qualitative analysis. SAGE Publications, Thousand Oaks, CA
65. Miles MB, Huberman AM (1994) Qualitative data analysis: an expanded sourcebook. SAGE Publications, Thousand Oaks, CA
66. Boyatzis RE (1998) Transforming qualitative information: thematic analysis and code development. SAGE Publications, Thousand Oaks, CA
67. Chazette L, Brunotte W, Speith T (2022) Explainable software systems: from requirements analysis to system evaluation. Suppl Mater J Article https://figshare.com/s/f73d41c41345dd08cf39
68. Vilone G, Longo L (2021) Notions of explainability and evaluation approaches for explainable artificial intelligence. Inf Fusion 76:89–106. https://doi.org/10.1016/j.inffus.2021.05.009
69. Zhou J, Gandomi AH, Chen F, Holzinger A (2021) Evaluating the quality of machine learning explanations: a survey on methods and metrics. Electronics. https://doi.org/10.3390/electronics10050593
70. Carvalho DV, Pereira EM, Cardoso JS (2019) Machine learning interpretability: a survey on methods and metrics. Electronics. https://doi.org/10.3390/electronics8080832
71. Pieters W (2011) Explanation and trust: What to tell the user in security and AI? Ethics Inf Technol 13(1):53–64. https://doi.org/10.1007/s10676-010-9253-3

72. Holzinger A, Langs G, Denk H, Zatloukal K, Müller H (2019) Causability and explainability of artificial intelligence in medicine. Wiley Interdiscipl Rev Data Min Knowl Discov 9(4):1–13. https://doi.org/10.1002/widm.1312
73. Hois J, Theofanou-Fuelbier D, Junk AJ (2019) How to achieve explainability and transparency in human AI interaction. International conference on human-computer interaction (HCI). Springer, Cham, CH, pp 177–183. https://doi.org/10.1007/978-3-030-23528-4_25
74. Glass A, McGuinness DL, Wolverton M (2008) Toward establishing trust in adaptive agents. In: Proceedings of the 13th international conference on intelligent user interfaces (IUI). ACM, New York, NY, pp 227–236. https://doi.org/10.1145/1378773.1378804
75. Liao QV, Gruen DM, Miller S (2020) Questioning the AI: informing design practices for explainable AI user experiences. In: Proceedings of the 2020 conference on human factors in computing systems (CHI). ACM, New York, NY, pp 1–15. https://doi.org/10.1145/3313831.3376590
76. Dourish P (2004) What we talk about when we talk about context. Pers Ubiquit Comput 8(1):19–30. https://doi.org/10.1007/s00779-003-0253-8
77. Ribeiro MT, Singh S, Guestrin C (2016) "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. ACM, New York, NY, pp 1135–1144. https://doi.org/10.1145/2939672.2939778
78. Miller T (2019) Explanation in artificial intelligence: insights from the social sciences. Artif Intell 267:1–38. https://doi.org/10.1016/j.artint.2018.07.007
79. Pacey A (1983) The culture of technology. MIT Press, Cambridge, MA
80. Kummer T-F, Leimeister JM, Bick M (2012) On the importance of national culture for the design of information systems. Bus Inf Syst Eng 4(6):317–330. https://doi.org/10.1007/s12599-012-0236-2
81. Thomsen S (2004) Corporate values and corporate governance. Corp Gov 4(4):29–46. https://doi.org/10.1108/14720700410558862
82. Chung L, Nixon BA (1995) Dealing with non-functional requirements: three experimental studies of a process-oriented approach. In: 1995 17th international conference on software engineering. IEEE, p 25
83. Carvallo JP, Franch X, Quer C (2006) Managing non-technical requirements in cots components selection. In: 14th IEEE international requirements engineering conference (RE). IEEE, New York, NY, pp. 323–326. https://doi.org/10.1109/RE.2006.40
84. Freeman S, Pryce N (2009) Growing object-oriented software. Guided by tests. Addison-Wesley, Boston, MA
85. Cambridge dictionary: spectrum. Cambridge Dictionary (2022). https://dictionary.cambridge.org/dictionary/english/spectrum Accessed 05 May 2022
86. McConnell S (2004) Code complete. Microsoft Press, Redmond, WA
87. Henin C, Daniel LM (2019) Towards a generic framework for black-box explanation methods. In: Proceedings of the IJCAI workshop on explainable artificial intelligence (XAI). pp 28–34
88. Chen L, Yan D, Wang F (2019) User evaluations on sentiment-based recommendation explanations. ACM Trans Interact Intell Syst (TiiS) 9(4):1–38. https://doi.org/10.1145/3282878
89. Preece AD, Harborne D, Braines D, Tomsett R, Chakraborty S (2018) Stakeholders in explainable AI. CoRR arXiv:1810.00184
90. Weller A (2019) Transparency: motivations and challenges. In: Samek W, Montavon G, Vedaldi A, Hansen LK, Müller K-R (eds) Explainable AI: interpreting, explaining and visualizing deep learning. Springer, Cham, CH, pp 23–40. Chap. 2. https://doi.org/10.1007/978-3-030-28954-6_2
91. Glinz M, Wieringa RJ (2007) Guest editors' introduction: stakeholders in requirements engineering. IEEE Softw 24(2):18–20. https://doi.org/10.1109/MS.2007.42
92. Riedl MO (2019) Human-centered artificial intelligence and machine learning. Hum Behav Emerg Technol 1(1):33–36. https://doi.org/10.1002/hbe2.117
93. McInerney J, Lacker B, Hansen S, Higley K, Bouchard H, Gruson A, Mehrotra R (2018) Explore, exploit, and explain: personalizing explainable recommendations with bandits. In: Proceedings of the 12th ACM conference on recommender systems (RecSys). ACM, New York, NY, pp 31–39. https://doi.org/10.1145/3240323.3240354
94. Cai CJ, Jongejan J, Holbrook J (2019) The effects of example-based explanations in a machine learning interface. In: Proceedings of the 24th international conference on intelligent user interfaces (IUI). ACM, New York, NY, pp 258–262. https://doi.org/10.1145/3301275.3302289
95. Zanker M (2012) The influence of knowledgeable explanations on users' perception of a recommender system. In: Proceedings of the sixth ACM conference on recommender systems (RecSys). ACM, New York, NY, pp 269–272. https://doi.org/10.1145/2365952.2366011
96. Pu P, Chen L (2006) Trust building with explanation interfaces. In: Proceedings of the 11th international conference on intelligent user interfaces (IUI). ACM, New York, NY, pp 93–100. https://doi.org/10.1145/1111449.1111475
97. Kizilcec RF (2016) 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). ACM, New York, NY, pp 2390–2395. https://doi.org/10.1145/2858036.2858402
98. Cramer H, Evers V, Ramlal S, Maarten VS, Rutledge L, Stash N, Aroyo L, Wielinga B (2008) The effects of transparency on trust in and acceptance of a content-based art recommender. User Model User-Adap Inter 18(5):455. https://doi.org/10.1007/s11257-008-9051-3
99. Tintarev N, Masthoff J (2007) Effective explanations of recommendations: user-centered design. In: Proceedings of the 2007 ACM conference on recommender systems (RecSys). ACM, New York, NY, pp 153–156. https://doi.org/10.1145/1297231.1297259
100. Tsai C, Brusilovsky P (2019) Explaining recommendations in an interactive hybrid social recommender. In: Proceedings of the 24th international conference on intelligent user interfaces (IUI). ACM, New York, NY, pp 391–396. https://doi.org/10.1145/3301275.3302318
101. Tintarev N, Masthoff J (2012) Evaluating the effectiveness of explanations for recommender systems. User Model User-Adap Inter 22(4–5):399–439. https://doi.org/10.1007/s11257-011-9117-5
102. Darlington K (2013) Aspects of intelligent systems explanation. Univ J Control Autom 1(2):40–51. https://doi.org/10.13189/ujca.2013.010204
103. Kumar PS, Saravanan M, Suresh S (2019) Explainable classification using clustering in deep learning models. In: Proceedings of the IJCAI workshop on explainable artificial intelligence (XAI). pp 115–121
104. Dodge J, Liao QV, Zhang Y, Bellamy RKE, Dugan C (2019) Explaining models: an empirical study of how explanations impact fairness judgment. In: Proceedings of the 24th international conference on intelligent user interfaces (IUI). ACM, New York, NY, pp 275–285. https://doi.org/10.1145/3301275.3302310

105. Putnam V, Conati C (2019) Exploring the need for explainable artificial intelligence (XAI) in intelligent tutoring systems (ITS). In: Joint proceedings of the ACM IUI 2019 workshops. CEUR, Aachen, DE
106. Schneider J, Handali J (2019) Personalized explanation in machine learning: a conceptualization. In: Proceedings of the 27th European conference on information systems (ECIS)
107. Gilpin LH, Testart C, Fruchter N, Adebayo J (2018) Explaining explanations to society. In: NIPS workshop on ethical, social and governance issues in AI. pp 1–6
108. Monteath I, Sheh R (2018) Assisted and incremental medical diagnosis using explainable artificial intelligence. In: Proceedings of the IJCAI/ECAI workshop on explainable artificial intelligence (XAI). pp 104–108
109. Binns R, Van Kleek M, Veale M, Lyngs U, Zhao J, Shadbolt N (2018) '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). ACM, New York, NY, pp. 1–14. https://doi.org/10.1145/3173574.3173951
110. McCarthy K, Reilly J, McGinty L, Smyth B (2004) Thinking positively-explanatory feedback for conversational recommender systems. In: Proceedings of the European conference on case-based reasoning (ECCBR) explanation workshop. pp 115–124
111. Lage I, Lifschitz D, Doshi-Velez F, Amir O (2019) Exploring computational user models for agent policy summarization. In Proceedings of the IJCAI workshop on explainable artificial intelligence (XAI). pp 59–65
112. Borgo R, Cashmore M, Magazzeni D (2018) Towards providing explanations for AI planner decisions. In: Proceedings of the IJCAI/ECAI workshop on explainable artificial intelligence (XAI). pp 11–17
113. Guidotti R, Monreale A, Ruggieri S, Turini F, Giannotti F, Pedreschi D (2019) A survey of methods for explaining black box models. ACM Comput Surv 51(5):1–42. https://doi.org/10.1145/3236009
114. Hohman F, Head A, Caruana R, DeLine R, Drucker SM (2019) 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. ACM, New York, NY, pp 1–13. https://doi.org/10.1145/3290605.3300809
115. Alexander IF (2004) A better fit - characterising the stakeholders. In: Grundspenkis J, Kirikova M (eds) CAiSE'04 workshops in connection with the 16th conference on advanced information systems engineering, Riga, Latvia, 7–11 June, 2004, knowledge and model driven information systems engineering for networked organisations, proceedings, vol 2. Riga Technical University, Riga, Latvia, Faculty of Computer Science and Information Technology, pp 215–223
116. Zhou J, Chen F (2019) Towards trustworthy human-AI teaming under uncertainty. In: Proceedings of the IJCAI workshop on explainable artificial intelligence (XAI). pp 143–147
117. Chen J, Lécué F, Pan JZ, Horrocks I, Chen H (2018) Knowledge-based transfer learning explanation. In: Proceedings of the sixteenth international conference for principles of knowledge representation and reasoning (KR). AAAI, Palo Alto, CA, pp 349–358
118. Jacovi A, Marasović A, Miller T, Goldberg Y (2021) Formalizing trust in artificial intelligence: prerequisites, causes and goals of human trust in AI. In: Proceedings of the 2021 ACM conference on fairness, accountability, and transparency (FAccT). Association for Computing Machinery, New York, NY, pp 624–635. https://doi.org/10.1145/3442188.3445923
119. Nascimento N, Alencar P, Cowan D, Lucena C (2020) A reference model for iot embodied agents controlled by neural networks. In: 2020 IEEE international conference on big data (Big Data). pp 3500–3505. https://doi.org/10.1109/BigData50022.2020.9377936
120. Schneider K (2012) Abenteuer softwarequalität: grundlagen und verfahren für qualitätssicherung und qualitätsmanagement. dpunkt.verlag, Heidelberg, DE
121. Wagner S, Goeb A, Heinemann L, Kläs M, Lampasona C, Lochmann K, Mayr A, Plösch R, Seidl A, Streit J, Trendowicz A (2015) Operationalised product quality models and assessment: the quamoco approach. Inf Softw Technol 62:101–123. https://doi.org/10.1016/j.infsof.2015.02.009
122. Glinz M (2017) A glossary of requirements engineering terminology. Stand Gloss Certif Prof Requir Eng (CPRE) Stud Exam Version 1:56
123. Chazette L, Klünder J, Balci M, Schneider K (2022) 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 (ICSSP'22). ICSSP '22. Association for Computing Machinery, New York, NY. https://doi.org/10.1145/3529320.3529321
124. Miller T, Howe P, Sonenberg L (2017) Explainable AI: beware of inmates running the asylum. or: how i learnt to stop worrying and love the social and behavioural sciences. In: Aha DW, Darrell T, Pazzani M, Reid D, Sammut C, Stone P (eds) Proceedings of the IJCAI 2017 workshop on explainable artificial intelligence (XAI). IJCAI, Santa Clara County, CA, pp 36–42
125. Hall M, Harborne D, Tomsett R, Galetic V, Quintana-Amate S, Nottle A, Preece A (2019) A systematic method to understand requirements for explainable AI (XAI) systems. In: Proceedings of the IJCAI 2019 workshop on explainable artificial intelligence (XAI), pp 21–27
126. Dam HK, Tran T, Ghose A (2018) Explainable software analytics. In: Proceedings of the 40th international conference on software engineering: new ideas and emerging results (ICSE-NIER). Association for Computing Machinery, New York, NY, pp 53–56. https://doi.org/10.1145/3183399.3183424
127. Weber E, Van Bouwel J, Vanderbeeken R (2005) Forms of causal explanation. Found Sci 10(4):437–454
128. Halpern JY, Pearl J (2005) Causes and explanations: a structural-model approach. Part ii: explanations. Br J Philos Sci 56(4):889–911
129. Byrne RM (2019) Counterfactuals in explainable artificial intelligence (xai): evidence from human reasoning. In: IJCAI. pp 6276–6282
130. Bechtel W (1994) Levels of description and explanation in cognitive science. Mind Mach 4(1):1–25
131. Bechtel W, Abrahamsen A (2005) Explanation: a mechanist alternative. Stud History Philos Sci Part C Stud History Philos Biol Biomed Sci 36(2):421–441. https://doi.org/10.1016/j.shpsc.2005.03.010 (**Mechanisms in Biology**)
132. Brinton C (2017) A framework for explanation of machine learning decisions. In: Proceedings of the IJCAI workshop on explainable artificial intelligence (XAI). pp 14–18
133. Gregor S, Benbasat I (1999) Explanations from intelligent systems: theoretical foundations and implications for practice. MIS Q 23(4):497–530. https://doi.org/10.2307/249487
134. Doran D, Schulz S, Besold TR (2017) 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. CEUR, Aachen, DE
135. Kim B, Wattenberg M, Gilmer J, Cai CJ, Wexler J, Viégas FB, Sayres R (2018) Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV). In: Dy JG, Krause A (eds) Proceedings of the 35th international conference on machine learning. ICML 2018. Microtome

Publishing, Brookline, MA, pp 2668–2677. http://proceedings.mlr.press/v80/kim18d.html
136. Speith T (2021) How to evaluate explainability – a case for three criteria. In: 30th IEEE international requirements engineering conference workshops. REW 2022. IEEE, Piscataway, NJ, USA
137. Doshi-Velez F, Kim B (2017) Towards a rigorous science of interpretable machine learning. CoRR arxiv:1702.08608
138. Friedrich G, Zanker M (2011) A taxonomy for generating explanations in recommender systems. AI Mag 32(3):90–98
139. Eiter T, Saribatur ZG, Schüller P (2019) Abstraction for zooming-in to unsolvability reasons of grid-cell problems. In: Proceedings of the IJCAI workshop on explainable artificial intelligence (XAI 2019). pp 7–13
140. Vorm ES (2018) Assessing demand for transparency in intelligent systems using machine learning. In: 2018 innovations in intelligent systems and applications (INISTA). IEEE, pp. 1–7. https://doi.org/10.1109/INISTA.2018.8466328
141. Juozapaitis Z, Koul A, Fern A, Erwig M, Doshi-Velez F (2019) Explainable reinforcement learning via reward decomposition. In: Proceedings of the IJCAI workshop on explainable artificial intelligence (XAI 2019). pp 47–53
142. Hoffman RR, Mueller ST, Klein G, Litman J (2018) Challenges and prospects, metrics for explainable AI
143. Herlocker JL, Konstan JA, Riedl J (2000) Explaining collaborative filtering recommendations. In: Proceedings of the 2000 ACM conference on computer supported cooperative work (CSCW). ACM, New York, NY, pp 241–250. https://doi.org/10.1145/358916.358995
144. Robson C, McCartan K (2016) Real world research: a resource for users of social research methods in applied settings, 4th edn. Wiley, Chichester
145. Ehsan U, Tambwekar P, Chan L, Harrison B, Riedl MO (2019) 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. pp 263–274
146. Olson ML, Neal L, Li F, Wong W-K (2019) Counterfactual states for atari agents via generative deep learning. In: Proceedings of the IJCAI workshop on explainable artificial intelligence (XAI 2019). pp 87–93
147. Sato M, Nagatani K, Sonoda T, Zhang Q, Ohkuma T (2019) Context style explanation for recommender systems. J Inf Process 27:720–729. https://doi.org/10.2197/ipsjjip.27.720
148. Wang N, Wang H, Jia Y, Yin Y (2018) Explainable recommendation via multi-task learning in opinionated text data. In: The 41st international ACM SIGIR conference on research & development in information retrieval. pp 165–174
149. Aydemir FB, Dalpiaz F (2018) A roadmap for ethics-aware software engineering. In: Proceedings of the international workshop on software fairness (FairWare). ACM, New York, NY, pp 15–21. https://doi.org/10.1145/3194770.3194778