

# Explanations in Everyday Software Systems: Towards a Taxonomy for Explainability Needs

Jakob Droste  
*Leibniz Universität Hannover*  
*Software Engineering Group*  
Hannover, Germany  
jakob.droste@inf.uni-hannover.de

Hannah Deters  
*Leibniz Universität Hannover*  
*Software Engineering Group*  
Hannover, Germany  
hannah.deters@inf.uni-hannover.de

Martin Obaidi  
*Leibniz Universität Hannover*  
*Software Engineering Group*  
Hannover, Germany  
martin.obaidi@inf.uni-hannover.de

Kurt Schneider  
*Leibniz Universität Hannover*  
*Software Engineering Group*  
Hannover, Germany  
kurt.schneider@inf.uni-hannover.de

**Abstract**—Modern software systems are becoming increasingly complex and opaque. The integration of explanations within software has shown the potential to address this opacity and can make the system more understandable to end-users. As a result, explainability has gained much traction as a non-functional requirement of complex systems.

Understanding what type of system requires what types of explanations is necessary to facilitate the inclusion of explainability in early software design processes. In order to specify explainability requirements, an explainability taxonomy that applies to a variety of different software types is needed. In this paper, we present the results of an online survey with 84 participants. We asked the participants to state their questions and confusions concerning their three most recently used software systems and elicited both explicit and implicit explainability needs from their statements. These needs were coded by three researchers. In total, we identified and classified 315 explainability needs from the survey answers.

Drawing from a large pool of explainability needs and our coding procedure, we present two major contributions of this work: 1) a taxonomy for explainability needs in everyday software systems and 2) an overview of how the need for explanations differs between different types of software systems.

**Index Terms**—Requirements Engineering, Explainability, Taxonomy, User Feedback

## I. INTRODUCTION

The development of software systems is becoming increasingly complex [1], [2]. This is reflected in the fact that development is carried out by teams of developers, some of whom are distributed globally [3]. Consequently, developers are confronted with the challenge of designing and implementing software solutions that not only meet functional requirements but also adhere to a multitude of non-functional requirements (NFRs) like security or usability. A relatively new non-functional requirement is explainability [4]–[6]. Explainability usually aims to make the decisions and results of algorithms transparent and understandable, especially in highly complex applications, such as in the fields of artificial intelligence (AI) and machine learning [7]–[10]. But even beyond machine learning, there are areas where users need

or demand an explanation, such as in relation to the quality aspect of privacy [11]. For example, if an app wants to access the smartphone’s contacts, many users want an explanation of why the app needs this access.

Explanations should be integrated carefully, as a random insertion of explanations at any point can have a negative effect on the user experience [4]. An understanding of the specific types of explanations required by different users in different software contexts is essential. It has been found that the elicitation of requirements is supported by having tools like quality models or checklists that can be used as a guide [12]. For example, the quality model ISO 25010 is an established tool for identifying non-functional requirements in the elicitation process. To simplify the elicitation of specific explanation requirements, a reference is needed that describes what kind of explanation needs exist.

A taxonomy that contains the different types of explanation needs can serve as a checklist, giving the requirements engineers guidance to discuss with the customer or users which explanations are desired [13]. In addition, a taxonomy offers a clearly defined terminology which helps to express explanation requirements in the further requirements engineering process. In order to create such a taxonomy, however, it is necessary to find out which types of explanation requirements actually exist in different software systems.

In this paper, we develop a taxonomy to categorize explanation needs of users facing everyday software systems. To achieve this goal, we conducted an online study with 84 participants and asked them about their three most recently used softwares. Based on their feedback, we identified explanation needs and categorised them throughout multiple labeling rounds.

Through our study and analysis we provide two main contributions:

- 1) an approach for identifying the need for explanation.
- 2) a taxonomy that can be used to categorize explanatory needs.

These contributions can help requirements analysts to specify explainability requirements in the development process.

The rest of the paper is structured as follows: In Section II, we present related work and background details. The study design is introduced in Section III. Section IV summarizes the results that are discussed in Section V, before concluding the paper in Section VI.

## II. BACKGROUND AND RELATED WORK

### A. Explainability

Explainability is a non-functional requirement (NFR) that is often considered in the context of AI systems [4]. However, systems without AI are also becoming increasingly complex, which is why explainability is also considered outside the explainable artificial intelligence (XAI) domain. Simplifying the definition by Chazette et al. [14], a system is described as explainable with respect to an aspect X if a corpus of information (the explanation) is given to an addressee A which enables A to understand X. Chazette et al. [14] do not define which aspects X should be explained, but give examples such as the behavior of the system or knowledge about the user. XAI methods focus on explaining the underlying model, more precisely the internal operations to justify decisions made [15]. Brunotte et al. [11] emphasize the importance of privacy explanations that reveal the purpose of using personal information. Besides algorithms and privacy, Deters et al. [16] also mention information regarding interactions with the system as a possible aspect that should be explained. Interaction explanation are intended to guide the user in using complex systems.

Explainability has an impact on many other NFRs. For example, explainability is often regarded as a means to increase trust in a system [15], [17], [18]. However, it was also found that explanations do not necessarily facilitate trust [19]. Likewise, explanations can have both a positive and a negative effect on usability and understandability [14]. The effect on transparency and learnability was found to be mainly positive [14]. Different types of explanations can therefore have different impacts on other NFRs.

### B. Taxonomies

A taxonomy is a classification system that is mostly used for animals and plants, but is also applied in the field of software engineering [20]. Classifications can help to identify gaps in a knowledge field, and they can enable a better understanding of the connections between objects [21]. Hierarchical taxonomies consist of one top class with several sub-classes which can also have further sub-classes. A true hierarchy ensures mutual exclusivity, which means that an entity belongs to exactly one class. This type of taxonomy is the most common in the software engineering sector [20]. Taxonomies in software engineering are most frequently developed for the knowledge areas of software construction, software design, software requirements and software maintenance [20]. Since explainability is considered an NFR, our taxonomy belongs to the group of software requirements.

### C. Related Work

Speith [22] conducted a review of taxonomies for XAI methods. He identified eleven representative papers that reference or propose a taxonomy for explainability methods. Ferreira et al. [23] developed a general taxonomy for explainability using an SLR. They describe three aspects of explainability. Firstly, they describe the reasons why explainability is introduced, secondly, they describe who receives explanations and thirdly, they describe the domain in which explainability is integrated. There are also several other taxonomies for designing and evaluating explainability [24]–[26]. However, all the above mentioned taxonomies do not refer to what types of explanations users need.

Unterbusch et al. [27] created a taxonomy for the need for explanation by analyzing 1730 app reviews of eight smartphone apps. They divide the need for explanation into primary and secondary concern. By primary concern, they mean that the need for explanation is the user's only issue. Secondary concern means that there is another issue that could be explained to increase the understanding of the situation, but cannot be solved by an explanation. For example, an error cannot necessarily be solved by an explanation, but understanding the error can reduce frustration. Using app reviews as the foundation for the taxonomy has two limitations that we try to overcome with our methodology. Firstly, the evaluation is limited to smartphone apps, which may have a different need for explanation than general software systems. Secondly, the threshold for writing an app review is quite high, which makes a bias in the reported problems likely. That means that mainly large problems, which create a high level of frustration that leads to writing a review, were considered. With our methodology, we try to cover everyday software systems and deal with all kinds of problems that lead to a need for explanation.

## III. STUDY DESIGN

### A. Research Questions

The main objective of our research is to identify and classify explanation needs of users regarding everyday software systems. Accordingly, our research in this work was framed by two research questions:

- RQ1 What types of needs for explanations do end-users have in everyday software systems?
- RQ2 How does the need for explanation differ between different types of software systems?

To answer *RQ1*, we conduct our online survey and process the resulting data set to build our taxonomy for explainability needs. We then apply the taxonomy to the same data set to examine differences between software types and answer *RQ2*.

### B. Instrument Development

We conducted a qualitative survey in the form of an online questionnaire to collect a comprehensive set of data. Earlier studies on explanation needs observed that explicit questions of whether explanations are desired in specific cases may lead

to an affirmation bias [28]. In order to elicit the needs for explanations without asking suggestively, we designed open-ended questions aimed at identifying needs for explanation that emerged in past uses. These questions allowed the participants to put themselves in a real context and thus capture their actual needs for explanation. To test the comprehensibility and effectiveness of the questionnaire, we conducted a pilot study with 8 participants. As a result, some questions were revised to improve their comprehensibility, whereas the structure of the questionnaire remained the same.

### C. Survey Structure

The survey can be divided into three sections. In the first section, participants were asked to indicate which three software systems they had used most recently. In the second section, the participants were then asked six questions about each of the software systems they indicated. The first question was designed to get participants to think back to when they used the system. The remaining five questions were aimed at determining at which points explanations were needed. The questions were posed in different ways in order to activate the participants' memories. Lastly, the third section captured demographics. The entire questionnaire is included in our supplementary material [29].

### D. Data Collection

We offered the survey in both German and English via LimeSurvey<sup>1</sup>. The survey was open from January 2023 to March 2023 and was distributed via networking platforms (like LinkedIn) and the institution's bulletin board. Since our target group was all adults with access to technology and there were no other requirements for participation, convenience sampling was appropriate.

Three types of data were collected - firstly, demographic data such as gender, age and field of work. Secondly, software systems that were most recently used were collected - this data consists of free text responses, usually consisting of one or two words. Finally, we asked users about various aspects of using their software. For example, what frustrated them about using the software, what confused them, whether they had or have questions about the software and whether they would like to know anything else about the software. By asking these questions, we hope to be able to indirectly elicit the need for explanations from users. This data was collected in the form of free text answers consisting of complete sentences or bullet points.

### E. Demographics

A total of 84 participants completed the survey. Surveys that were not fully completed were removed from the data set. This was done to ensure that answers to certain parts of the questionnaire would not outweigh others throughout the later coding procedure. 35% of the participants were female, 64% were male and 1% was diverse. The average age was 37,8 (min: 18 years, max: 72 years, SD: 16,3). 17% of

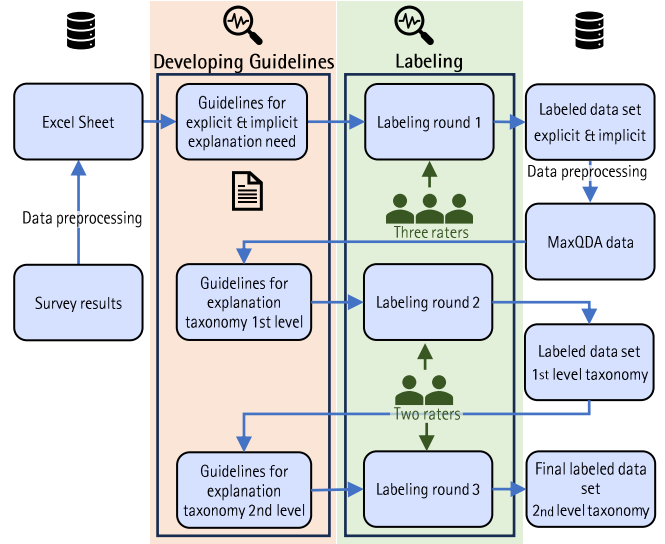


Fig. 1. Approach of our data analysis

participants were studying, 55% were working and 14% were both studying and working. 14% of the participants stated that they were neither studying nor working.

### F. Data Analysis

An overview of our data analysis approach can be seen in Figure 1. We exported the results of our survey via LimeSurvey into an Excel spreadsheet. Three of the authors of this paper then pre-processed the survey answers for the need for explanation. The three raters labeled each of the participants' responses that could contain needs for explanation. The label categories were: explicit explanation need, implicit explanation need, unspecific explanation need and no explanation need.

Explicit explanation need involves the explicit statement of a need for explanation. An example would be if a user gave the following answer when asked if he or she had any questions about a software: "Yes, how can I quote a group message and then reply to it?". An implicit explanation need, on the other hand, can be recognized by certain trigger words, e.g. when something is not obvious to the participant, is not comprehensible or seems questionable. This could be, for example, the following feedback of an user regarding a software: "Some functions are not immediately obvious and you need to follow specific instructions". Unspecific explanation need refers to explanation need where a participant responds that he or she has some kind of explanation need without specifying what the need is. In order to be as objective as possible when labeling according to these categories, we have developed guidelines containing indicators and examples for the four labeling categories. These guidelines were derived by labeling the first five participants' responses in the data set and then discussing the labeling results.

The three raters then labeled the entire data set in two cycles (50% each). In order to prepare for the second labeling round, we also tried to find suitable categories during the first labeling

<sup>1</sup><https://www.limesurvey.org/>

round. Therefore, during the first labeling round, the three raters noted possible categories for the identified explanation needs. After labeling, the raters reviewed the categories and developed a taxonomy based on these categories. Using this taxonomy, the raters reanalyzed the first five data points, discussed the labeling results and adjusted the taxonomy accordingly. We used Excel for the first labeling round.

The entire data set was then re-labeled in two cycles (again 50% in each cycle), this time by two raters. If there was disagreement between the two, the third rater was consulted in order to reach an agreement. During this second labeling process, the raters wrote down possible subcategories of the taxonomy categories from the previous labeling round. Finally, the two raters re-labeled these subcategories based on the first five participant responses. In the end, the complete data set was re-categorized into these subcategories in one cycle by two raters. Again, the third rater was consulted in order to reach an agreement in case of disagreement. We used the coding software MAXQDA<sup>2</sup> for second and third labeling round.

Finally, all software systems mentioned by the participants were categorized into software type categories by two raters according to an existing taxonomy for software types [30]. There were no disagreement between the two raters.

#### G. Interrater Agreement

After each labeling step, we resolved any disagreement between raters. We used two methods to measure disagreement. To measure the extent to which the three raters agree in their labeling results during the first labeling round, we calculated *Fleiss' Kappa* ( $\kappa$ ) [31] as agreement value. Fleiss'  $\kappa$  measures the interrater agreement between more than two raters [31]. For the other labeling rounds, where two raters labeled, we calculated Brennan & Prediger  $\kappa$  (B & P  $\kappa$ ) [32], an adapted kappa from Cohen's  $\kappa$  [33].

We classify the  $\kappa$  value according to Landis and Koch [34]. Furthermore, we calculated the proportional agreement between the raters during the first labeling round. That is the proportion between the agreement of the three raters and the total amount of data to be labeled. For example, if the raters labeled a data set and agreed 70 times on the label and disagreed 30 times, the proportional agreement between them would be 70%.

#### H. Data Availability Statement

To enable the verification and replication of our work, we provide a supplementary material [29]. In this material, the questionnaire and coding guidelines used in our study are available. Furthermore, the material includes a CSV-file that contains all software systems mentioned by the participants of our study with the corresponding system type and explainability need codes, including categories and subcategories. Participants' uncoded survey answers cannot be shared openly, due to privacy concerns. However, they are available from the first author of this paper on reasonable request. All samples

TABLE I  
INTERRATER AGREEMENT VALUES DURING LABELING

	Round 1			Round 2			Round 3
% Data	0-50	50-100	100	0-50	50-100	100	100
# Raters	3	3	3	2	2	2	2
Fleiss $\kappa$	0.87	0.81	0.84				
B & P $\kappa$				0.81	0.74	0.77	0.76
Proportion	0.87	0.81	0.84	0.83	0.76	0.79	0.77

of uncoded survey answers found within this paper have been adapted to ensure the anonymity of our participants.

## IV. RESULTS

### A. Types of Software Systems

Participants answered the questionnaire with regard to their three most recently used software systems. In accordance with our sample size of 84 respondents, we collected 252 names of software systems. Notably, the same application could be mentioned by multiple participants. We labeled the reported systems in accordance with the software type taxonomy proposed by Forward and Lethbridge [30]. An overview of the software types can be seen in Figure 2. Due to the large number of different software types, we classified software types that appeared less than 10 times as "other", to increase the readability of this paper. The detailed classification of each individual software named by our participants can be found in our supplementary material [29].

The participants named systems that they use in their daily lives. Unsurprisingly, the majority of reported systems were consumer-oriented software (162). This category includes communication and information software (94) such as messenger apps and e-mail programs, software for productivity and creativity (37) such as office applications, and entertainment and education software (26) like media streaming services. The second most prevalent software type was applications for information display and transaction entry (57). Examples of these are web applications (56) such as social networks, e-finance and user-generated content like image boards. The remaining prominent categories were systems software (11) such as kernels, password managers and anti-virus applications, and design and engineering software (10) like video and music composition tools.

### B. Taxonomy for Explainability Needs

The main contribution of this work is the taxonomy that we developed as a result of our coding procedure. We report our interrater agreement in Table I.

The  $\kappa$  values are always above 0.70, often even above 0.80. This means that we interpret the kappa values as substantial to almost perfect agreement [34]. The proportional agreement values are also over 75%, which show that in most of the time the two or three authors agreed.

We found five categories of explainability needs within our data set, which we divided into 11 subcategories. These categories and subcategories are displayed in Figure 3. We

<sup>2</sup><https://www.maxqda.com/>

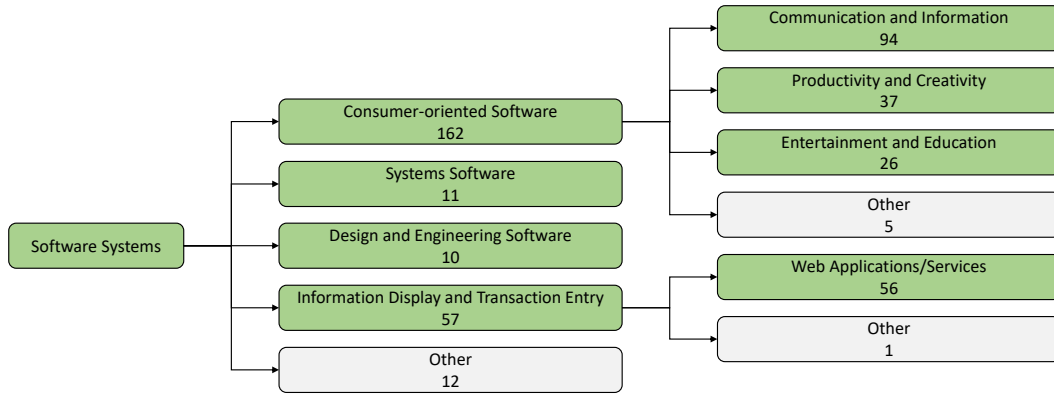


Fig. 2. Software types examined in our study (classified according to Forward and Lethbridge [30])

were able to assign each explicit and implicit explainability need to at least one subcategory. Examples of each kind of explainability need are provided in Table II. The classification of each individual need and the coding guidelines for each category can be found in our supplementary material [29].

1) *System Behavior*: The first category is the need to explain *System Behavior*. This covers explaining how the software works and why it behaves in certain ways. *Unexpected System Behavior* can be a major cause of confusion and frustration for end-users. This need for explanation arises when the observed system behavior deviates from what the user expects. In specific cases, users may directly relate unwanted system behavior to *Bugs & Crashes*. Explanations can help users understand these errors and enable them to work on a solution. On other occasions, users might wonder about the *Consequences* of their inputs, i. e., how the system will respond if they perform a certain action. Commonly used in XAI are explanations of an *Algorithm* and its outputs. However, these explanations may also be required for complex algorithms outside of AI. Explaining how and why a software behaves in certain ways could increase the understandability and transparency of the system.

2) *Interaction*: The second category is the need to explain *Interactions* between the end-user and the system. Specifically, users want to know how a certain *Operation* with the software can be performed. Operations may concern all kinds of software functionalities, such as the user making inputs or ordering the system to perform tasks. Operation explanations are differentiated from explanatory needs concerning the *Navigation* within the software. Instead of specific operations on software functionalities, users want to learn how to reach and access different views or subsystems within a system. In other cases, where users are concerned with learning how to use a new system or new features from scratch rather than understanding a specific operation, we categorize the explanatory need as a need for a *Tutorial*. Explaining how a system can be used and navigated has the potential to increase the usability and learnability of the system.

3) *Domain Knowledge*: The third category of explainability needs is concerned with *Domain Knowledge*. Even experi-

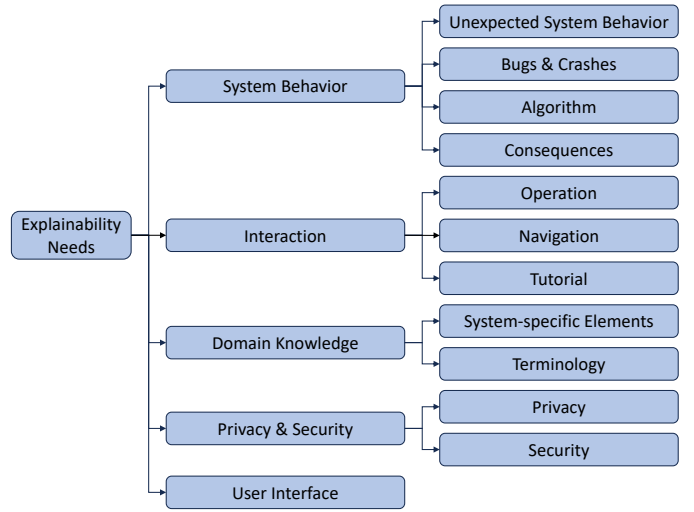


Fig. 3. Taxonomy of explainability needs in everyday software systems

enced software users might run into problems if they lack knowledge about the domain of the system. For example, a tech-savvy user might still need help when using tax preparation software. This could be the case if the user needs explanations for certain *Terminology* within the software. Furthermore, explanations for *System-specific Elements* might be required. This is a broad subcategory as it includes all kinds of elements unique to an individual system. Examples could be explaining the privileges of different types of accounts or why specific kinds of information are needed for a registration. Providing explanations concerning domain knowledge could increase end-users' understanding of the software and improve usability.

4) *Privacy & Security*: The fourth category contains the need to explain *Privacy & Security* aspects of a system to end-users. Privacy and security are closely related non-functional requirements. Needs for *Privacy* explanations may include explaining if and how the user's data is stored and processed within a software system. The *Security* subcategory deals with explaining what measures are taken to ensure the

TABLE II  
DISTRIBUTION OF EXPLAINABILITY NEEDS ACROSS ALL SOFTWARE TYPES AND EXAMPLE NEEDS

Type of need	#	%	Example adapted from the data set
All needs combined	315	100%	
<b>Interaction</b>	<b>159</b>	<b>50.5%</b>	
Operation	105	33.3%	“How can I set filters to automatically assign my e-mails to certain folders?”
Navigation	30	9.5%	“Sometimes it is unclear where my podcasts are being saved.”
Tutorials	24	7.6%	“Due to the large number of features, it took me a long time to learn to use it efficiently.”
<b>System behavior</b>	<b>95</b>	<b>30.2%</b>	
Unexpected system behavior	50	15.9%	“I was wondering why I could not access the events of a shared calendar.”
Bugs & crashes	24	7.6%	“The error messages are not comprehensible.”
Algorithm	15	4.8%	“I am unsure about the capabilities of the search functionality.”
Consequences	6	1.9%	“I was wondering if my messages would arrive if the person unblocked me.”
<b>Domain knowledge</b>	<b>27</b>	<b>8.6%</b>	
System-specific elements	14	4.4%	“Why aren’t all videos available through my business account?”
Terminology	13	4.1%	“The names of settings are unclear. Who simply knows what ‘experimental sharpening’ is?”
<b>Privacy &amp; security</b>	<b>23</b>	<b>7.3%</b>	
Privacy	15	4.8%	“It is unclear to me exactly which parts of my personal data are stored.”
Security	8	2.5%	“How safe is the end-to-end encryption?”
<b>User Interface</b>	<b>11</b>	<b>3.5%</b>	“I got confused after an update, as the positions of some functions were shifted.”

security of the data and the system. This covers explaining protective measures such as encryption, but also informing users about vulnerabilities. Explaining aspects of privacy and security increases transparency and has the potential to foster end-user trust.

5) *User Interface*: The fifth and final category is the need to explain the *User Interface*. Users may wonder why certain design decisions were made within the user interface. Furthermore, changes in the interface, which may be caused by updates or patches, can be a cause for confusion. Explaining how and why an interface changed could potentially increase the usability and learnability of the interface.

### C. Explainability Needs in the Data Set

The prevalence of different types of explainability needs within our taxonomy varied in general, but there were also distinct differences between different software types. Notably, we managed to categorize 100% of the previously identified implicit and explicit explainability needs using our taxonomy.

1) *Explainability Needs across all Software Types*: The distribution of the explainability needs across all software types, ordered by numbers of appearance, is shown in Table II.

50.5% of the explainability needs stated by our participant were categorized as *Interaction* explanation needs. At 33.3%, the most ubiquitous subcategory was *Operation* explanations. The other two subcategories appeared much less often, with *Navigation* at 9.5% and *Tutorials* at 7.6%.

The need to explain *System Behavior* was the second most prevalent category at 30.2%. Here, the most common type of need was the explanation of *Unexpected System Behavior*. Needs to explain *Bugs & Crashes* were expressed at 7.6%, while the need to explain an *Algorithm* or its output appeared only at 4.8%. The need to explain *Consequences* was the least common subcategory overall at only 1.9%.

The remaining three categories of explanatory needs were much less prevalent than the previous two. Explanations on *Domain Knowledge* made up only 8.6% in total. The subcategories were almost evenly spread, with the need to explain *System-specific Elements* at 4.4% and *Terminology* explanations at 4.1%.

Explanations for *Privacy & Security* aspects made up 7.3% of explainability needs. At 4.8%, most of these were *Privacy* concerns. The remaining 2.5% covered the need for *Security* explanations. *User Interface* explanations as a standalone category appeared only at 3.5%.

TABLE III  
RELATIVE NUMBER OF EXPLAINABILITY NEEDS  
FOR EACH SOFTWARE TYPE

Software type	#Needs / #Type	Rel. #Needs
<b>Consumer-oriented Software</b>	<b>207/162</b>	<b>1.28</b>
Communication and Information	118/94	1.26
Productivity and Creativity	44/37	1.19
Entertainment and Education	38/26	1.46
<b>Info. Display and Transaction Entry</b>	<b>70/57</b>	<b>1.23</b>
Web Applications/Services	70/56	1.25
<b>Systems Software</b>	<b>10/11</b>	<b>0.91</b>
<b>Design and Engineering Software</b>	<b>14/10</b>	<b>1.4</b>

2) *Explainability Needs between Software Types*: We found explainability needs for all software types that we examined (see Figure 2). The relative number of explainability needs, depending on the software type, is displayed in Table III.

At 0.91 needs per software mentioned, systems software had the lowest number of needs per appearance. That means that when a participant reported about a systems software, we identified 0.91 explainability needs on average. The next highest value is found in software for communication and creativity, for which we identified 1.19 needs on average. We

TABLE IV  
DISTRIBUTION OF EXPLAINABILITY NEEDS ACROSS DIFFERENT SOFTWARE TYPES.  
(\* THIS ALSO INCLUDES SUBCATEGORIES APPEARING LESS THAN 10 TIMES, CATEGORIZED AS “OTHER”.)

	System behavior		Interaction		Domain Knowledge		Privacy & Security		User Interface		Total
	#	%	#	%	#	%	#	%	#	%	#
<b>Consumer-oriented Software *</b>	<b>64</b>	<b>30.9%</b>	<b>101</b>	<b>48.8%</b>	<b>18</b>	<b>8.7%</b>	<b>18</b>	<b>8.7%</b>	<b>6</b>	<b>2.9%</b>	<b>207</b>
Communication and Information	40	33.9%	52	44.1%	5	4.2%	16	13.6%	5	4.2%	118
Productivity and Creativity	7	15.9%	33	75%	3	6.8%	0	0%	1	2.2%	44
Entertainment and Education	15	39.5%	15	39.5%	6	15.8%	2	5.3%	0	0%	38
<b>Information Display and Transaction Entry *</b>	<b>22</b>	<b>31.4%</b>	<b>37</b>	<b>52.9%</b>	<b>5</b>	<b>7.1%</b>	<b>2</b>	<b>2.9%</b>	<b>4</b>	<b>5.7%</b>	<b>70</b>
Web Applications/Services	22	31.4%	37	52.9%	5	7.1%	2	2.9%	4	5.7%	70
<b>Systems Software</b>	<b>4</b>	<b>40%</b>	<b>2</b>	<b>20%</b>	<b>1</b>	<b>10%</b>	<b>3</b>	<b>30%</b>	<b>0</b>	<b>0%</b>	<b>10</b>
<b>Design and Engineering Software</b>	<b>3</b>	<b>21.4%</b>	<b>10</b>	<b>71.4%</b>	<b>1</b>	<b>7.1%</b>	<b>0</b>	<b>0%</b>	<b>0</b>	<b>0%</b>	<b>14</b>

found a similar number for web applications and services, as well as software for communication and information, which had an average of 1.25 and 1.26 needs respectively. The highest needs were identified for design and engineering software at 1.4 and for entertainment and education software at 1.46.

The distribution of the explainability needs between different software types is shown in Table IV. We do not display software systems categorized as “other” individually, as they would hardly be comparable to the defined categories and subcategories. However, we included the “other” subcategories in the numbers shown for the overarching categories, i.e., consumer-oriented software and software for information display and transaction entry. In the case of software for information display and transaction entry, this made no difference, as the single data point from the “other” subcategories added no new explainability needs.

In this paper, we report the relation of software types to the categories of explanation need, but not the subcategories. This is necessary as the display of more detailed data would be much more space-consuming and hard to read. For more specific results, please refer to the detailed coding data in our supplementary material [29]. The percentages stated hereinafter refer to the distribution of the different explainability needs within each type of software system.

The need to explain system behavior is prevalent across all software types. Systems software has 40% of its explanatory needs concerned with system behavior. Closely following is entertainment and education software at 39.5%. Software for communication and education has 33.9% share of behavior explanation needs, whereas web applications and services come in at 33.4%. The category is less prevalent in design and engineering software (21.4%), as well as in software for productivity and creativity (15.9%).

Interaction explanations, which are the largest category overall, were by far the most prevalent in software for productivity and creativity (75%), and in design and engineering software (71.4%). Web applications and services have approximately half (52.9%) of their needs in the interaction category,

whereas communication and information software has 44.1% of theirs. Entertainment and education software has 39.5% interaction explanation needs. Systems software only has 20% of its explainability needs concerned with interaction, and is the only type of system where the category does not cover the largest share of needs.

Domain related explanations covered a far smaller share of the explanation needs, also between software types. Their largest share was in software for entertainment and education, where they covered 15.8% of needs. Systems software and software for design and engineering both had a single occasion of domain explanation need, which corresponds to 10% and 7.1% respectively. Web applications and services also had a 7.1% share of their needs in the domain category, but a larger total number of occurrences. Finally, domain explanation needs were less prevalent in software for productivity and creativity at 6.8% and least prevalent in communication and information software at 4.2%.

Explanation needs concerning privacy and security did not appear for all software types. More specifically, we found no need in software for productivity and creativity, as well as in design and engineering software. Their largest shares are found in systems software at 30% and in software for communication and information at 13.06%. They also appeared in software for entertainment and education at 5.3% and in web applications and services at 2.9%.

The smallest category of needs, explanations about the user interface, was only found in three types of software systems. In web applications and services, they covered 5.7% of explanation needs. Software for communication and information had a 4.2% share of interface explanation needs, whereas software for productivity and creativity only had a 2.2% share, which was a single occurrence. Software for entertainment and education, as well as systems software and design and engineering software did not have any need assigned to them.

## V. DISCUSSION

### A. Answering the Research Questions

1) *RQ1: What types of needs for explanations do end-users have in everyday software systems?*: Our findings show that end-users have a variety of explainability needs regarding everyday software systems. We found the most prominent category of needs to be the need for *Interaction* explanations. While not typically in the focus of contemporary explainability research, the majority of our participants' concerns and questions were related to issues with the operation or navigation of a software, or our participants expressed the need for a tutorial. Explaining how to operate a system and providing appropriate tutorials could be an effective measure to increase the learnability and usability of a system.

Explaining *System Behavior* is more common in explainability research, specifically in the field of XAI. This covers algorithms and their outputs, but also bugs and crashes or otherwise unexpected system behavior. Furthermore, some of our participants wondered about the possible consequences of inputs they could make. The goal of explaining system behavior is typically to increase the understandability and transparency of a software, and thereby foster trust in end-users.

Besides these two large categories, we also found needs for explanations in the areas of *Domain Knowledge*, *Privacy & Security* and for *User Interfaces*. Domain-specific terminology or system-specific elements can pose a barrier to entry for users, even if they are generally well-versed in technology. Explaining these terms and elements could mitigate this and enable users to effectively use the systems. Similarly, concerns about privacy and security might push users not to engage with a system, even if the concerns are unwarranted. Providing privacy and security explanations could address these concerns and foster trust in the stakeholders of the system. Confusing or non-intuitive *User Interfaces* can frustrate users and impede usability. However, in the case of complex software systems, this might not always be easily avoidable. In those cases, explaining why an interface is designed in a certain way, or why and how specific elements were updated could help the user learn to use the system.

2) *RQ2: How does the need for explanation differ between different types of software systems?*: The results of our coding procedure show that there is a notable difference in explainability needs between different types of software systems. For instance, while the need for interaction explanations constitutes the largest category, it is not the most prevalent type of need in systems software and only tied for most prevalent in entertainment and education software. While the sample size for systems software is admittedly low, it makes sense that the interaction explanations are less needed in entertainment and education software, which is supposedly easy to use.

Indeed, the share of interaction explanation needs is largest in software for productivity and creativity, and in software for design and engineering. This is consistent with these systems offering a variety of tools to work with, some of which might

not be intuitive to use. Incidentally, these are also the two types of systems in which the need to explain system behavior is the lowest. This makes sense as most of the system's behavior would be a direct consequence of the user's inputs and the tools they chose to process their inputs.

Software for entertainment and education often comes with system-specific elements like account and subscription models, or different forms of organizing content. Explaining the difference between these models and the terminology behind them makes the software more understandable and transparent to end-users. Accordingly, the need for domain explanations was most prevalent in this type of software. While a share of 10% in systems software is also relatively high, this only accounts for a single need and is therefore caused by the low sample size of system softwares.

Compared to other software types, systems software had a large share of privacy and security explanation needs. This makes sense as systems software also includes anti-virus applications and password managers. Software for communication and information also shows a high need for privacy and security explanations. This includes software such as messenger apps and e-mail systems, through which end-users might store and share sensible, personal information. Privacy and security explanations are needed to provide users with the necessary transparency on how their data is being stored and processed.

With one exception in software for productivity and creativity, all needs for explanations concerning the user interface appeared in either software for communication and information, or in web applications and services. Arguably, this could be caused by both communication software as well as web applications being highly relying on their visual interfaces. However, as this is the least prevalent category of explainability need, it is hard to say whether this conclusion holds any merit or if the observation occurred by chance.

### B. Discussion of the Results

1) *Applicability of the Taxonomy*: Constructing a taxonomy from empirical data is typically subject to a number of limitations:

- the content of the statements made by our study participants is influenced by the questions we asked
- the applicability of the taxonomy is limited by the sample of study participants
- the applicability of the taxonomy is limited by the context of the study
- the robustness of the taxonomy depends on the quality of the coding procedure

We designed the questions in our survey as open as possible, and avoided leading questions to influence our participants' answers as little as possible. For example, asking if system behavior had confused or frustrated our participants would likely have led to a larger share of explanation need concerning system behavior. To avoid this, we instead asked our participants if *anything* concerned with the software had frustrated or confused them before.

The demographic distribution of our participants is not entirely proportionate to society at large (two thirds male, one third students). Our target demography was adults with access to technology, from which we were able to cover the most prevalent subgroups in terms of gender identity, employment status and age. Consequently, we are confident that our taxonomy covers the software types and explainability needs that are most relevant to our target demography.

Our taxonomy is mainly applicable to everyday software systems, which was the context of our study. Within the context of everyday systems, our taxonomy is limited by the software types mentioned by our participants. As the responses to our study cover a variety of different software types, we are confident that our taxonomy is applicable with the desired context of everyday software systems. In a time of mobile applications and internet, this notion is underlined by the fact that the most frequently mentioned software types were consumer-oriented software, as well as web applications and services.

Lastly, the taxonomy is only applicable if the underlying coding procedure is well performed and results in sufficiently high interrater agreements. Before performing each of our three labeling rounds, we designed and discussed examples and our guidelines until there was no longer disagreement between the raters. During the first two labeling rounds, we coded the data set in two halves and re-evaluated our guidelines in between, to ensure the robustness of our coding procedure. During the third labeling round, we were confident to label the whole data set at once, without re-evaluating the guidelines in between. This decision was justified by the high interrater agreements that we achieved in the previous two labeling rounds. Throughout our entire coding procedure, our interrater agreements ranged between 0.74 and 0.87, which is considered to be substantial to almost perfect agreement [34]. Considering the large amount of statements to be coded and the number of categories and subcategories, we are confident that our taxonomy is build upon a robust data basis. We attribute these results to our rigorous labeling procedure and detailed coding guidelines.

In summary, our findings show that our taxonomy is applicable to everyday software systems used by adults, and that it could be a valuable tool for requirements engineers that are looking to integrate explainability into everyday software systems.

2) *Software Types as a Factor in Explainability Engineering*: Whether an explanation is appropriate for a certain scenario depends on a variety of factors. Previous works in explainability research have stated the importance of considering the addressee when providing explanations [35], [36]. In essence, they argue that the characteristics of the end-user determine what kind of explanation is required. Other works have highlighted the importance of the context of use [37], [38]. Here, the idea is that whether an explanation is appropriate or not depends on what context the software is used in. For example, in safety-critical situations, short and concise explanations might be preferred over long and

detailed explanations. Yet another work proposes that the goal of the explainer plays an important role when integrating explanations [39].

To our knowledge, this work is the first to examine different types of software systems as a factor in explainability engineering. While limited to everyday software systems, our results show that explainability needs indeed vary between different software types. Naturally, one would assume that more complex software needs more explanations concerning its behavior, and that more data-driven software comes with a larger need for privacy and security explanations. However, these factors have not been systematically examined through empirical data before.

3) *Explainability beyond System Behavior*: A large section of explainability focuses on the explanation of system behavior. Specifically, the field of XAI provides a large part of the existing explainability literature. AI systems are typically very complex and often considered to be opaque black-boxes [40]. As a result, explainability in XAI is sometimes equated to interpretability [14], which describes how well the decisions and outputs of a system can be comprehended [41]. Following our taxonomy, these are aspects of explaining system behavior, more specifically the XAI algorithms and their outputs.

Our results show that explainability requirements are not limited to AI, but apply to different kinds of software systems with varying degrees of complexity. Furthermore, not all explainability requirements are concerned with the explanation of system behavior. While system behavior needs are a large share of the needs we identified, they are second to the category of interaction explanations. According to our data, explanations for interactions need to move into the focus of requirements engineering for everyday software. Especially in the case of complex interactions, simply providing good usability is not the same as explaining how to operate or navigate a system. A high degree of usability may decrease the need for interaction explanations, but developing a software system that is highly usable for every user group is unrealistic. However, carefully elicited requirements for interaction explanations can support inexperienced users by providing guidance if needed and would not impede the experience of long-time users as long as they are optional.

4) *Addressing Bias in Requirements Elicitation*: Raising explainability requirements in the early development stages is challenging. Without the completed software at hand, stakeholders would have to rely on tacit knowledge to express the need for explanations. Mentally putting oneself into a role or scenario that has not been experienced before may introduce hypothetical bias [16], [42], and hinder the requirements elicitation process. Furthermore, asking if explanations are needed in specific cases may lead to affirmation bias [28], which could wrongfully motivate the implementation of unneeded explanations.

We addressed these challenges by focusing on participants' past experiences with software systems they actually use. We let them set the context of use for themselves and motivated them to report past experiences that typically coincide with

explicit and implicit needs for explanations. This way, we were able to raise a variety of explainability requirements that are based on actual user needs, rather than tacit knowledge.

### C. Limitations and Threats to Validity

In the following, we present threats to validity according to our study. We categorize the threats according to Wohlin et al. [43] as construct, internal, external, and conclusion validity.

*Construct Validity.* Categorizing software systems into domain categories might introduce subjectivity. The software types were coded by two authors, and there was no disagreement, but the coding could still be influenced by the raters' perspectives. The categories developed during the labeling of explainability needs and the subsequent taxonomy may not fully capture the complexity and diversity of explanation needs in software systems. As we achieved overall high interrater agreements, and were able to code all requirements, we are confident that our taxonomy is complete within the context of this work. As we used a survey as the only method of data collection, our results might be influenced by mono-method bias. In the future, we plan to not only elicit explainability requirements via self-reported needs, but also via other triggers, such as behavioral patterns and physiological triggers.

*Internal Validity.* The conclusions drawn from the self-reported needs of our participants and the developed taxonomy might be influenced by our coding procedure. Our labeling process, despite guidelines and discussions, and despite our high interrater agreement, may still introduce subjectivity. The interrater agreement measures (Fleiss' Kappa and Brennan & Prediger Kappa) should be scrutinized for potential disagreements. Participants were asked to recall their experiences with software systems that they used in the past. Memory inaccuracies or biases may affect the reliability of their responses. We tried to address this by not focusing our study on a specific type of system, but instead having them report about their three most recently used applications.

*Conclusion Validity.* A sample size of 84 participants might not be sufficient to cover all explainability needs in the entire area of "everyday" software systems. However, every participant reported three systems, making for a total amount of 252 reported systems, and we found a variety of different software types within this data. With regard to these systems, we identified a total of 315 explainability needs, which we consider a sufficient number for the purposes of our analysis. Hence, while we cannot claim that our conclusion apply to *all* types of everyday software systems, we are confident that they cover a significant number of system types.

*External Validity.* We did not perform a hands-on validation of our taxonomy in practice or on other user feedback. Thus, we cannot claim the applicability of the taxonomy in practice. However, as our results are based on participants' real experiences, we are confident that our results are robust within the context of the software types we examined. Convenience sampling was used, and the target group was the entire population. Still, the results may not be generalizable to the broader population, particularly if the sample has specific

characteristics. Furthermore, the distribution of participants is not always proportionate (e.g., two thirds are male). The only constraint for participation was that participants had to be adults with access to technology, which we adhered to, but there might still be unknown demographic factors that should be considered. Furthermore, we did not collect any data for software developed specifically for minors (under 18 years old) as part of the study, so our results cannot be generalized for this group of people. The use of networking platforms and an institution's bulletin board might introduce bias, as certain demographics may be overrepresented or underrepresented. Certain software domains may be underrepresented or not present at all. As this work reports on a diverse variety of software types and due to our high interrater agreements, we are still confident that our results are applicable in the context of everyday software applications.

## VI. CONCLUSION

For this paper, we conducted an exploratory study to identify the explainability needs of end-users in software systems that they actually use in their daily lives. Through an online survey with 84 participants, we identified 315 explainability needs in various types of software systems. Through a three-step coding procedure with three raters, we categorized the explainability needs stated by our participants into five categories and 11 subcategories. These categories and subcategories form the basis for our taxonomy of explainability needs. To examine explainability needs in relation to software type, we categorized the software systems into software types according to an existing taxonomy [30] and applied our own taxonomy to the categorized data set.

Our results show that explainability needs exist for various types of everyday software, not only for AI systems. Furthermore, we identified a variety of explainability needs beyond explaining how a system works or why it arrived at a certain output. Specifically, we find that the need to explain system behavior, which is typically in the focus of XAI research, is not the most prevalent type of explainability need in everyday software systems. Instead, everyday software has its most prevalent explainability need in the need to explain end-users' interactions with the software. Different types of software systems lead to different kinds of explainability requirements. In this light, explainability needs explicit consideration in the requirements engineering process, in order to provide appropriate explanations for each type of software.

As the next step, we plan to validate the applicability of our taxonomy by applying it in a practical setting. To this end, we will integrate the taxonomy into the development process of small-scale software projects. Furthermore, we want to investigate the influence of cultural differences on the need for explanations, as well as the influence of common demographic factors such as age and affinity for technology. Lastly, we will research the elicitation of explainability requirements via triggers such as behavioral patterns and physiological triggers.

## ACKNOWLEDGMENTS

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Grant No.: 470146331, project softXplain (2022-2025).

## REFERENCES

- [1] R. E. Kraut and L. A. Streeter, "Coordination in software development," *Commun. ACM*, vol. 38, no. 3, p. 69–81, Mar. 1995.
- [2] D. E. Perry, N. A. Staudenmayer, and L. G. Votta, "People, organizations, and process improvement," *IEEE Software*, vol. 11, no. 4, pp. 36–45, 1994.
- [3] M. Kuhrmann, P. Tell, J. Klünder, R. Hebig, Sherlock A. Licorish, and S. G. MacDonell, "Helena stage 2 results," 2018.
- [4] L. Chazette and K. Schneider, "Explainability as a non-functional requirement: challenges and recommendations," *Requirements Engineering*, vol. 25, no. 4, pp. 493–514, Dec 2020. [Online]. Available: <https://doi.org/10.1007/s00766-020-00333-1>
- [5] L. Chazette, J. Klünder, M. Balci, and K. Schneider, "How can we develop explainable systems? insights from a literature review and an interview study," in *Proceedings of the International Conference on Software and System Processes and International Conference on Global Software Engineering*, ser. ICSSP'22. New York, NY, USA: Association for Computing Machinery, 2022, p. 1–12. [Online]. Available: <https://doi.org/10.1145/3529320.3529321>
- [6] M. A. Köhl, K. Baum, M. Langer, D. Oster, T. Speith, and D. Bohlender, "Explainability as a non-functional requirement," in *2019 IEEE 27th International Requirements Engineering Conference (RE)*, 2019, pp. 363–368.
- [7] R. Confalonieri, L. Coba, B. Wagner, and T. R. Besold, "A historical perspective of explainable artificial intelligence," *WIREs Data Mining and Knowledge Discovery*, vol. 11, no. 1, p. e1391, 2021. [Online]. Available: <https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/widm.1391>
- [8] P. P. Angelov, E. A. Soares, R. Jiang, N. I. Arnold, and P. M. Atkinson, "Explainable artificial intelligence: an analytical review," *WIREs Data Mining and Knowledge Discovery*, vol. 11, no. 5, p. e1424, 2021. [Online]. Available: <https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/widm.1424>
- [9] N. L. Goodwin, S. R. Nilsson, J. J. Choong, and S. A. Golden, "Toward the explainability, transparency, and universality of machine learning for behavioral classification in neuroscience," *Current Opinion in Neurobiology*, vol. 73, p. 102544, 2022. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0959438822000381>
- [10] N. Burkart and M. F. Huber, "A survey on the explainability of supervised machine learning," *Journal of Artificial Intelligence Research*, vol. 70, pp. 245–317, 2021.
- [11] W. Brunotte, A. Specht, L. Chazette, and K. Schneider, "Privacy explanations – a means to end-user trust," *Journal of Systems and Software*, vol. 195, p. 111545, 2023. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0164121222002217>
- [12] J. Doerr, D. Kerkow, T. Koenig, T. Olsson, and T. Suzuki, "Non-functional requirements in industry - three case studies adopting an experience-based nfr method," in *13th IEEE International Conference on Requirements Engineering (RE'05)*, 2005, pp. 373–382.
- [13] K. Lauenroth, E. Kamsties, and O. Hehlert, "Do words make a difference? an empirical study on the impact of taxonomies on the classification of requirements," in *2017 IEEE 25th International Requirements Engineering Conference (RE)*, 2017, pp. 273–282.
- [14] L. Chazette, W. Brunotte, and T. Speith, "Exploring explainability: A definition, a model, and a knowledge catalogue," in *2021 IEEE 29th International Requirements Engineering Conference (RE)*, 2021, pp. 197–208.
- [15] A. Das and P. Rad, "Opportunities and challenges in explainable artificial intelligence (XAI): A survey," *CoRR*, vol. abs/2006.11371, 2020. [Online]. Available: <https://arxiv.org/abs/2006.11371>
- [16] H. Deters, J. Droste, M. Fechner, and J. Klünder, "Explanations on demand - a technique for eliciting the actual need for explanations," in *2023 IEEE 31st International Requirements Engineering Conference Workshops (REW)*, 2023, pp. 345–351.
- [17] F. Rossi, "Building trust in artificial intelligence," *Journal of international affairs*, vol. 72, no. 1, pp. 127–134, 2018.
- [18] D. Shin, "The effects of explainability and causability on perception, trust, and acceptance: Implications for explainable ai," *International Journal of Human-Computer Studies*, vol. 146, p. 102551, 2021. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1071581920301531>
- [19] L. Kästner, M. Langer, V. Lazar, A. Schomäcker, T. Speith, and S. Sterz, "On the relation of trust and explainability: Why to engineer for trust-worthiness," in *2021 IEEE 29th International Requirements Engineering Conference Workshops (REW)*, 2021, pp. 169–175.
- [20] M. Usman, R. Britto, J. Börstler, and E. Mendes, "Taxonomies in software engineering: A systematic mapping study and a revised taxonomy development method," *Information and Software Technology*, vol. 85, pp. 43–59, 2017. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0950584917300472>
- [21] S. Vegas, N. Juristo, and V. R. Basili, "Maturing software engineering knowledge through classifications: A case study on unit testing techniques," *IEEE Transactions on Software Engineering*, vol. 35, no. 4, pp. 551–565, 2009.
- [22] T. Speith, "A review of taxonomies of explainable artificial intelligence (xai) methods," in *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, ser. FAccT '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 2239–2250. [Online]. Available: <https://doi.org/10.1145/3531146.3534639>
- [23] J. J. Ferreira and M. S. Monteiro, "What are people doing about xai user experience? a survey on ai explainability research and practice," in *Design, User Experience, and Usability. Design for Contemporary Interactive Environments*, A. Marcus and E. Rosenzweig, Eds. Cham: Springer International Publishing, 2020, pp. 56–73.
- [24] K. Sokol and P. Flach, "Explainability fact sheets: A framework for systematic assessment of explainable approaches," in *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, ser. FAT\* '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 56–67. [Online]. Available: <https://doi.org/10.1145/3351095.3372870>
- [25] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," 2017. [Online]. Available: <https://arxiv.org/abs/1702.08608>
- [26] I. Nunes and D. Jannach, "A systematic review and taxonomy of explanations in decision support and recommender systems," *User Modeling and User-Adapted Interaction*, vol. 27, no. 3, pp. 393–444, 2017.
- [27] M. Unterbusch, M. Sadeghi, J. Fischbach, M. Obaidi, and A. Vogelsang, "Explanation needs in app reviews: Taxonomy and automated detection," in *2023 IEEE 31st International Requirements Engineering Conference Workshops (REW)*, 2023, pp. 102–111.
- [28] J. Droste, H. Deters, J. Puglisi, and J. Klünder, "Designing end-user personas for explainability requirements using mixed methods research," in *2023 IEEE 30th International Requirements Engineering Conference Workshops (REW)*, 2023.
- [29] Anonymous, "Supplementary material for research paper "explanations in everyday software systems: Towards a taxonomy for explainability needs"," January 2024, our supplementary material is available online at: <https://figshare.com/s/4991cdf38f765e8c9906>.
- [30] A. Forward and T. C. Lethbridge, "A taxonomy of software types to facilitate search and evidence-based software engineering," in *Proceedings of the 2008 conference of the center for advanced studies on collaborative research: meeting of minds*, 2008, pp. 179–191.
- [31] J. L. Fleiss, "Measuring nominal scale agreement among many raters," *Psychological Bulletin*, vol. 76, no. 5, pp. 378–382, 1971.
- [32] R. L. Brennan and D. J. Prediger, "Coefficient kappa: Some uses, misuses, and alternatives," vol. 41, no. 3, pp. 687–699, 1981.
- [33] J. Cohen, "A coefficient of agreement for nominal scales," *Educational and Psychological Measurement*, vol. 20, no. 1, pp. 37–46, 1960.
- [34] J. R. Landis and G. G. Koch, "The measurement of observer agreement for categorical data," *Biometrics*, vol. 33, no. 1, pp. 159–174, 1977.
- [35] K. Sokol and P. Flach, "One explanation does not fit all: The promise of interactive explanations for machine learning transparency," *KI-Künstliche Intelligenz*, vol. 34, no. 2, pp. 235–250, 2020.
- [36] N. Tintarev and J. Masthoff, "Effective explanations of recommendations: user-centered design," in *Proceedings of the 2007 ACM conference on Recommender systems*, 2007, pp. 153–156.
- [37] W. Brunotte, J. Droste, and K. Schneider, "Context content consent—how to design user-centered privacy explanations," in *The 35th International Conference on Software Engineering & Knowledge Engineering*, 2023.

- [38] M. Langer, D. Oster, T. Speith, H. Hermanns, L. Kästner, E. Schmidt, A. Sesing, and K. Baum, "What do we want from explainable artificial intelligence (xai)?—a stakeholder perspective on xai and a conceptual model guiding interdisciplinary xai research," *Artificial Intelligence*, vol. 296, p. 103473, 2021.
- [39] H. Deters, J. Droste, and K. Schneider, "A means to what end? evaluating the explainability of software systems using goal-oriented heuristics," in *Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering*, 2023, pp. 329–338.
- [40] A. Adadi and M. Berrada, "Peeking inside the black-box: a survey on explainable artificial intelligence (xai)," *IEEE access*, vol. 6, pp. 52 138–52 160, 2018.
- [41] Y. Zhang, P. Tiño, A. Leonardis, and K. Tang, "A survey on neural network interpretability," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 5, no. 5, pp. 726–742, 2021.
- [42] G. W. Harrison and E. E. Rutström, "Chapter 81 experimental evidence on the existence of hypothetical bias in value elicitation methods," in *Handbook of Experimental Economics Results*, C. R. Plott and V. L. Smith, Eds. Elsevier, 2008, vol. 1, pp. 752–767. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1574072207000819>
- [43] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering*. Springer Science & Business Media, 2012.