A Conceptual Framework for Explainability Requirements in Software-Intensive Systems

Marcello M. Bersani*, Matteo Camilli*, Livia Lestingi*, Raffaela Mirandola*, Matteo Rossi†, Patrizia Scandurra‡
* Department of Electronics, Information and Bioengineering (DEIB),
Politecnico di Milano, Italy
Email: {name}.{surname}@polimi.it
† Department of Mechanical Engineering (DMec),
Politecnico di Milano, Italy
Email: matteo.rossi@polimi.it
‡ Department of Management, Information, and Production Engineering (DIGIP),
University of Bergamo, Italy
Email: patrizia.scandurra@unibg.it

***Abstract*—Software-intensive systems include enterprise systems, IoT systems, cyber-physical systems, and industrial control systems where software plays a vital role. In such systems, the software is increasingly responsible for autonomous decision-making. However, trust can be hindered by the black-box nature of these systems, whose autonomous decisions may be confusing or even dangerous for humans. Thus, explainability emerges as a crucial non-functional property to achieve transparency and increase the understanding of the systems' behavior, fostering their acceptance in our society.**

This paper introduces a conceptual framework for eliciting explainability requirements at different granularity levels. Each level is associated with a set of meta-requirements and means for instantiating the framework within a system to make it capable of producing explanations in a given application domain. We illustrate our conceptual framework using a running example from the robotics domain.

***Index Terms*—Explainability, explainable software systems, explainability requirements**

I. Introduction

The pervasiveness of software-intensive systems in our daily activities makes our lives increasingly influenced by software-based decisions. This, together with the growing complexity of software, calls for understandable and trustworthy software behavior. However, the widespread adoption of Artificial Intelligence (AI) technologies and black-box Machine Learning (ML) often makes the system behavior completely opaque, making it difficult to understand how decisions are taken and their dependability. This lack of transparency can potentially lead to a lack of human *trust* in the results of software systems.

In this context, *explainability*, that is, the ability to provide human-interpretable explanations, can be seen as a way to achieve transparency and increase awareness of the system's behavior. This awareness, combined with guarantees about the correctness and quality of the system's behavior, can lead to the creation of *trust* in software systems. However, there is still a limited understanding of systematic engineering methods that can generate valuable explanations for human stakeholders.

This paper builds upon the idea initially introduced by Köhl et al. [15], and also adopted in [9], [10], where explainability is treated as a non-functional requirement. We present a conceptual framework defining explainability requirements in terms of phenomena to be explained (the *explananda*), interpretable and measurable *factor*s of the observed phenomena, the *context* in which such phenomena are observed, and recipient *stakeholder*s of the explanation. Furthermore, we introduce increasing levels of explainability that a system may offer. The framework also comprises a general notion of explainability metric, which is used to formulate the problem of determining the satisfaction of explainability requirements: more precisely, the metric captures the measurable condition the system should fulfill to meet the corresponding explainability level in a specific application domain.

To promote the engineering of explainability requirements, defining a reference framework for explainability is a crucial first step towards a shared language, reusable structures, and established practices. In line with such a vision, the proposed conceptual framework aims at also providing guidelines to software and requirements engineers for developing explainable software-intensive systems in a given application domain. The framework is complemented by a set of meta-requirements and means to be engineered within a system to make it capable of producing explanations. We illustrate the applicability of the proposed framework by instantiating its main conceptual aspects in a *Human-Machine-Teaming* (HMT) [3], [19] application scenario where service robots assist patients and hospital staff during daily operations.

The proposed framework leverages our preliminary work [3], [7] evaluating the feasibility of a software system for producing human-interpretable explanations in specific application domains. Specifically, in [7], we introduce our idea of explainable self-adaptation for endowing a self-adaptive system capable of providing human-understandable explanations for successful and unsuccessful adaptations in critical scenarios. In [3], we present an emerging idea for generating dependability-related explanations and partially evaluate it on the estimation of success/failure of service robot missions in the HMT domain [2]. We also conducted preliminary feasibility experiments in selected domains. In this respect, the

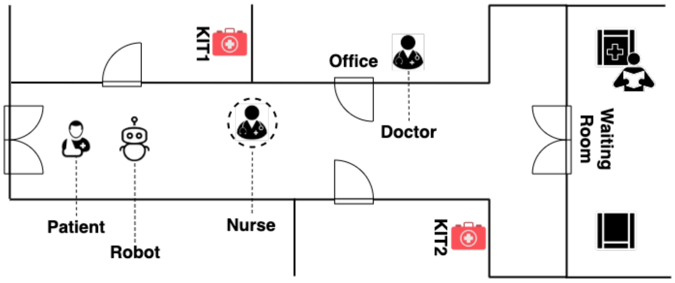

Fig. 1. High-level representation of the operational layout for the illustrative example. The agents (patient, nurse, doctor, and robot) are represented in their starting positions.

novel contributions of this paper are summarized as follows:

- a conceptual general framework for explainability elicitation, comprising the notion of explainability, levels of explainability, and a metric to assess the satisfaction of explainability requirements;
- the instantiation of the main concepts of the framework to a system example from the robotics domain;
- the identification of a set of meta-requirements and means associated with our framework to be engineered and incorporated within a target software system to make it (self-)explainable.

The remainder of the paper is organized as follows. In Sec. II, we introduce an illustrative example in the healthcare domain, including multiple service robots interacting with doctors and patients. In Sec. III, we present our conceptual framework for explainability requirements, including four levels of explainability and corresponding meta-requirements. In Sec. IV, we discuss existing tools that can be used to engineer such meta-requirements. In Sec. V, we survey related work. Finally, Sec. VI concludes the paper and outlines future research directions.

II. Illustrative Example

As an illustrative example for our conceptual framework, we use a scenario from the healthcare domain, including multiple service robots interacting with doctors and patients [16]. Figure 1 illustrates a high-level schema of the scenario's setting: a hospital ward with a doctor's office serving as an examination room, a patient waiting room, and storage rooms with medical equipment. An assistive robot is deployed on the floor to provide services requiring interaction with a human. The specific sequence of services (i.e., the mission) shown in Fig. 2 begins with the robot *escorting* a patient to the waiting room. While the patient waits for the examination room to be set up, an emergency occurs, causing the robot and a healthcare professional (i.e., nurse agent in Fig. 1 and 2) to *compete* for the same medical kit (KIT1 in Fig. 1). The outcome of the competition determines alternative plans. If the robot retrieves the resource first, it *delivers* it to the doctor in the examination room. Otherwise, the doctor *leads* the robot to another storage room where it can enter only if in the presence of authorized personnel to retrieve the required resource (KIT2 in Fig. 1). Once the examination room is set up, the robot *escorts* the patient from the waiting room and *assists* the doctor in administering the medication.

If the robot completes the mission successfully (i.e., the robot completes all services in the sequence in Fig. 2), we say the robot is *dependable*. Dependability and other phenomena of interest, such as fatigue of the human agents and the outcome of the competition, are typically affected by uncertain and changing factors that shall be measured during the mission [5], [6]. Thus, both the robot and the human agents are equipped with sensors gathering information about their current state; specifically measuring their position inside the layout, the robot's level of charge, and the humans' level of muscular fatigue, which provides a measure of the physical effort they are currently enduring. Furthermore, it is also possible to infer from field data the frequency (also referred to as their *free will profile*) with which humans ignore instructions concerning the tasks they are carrying out in coordination with the robot.

III. Explainability Framework

This section provides the main concepts of the proposed explainability framework. In particular, after preliminary definitions, we introduce the notion of explainability level and the satisfiability of explainability requirements.

A. Basic Definitions

First, we provide a working definition of explainability for the domain of software and requirements engineering. We build upon the conceptual analysis by Köhl et al. [15] and Chazette et al. [9], which, by leveraging results from psychology and the cognitive sciences, propose to treat explainability by measuring understanding for different stakeholders (system engineers, target users, etc.). According to this conceptualization, we consider as main elements that characterize an explanation the following ones: (i) phenomena—or *explananda*—of the system of interest, (ii) a set of *factor*s (i.e., interpretable and measurable aspects) of the observed explananda in a given *context*,[1] (iii) the recipient *stakeholder*s of the explanation, and (iv) the *means* for producing the explanation. More formally, we define the notion of explanation as follows.

Definition 1 (Explanation): An explanation E for a given explanandum X and a target group G of stakeholders is a piece of information (or evidence) that makes the explanandum X interpretable by G.

We hereafter refer to the notion of an explainable system according to [15] as follows.

Definition 2 (Explainable system): A system S is explainable if, and only if, it is able by a means M to produce an explanation E of an explanandum X for a target group G in a certain operating context C.

The means M in Definition 2 may be provided by a domain expert or an external system. Alternatively, the means M can be integrated into the system of interest. In the latter case, the

[1] A context represents the environmental entities that interact with the system and influence its behavior.

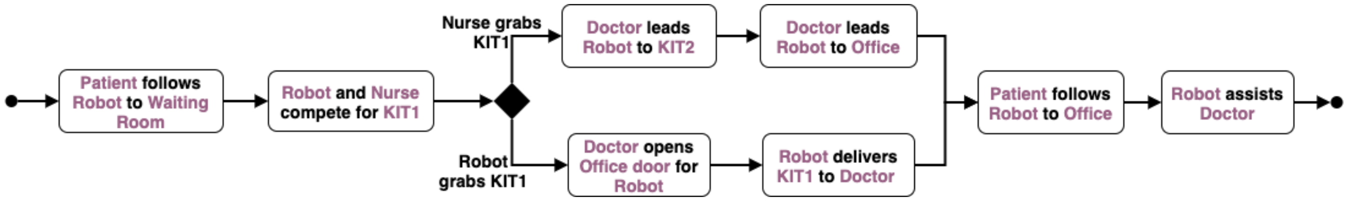
Patient follows Robot to Waiting Room
Robot and Nurse compete for KIT1
Nurse grabs KIT1
Robot grabs KIT1
Doctor leads Robot to KIT2
Doctor leads Robot to Office
Doctor opens Office door for Robot
Robot delivers KIT1 to Doctor
Patient follows Robot to Office
Robot assists Doctor

Fig. 2. Sequence of services constituting the mission for the illustrative example. The diagram highlights the alternative plans whose selection depends on the outcome of the competition for KIT1.

TABLE I
LEVELS OF EXPLAINABILITY.

Level	Description	Meta-requirements
L1	**No explanability**: The system ignores any possible explanandum X.	
L2	**Recognition of explainability needs**: The system is aware that an explanandum X for stakeholders G exists. Thus, it collects knowledge about the context C either passively or actively, by means that are deliberately designed to increase explainability through exploration.	- **MR 2.1**: the context C shall be defined in terms of selected (independent/uncorrelated) factors that are interpretable by G and may affect the explanandum X in the system's current mission. Each factor has its exposing agent, type, and domain. - **MR 2.2**: the context C, defined in terms of selected interpretable and measurable factors, shall be observable and measurable at runtime during the system operations. - **MR 2.3**: a measurement of the factors of a context C shall be available when the behavioral aspects of interest occur.
L3	**Local explainability**: - **Single agent (SA)**: The system provides an explanation E for an explanandum X by considering a specific (punctual) operating context C to make G able to understand how the relevant individual elements of C influence X. - **Multiple agents (MA)**: The local explainability (L3) process is realized by multiple cooperating agents that collectively achieve the mission objectives. Each agent has a partial view of the operating context C whose relevant elements are collected (and possibly analyzed) in a decentralized manner.	- **MR 2.1, 2.2, 2.3** - **MR 3.SA.1**: a tangible manifestation of the explanandum X shall be measurable at runtime during the system operations. - **MR 3.SA.2**: a local explanation E of an individual explanandum X shall be computed. The explanation shall be expressed as the sum of the effects of the observable factors in the context C. - **MR 3.MA.1**: Each partial context C shall be measurable by the corresponding agent participating in the mission. - **MR 3.MA.2**: A local explanation E for X shall be computed by considering the factors in all partial views.
L4	**Global explainability**: - **Single agent (SA)**: The system provides an explanation E for an explanandum X by considering a varying operating context C to make G able to understand the extent to which changes of relevant elements of C influence X on average. - **Multiple agents (MA)**: Global explainability (L4) is realized by multiple cooperating agents collectively achieving the system's mission objectives. Each agent has only a partial view of the operating context C whose relevant elements are collected (and possibly analyzed) in a decentralized manner.	- **MR 2.1, 2.2, 2.3, 3.1** - **MR 4.SA.1**: a global explanation E of the average behavior of the explanandum X shall be computed. The explanation shall be expressed as the expected distribution of X based on the factors in the context C. - **MR 4.MA.1**: A global explanation E for X shall be computed by considering the factors in all partial views.

system itself produces the explanations; thus, we say that the system is *self-explainable*.

Definition 3 (Self-explainable system): A system S is self-explainable if, and only if, the means M for producing the explanation E is part of S.

Example 1 (Basic definitions): Consider our illustrative example in Sec. II. The nurse participating in the mission would like to be guided in making decisions in the future since they may decide to drop the competition and focus on other urgent tasks in case he/she has little chance of winning under certain circumstances. In this case, the outcome of the competition (e.g., the robot wins the competition with a probability of 0.99) represents the explanandum X, while the nurse is the target stakeholder G. Thus, the system S implementing the robotic mission is explainable if there exists a means M (either embedded into S or not) that produces a human-interpretable explanation E that the nurse can use to understand the main reason(s) for the observed outcome. The explanation E includes the main factors that determine the actual outcome. For instance, the speed and the starting position of the two agents (robot and nurse), as well as the charge level of the robot, may be important factors affecting the competition. In this case, the actual value of these measurable aspects yields the context C of the ongoing mission.

B. Explainability Levels

We here characterize the context C and means M, according to different (increasing) *levels* of explainability. We take inspiration from our previous classification [7] that identifies levels of explainability of self-adaptive systems based, in turn, on the guidelines introduced by the roadmap for robotics in Europe [11]. This roadmap identifies various abilities of autonomous robotics (but does not include explainability) and defines levels for each of them as an instrument to perform an evidence-based assessment of a system under a specific lens. In this respect, a reference conceptual framework for explainability may guide software engineers in the concrete realizations of explainability software layers to be incorporated within any software system to increase the reliability and trustability of its automated/autonomous decision-making.

Table I lists and describes the devised levels of explainability that a software system can feature in any application domain. With this characterization, we introduce increasing degrees of explainability, starting from absence to recognition of the need and then from local to global explainability. Each level, in this view, can be further refined taking into account other relevant dimensions. As an illustrative example, we consider how explainability is achieved (i.e., single-agent, or multi-agent). Note that the framework is general, and other dimensions may be added depending on the needs of the application domain at hand.

Table I also includes a set of meta-requirements an explainable system should be able to satisfy to meet the corresponding explainability level.

To instantiate the abstract notions introduced in Table I, we illustrate levels L3-L4 with two scenarios in our running example from Sec. II. These scenarios include multiple mission agents, stakeholders, explananda (mainly related to the dependability of the system mission and patient fatigue), and different mission contexts (composed of different factors).

Example 2 (L3 local explainability, single agent for patient fatigue): In case the patient is particularly vulnerable (e.g., unsteady health status), the doctor (G) may want to constantly monitor the ongoing missions to spot specific (punctual) combinations of patient-related factors that systematically lead to (a tangible manifestation of) high stress (X). In this case, the doctor understands the factors having the highest impact (E) and, based on this, can steer the patient's behavior to reduce the overall level of fatigue. The factors characterizing the patient yield the context (C) of this scenario. During an ongoing mission, E may suggest to the doctor that an inattentive free will profile is the primary cause of high stress. In this case, the doctor can pay special attention and help the patient stay focused.

Example 3 (L3 local explainability, multiple agents for competition outcome): The nurse (G) wants to understand the main characteristics of some of the agents—including robot(s) and nurse(s)—that currently affect the outcome of the competition to access the shared medical kit (X). Understanding the positive/negative impact (E) of these characteristics can influence future decisions of the nurse(s) as anticipated in Example 1. In this case, the context C consists of the factors describing the agents participating in the competition. For example, the explanation E may reveal that high robot speed reduces the chance for the nurse only under certain location and charge level constraints.

Example 4 (L4 global explainability, single agents for mission dependability): The doctor (G) wants to understand which characteristics of his/her behavior (e.g., walking speed, position) are important. The extent to which changes in his/her nominal habits affect the likelihood (E) of a successful mission (X); that is, the probability that the robot successfully executes the whole sequence of services is higher than 0.9. In this scenario, the partial view of the doctor on the whole mission yields the context (C), that is, the subset of factors describing his/her behavior. For example, E may suggest to the doctor that specific locations in the shared space and too high walking speed, combined with unsteady health status, reduce the probability of success.

Example 5 (L4 global explainability, multiple agents for mission dependability): The system administrator (G) wants to understand what are the important configuration options of the software components (e.g., minimum and maximum distance) and how the interactions between them and the other characteristics of the agents affect the likelihood (E) of satisfying the dependability requirements of the mission (X). Here, context C comprises all factors, including those concerning the controller configuration. For example, the explanation E may suggest to the administrator that the maximum distance configuration has almost no impact. At the same time, on average, there is a linear dependency between maximum fatigue and the likelihood of mission success.

To support achieving a certain explainability level illustrated in the examples above, the system shall meet the corresponding meta-requirements reported in Table I. It is worth noting that meta-requirements are generic and do not refer to a specific class of systems or domain. Therefore, meta-requirements must be instantiated by defining the elements X, G, C, and E according to the system of interest.

Example 6 (Meta-requirements for L3 local explainability, multiple agents): Consider the explainability level 3, multiple agents, instance introduced in Example 3. In this case, each agent (nurse and robot) must be equipped with proper sensors to sample the factors of interest, which determine the context C of the ongoing mission. A local monitor can construct each partial view collected from the corresponding agents. Then, a local *explainer* component that represents the means M aggregates the local views and produces E. In this example, the explainer component receives the relevant data collected by the local monitors to explain the outcome of the competition. Namely, the explainer can receive data from all local monitors except the doctor's since the factors characterizing this latter agent are irrelevant for X.

C. Satisfiability of explainability requirements

We focus here on the quantitative nature of an explanation characterized by a certain explainability level.

Following the definitions given in Section III-A, we denote an explanation E, at level L_i, that concerns an explanandum X, in a context C, for a stakeholder G, as $E(L_i, X, C, G)$. To assess the quality of E, we introduce the notion of *explainability metric* $Q_E(M) \in \mathcal{M}$ as a measure of the degree of satisfaction of the explanation $E(L_i, X, C, G)$ using a means M, where $\mathcal{M}$ is some suitable preorder (with an associated ordering relation $\preceq$), which allows the comparison between different values of $Q_E(M)$.

In particular, given a means M and a minimum explainability quality threshold $\epsilon \in \mathbb{R}_+$ chosen by stakeholder G, the explanation $E(L_i, X, C, G)$ satisfies the stakeholder's expectation if the quality of the explanation $Q_E(M)$ is greater than the given threshold. More precisely, we define the explainability requirement R_E as follows.

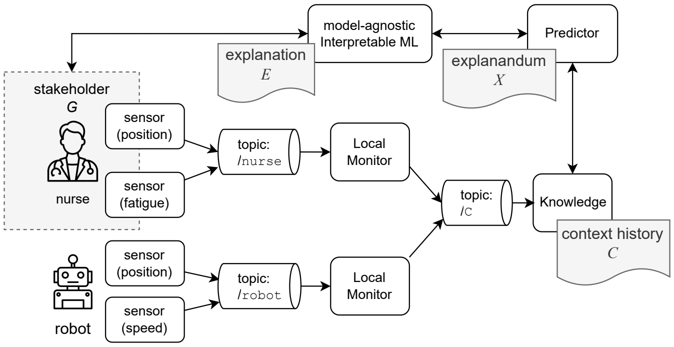

Fig. 3. High-level schema of our explainability framework.

Definition 4 (Explainability requirement R_E): The explanation $E(L_i, X, C, G)$ at a level L_i, derived for an explanandum X, in a context C for stakeholders G, shall have quality greater than a threshold $\epsilon \in \mathbb{R}_+$ established by G:

$$Q_E(M) \geq \epsilon$$

Example 7 (Explainability metric): Consider again the scenario introduced in Example 3. The elements X, G, and C are the outcome of the competition, the nurse stakeholder, and the set of factors characterizing the nurse and the service robot, respectively. For instance, the quality metric Q_E for explanation E can be constructed using a Likert scale [17] (e.g., 1 to 5). In this case, the nurse is asked to evaluate a sample of explanations produced by the local explainer component M. The nurse gives a quantitative value on a subjective matter, that is, the quality level of the explanations is either low or high according to his/her perception. For instance, the means M satisfies the requirement if the average score exceeds the selected threshold (e.g., $\epsilon = 4$).

Alternatively, a quality metric Q_E can be constructed using multiple explainer components inspired by uncertainty estimation based on *ensembling* [20]. In this case, multiple explanations E can be generated for a single context C. By calculating the variance σ of these explanations,[2] an approximation of Q_E can be obtained since higher variance yields higher uncertainty and, consequently, lower usefulness of E. Thus, the ensemble satisfies the requirement if $1 - \sigma$ is higher than the selected threshold (e.g., $\epsilon = 0.9$).

IV. Engineering Explainability

In Table I (right-hand side), we identify and describe the meta-requirements to be engineered to make a target system (self-)explainable. These meta-requirements are associated with the level of explainability to meet up to L4. Several suitable tools shall also be selected and implemented to engineer such meta-requirements. We here discuss some of them as they emerge from the current state of practice and the literature.

To implement MR 2.1 and MR 2.2 the system must be made *context-aware*, i.e., endowed with a context manager or dedicated middleware responsible for sensing and dealing with context changes. Sensing also implies the capability of measuring with appropriate metrics the context factors of interest at intervals or continuously and persisting such values into a shared knowledge base for future reference, as stated by MR 2.3. Such sensing infrastructure may adopt a request/response messaging pattern (pull or push mode) to update data in the knowledge base or an event-driven architecture with publish/subscribe interaction pattern to allow efficient measuring. To realize MR 3.SA.1, an analyzer component shall compare event data against patterns in the knowledge base to diagnose symptoms for an explanandum X and then store the signs for future reference in the knowledge base. Predictive models (e.g., neural network regressors/classifiers) can be trained/tested to forecast a certain explanandum X based on the context factors. The predictions can then be explained using state-of-the-art model-agnostic interpretable ML techniques, thus realizing MR 3.SA.2 and MR 4.SA.1 using *local* and *global* techniques, respectively [18].

Global explanations describe the average behavior of a given model. Partial Dependence Plot [18] (PDP) is an example of a global model-agnostic method that shows the marginal effect that selected features have on the predicted outcome of a model. Local explanations, such as those produced by Local Interpretable Model-agnostic Explanation [18] (LIME), explain each individual prediction. The so-built model has the local fidelity property; that is, it represents a good approximation of local predictions, but it does not have to be a good global approximation. These methods may be complemented with distributed communication techniques for realizing collective explainability as stated by MR 3.MA.1, MR 3.MA.2, and MR 4.MA.1. In this case, the *Event Sourcing* pattern[3] can be adopted whereby explanations are determined and possibly reconstructed on demand by storing all messages exchanged among agents over publish/subscribe topics related to a specific explanation. Persisting these messages would enable a complete history of context changes and explanations over time.

Example 8 (Engineering L3 local explainability, multiple agents): Figure 3 illustrates a possible high-level workflow and the main elements involved in realizing the explainability level 5 in Example 3. The two agents involved in this scenario (i.e., nurse, robot) have proper sensors that gather data to monitor the context factors composing C. The sensors publish the data to the topic associated with the corresponding agent with a Local Monitor component in charge of creating a partial view of the context and then publish the partial view to the topic C. All the partial views are then aggregated, and the entire history of C is stored in the Knowledge component using event sourcing. As illustrated in Fig. 3, the scenario has a Predictor and a model-agnostic interpretable ML component. These elements represent the means M used to produce

[2] Note that σ represents the spread between explanations. This measure shall be carefully defined according to the nature of E. Indeed, E may have a complex structure rather than a simple numeric value.

[3] https://martinfowler.com/eaaDev/EventSourcing.html

the explanations E. In this example, the predictor receives the context factors and produces the expected result of the competition as output. Then, a LIME explainer can produce local explanations through an interpretable surrogate model. For instance, an explanation E may reveal that considering the current location of the agents, the robot speed is the most critical context factor in reducing the chance for the nurse actor.

V. Related Work

In recent years, explainability, seen as the ability to provide a human with understandable explanations of the results produced by AI and ML algorithms, has become an essential aspect of designing tools based on these techniques [1], especially in critical areas such as healthcare [26]. Even if explainability is a term coined in the area of AI, interest in it is also growing in the software engineering and requirement engineering communities [9], [25]; researchers in these communities have proposed, for example, explainable analytical models for predictions and decision-making [25], explainable counterexamples [14], explainable quality attribute trade-offs in software architecture selection [4], the analysis of explainability as a non-functional requirement and its trade-off with other quality attributes [9], [15] and in relation to human-machine teaming [3]. Work describing the theoretical basis of explainability, exploiting concepts from philosophy, psychology, and sociology can be found, for example, in [8], [21], [22], [24].

Another research direction on which a lot of work has focused recently concerns the definition of metrics and properties for explainability [12], [21], [23]. The proposed metrics could be complex and tightly related to a specific method. The explanation methods can be categorized as *attributive* and *counterfactual* [23]. The former category produces metrics or visualizations based on importance scores or weights; the latter, instead, allows for the investigation of other possibilities through the modification of the prediction function. A discussion of different aspects of explanations is presented in [12], where aspects like the goodness of explanation, user understanding and satisfaction, and the impact of human curiosity are considered key measurement factors. A different approach for eXplainable AI (XAI) is presented in [21], where four metrics are proposed based on the difference between the expected and actual performance, the number of rules produced by the explanation, the number of features used to generate that explanation and the stability of the explanation.

With respect to the existing work, in this paper, we introduce increasing levels of explainability and a global satisfaction metric for system explainability and provide guidelines for engineering explainability.

VI. Conclusion

This paper addresses the problem of providing meaningful explanations of software-based decisions and shapes a conceptual framework for explainability elicitation. We provide increasing levels of explainability and a metric for quantitatively measuring a system's explainability at a certain level. These abstract concepts have been instantiated using human-machine teaming scenarios where explanations are highly demanding since a failure in decision-making can lead to severe consequences. We also envision a set of meta-requirements and means that could help software and requirements engineers in developing (self-)explainable software-intensive systems in a given application domain.

As future work, we plan to investigate several human-centered factors that may impact the quality of an explanation and its underlying production process. We also plan to elaborate further on the introduced explainability metric by defining a quality model to help evaluate the quality of explanations depending on explainability-related attributes in a system [13]. Some applications (e.g., autonomous driving and clinical diagnosis) introduce considerably more risks than others (e.g., language translation and web searches) regarding error occurrences. Some applications are expected to deliver immediate decisions in near-real-time, whereas others may respond in a lazy fashion. Finally, specific applications can be highly autonomous, while others may require human supervision. All these differences could constitute a set of attributes' values for applications, for which we can define suitable explainability strategies according to their specificity.

We plan to instantiate the proposed framework in different application domains (e.g., human-machine teaming) and different target systems (e.g., service robots). We are currently developing a software architectural solution integrated with the target system and capable of providing stakeholders with human-interpretable explanations based on user-specified explainability requirements.

References

[1] Plamen P. Angelov, Eduardo A. Soares, Richard Jiang, Nicholas I. Arnold, and Peter M. Atkinson. Explainable artificial intelligence: an analytical review. *WIREs Data Mining and Knowledge Discovery*, 11(5):e1424, 2021.

[2] Marcello M. Bersani, Matteo Camilli, Livia Lestingi, Raffaela Mirandola, and Matteo Rossi. Explainable human-machine teaming using model checking and interpretable machine learning. In *2023 IEEE/ACM 11th International Conference on Formal Methods in Software Engineering (FormaliSE)*, pages 18–28, 2023.

[3] Marcello M. Bersani, Matteo Camilli, Livia Lestingi, Raffaela Mirandola, Matteo Rossi, and Patrizia Scandurra. Towards better trust in human-machine teaming through explainable dependability. In *ICSA Companion*, pages 86–90. IEEE, 2023.

[4] Javier Cámara, Mariana Silva, David Garlan, and Bradley R. Schmerl. Explaining architectural design tradeoff spaces: A machine learning approach. In *ECSA*, volume 12857 of *LNCS*, pages 49–65. Springer, 2021.

[5] Matteo Camilli, Raffaela Mirandola, and Patrizia Scandurra. Taming model uncertainty in self-adaptive systems using bayesian model averaging. In *Proceedings of the 17th Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS '22, page 25–35, New York, NY, USA, 2022. ACM.

[6] Matteo Camilli, Raffaela Mirandola, and Patrizia Scandurra. Enforcing resilience in cyber-physical systems via equilibrium verification at runtime. *ACM Trans. Auton. Adapt. Syst.*, feb 2023. Just Accepted.

[7] Matteo Camilli, Raffaela Mirandola, and Patrizia Scandurra. XSA: Explainable self-adaptation. In *Intl. Conf. on Automated Software Engineering*, ASE'22. ACM, 2023.

[8] Shruthi Chari, Daniel M. Gruen, Oshani Seneviratne, and Deborah L. McGuinness. Directions for explainable knowledge-enabled systems, 2020.

[9] Larissa Chazette, Wasja Brunotte, and Timo Speith. Exploring explainability: A definition, a model, and a knowledge catalogue, 2021.

[10] Larissa Chazette and Kurt Schneider. Explainability as a non-functional requirement: challenges and recommendations. *Requirements Engineering*, 25, 12 2020.

[11] EU. Robotics 2020 Multi-Annual Roadmap For Robotic in Europe. https://www.eu-robotics.net/sparc/upload/about/files/H2020-Robotics-Multi-Annual-Roadmap-ICT-2016.pdf, 2016.

[12] Robert R. Hoffman, Shane T. Mueller, Gary Klein, and Jordan Litman. Metrics for explainable AI: Challenges and prospects, 2019.

[13] Mladan Jovanović and Mia Schmitz. Explainability as a user requirement for artificial intelligence systems. *Computer*, 55(2):90–94, 2022.

[14] Arut Prakash Kaleeswaran, Arne Nordmann, Thomas Vogel, and Lars Grunske. A systematic literature review on counterexample explanation. *Inf. Softw. Technol.*, 145:106800, 2022.

[15] Maximilian A. Köhl, Kevin Baum, Markus Langer, Daniel Oster, Timo Speith, and Dimitri Bohlender. Explainability as a non-functional requirement. In *RE*, pages 363–368. IEEE, 2019.

[16] Livia Lestingi, Davide Zerla, Marcello M. Bersani, and Matteo Rossi. Specification, stochastic modeling and analysis of interactive service robotic applications. *Robotics and Autonomous Systems*, 163, 2023.

[17] Rensis Likert. A technique for the measurement of attitudes. *Archives of psychology*, 1932.

[18] Christoph Molnar. *Interpretable Machine Learning*. 2 edition, 2022.

[19] Ipek Ozkaya. The behavioral science of software engineering and human–machine teaming. *IEEE Software*, 37(6):3–6, 2020.

[20] Tim Pearce, Mohamed Zaki, Alexandra Brintrup, N Anastassacos, and A Neely. Uncertainty in neural networks: Bayesian ensembling. *stat*, 1050:12, 2018.

[21] Avi Rosenfeld. Better metrics for evaluating explainable artificial intelligence: Blue sky ideas track. 05 2021.

[22] Mersedeh Sadeghi, Verena Klös, and Andreas Vogelsang. Cases for explainable software systems: Characteristics and examples. In *2021 IEEE 29th International Requirements Engineering Conference Workshops (REW)*, pages 181–187. IEEE, 2021.

[23] Vandita Singh, Kristijonas Cyras, and Rafia Inam. Explainability metrics and properties for counterfactual explanation methods. In *Explainable and Transparent AI and Multi-Agent Systems: 4th International Workshop, EXTRAAMAS 2022, Virtual Event, May 9–10, 2022, Revised Selected Papers*, page 155–172, Berlin, Heidelberg, 2022. Springer-Verlag.

[24] Chakkrit Tantithamthavorn and Jirayus Jiarpakdee. *Explainable AI for Software Engineering*. Monash University, 2021.

[25] Chakkrit Kla Tantithamthavorn and Jirayus Jiarpakdee. Explainable AI for software engineering. In *ASE*, pages 1–2. ACM, 2021.

[26] Erico Tjoa and Cuntai Guan. A survey on explainable artificial intelligence (XAI): Toward medical XAI. *IEEE Transactions on Neural Networks and Learning Systems*, 32(11):4793–4813, 2021.