Explainable Human-Machine Teaming using Model Checking and Interpretable Machine Learning

Marcello M. Bersani*
Matteo Camilli*
Livia Lestingi*
Raffaela Mirandola*
* Department of Electronics, Information and Bioengineering (DEIB), Politecnico di Milano, Italy
Email: {name}.{surname}@polimi.it

Matteo Rossi†
† Department of Mechanical Engineering (DMec), Politecnico di Milano, Italy
Email: matteo.rossi@polimi.it

***Abstract*—The human-machine teaming paradigm promotes tight teamwork between humans and autonomous machines that collaborate in the same physical space. This paradigm is increasingly widespread in critical domains, such as healthcare and domestic assistance. These systems are expected to build a certain level of trust by enforcing dependability and exhibiting interpretable behavior. However, trustworthiness is negatively affected by the black-box nature of these systems, which typically make fully autonomous decisions that may be confusing for humans or cause hazards in critical domains.**

We present the EASE approach, whose goal is to build better trust in human-machine teaming leveraging statistical model checking and model-agnostic interpretable machine learning. We illustrate EASE through an example in healthcare featuring an infinite (dense) space of human-machine uncertain factors, such as diverse physical and physiological characteristics of the agents involved in the teamwork. Our evaluation demonstrates the suitability and cost-effectiveness of EASE in explaining dependability properties in human-machine teaming.

***Index Terms*—Human-machine teaming, formal analysis, statistical model checking, interpretable machine learning**

I. Introduction

The widespread diffusion and pervasiveness of autonomous systems pave the way for their inclusion in critical domains, such as healthcare and domestic assistance. In this context, Human-Machine Teaming [8] (HMT) is an emerging paradigm in which machines and humans can be seen as teammates that collaborate, leveraging their strengths and reducing their weaknesses to achieve a common goal. One of the primary challenges for a successful collaboration between humans and machines is the establishment of a certain level of (mutual) trust. However, trustworthiness is negatively affected by the black-box nature of these systems, which typically make fully autonomous decisions that may be confusing for humans or cause hazards in critical domains.

In our vision, to achieve trust, the machine shall exhibit dependable and transparent behavior to offer strong, ideally provable assurances along with human interpretable explanations for the expected scenario outcome. A natural starting point is to consider formal verification as a crucial enabler of some facets that collectively compose the notion of trust in this context. Indeed, the adoption of formal approaches to verification, such as model checking, remains essential, yet not conclusive. Humans not only require knowing whether the teaming is going to succeed and certain dependability properties hold, but they also need to understand the main reasons in terms of many *teaming factor*s that are typically uncertain and evolve as the scenario of interest develops over time. For example, consider an interactive service robotics application in the healthcare domain. Here, doctors, patients, and robots collaborate in highly volatile conditions. Patients may be in discomfort, affecting their mobility, while doctors may be subject to stress factors affecting their actions due to physical or mental fatigue.

In this context, there exist a number of open challenges. Teaming factors are affected by sources of uncertainty and vary over time. Thus, the set of factors typically determines a nontrivial space of possible values for a parametric specification describing the target teaming scenario. Each value assignment sampled from the space of teaming factors may lead to different verification outcomes in terms of satisfaction of dependability properties of interest. However, running model checking tasks for all assignments to explain what happens under all possible conditions is unfeasible since the space of teaming factors may be huge or even dense. Furthermore, runtime usage of model checking when changes occur may be prohibitive since verification outcomes may be produced at a slower rate compared to changes in the factors. On the contrary, explanations must be constructed quickly and, at the same time, must follow rigorous approaches.

To deal with these challenges, we propose EASE[1], a novel approach that combines formal verification, based on Statistical Model Checking [10] (SMC), and interpretable Machine Learning [30] (ML). Formal verification provides rigor to the characterization of the system's operation, while ML models trained on the verification results can predict the satisfaction of dependability properties during an ongoing teaming scenario with high accuracy and a small amount of time (orders of magnitude faster than model checking). Furthermore, interpretable ML can be used to build on-the-fly explanations to track

[1]EASE stands for "Exploration, AnalySis, and Explanation."

scenario outcomes down to human interpretable root causes in terms of relevant teaming factors. The operator may exploit such explanations to apply recovery measures or redesign the teaming scenario.

The combination of two such different topics requires special attention, especially when the result of a formal tool, which is correct by definition and construction, is replaced by a potentially inaccurate result. Thus, an essential aspect to be assessed in this approach is the cost-effectiveness, that is, the trade-off between the accuracy of ML predictions as a surrogate of model checking results and the resources (time and memory) required by the two tools.

We address these questions by introducing and evaluating the EASE framework, which builds on the modeling and analysis framework presented in [22]–[24] with the ultimate goal of building better trust in HMT. EASE includes three stages: *offline* (1) HMT modeling using Stochastic Hybrid Automata [1] (SHA) and (2) exploration of the teaming factors and SMC of the teaming scenarios of interest; and *online* (3) explanation of target dependability properties. Since SMC may be computationally expensive, we keep it in pre-production as an offline stage to not interfere with the running applications. SMC feeds a binary classifier estimating the relationships between uncertain and changing teaming factors and the scenario outcome in terms of satisfaction of dependability properties. The classifier is then used online to predict the scenario outcomes and build human-interpretable explanations. Explanations can be used while designing a new HMT scenario (or reconfiguring an existing one) to guide value selection in the space of teaming factors.

The approach is illustrated using a running example in the healthcare domain featuring multiple human-machine uncertain factors. The ability of EASE to explain the dependability in the HMT featured scenario and its cost-effectiveness is shown through an empirical evaluation.

The remainder of this paper is as follows. In Sec. II we introduce background notions we use in the rest of the paper. In Sec. III we introduce our illustrative example. In Sec. IV we describe our novel approach EASE. Then, we describe our evaluation results in Sec. V. In Sec. VI we discuss major strengths and weaknesses of EASE as well as threats to validity. Section VII summarizes related work, while Sec. VIII concludes the paper and presents our future work.

II. Preliminaries

This section introduces preliminary theoretical concepts on SHA, SMC, classification, and interpretable ML.

A. Stochastic Hybrid Automata

We define SHA [12] as an extension of Hybrid Automata (HA) [1]. Let W be a set of symbols, $\Gamma(W)$ is the set of guard conditions, $\Xi(W)$ the set of updates on elements of W.

Definition 1 (Stochastic Hybrid Automaton)**.** An SHA is a tuple $\langle L, W, \mathcal{F}, \mathcal{D}, \mathcal{I}, C, \mathcal{E}, \mu, \mathcal{P}, l_{\text{ini}} \rangle$, where:

1) tuple $\langle L, W, \mathcal{F}, \mathcal{I}, C, \mathcal{E}, l_{\text{ini}} \rangle$ is an HA;
2) $\mathcal{D} : L \rightharpoonup \{\mathbb{R} \to [0,1]\}$ is the partial function assigning a probability distribution from $\{\mathbb{R} \to [0,1]\}$ to locations;
3) $\mu : (L \times \mathbb{R}^W) \to \{\mathbb{R}_+ \to [0,1]\}$ is the function assigning a probability distribution from $\{\mathbb{R}_+ \to [0,1]\}$ to each *valuation* of the SHA;
4) $\mathcal{P} : L \rightharpoonup \{(C_{!?} \times \Gamma(W) \times \wp(\Xi(W)) \times L) \rightharpoonup [0,1]\}$ is the *partial* function assigning a probability weight to the *defined* edges outgoing a location, such that $\sum_{\alpha=(\mathtt{c}!,\gamma,\xi,l') \in \mathcal{E}(l), \mathtt{c} \in C} \mathcal{P}(l)(\alpha) = 1$ holds.

While in location $l \in L$, real-valued variables in W evolve in time according to the expressions $\mathcal{F}(l)$. These expressions are called *flow condition*s [1] and are defined through sets of Ordinary Differential Equations (ODEs), making SHA suitable to model systems with complex dynamics. Special cases of flow conditions constrain clocks ($\dot{x} = 1$ holds for all $x \in X \subset W$), dense-counter variables, and constants ($\dot{v} = 0$ holds for all $v \in (V_{\text{dc}} \cup K) \subset W$).

If dense-counter $\theta \in V_{\text{dc}}$ is an independent term for flow $f \in \mathcal{F}(l)$ on location $l \in L$, i.e., $f = f(t, \theta)$, and parameter θ is randomly distributed, then f is a stochastic process [14]. We limit the analysis to flow conditions depending on, at most, one random parameter, according to Definition 1.

Probability measures are also associated with delays to model the wait between the firing of two edges. Specifically, given configuration (l, v_{var}), distribution $\mu(l, v_{\text{var}})$ governing the associated time delay is either uniform or exponential [12].

Multiple SHA modeling different entities forming a system can be combined into a *network*. Different automata of a network synchronize through the channels of set C [20]. Given channel $\mathtt{c} \in C$ and two edges of two distinct automata, whose events are $\mathtt{c}!$ (the *sender*) and $\mathtt{c}?$ (the *receiver*), triggering an event through channel $\mathtt{c}$ causes both edges to fire simultaneously. Synchronization always requires at most one sender and possibly many receivers (or none) [11].

B. Statistical Model Checking

SMC applies statistical techniques to a set of runs of the formal model, expressed as a network of SHA, to estimate the probability of a desired property holding. SMC is cheaper than exhaustive state space exploration since it is based on a finite number of simulations of the target system [10]. Given the stochastic nature of the formalism, SMC estimates the probability that a certain property ψ holds for the system. Specifically, the value of expression $\mathbb{P}_{\mathcal{M}}(\psi)$ is an estimate for the probability of property ψ holding for a given SHA network $\mathcal{M}$ [11]. Formulae ψ are Metric Temporal Logic (MTL) properties that represent an expressive formal language adopted in the context of Cyber-Physical Systems [19].

In our framework, the property ψ is of the form: $\Diamond_{\leq \tau} \mathtt{ap}$, where $\Diamond$ is the "eventually" operator and $\mathtt{ap}$ is an atomic proposition. Formula $\Diamond_{\leq \tau} \mathtt{ap}$ is true if $\mathtt{ap}$ holds within $\tau \in \mathbb{N}$ times units from the onset. SMC computes the *confidence interval* $[p - \epsilon, p + \epsilon]$ for the probability of ψ holding for $\mathcal{M}$, estimated through the Clopper-Pearson method [9].

C. Classification

In the field of Machine Learning (ML), *classification* [18] refers to a predictive modeling problem where the class label y_i is anticipated for a specific input data point $x^{(i)}$. Classification models (either binary or multiclass) are built by using supervised learning techniques to create a concise representation of the distribution of class labels in terms of quantifiable properties, known as features (or explanatory variables). Thus, a data point $x^{(i)}$ is a vector that contains a value $x_j^{(i)}$ for each feature j. A supervised learning algorithm that implements classification is referred to as *classifier*. Supervised learning uses a *training set* that includes pre-labeled data points $\langle x^{(i)}, y_i \rangle$ to "learn" the desired classification function $\hat{f}$. The classification function applied to a data point $\hat{f}(x^{(i)})$ is referred to as *prediction*. There exist several popular classifiers (either binary or multiclass) in supervised ML, including, for instance, Logistic Regression and Neural Networks [18].

The evaluation and comparison of alternative classifiers represent crucial steps after training. The evaluation is based on predictive accuracy measures taking into account True/False Positive and True/False Negative rates collected by comparing the outcome of the classifiers on new data points (not in the training set) and the corresponding oracle (i.e., ground truth class labels). Unseen data points used for evaluation purposes collectively compose the so-called *test set*. A popular predictive accuracy measure widely suggested by recent research is the Area Under the receiver operator characteristic Curve [15] (AUC). The AUC measures the discriminatory power of classifiers capturing the true positive (TP) rate against the false positive (FP) rate at various threshold settings. The AUC measure ranges between 0 (worst), 0.5 (no better than random guessing), and 1 (best).

D. Interpretable Machine Learning

Interpretable ML [30] refers to the extraction of relevant knowledge from an ML model concerning existing relations contained in data or learned by the model itself. In this sense, interpretability is the ability of a model to be understood and explained by humans. According to the terminology introduced by Miller [29], we use terms *interpretable* and *explainable* interchangeably. Some predictive models are designed to have a clear and simple structure, and their predictions are inherently explained (e.g., Linear Regression, Decision Trees). More complex models (e.g., Neural Networks, Random Forests) that do not explain their predictions are referred to as *black box* (or *non-interpretable*) models.

The scope of interpretability is either *global* (i.e., holistic model interpretability) or *local* (i.e., interpretability for a single prediction). Global explanations describe the average behavior of a given model. They give a holistic view of the distribution of the target outcome (e.g., class labels) based on the features. Partial Dependence Plot [30] (PDP) is a global model-agnostic method that shows the marginal effect that selected features have on the predicted outcome of a model. A PDP shows the relationship between the predictions and one or more features (e.g., linear, monotonic or more complex). The PDP function is defined as follows:

$$\hat{f}_S(x_S) = E_{X_C}[\hat{f}(x_S, X_C)] = \int \hat{f}(x_S, X_C)\ d\mathbb{P}(X_C) \quad (1)$$

where x_S are the features for which we want to know the effect on the prediction (usually one or two features), and X_C are the other features treated here as random variables. PDP marginalizes the model output over the distribution of features X_C to show the relationship between x_S and the predictions. For classifiers, PDP calculates the probability for a certain class label given different values for feature(s) x_S.

Local explanations take into account an individual data point of interest $x^{(i)}$ and examine the prediction $\hat{f}(x^{(i)})$ to explain possible reasons usually based on a local surrogate model. The Local Interpretable Model-agnostic Explanation [30] (LIME) method starts from $x^{(i)}$ and generates a new dataset consisting of perturbed samples mapping to the corresponding predictions of the original model. LIME uses the new dataset to train an interpretable model, which is weighted by the proximity of the samples to $x^{(i)}$. The local model built this way 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. Formally, the local model can be expressed as:

$$g^*(x^{(i)}) = \arg\min_{g \in G} L(\hat{f}, g, \pi_{x^{(i)}}) + \Omega(g) \quad (2)$$

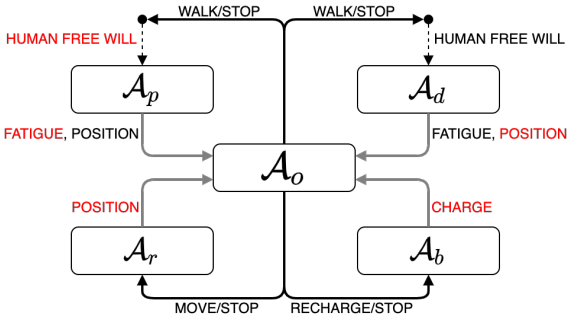
where the explanation model g^* is the local surrogate belonging to G (i.e., the set of all possible local surrogate models) that minimizes the loss function L measuring the distance between $g(x^{(i)})$ and $\hat{f}(x^{(i)})$ for all perturbed samples defined by the proximity measure $\pi_{x^{(i)}}$. The function Ω defines the model complexity that shall be minimized (e.g., models with fewer features are preferred over more complex models).

III. Illustrative Example in Healthcare

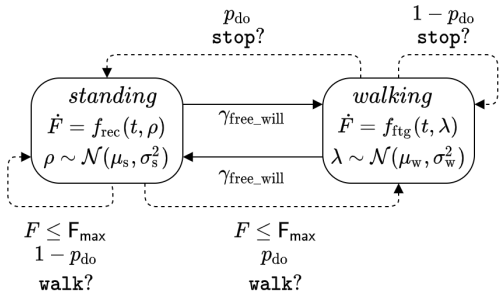
The illustrative example targets interactive service robotics in the healthcare domain. In such applications, humans and robots collaborate to achieve a common goal while operating in highly volatile conditions. The selected scenario features a hospital ward with an analysis room where doctor visits take place, a waiting room for patients to wait until the doctor is ready, and a storage room with medical equipment. A robot is deployed on the floor to assist patients and doctors during daily operations. Such assistance consists of *services* provided by the robot requiring interaction and synchronization with a human subject. The specific service sequence (i.e., the *scenario*) we consider is: 1) the robot *escorts* a patient from the entrance to the waiting room; 2) the doctor *leads* the robot to a storage room to retrieve the equipment required to visit the patient; 3) the robot *follows* the doctor to the analysis room while carrying the equipment; 4) the robot *escorts* the patient from the waiting room to the analysis room that has been set up for a visit.[2]

Following the modeling approach in [22]–[24], this scenario can be formalized by using a SHA network composed of

[2]Scenario specification available at https://github.com/LesLivia/hri_dsl.



(a) High-level representation of how the SHA from the use case scenario communicate. Synchronizations through channels are solid black; probabilistic transitions are dashed; data shared through variables are solid grey. Selected HMT factors are highlighted in red.



(b) Extract of $\mathcal{A}_p$ SHA modeling a human walking.

Fig. 1: SHA network of the illustrative example.

five automata as illustrated in Fig. 1a. Automata $\mathcal{A}_p$ and $\mathcal{A}_d$ model the human subjects (i.e., the patient and the doctor, respectively), while $\mathcal{A}_r$, $\mathcal{A}_b$, and $\mathcal{A}_o$ model the robotic system (i.e., the robotic platform, the battery, and the orchestrator, respectively). The latter serves as a controller since it monitors the state of the system and issues commands for the robot (i.e., start/stop moving and start/stop recharging) or suggestions for the human (i.e., start/stop walking). To this end, agents share data with the orchestrator through global dense counters in V_{dc}. Robots share updates on their position and state of charge, while humans share updates on position and fatigue. An extract of the SHA modeling the patient is described in the following, while we refer the reader to [22]–[24] for a thorough description of the complete SHA.

The extract of the patient model in Fig. 1b includes two locations capturing the subject standing or walking. Real-valued variable $F \in W$ models fatigue, which increases while walking, whereas *standing* corresponds to a recovery phase. Flows $\mathcal{F}(standing) = f_{\text{rec}}(t,\rho)$ and $\mathcal{F}(walking) = f_{\text{ftg}}(t,\lambda)$ constrain the time derivative of F, where t measures the duration of the current fatigue/recovery phase. Fatigue/recovery rates λ and ρ are randomly distributed according to $\mathcal{D}(standing) = \mathcal{N}(\mu_{\text{s}}, \sigma_{\text{s}}^2)$ and $\mathcal{D}(walking) = \mathcal{N}(\mu_{\text{w}}, \sigma_{\text{w}}^2)$. In more detail, upon entering the *standing* (resp. *walking*) location, a sample of $\mathcal{N}(\mu_{\text{s}}, \sigma_{\text{s}}^2)$ (resp. $\mathcal{N}(\mu_{\text{w}}, \sigma_{\text{w}}^2)$) is extracted and assigned to ρ (resp. λ). Thus, alternating fatigue/recovery phases may yield different values of the corresponding rate. According to [17], [31], such distributions depend on the health status and age group of the involved subject. Probabilistic edges model visible manifestations of human free will. When the human is standing with fatigue smaller than a critical threshold (guard $F \leq \mathsf{F}_{\max}$ holds), and the orchestrator fires a suggestion through channel $\texttt{walk} \in C$, they begin walking with probability $p_{\text{do}} \in K$, or ignore it with probability $1-p_{\text{do}}$ (similarly for the edges from *walking* to *standing* with channel $\texttt{stop} \in C$). The value of p_{do} depends on the specific free will profile. A human can also start walking independently of the orchestrator's suggestions. The edges between locations *standing* and *walking* labeled with guard $\gamma_{\text{free\_will}}$ model haphazard human decisions in terms of a dice roll.

TABLE I: Selected HMT factors.

Factor	Agent	Type	Domain
Free will profile	Patient	Categorical	{*focused*, *nominal*, *inattentive*}
Health status	Patient	Categorical	{*healthy*, *sick*, *unsteady*}
Age group	Patient	Categorical	{*young*, *elderly*}
Initial position x	Doctor	Continuous	$[0.0, 50.0]$ m
Initial position y	Doctor	Continuous	$[0.0, 8.0]$ m
Speed	Robot	Continuous	$[30.0, 100.0]$ cm/s
Battery charge	Robot	Continuous	$[11.1, 12.4]$ V

As anticipated above, several factors in this highly dynamic context affect the outcome of human-robot teaming. Indeed, the number of people operating in the ward and the level of bustle vary significantly during the day or based on seasonal factors, larger-scale emergencies (e.g., the COVID-19 pandemic), or unexpected local emergencies. In these cases, human behavior can be highly uncertain. For instance, unexpected decisions often stray from the original plan. Patients may be distressed or in discomfort, which may affect their mobility, whereas professionals may be subject to stress factors. Therefore, the duration and efficiency of their action due to either physical or mental *fatigue* are highly variable.

Table I lists selected HMT factors that may affect the agents in our illustrative example. Some apply to the patient only, while others to the doctor only (see the color-coding in Fig. 1a). These factors potentially hinder service provision and, in turn, the dependability of the robotic application and people's trust in the technology.

IV. The EASE Approach

In this section, we present the main stages of our approach illustrated in Fig. 2: offline (A) HMT modeling, (B) exploration and analysis; and online (C) prediction and explanation. In the following, we describe these stages and we illustrate the key concepts through our example in Sec. III.

A. HMT Modeling

The first activity of the offline stage envisages the modeler defining one or more HMT scenarios of interest. An HMT scenario is specified through a user-friendly Domain Specific Language[3] (DSL) to specify the geometrical boundaries of the

[3]DSL sources available at https://github.com/LesLivia/hri_dsl.

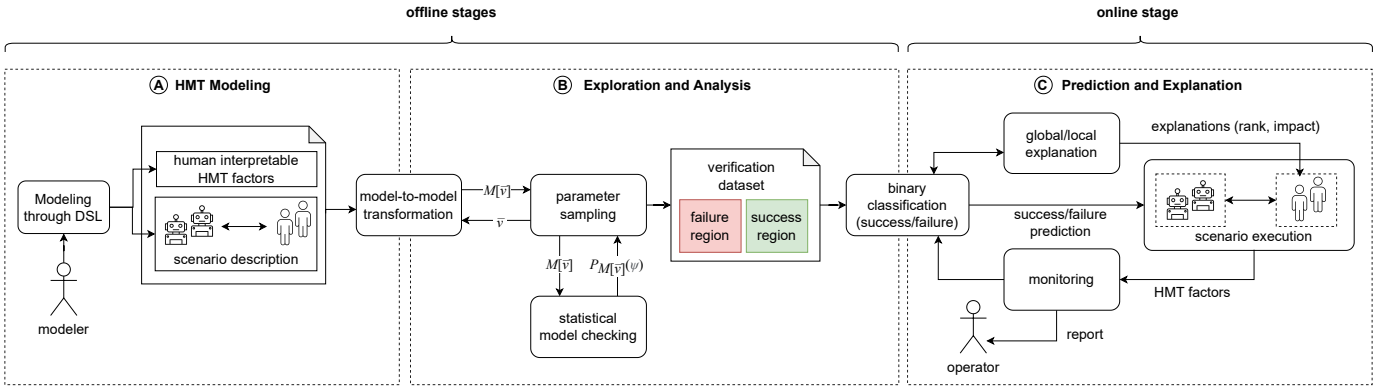


Fig. 2: EASE workflow.

teaming area, the scenario workflow, and the dependability requirements of interest. The DSL, presented in detail in [25], supports the specification of a broad range of human-machine teaming patterns, including cooperative tasks that require close contact and precise synchronization (e.g., robot feeding the human or support while walking), as well as competitive patterns capturing the human and the robot simultaneously requiring a critical resource, which could happen during emergencies.

The modeler also specifies the agent's characteristics, such as the humans' health status and age group, the initial battery level of the robots, and the initial position of all the agents in the teaming area. Table I presents the list of HMT factors we consider in this work. We define them as a set of variables $\bar{v}$ characterizing human agents (both patients and doctors) and robot agents. Each variable has its own type and domain. The value of these variables influences the phenomena of interest in the target HMT scenario possibly affecting, in turn, the satisfaction of dependability properties.

Firstly, during the modeling phase, the described set of variables and the dependability requirements are specified through the custom DSL. The DSL model is then validated to ensure its well-formedness against a fixed set of rules (e.g., that all the agent's starting positions belong to the teaming area) and automatically converted into an intermediate JSON file to decouple the specification language from the selected verification tool. In this work, each generated JSON file is automatically converted into a verification-ready UPPAAL model. Given a teaming pattern and a value assignment $\bar{v}$, the EASE framework automatically processes the JSON file into the corresponding UPPAAL model instance, that is, an SHA network $\mathcal{M}[\bar{v}]$ built through a mechanical model-to-model transformation procedure [24].

Example 1 (Model instance)**.** The chosen HMT factors and their possible values, selectable during the modeling phase, are shown in Table I. Different assignments $\bar{v}$ and $\bar{v}'$ correspond to different SHA networks $\mathcal{M}[\bar{v}]$ and $\mathcal{M}[\bar{v}']$ exhibiting, thus, different behaviors. For instance, with reference to the SHA in Fig. 1b, if the free will profile is set to *focused*, $p_{\text{do}} = 0.99$ holds. In contrast, the *nominal* profile corresponds to 0.95, thus impacting the probability with which the human abides by or ignores the robot's command.

B. Exploration and Analysis

Given an SHA network instance $\mathcal{M}[\bar{v}]$, dependability properties are verified using SMC using UPPAAL [10] model checker. We estimate the probability $\mathbb{P}_{\mathcal{M}[\bar{v}]}(\psi)$, where the atomic proposition ap in ψ is a Boolean variable that becomes true when the scenario is completed. Specifically, every service in our illustrative example (e.g., the robot escorts a patient) is associated with a Boolean value $supplied_i$ that is true if the i-th service in the sequence succeeds. So, the generated dependability property ψ has the form:

$$\psi := \Diamond_{\leq \tau} \bigwedge_i supplied_i \tag{3}$$

The SMC tool checks whether ψ holds for all traces and estimates the confidence interval $[p - \epsilon, p + \epsilon]$ for the actual probability. Then, we say that the instance $\mathcal{M}[\bar{v}]$ satisfies the property ψ, denoted by $\mathcal{M}[\bar{v}] \models \psi$, if the lower bound value $p - \epsilon$ is greater than a user-defined probability threshold π.

Example 2 (Verification of dependability properties)**.** Our illustrative example includes a number of services such as: "robot escorts a patient", and "doctor leads the robot". The scenario succeeds if they all terminate successfully. For instance, the robot successfully escorts a patient if it is simultaneously sufficiently close (i.e., within a fixed threshold) to the destination and to the human subject, and human fatigue is smaller than $\mathsf{F}_{\mathsf{max}}$ (see Fig. 1b).

The set of human interpretable HMT factors in Table I induces a large, or even infinite, SHA model space. As anticipated in Sec. III, the space of all possible changes that may break dependability requirements cannot be reasonably explored exhaustively. As shown in Fig. 2, the offline stages of EASE include *parameter sampling*, which is designed to increase knowledge through the exploration of the space of HMT factors. At the current stage, EASE adopts a simple exploration strategy based on uniform random sampling. Each sample is an assignment $\bar{v}$ that yields the corresponding instance $\mathcal{M}[\bar{v}]$. The sampling feeds the SMC that verifies the property ψ and produces the *verification dataset* that includes

the so-called *success* and *failure* regions. The success region contains the points $\bar{v}$ such that $\mathcal{M}[\bar{v}] \models \psi$ holds. The failure region contains instead the points $\bar{v}$ such that $\mathcal{M}[\bar{v}] \not\models \psi$ holds.

Other exploration strategies may be adopted depending on the search space's characteristics. Uniform random sampling is suitable in case the likelihood of successful and failing runs is comparable. If one of the two cases is a rare event, other search strategies can be adopted, such as metaheuristic optimizing search [2]. In this latter case, the search process can either maximize the lower bound $p - \epsilon$ or minimize the upper bound $p + \epsilon$, selectively pushing the evolutionary search towards successful or failing runs, respectively.

C. Prediction and Explaination

The verification dataset produced by the search process is then used to train and validate a *binary classifier* [35] to bridge the gap between offline and online stages. As shown in Fig. 2, we build the model offline, then use it online along with the running HMT to predict and explain the outcome of the ongoing scenario. The classifier estimates the relationships between features (i.e., HMT factors) and the corresponding class label (i.e., the verification outcome, that is, the property ψ holds or not). Training and validation are carried out considering different classifiers built using the verification dataset split into 80% training and 20% test using *stratified sampling* [18]. As described in Sec. II, the evaluation is based on the AUC measure [15].

Example 3 (Training and validation)**.** Training and validation may consider, for instance, Random Forests (RF) and Neural Network (NN) classifiers trained on 800 data points belonging to the training set. The AUC measure computed on 200 data points belonging to the test set is 0.95 and 0.6 for RF and NN classifiers, respectively. The interpretation of the AUC measure is that NN is slightly better than random guessing (AUC $=$ 0.5). RF exhibits higher accuracy and is close to optimal (AUC $= 1.0$). In this case, RF represents a better choice.

After the validation process, the most accurate classifier can support operators during the execution of the HMT scenario of interest. According to Fig. 2, the actual value of the HMT factors is monitored and used to predict whether the ongoing scenario is going to meet the dependability property in Eq. 3. The operator can then examine the predictions taking into account the policies and safety regulations of the facility (e.g., the minimum accepted success range within a specific ward). In case the estimated likelihood of observing a dependable scenario is low, the operator may apply manual emergency procedures or the robot enforce automated graceful degradation.

EASE also adopts global and local model-agnostic interpretable ML techniques to build human interpretable explanations for the predictions. Concerning global explainability, we rely on PDP introduced in Sec. II. We use PDP to understand the average relationship between the predictions and one or more selected features according to their importance.

TABLE II: Mapping between RQs and measurements.

RQ	Study subject	Activities	Measurements/criteria
RQ1	Best classifier	Offline analysis Offline training	Cross-validation AUC
RQ2	Online cost-effectiveness	Offline analysis Online prediction	Execution time Memory consumption Confusion matrix
RQ3	Types of explanations	Online explanation	PDP plots LIME plots

Example 4 (PDP explanations)**.** PDP can help operators to interpret the dependency between the features and the outcome (e.g., linear, monotonic, or more complex). As an example, global PDP explanations for the joint effect of robot speed and its charge level may reveal that, for speed values between 60 and 70 cm/s, the probability of success is nearly independent of the charge level when its value is higher than 11.5.

Concerning local explainability, we rely on LIME to probe repeatedly the classifier and understand each individual predicted outcome (either success or failure). Starting from the current assignment $\bar{v}$, the binary classifier uses LIME to generate a new dataset made of synthetic perturbations of $\bar{v}$ paired with the corresponding predictions. The new dataset is used to train an interpretable regression model weighted by the proximity of the perturbations to the original data point as introduced in Sec. II. The explanations are directed to operators to illustrate the reason why the current scenario is going to fail (or succeed) in terms of HMT factors.

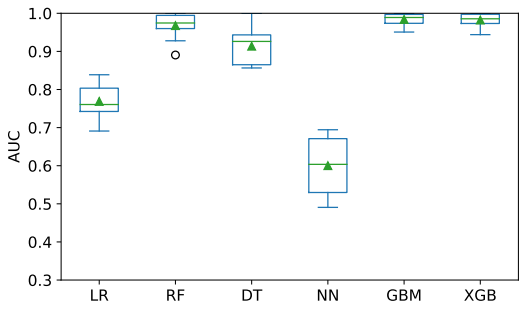
Example 5 (LIME explanations)**.** LIME can be used to understand the relative importance of the HMT factors and quantify the extent to which their value contributes to the likelihood of observing a failure (or a success). For instance, a LIME explanation may reveal that the patient profile *inattentive* represents, under the current assignment $\bar{v}$, the main reason for the failure, that is, the surrogate local model has the highest weight 0.56 associated to this feature. Other features, like the fatigue profile, may instead reduce the probability of failure. In this case, the corresponding feature in the local surrogate has a negative weight, such as -0.19.

Global/local explanations may be adopted by modelers to guide the selection of the most suitable configurations (i.e., value assignments to factors) that maximize the likelihood of observing a successful scenario according to the results of the classifier. Suggestions should then be validated through SMC to achieve stronger dependability guarantees. This way, the number of assignments to be verified is limited, which makes the approach more practical compared to exhaustive enumeration and verification of all possible assignments.

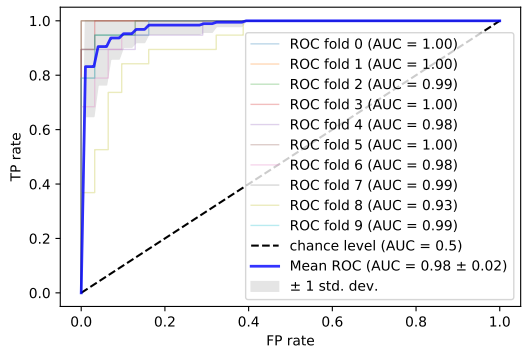
V. Evaluation

In this section, we describe the evaluation of EASE. We introduce our research questions, the design of the evaluation, and then we present the major results[4].

[4]Replication package available at https://doi.org/10.5281/zenodo.7614649



(a) Cross validation AUC comparison



(b) GBM ROC analysis

Fig. 3: Cross-validation AUC.

A. Research questions and design

The purpose of our evaluation is to study the extent to which EASE can be adopted to explain the dependability of HMT and its cost-effectiveness. In particular, we aim at answering the following research questions:

RQ1: What is the best classifier we can build in the offline stage of EASE?

RQ2: What is the cost-effectiveness of the EASE in the online stage?

RQ3: What kind of human interpretable explanations can we provide using EASE?

To answer the research questions, we carried out a number of experiments by executing the activities of EASE (both offline and online stages) on the example in Sec. III. The structural complexity of the SHA network used in our evaluation is approximately 176×10^3 (i.e., the cumulative size of each SHA, calculated as the product of the number of locations, edges, and the cardinality of state variables' domains). Table II maps each research question to the corresponding study subject, the executed activities, and the collected measurements or criteria used to answer the question.

In our experiments, we took control over the HMT factors in Table I and we generated two random samples S and S', each one with 500 unique assignments ($1k$ assignments in total). Then, we used the UPPAAL model checker[5] to verify the dependability property in Eq. 3 under the specification $\mathcal{M}[\bar{v}]$ for all $\bar{v} \in S \cup S'$. The results have been used to create two datasets D_S and $D_{S'}$ mapping all $\bar{v}$ to the corresponding Boolean outcome y. We used the dataset D_S to train and validate a number of binary classifiers to compare them and identify the best one in our problem domain. Then, we used the dataset $D_{S'}$ to study the cost-effectiveness of the online predictions of the best classifier compared to the actual outcome of the model checker. Finally, we collected the output of the best classifier to feed model agnostic interpretable ML techniques. In particular, we identified the most important HMT factors to build PDP global explanations and then we sampled the dataset $D_{S'}$ to build LIME local explanations to understand the extent to which these techniques can be used to provide scenario designers with human interpretable feedback.

All the SMC experiments have been conducted by using a commodity hardware machine running UBUNTU OS v22.04 with 64GB of memory and 4 CPU cores.

[5]For each run, we estimated the probability of success p with confidence interval magnitude equal to 0.1 (i.e., with $\epsilon = 0.05$).

TABLE III: Model rank through Scott-Knott ESD test.

Classifier	Median AUC	Rank
Gradient Boosting Machine (GBM)	0.988	Rank-1
eXtreme Gradient Boosting Tree (XGB)	0.985	
Random Forests (RF)	0.974	Rank-2
Decision Tree (DT)	0.926	Rank-3
Logistic Regression (LR)	0.760	Rank-4
Neural Network (NN)	0.603	Rank-5

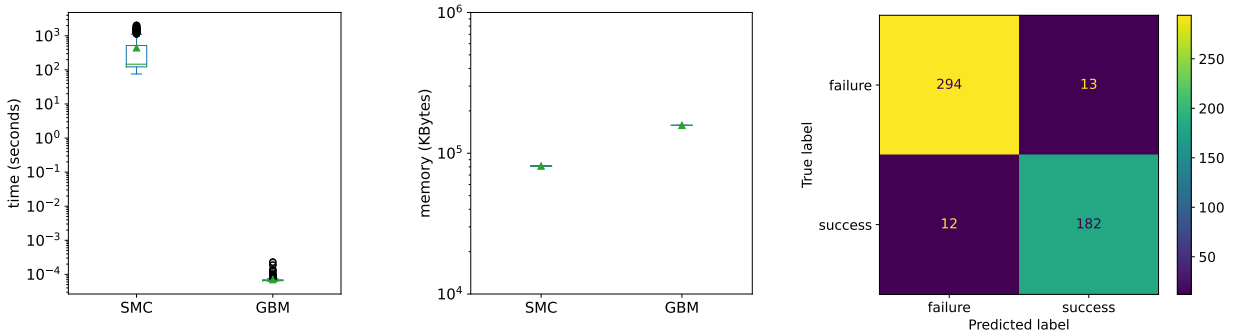
B. Results

1) RQ1 (best classifier): We executed the offline analysis and training activities multiple times to find the best classification technique in our problem domain. We selected and then compared 6 common classification techniques [34] listed in Table III. These techniques include interpretable models (DT), techniques having built-in model-specific explanations (LR, RF) and other well-known approaches (NN, GBM, XGB).

All the classifiers have been trained using the dataset D_S (split into 80% training set and 20% test set). To reduce the risk of overfitting on the test set, we used k-fold cross-validation [33]. According to Table II, we determine the best classifier using the Area Under the receiver operator characteristic Curve (AUC) to measure the discriminatory power of predictive models [15]. The Receiver operating characteristic (ROC) [15] is a curve that plots the True Positive (TP) rates against False Positive (FP) rates for all possible FP thresholds in $[0, 1]$. The best possible model has ROC close to $y = 1$. Thus, the values of AUC range between 0 (worst), 0.5 (no better than random guessing), and 1 (best).

Figure 3a illustrates the AUC measure obtained by applying 10-fold cross-validation. We can observe that the top three classifiers (RF, GBM, and XGB) have average and median AUC values close to 1. As shown in Table III, the results of model ranking using the non-parametric Scott-Knott effect size difference (ESD) test[6] [32] identify GBM and XGB as first-

[6]Comparison approach that leverages hierarchical clustering to partition the set of AUC values into distinct groups with a non-negligible difference.



(a) Execution time (b) Memory consumption (c) Confusion matrix

Fig. 4: Cost-effectiveness comparison (SMC vs GBM).

rank models. Even though the difference is not statistically significant, we finally selected GBM as the best classifier since it has a higher median AUC compared to XGB (0.988 vs 0.985). Figure 3b shows the ROC curves of the GBM (first-rank) classifier. Each curve (one for each fold) features the TP rate on the y axis, and the FP rate on the x axis. The top left corner represents the optimum (i.e., 0 FP rate of zero, and 1 TP rate). The mean ROC yields a steep curve, meaning that the GBM model exhibits a behavior close to ideal (i.e., it maximizes the TP rate while minimizing the FP rate).

RQ1 Summary: We identified GBM and XGB as first-rank models, according to the AUC measure with 10-fold cross-validation. GBM has a slightly higher median AUC (0.988) compared to XGB (0.985) even though there is a negligible effect size difference.

2) RQ2 (online cost-effectiveness): We executed the online prediction activity multiple times using the best classifier GBM. We observed 500 predictions given new assignments to HMT factors belonging to $D_{S'}$ (i.e., new data points outside the training and validation sets). For each assignment $\bar{v}$, we compared the predicted outcome and the oracle (i.e., actual outcome $y \in D_{S'}$) to measure the effectiveness in terms of TP, TN, FP, and FN rates. In addition to the effectiveness, we measured the cost as the number of required resources: execution wall-clock time and resident memory.

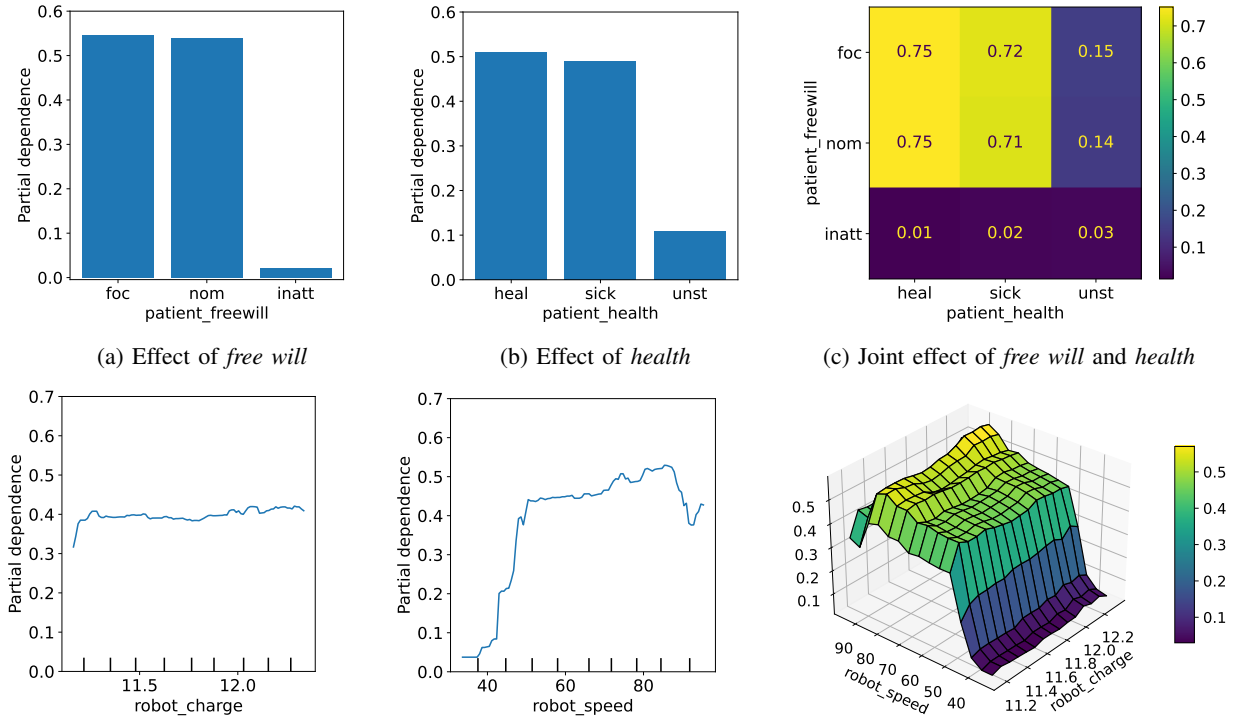
Figure 4 summarizes the cost-effectiveness results collected in these experiments. Figure 4a illustrates the execution time (seconds) required by the SMC and the GBM classifier to verify the property and predict the verification outcome, respectively. We can observe that the time required by the SMC is, in general, several orders of magnitude higher compared to GBM. The SMC takes on average $436s$, while the GBM takes on average $72\mu s$. Concerning memory consumption, the results in Fig. 4b show that SMC requires on average 81 MBytes, while GBM uses on average 130 MBytes. While the cost is comparable in terms of memory consumption, the execution time required by SMC is prohibitive and it makes it inadequate for online usage in our context. Considering execution time, GBM represents indeed a better option since it takes only a few microseconds for each prediction. Figure 4c illustrates the confusion matrix built by comparing predictions against the oracle. It shows that both TN and TP values are high: 294 out of 307 negative outcomes and 182 out of 194 positive outcomes, respectively. On the contrary, both FN and FP values are small: 13 out of 307 negative outcomes and 12 out of 194 positive outcomes, respectively. We can observe that the effectiveness in terms of correct predictions is high and the probability of observing mispredictions is low. Overall, we observed an FP rate of 0.026 and an FN rate of 0.024.

RQ2 Summary: The cost of GBM is lower compared to SMC. On average, each prediction takes $72\mu s$, while SMC requires $436s$. The efficacy of GBM in terms of correct predictions is high, as FP and FN rates are less than 0.03.

3) RQ3 (types of explanations): To answer this RQ, we first used *permutation feature importance*[7] [4] to detect the most important HMT factors. We then analyzed the four top factors (i.e., patient *free will*, patient *health*, robot *speed*, and robot *battery charge*) using PDP to extract global explanations for the average effect of these factors on the scenario outcome. According to these results, we sampled new assignments and then analyzed them through LIME to understand the extent to which local changes applied to controllable factors can increase the chance of success.

Figure 5 shows the results of PDP. The effect of the two categorical factors patient *free will* and *health* is shown in Fig. 5a and Fig. 5b, respectively. The interpretation of these plots is causal. Namely, we can quantify the extent to which changes to these two factors influence success on average. We can observe that, when the patient is either focused or nominal, the probability of success is higher than 0.5 on average. The probability drops below 0.05 when the patient is inattentive. Similarly, the probability is around 0.5 when the patient is either healthy or sick, but it drops to 0.1 when the health

[7]Technique that randomly shuffles the values of each feature to assess how much the classifier depends on them according to the drop in accuracy.



(a) Effect of *free will*

(b) Effect of *health*

(c) Joint effect of *free will* and *health*

(d) Effect of *battery charge*

(e) Effect of *speed*

(f) Joint effect of *charge* and *speed*

Fig. 5: PDP global explanations of the average effect of HMT factors.

status is unsteady (i.e., more critical than "sick"). The two-variable PDP in Fig. 5c shows the probability of success on joint values. For focused and nominal patients, the probability is nearly independent of free will, while there is a dependence on the health status. For inattentive patients, the probability of success is low and nearly independent of health status.

The effect of the two continuous factors robot *charge* and *speed* is shown in Fig. 5d and Fig. 5e, respectively. Considering the speed, we can see that there is a peak in the average probability between 75 and 85 cm/s. The probability drops when the speed is either less than 50 or higher than 85 cm/s. The joint effect in Fig. 5f shows that, for speed values between 60 and 70 cm/s, there is a plateau nearly independent of the charge level (if it is between 11.5 and 12.4 V).

In general, we can also observe that we can increase the average probability of success by controlling the speed of the robot. We tested this last observation by running the scenario with the following assignment of HMT factors: elderly patient with nominal free will profile and healthy status; doctor starting from $x = 3062.5$ and $y = 761.2$; robot with 40 cm/s cruise speed, and 12.2 V charge level. According to Fig. 5e and Fig. 5f, the expected probability of success is very small mainly because of a small speed value. The local explanation extracted using LIME in Fig. 6a confirms this result by showing the relative importance of the factors and by quantifying the extent to which each value of the current assignment contributes to the outcome. The cruise speed smaller than 49.6 cm/s represents the main reason in this case (highest weight equal to 0.35). Other factors, such as nominal free will profile and healthy status (weight -0.24), increase a bit the chance of success, estimated by LIME as 0.028 under the current assignment. We then changed the speed from 40 to 85 cm/s. The results in Fig. 6b show that by controlling this factor, the probability of success increases to 0.969. Under the new assignment, the success is explained by LIME in terms of two main reasons: nominal free will and healthy patient (highest weight 0.21). In both cases (scenario under the former and the new assignment), the success probability estimated by LIME is consistent with the confidence interval computed by the SMC: $[0.0, 0.1]$ and $[0.87, 0.97]$, respectively.

RQ3 Summary: PDP explanations show there is a partial interaction between patient free will and health status. The joint effect of robot charge and speed shows the extent to which the cruise speed can be controlled to increase the average probability of success. LIME explanations for selected HMT assignments are consistent with this result.

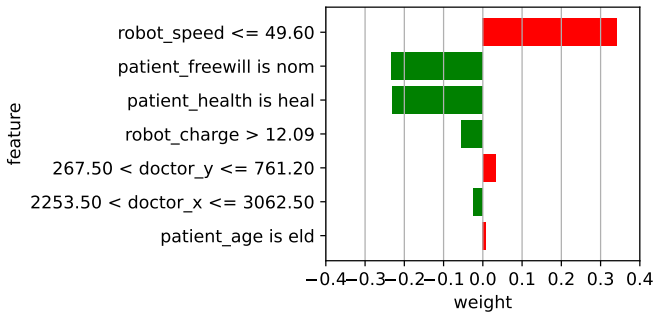
VI. Discussion and Threats to Validity

In this section, we discuss the major advantages and shortcomings of EASE and threats to the validity of our findings.

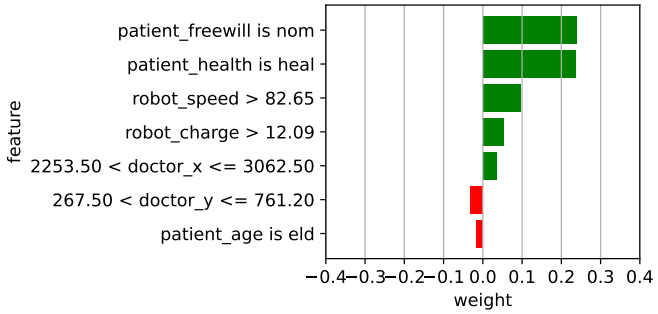
A. Strenghts and Weaknesses

In our experience, EASE offers the following advantages:

- EASE deals with infinite spaces of teaming factors whose effects cannot be reasonably explored exhaustively.



(a) Estimated success probability 0.028 (scenario failure)



(b) Estimated success probability 0.969 (scenario success)

Fig. 6: Examples of LIME explanations.

- Parameter sampling in EASE is deliberately designed to increase knowledge by exploring the space of teaming factors. The search strategy may be selected based on the characteristics of the search space at hand.
- EASE deals with the prohibitive usage of model checking online. The cost of making online predictions in terms of execution time is 7 orders of magnitude lower compared to running the statistical model checker.
- EASE builds global and local explanations to help operators track the outcomes down to human interpretable root causes in terms of relevant teaming factors.

Based on our evaluation, a few disadvantages may arise:

- The training phase may be costly since it requires the analysis of a large number of model instances. However, we keep this phase offline not to interfere with the running teaming scenario.
- Online predictions made by classifiers are not exact, thus, we may observe wrong predictions. However, according to our experience, both true-positive and false-positive rates are low (both values less than 0.03).

B. Threats to validity

1) Construct validity: Threats in this category arise if selected measurements/criteria do not reflect the properties of interest of our study subjects. We limit this threat by assessing their validity before using them in our experimental campaign. The AUC measures the discriminatory power of predictive models and is widely suggested by recent research [21] to carry out a model evaluation. We measure the computational cost by taking into account standard metrics based on execution time and memory consumption. The effectiveness of the best classifier is assessed by considering common predictive accuracy measures (TP, TN, FP, and FN), as described in [15]. We then interpret PDP and LIME results in the context of our case study. Both PDP and LIME are model-agnostic techniques we selected due to their flexibility [30].

2) Conclusion validity: We mitigate conclusion validity threats by reducing the possibility of overfitting on the test set by applying 10-fold cross validation [33]. Thus, we increase the possibility of generalizing the classifiers to observed data in the training set and unseen data in the test set. To reduce the possibility of obtaining results by chance in the context of RQ2, we execute both SMC and GBM 500 times for all assignments in the dataset $D_{S'}$.

3) Internal validity: Threats may be caused by bias in establishing cause-effect relationships in our experiments. We extract a large sample for all the HMT factors considered in our case study to limit these threats. Fine-grained access to these factors increases internal validity compared to observations without manipulation. During the training and validation of the classifiers, we adopt stratified sampling to reduce the risk of obtaining underrepresented HMT factors.

4) External validity: Threats may exist if the characteristics of our case study are not indicative of the characteristics of other HMT systems. We mitigate these threats by considering an existing case study from the literature with non-trivial space of HMT factors, including discrete and continuous variables. The generalization of our findings to other systems having more complex SHA specifications and a comprehensive scalability assessment of EASE require additional experiments.

VII. Related Work

The combination of ML and formal verification has been the subject of previous studies having the aim of endowing the results derived from an ML model with formal guarantees of correctness [3], [13], [16]. As an example, ML has been exploited to solve the problem of forming ensembles of cooperating autonomous components [5]. This problem is classically solved by solving complex constraints, that is, reducing it to a Constraint Satisfaction Problem (CSP). The use of CSPs, however, does not scale given the inherent complexity of the problem, and the introduction of classifiers suitably trained with instances (and solutions) of pre-computed CSP problems proves effective in reducing computational costs.

The design, development, and deployment of robotic applications that realize collaborative scenarios involving humans and service robots are addressed in [22]–[24]. The scenarios are sequences of collaborative activities, each satisfying an interaction pattern. Different parameters characterize a scenario, including, for example, the speed of the robot and human traits such as the tendency to fatigue of the subjects involved in the action. Formal guarantees on the feasibility of the scenarios are obtained through stochastic model checking.

Along these lines, it lies the most recent work of Garlan et al. [6], [26], [27], where the inclusion of the Human-in/on-

the-loop is formalized through stochastic models incorporating human personality traits and including explanations to facilitate the human understanding of the system operation through model checking. Research on the explainability of autonomous and robotic systems is recognized as an important topic since the lack of transparency makes their decisions and effects on the world hard to interpret for humans [7].

Another related research line recently started exploring a more advanced form of collaboration between humans and machines, which has been referred to as HMT [8], [28], where the interaction is perceived as a partnership exploiting the strengths of both actors. Specifically, the approach presented in [28] proposes a framework to help investigate different HMT options in a set of simulated operational contexts.

VIII. Conclusion and Future Work

In this paper, we introduce EASE, a novel approach that combines statistical model checking and interpretable ML to achieve better trust in HMT domains. Our evaluation shows the ability of EASE to predict and explain the satisfaction of dependability properties in HMT. We assessed the cost-effectiveness through a trade-off analysis between the accuracy of ML predictions and the required resources. We found that the cost of using the classifiers online is 7 orders of magnitude lower compared to statistical model checking. Furthermore, the best classifier yields high accuracy (i.e., both false-positive and false-negative rates are less than 0.03).

We plan to study the effectiveness of alternative factor sampling strategies based on metaheuristic optimization to guide the generation of the verification dataset in presence of rare events. We also plan to carry out a comprehensive scalability assessment of the approach targeting SHA models having higher structural complexity.

References

[1] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *TCS*, 138(1):3–34, 1995.

[2] C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.*, 35(3):268–308, sep 2003.

[3] T. Brázdil, K. Chatterjee, M. Chmelík, V. Forejt, J. Křetínský, M. Kwiatkowska, D. Parker, and M. Ujma. Verification of Markov Decision Processes using learning algorithms. In *Automated Technology for Verification and Analysis*, pages 98–114. Springer, 2014.

[4] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[5] T. Bureš, I. Gerostathopoulos, P. Hnětynka, and J. Pacovský. Forming ensembles at runtime: A machine learning approach. In *Leveraging Applications of Formal Methods, Verification and Validation: Engineering Principles*, pages 440–456. Springer, 2020.

[6] J. Cámara, M. Silva, D. Garlan, and B. R. Schmerl. Explaining architectural design tradeoff spaces: A machine learning approach. In *ECSA*, volume 12857 of *LNCS*, pages 49–65. Springer, 2021.

[7] M. Camilli, R. Mirandola, and P. Scandurra. XSA: Explainable self-adaptation. In *37th IEEE/ACM International Conference on Automated Software Engineering*, ASE22. ACM, 2023.

[8] J. Cleland-Huang, A. Agrawal, M. Vierhauser, M. Murphy, and M. Prieto. Extending mape-k to support human-machine teaming. SEAMS '22, page 120–131. ACM, 2022.

[9] C. J. Clopper and E. S. Pearson. The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika*, 26(4):404–413, 1934.

[10] A. David, K. G. Larsen, A. Legay, M. Mikučionis, and D. B. Poulsen. Uppaal SMC tutorial. *Intl. Journal on Software Tools for Technology Transfer*, 17(4):397–415, Aug 2015.

[11] A. David, K. G. Larsen, A. Legay, M. Mikučionis, and D. B. Poulsen. Uppaal SMC tutorial. *STTT*, 17(4):397–415, 2015.

[12] A. David, K. G. Larsen, A. Legay, M. Mikučionis, D. B. Poulsen, J. Van Vliet, and Z. Wang. Statistical model checking for networks of priced timed automata. In *Intl. Conf. on Formal Modeling and Analisys of Timed Systems*, pages 80–96. Springer, 2011.

[13] J. GIRARD-SATABIN. Verification and validation of machine learning techniques. Phd thesis. Universite' Paris-Saclay. 2021.

[14] U. Grenander. Stochastic processes and statistical inference. *Arkiv för matematik*, 1(3):195–277, 1950.

[15] J. A. Hanley and B. J. McNeil. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143(1):29–36, 1982. PMID: 7063747.

[16] X. Huang, M. Kwiatkowska, S. Wang, and M. Wu. Safety verification of deep neural networks. In R. Majumdar and V. Kunčak, editors, *Computer Aided Verification*, pages 3–29. Springer, 2017.

[17] H. G. Kang and J. B. Dingwell. Differential changes with age in multiscale entropy of electromyography signals from leg muscles during treadmill walking. *PloS one*, 11(8):e0162034, 2016.

[18] S. B. Kotsiantis. Supervised machine learning: A review of classification techniques. In *Proceedings of the 2007 Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in EHealth, HCI, Information Retrieval and Pervasive Technologies*, page 3–24, NLD, 2007. IOS Press.

[19] R. Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255–299, Nov 1990.

[20] K. G. Larsen, P. Pettersson, and W. Yi. UPPAAL in a nutshell. *Int. J. on Softw. Tools for Tech. Transf.*, 1(1-2):134–152, 1997.

[21] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch. Benchmarking classification models for software defect prediction: A proposed framework and novel findings. *IEEE Trans. on Software Engineering*, 34(4):485–496, 2008.

[22] L. Lestingi, M. Askarpour, M. M. Bersani, and M. Rossi. Formal verification of human-robot interaction in healthcare scenarios. In *SEFM*, volume 12310 of *LNCS*, pages 303–324. Springer, 2020.

[23] L. Lestingi, M. Askarpour, M. M. Bersani, and M. Rossi. A deployment framework for formally verified human-robot interactions. *IEEE Access*, 9:136616–136635, 2021.

[24] L. Lestingi, C. Sbrolli, P. Scarmozzino, G. Romeo, M. M. Bersani, and M. Rossi. Formal modeling and verification of multi-robot interactive scenarios in service settings. FormaliSE '22, page 80–90. ACM, 2022.

[25] L. Lestingi, D. Zerla, M. M. Bersani, and M. Rossi. Specification, stochastic modeling and analysis of interactive service robotic applications. *Robotics and Autonomous Systems*, 2023.

[26] N. Li, J. Cámara, D. Garlan, and B. R. Schmerl. Reasoning about when to provide explanation for human-involved self-adaptive systems. In *IEEE ACSOS 2020,*, pages 195–204. IEEE.

[27] N. Li, J. Cámara, D. Garlan, B. R. Schmerl, and Z. Jin. Hey! preparing humans to do tasks in self-adaptive systems. In *SEAMS 2021*, pages 48–58. IEEE.

[28] A. M. Madni and C. C. Madni. Architectural framework for exploring adaptive human-machine teaming options in simulated dynamic environments. *Systems*, 6(4), 2018.

[29] T. Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38, 2019.

[30] C. Molnar. *Interpretable Machine Learning*. 2 edition, 2022.

[31] M. Paneroni, C. Simonelli, M. Saleri, L. Bertacchini, M. Venturelli, T. Troosters, N. Ambrosino, and M. Vitacca. Muscle strength and physical performance in patients without previous disabilities recovering from COVID-19 pneumonia. *American Journal of Physical Medicine & Rehabilitation*, 100(2):105–109, 2021.

[32] A. J. Scott and M. Knott. A cluster analysis method for grouping means in the analysis of variance. *Biometrics*, 30(3):507–512, 1974.

[33] M. Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society: Series B (Methodological)*, 36(2):111–133, 1974.

[34] C. Tantithamthavorn, S. McIntosh, A. E. Hassan, and K. Matsumoto. The impact of automated parameter optimization on defect prediction models. *IEEE Trans. on Software Engineering*, 45(7):683–711, 2019.

[35] V. Vapnik. An overview of statistical learning theory. *IEEE Trans. on Neural Networks*, 10(5):988–999, 1999.