Verify, Augment, Improve: Self-Adaptation Repair via Automated Knowledge Augmentation from Mistakes

Pietro Benecchi
Politecnico di Milano
Milan, Italy
pietro.benecchi@polimi.it

Luigi Cardone
Politecnico di Milano
Milan, Italy
luigi.cardone@polimi.it

Matteo Camilli
Politecnico di Milano
Milan, Italy
matteo.camilli@polimi.it

Livia Lestingi
Politecnico di Milano
Milan, Italy
livia.lestingi@polimi.it

Raffaela Mirandola
Karlsruhe Institute of Technology
Karlsruhe, Germany
raffaela.mirandola@kit.edu

Abstract

Cyber–Physical Systems (CPSs) operate under uncertainty and cannot always guarantee the satisfaction of dependability requirements. Proactive self-adaptation mitigates violations by planning corrective actions, often leveraging predictive models. These models can be inaccurate in underrepresented regions of the operational space, leading to ineffective or unsafe adaptations. We present Verify, Augment, and Improve (VAI), a framework that extends a standard MAPE–K architecture with an asynchronous self-improving loop. VAI intercepts ineffective adaptation actions and turns explanations from a descriptive aid into a mechanism for continual improvement of the self-adaptation process. Specifically, each ineffective adaptation is verified against a ground truth (e.g., a high-fidelity simulator); when a drift between surrogate and ground truth is detected, VAI explains the failure, augments the training data near the drift, and retrains the surrogate. We instantiate VAI on a human–machine teaming benchmark and two study subjects adopting alternative ground truths. Experimental results show that VAI consistently reduces the relative error of adaptation decisions and increases the success rate of meeting requirements, with average gains of 6.89% and 10.88% across the two selected subjects.

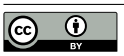
CCS Concepts

• **Computer systems organization** → *Embedded and cyber-physical systems*; *Self-organizing autonomic computing*; • **Software and its engineering** → *Extra-functional properties.*

Keywords

Self-improving, Adaptation Repair, Knowledge Augmentation, Cyber-Physical Systems

ACM Reference Format:
Pietro Benecchi, Luigi Cardone, Matteo Camilli, Livia Lestingi, and Raffaela Mirandola. 2026. Verify, Augment, Improve: Self-Adaptation Repair via Automated Knowledge Augmentation from Mistakes. In *21st International Conference on Software Engineering for Adaptive and Self-Managing Systems (SEAMS '26), April 13–14, 2026, Rio de Janeiro, Brazil.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3788550.3794874



SEAMS '26, Rio de Janeiro, Brazil

ACM ISBN 979-8-4007-2445-9/26/04
https://doi.org/10.1145/3788550.3794874

1 Introduction

Cyber-Physical Systems (CPSs) integrate computational, physical, and communication elements to deliver dependable services in dynamic environments. Yet uncertainties due to environmental variability and unforeseen interactions can compromise the sustained satisfaction of dependability requirements once systems are deployed [6, 37, 42]. A well-established response is *self-adaptation*: a managing system monitors the CPS, analyzes its state, plans corrective actions, and executes them to maintain compliance with requirements [35]. This closed loop is typically organized as MAPE-K architecture (Monitor, Analyze, Plan, and Execute over shared Knowledge). In cases where the managing system acts proactively, the adaptation process may rely on predictive models to make timely decisions before failures occur [9, 35]. Such predictive models serve as surrogates of the behavior of the real system and, by their very nature, are inherently imperfect. Their accuracy degrades in underrepresented regions of the operational space, which can lead to suboptimal or unsafe adaptations precisely when adaptation is most needed. Explainability has been proposed to increase transparency and trust in such decisions [4, 8], but explaining a decision does not, by itself, *repair* the underlying model when it proves inaccurate.

We address this gap with VAI, a framework that augments MAPE-K with an *asynchronous self-improving loop*. In VAI, each runtime adaptation is subsequently verified against a given *ground truth* (e.g., a high-fidelity simulator). When a *drift* between the surrogate prediction and the ground truth is detected, the framework (i) explains the failure to identify influential features, (ii) *augments* the training data with targeted new samples near the drift (via alternative augmentation methods), and (iii) retrains and redeploys the surrogate model(s). This *verify–augment–improve* cycle allows the system to operate continuously and with low latency—since real-time adaptation decisions still rely on the lightweight surrogate—while the more computationally intensive verification and retraining occur asynchronously in the background, progressively enhancing the surrogate and, consequently, the quality of future adaptations.

We instantiate VAI in a Human-Machine Teaming (HMT) scenario and empirically study how its effectiveness varies with the ground truth source and augmentation strategy. Results show that

VAI reduces the relative error of adaptation decisions and improves the success rate of meeting dependability goals, with a specific augmentation method we call eXplanation-driven Augmentation, yielding the largest gains at negligible cost compared to verification.

The major contributions of this paper are:

- VAI, an extension of MAPE-K with an asynchronous self-improving loop embedding a novel augmentation strategy for drift neighborhoods leveraging explanations to focus sampling on locally influential features;
- an empirical evaluation using an open source HMT benchmark, demonstrating consistent error reduction and higher adaptation success rates under fixed budgets; and
- a replication package with code, models, and experimental artifacts.

The remainder is organized as follows. Section 2 presents the problem addressed in the paper together with the description of a running example in a human-machine teaming context. Section 3 introduces background notions we use in the rest of the paper. The VAI approach is illustrated in Sec. 4, and the empirical evaluation is presented in Sec. 5 with a discussion of threats to validity. Section 6 summarizes related work and Sec. 7 concludes the paper.

2 Problem Statement

We illustrate the problem by considering a Cyber-Physical System (CPS) that operates within a dynamic environment—such as a HMT scenario set in a hospital ward, where a service robot collaborates with medical personnel to assist patients during daily operations. The hospital environment includes some rooms, a waiting area for patients, and a storage room containing medical equipment. During the mission, the robot performs a sequence of tasks in coordination with a doctor and the patients: it first escorts a patient from the entrance to the waiting room, then follows the doctor to the storage room to retrieve the necessary medical equipment, escorts the doctor to the analysis room while carrying the equipment, and finally escorts the patient from the waiting room to the analysis room prepared for the visit. The mission is considered successful when all these services are completed without failure.

The success of the mission depends on several *features*, such as the patient's health status, the robot's battery level, and the relative positions of the humans and the robot. The set of features is henceforth referred to as *semantic space* [7]. The semantic space includes n continuous or discrete variables $X \subseteq \mathcal{X}^n$ describing both environmental and system properties[1]. The full set of variables in our illustrative example is reported in Table 1. Among these, some correspond to *controllable* features (those that can be modified) while others are only *observable* and can be monitored at runtime.

We denote the current *state* of the system and its surroundings as $\bar{v} \in X$, corresponding to a point in the semantic space and representing a specific operational condition. The system S is said to be dependable under a given operational condition when the dependability requirements R are satisfied: $S[\bar{v}] \models R$ holds. For example, a requirement may state that "the probability of mission success must remain above 0.9."

To ensure dependable operation, the system monitors these factors and adapts its behavior at runtime. For example, if the robot's battery is low, the system may decide to reduce the maximum speed of the robot or even reduce the robot's workload to maintain mission success. We assume that the system adopts a proactive self-adaptation mechanism in which a managing subsystem continuously monitors relevant features in the semantic space and predicts whether the current configuration $\bar{v}$ may lead to a requirement violation. When a potential violation is detected, the managing system computes an adaptation action that transitions the system to a new configuration $\bar{v}'$ by changing the controllable features such that $S[\bar{v}'] \models R$ holds—that is, the adaptation goal is achieved.

The system adopts a MAPE-K feedback loop. The Monitor component observes the system and environment, the Analyze component predicts whether $\bar{v}$ is acceptable or not (e.g., the green and red dots in Fig. 1a, respectively), and the Plan component identifies a suitable target configuration $\bar{v}'$. Finally, the Execute component enacts the adaptation, moving the system from $\bar{v}$ to $\bar{v}'$ (e.g., the current and target configurations in Fig. 1b). For example, if reducing the speed of the robot increases the likelihood of success, the control loop shifts the configuration accordingly.

At runtime, the analysis and planning components must operate under strict timing constraints. To meet these constraints, the Analyze and Plan components rely on a surrogate predictive model—often a regressor or classifier—as described by Camilli et. al [9]. These models efficiently predict the satisfaction of R based on $\bar{v}$, rather than using more demanding approaches such as invoking a high-fidelity simulator. Pre-trained surrogate models are inherently imperfect. In regions of the semantic space that are underrepresented or uncertain in the training data, the surrogate may yield inaccurate or overconfident predictions.

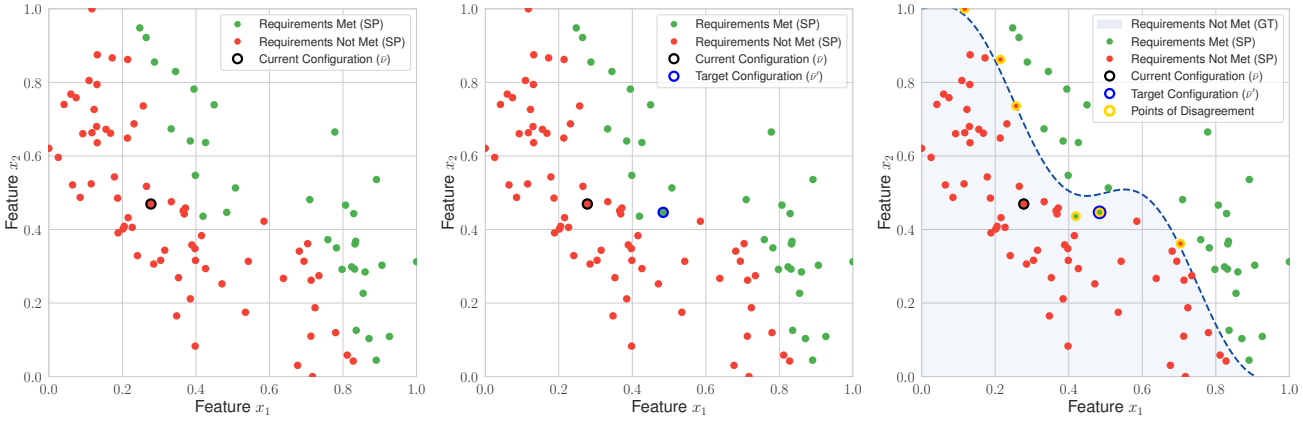
As shown in Fig. 1c, discrepancies between the surrogate model and the actual system behavior (i.e., ground truth) can lead to two types of mispredictions: *false negatives* in which the surrogate incorrectly predicts $S[\bar{v}] \not\models R$, triggering unnecessary adaptations; and *false positives* where the surrogate predicts $S[\bar{v}] \models R$, but in reality it does not—leading to ineffective adaptation actions. Although false negatives primarily affect efficiency, false positives are more critical, as they may cause the system to enter unsafe or undesirable operating conditions. For instance, a false positive might occur when the surrogate predicts that a new speed-position configuration for the robot ensures mission success, but in reality, a low battery prevents the robot from completing its task—potentially leaving the patient unattended.

This work addresses the problem of repairing self-adaptive behavior by automatically detecting and mitigating possible drifts in surrogate models that undermine system dependability. Specifically, we aim to improve the effectiveness of adaptation actions by identifying regions where the surrogate diverges from the ground truth and augmenting the knowledge of the managing system accordingly, with the ultimate goal of maintaining dependable adaptation despite imperfect models and evolving operational conditions.

3 Preliminaries

We introduce background concepts to make the paper self-contained: *supervised* ML, *interpretable* ML, and then *data augmentation*.

[1] $\mathcal{X}$ represents the joint domain of all feature types.



(a) Requirements not met with $\bar{\nu}$. **(b) Target $\bar{\nu}'$ to achieve goal.** **(c) Surrogate-ground truth drifts.**

Figure 1: Example with $n = 2$ features (SP=Surrogate Predictor, GT=Ground Truth).

Table 1: Semantic space of our illustrative example, including observable and controllable features.

Feature	Agent	Observable/ Controllable	Type	Domain	Description
Free will profile	Patient/Doctor	Observable	Categorical	{*focused, nominal, inattentive*}	Represents the human's attentional state, influencing how engaged or distracted they are during the collaborative task.
Health status	Patient/Doctor	Observable	Categorical	{*healthy, sick, unsteady*}	Indicates the general physical condition of the human, affecting mobility and task performance.
Age group	Patient/Doctor	Observable	Categorical	{*young, elderly*}	Distinguishes between demographic groups that may exhibit different motor and cognitive abilities.
Walking Speed	Patient/Doctor	**Controllable**	Continuous	$[30.0, 100.0]$ cm/s	Defines how fast the human can or should move during the collaboration, adaptable for coordination with the robot.
Initial position x	Patient	Observable	Continuous	$[0.0, 50.0]$ m	The patient's starting horizontal position in the shared workspace.
Initial position y	Patient	Observable	Continuous	$[0.0, 8.0]$ m	The patient's starting lateral position in the workspace.
Initial position x	Doctor	Observable	Continuous	$[0.0, 50.0]$ m	The doctor's initial horizontal position relative to the workspace or patient.
Initial position y	Doctor	Observable	Continuous	$[0.0, 8.0]$ m	The doctor's initial lateral position, influencing spatial coordination with others.
Translational Speed	Robotic Device	**Controllable**	Continuous	$[30.0, 100.0]$ cm/s	The robot's motion velocity, regulated to synchronize with the human's pace and task context.
Battery charge	Robotic Device	Observable	Continuous	$[11.1, 12.4]$ V	The robot's power level, determining available operational time and need for recharging.
Maximum Distance	Robot Controller	**Controllable**	Continuous	$[5.0, 7.5]$ m	The maximum allowed separation between robot and human to ensure safety and communication effectiveness.
Minimum Distance	Robot Controller	**Controllable**	Continuous	$[2.0, 4.5]$ m	The minimum safe distance to avoid collisions or discomfort in close-proximity collaboration.
Maximum Fatigue	Robot Controller	**Controllable**	Continuous	$[0.5, 0.8]$	Upper fatigue threshold above which the orchestrator instructs the human to stop walking.
Minimum Fatigue	Robot Controller	**Controllable**	Continuous	$[0.1, 0.4]$	Lower fatigue threshold below which it is considered safe for the human to resume walking.
Time Bound (τ)	-	Observable	Continuous	$[250, 700]$	Represents the temporal constraint or deadline within which the task must be completed.
Mission Progress	-	Observable	Discrete	$[0, 100]$	Indicates the percentage of the task completed by the team, reflecting overall mission advancement.

3.1 Supervised Machine Learning

Supervised ML [25] aims to learn a predictive function $\hat{f}$ that maps input data to an output variable based on labeled examples. A dataset $\mathcal{D} = (x_i, y_i)_{i=1}^m$ is used for training, where each input vector $x_i = [x_{i,1}, \ldots, x_{i,n}] \in \mathbb{R}^n$ represents feature values, and y_i is the corresponding output label or value.

In classification [20], the target variable y_i takes one of a finite set of discrete labels (e.g., 0, 1 for binary classification). The model $\hat{f} : \mathbb{R}^n \to y_1, \ldots, y_k$ learns decision boundaries separating the classes. Typical algorithms include Decision Trees, Random Forests, and Neural Networks. Probabilistic classifiers also estimate the likelihood of a class given an input, denoted as $\mathcal{P}(y = y_i \mid x)$.

In regression, the target variable $y_i \in \mathbb{R}$ is continuous, and the model $\hat{f} : \mathbb{R}^n \to \mathbb{R}$ estimates the expected value of y given x. Regression models are widely used to predict quantitative outcomes such as system performance metrics, probabilities of success, or reliability scores. Common regressors include Linear and Polynomial Regression, Random Forest Regression, and Neural Networks.

Some models, such as linear regressors or decision trees, are inherently interpretable due to their simple structure. More complex models—like deep neural networks or ensemble methods—are often treated as *black boxes*, as the relationships they learn are not directly understandable by humans.

3.2 Interpretable Machine Learning

Interpretable Machine Learning (ML) [26] seeks to make the decision processes of ML models transparent and understandable. Interpretability can be pursued at two complementary levels: *global interpretability*, which aims to capture the overall behavior of the model across the entire input space; and *local interpretability*, which focuses on explaining individual predictions for specific instances.

Local Interpretable Model-agnostic Explanations (LIME) [28] approximates the behavior of a model $\hat{f}$ around a particular instance x by learning an interpretable surrogate model $\hat{g}$ (e.g., a simple linear regressor) using a set of perturbed samples near x. The resulting explanation is derived from the weights of $\hat{g}$, which quantify the local importance of each feature, indicating how strongly and in which direction it contributes to the prediction $\hat{f}(x)$. This allows the interpretation of black-box models through simple approximations valid within a limited neighborhood of interest.

Formally, LIME solves the following optimization problem:

$$\hat{g} = \arg\min_{g \in G} \mathcal{L}(\hat{f}, g, \pi_x) + \Omega(g) \tag{1}$$

$\mathcal{L}(\hat{f}, g, \pi_x)$ measures the fidelity of surrogate g to the original model $\hat{f}$ within the locality of x (weighted by the proximity kernel π_x), while $\Omega(g)$ penalizes complexity to preserve interpretability.

3.3 Data Augmentation

Data augmentation refers to the process of expanding an existing dataset with additional synthetic or transformed samples that preserve the underlying data distribution [34]. Originally introduced in computer vision and natural language processing to improve model generalization, data augmentation has become a general strategy for addressing data scarcity and mitigating bias.

Formally, given an initial training dataset $\mathcal{D} = (x_i, y_i)_{i=1}^{m}$, the goal of augmentation is to generate an extended dataset

$$\mathcal{D}' = \mathcal{D} \cup \{(x'_j, y'_j)\}_{j=1}^{m'} \tag{2}$$

such that the new samples (x'_j, y'_j) are consistent with the underlying ground truth distribution.

In regression settings, augmentation can be achieved by introducing controlled perturbations to the input features or by synthetically sampling new data points guided by domain knowledge. For example, in the HMT scenario, the system may generate new data points by slightly varying the operator's workload or the robot's speed.

4 The VAI Framework

This section presents VAI, our novel approach to address the problem introduced in Sec. 2. In the following, we present our assumptions about the managing system and then describe in detail VAI, including the main components: `Verify`, `Augment`, and `Improve`.

4.1 Overview

Figure 2 illustrates the architecture of the Verify, Augment, and Improve (VAI) framework. VAI extends a standard managing system for self-adaptive systems organized around the MAPE-K control loop, as introduced in Sec.2. The managed system corresponds to a target CPS, such as the HMT illustrative example, characterized by

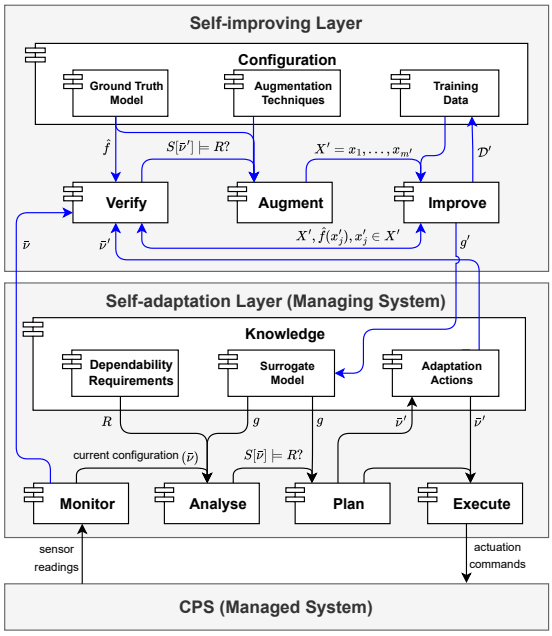


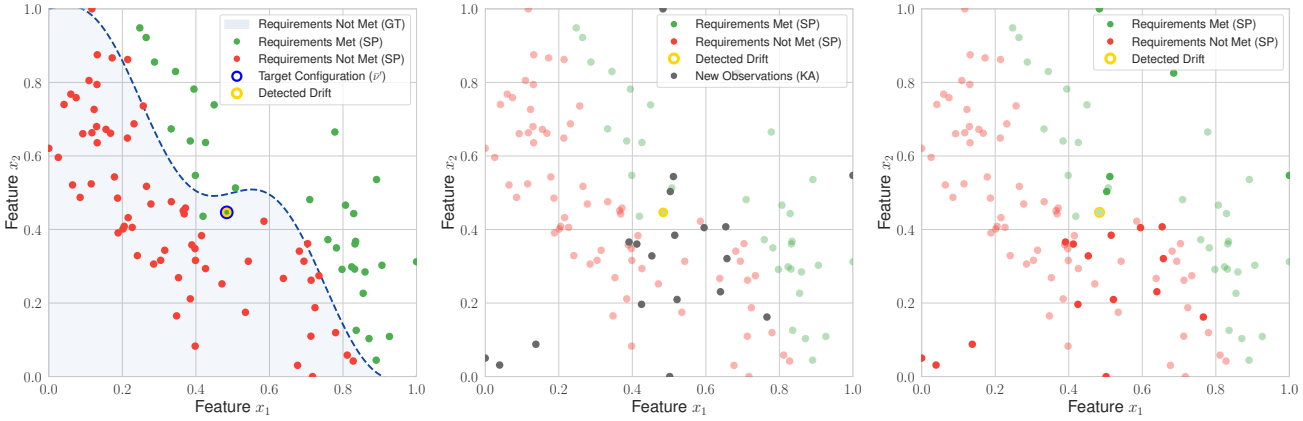
Figure 2: VAI framework with the operational workflow of the self-improving layer highlighted in blue.

a semantic space composed of observable and controllable features (see Table 1).

Beyond the standard control loop, VAI introduces an additional self-improving layer that operates *asynchronously* with respect to the main MAPE-K loop. This layer is triggered whenever a new adaptation action is available and analyzes the effectiveness of the adaptation actions executed on the managed system. Whenever an adaptation action is found to be ineffective, a repair process is triggered with the goal of reducing discrepancies between the surrogate model's predictions and the actual system behavior.

The repair process is realized through three main components, realizing the *verify-augment-improve* cycle. First, the `Verify` component identifies ineffective adaptation actions by comparing the expected probability of satisfying the dependability requirements—predicted by the surrogate model—with the actual probability computed using the high-fidelity *ground truth model*. These explanations guide the `Augment` component steers a targeted sampling process to select new data points within the semantic space. The newly sampled configurations are then passed to the `Improve` component, which labels them using the ground truth model (via `Verify`). Finally, the `Improve` component uses the labeled samples to retrain and redeploy the surrogate predictive model stored in the Knowledge component of the managing system.

As stated above, the goal of this self-repair layer is to iteratively refine the surrogate model, ensuring that future adaptation decisions become increasingly accurate and effective. Because the verification process operates asynchronously with respect to the MAPE-K loop, its potentially higher computational cost does not



(a) $\bar{v}'$ is identified as a drift. **(b) New samples in a neighborhood of $\bar{v}'$.** **(c) Processed new samples.**

Figure 3: Example of the Verify, Augment, and Improve stages.

interfere with the system's real-time responsiveness. When a repair iteration concludes while the system is still operational, the updated surrogate model immediately enhances subsequent adaptation decisions—allowing VAI to achieve continuous self-improvement.

4.2 Managing System

The `Monitor` component of the managing system maintains and continuously updates the vector $\bar{v}$, which represents the latest configuration of the semantic space and captures the relevant aspects of the operating conditions observed from both the managed system and its surrounding environment.

To enable proactive adaptation under strict time constraints, the `Analyze` component employs a pre-trained regression model $g : \mathcal{X}^n \to \mathbb{R}$, stored in the `Knowledge` component, to predict whether the current feature assignment $\bar{v}$ may lead to a violation of the dependability requirements R—that is, whether $S[\bar{v}] \not\models R$ holds. In our illustrative HMT example (see Sec. 2), the regressor estimates the probability of mission success, that is, the likelihood of completing the collaborative task within a specified time bound. As an example, if a requirement in R states that this probability must remain above 0.9, the system is considered dependable whenever $g(\bar{v}) > 0.9$ holds. Otherwise, a potential requirement violation is detected, and an adaptation process is initiated.

The `Plan` component identifies a new configuration $\bar{v}'$ such that $S[\bar{v}'] \models R$ holds, essentially improving the system's state, The identification of new configurations is formulated as a multi-objective optimization problem in which the system seeks to maximize or minimize a set of possibly conflicting objectives $O = O_1, \ldots, O_k$, subject to the operational constraints defined over the semantic space. In our HMT example, these objectives include maximizing the probability of mission success while minimizing the fatigue level of the human collaborators, both of which depend on the controllable features of the system [22].

The surrogate model g behaves as a black-box predictor—that is, it provides outcome estimates without exposing analytical gradients or explicit structure. Therefore, we assume the use of metaheuristic search techniques to solve the optimization problem. In particular, the multi-objective evolutionary algorithm NSGA-II [14] is used to explore the semantic space, identify trade-offs among competing objectives, and generate a set of Pareto-optimal candidate configurations from which $\bar{v}'$ is selected.

Finally, the `Execute` component computes and issues the necessary actuation commands to transition the system from $\bar{v}$ to $\bar{v}'$, enacting the planned adaptation. The pair $\langle \bar{v}, \bar{v}' \rangle$ and its outcome are recorded in the `Knowledge` component, enriching the historical context for subsequent analysis and improvement.

4.3 Self-improving Layer

Asynchronously with respect to the execution of the adaptation action, whenever the `Plan` component updates the `Knowledge` base in the managing system with a new adaptation pair $\langle \bar{v}, \bar{v}' \rangle$, a new run of the self-improving layer is triggered. This layer incrementally refines the shared knowledge and thereby repairs the predictive surrogate model used by the managing system to support analysis and planning. The underlying idea is to detect and mitigate drifts between the surrogate model and the actual system behavior through the comparison with a high-fidelity ground truth model.

Notice that a systematic and exhaustive exploration of the semantic space to label all possible configurations is generally infeasible. Moreover, relying solely on the high-fidelity model instead of the surrogate would violate the time constraints of the adaptation process, especially in critical domains such as our HMT example. For these reasons, the self-improving layer adopts a *reactive* strategy: each time a new configuration $\bar{v}'$ is produced by the planning phase, it is selectively verified using the ground truth model.

Because the repair process operates *asynchronously* with respect to the MAPE-K loop iteration that initiated it, the lower layers can immediately benefit from the refined knowledge and the more accurate surrogate model once the self-improving layer completes and deploys the updated model into the shared knowledge base. In our HMT example, if the self-improving layer concludes while the mission is still in progress, the refined surrogate regressor becomes available to guide subsequent adaptation decisions within the same mission.

The following section details the main components and workflow of the self-improving layer.

4.3.1 Verify. The `Verify` component is designed to detect ineffective adaptation actions. Namely, adaptation actions enacted by the `Execute` component that do not achieve the adaptation goals. As anticipated in Sec. 2, this reduces to new operating configurations $\bar{v}'$ where the system is not considered dependable: $S[\bar{v}'] \not\models R$. To this end, the `Verify` component leverages a *ground truth model* of the managed system stored in the `Configuration` component. The ground truth provides a reference for the system's behavior and can take different forms depending on the domain and available resources. The ground truth model is assumed to be sufficiently accurate to serve as a reliable reference for the actual behavior of the managed system. Notice that we do not address potential discrepancies between the ground truth and the managed system. Such drifts can be mitigated through the adoption of a *digital shadow* paradigm [21], where the ground truth is continuously synchronized with real-world observations to maintain its fidelity over time.

We denote the ground truth model as $\hat{f} : \mathcal{X}^n \to \mathbb{R}$, where $\hat{f}(\bar{v}')$ returns the estimated probability that the system operating under configuration $\bar{v}'$ satisfies the target dependability requirement R. The resulting value $\hat{f}(\bar{v}')$ thus represents the actual confidence level that $S[\bar{v}'] \models R$ holds, providing a reliable basis for assessing the effectiveness of the adaptation action.

As an example, the ground truth can be realized by using a high-fidelity simulator that accurately captures the system dynamics and environmental interactions[2]. Such simulators can be parametrized using the variables defined in the semantic space, allowing the evaluation of dependability requirements under specific feature assignments $\bar{v}'$. In this case, by executing multiple simulation runs of the target HMT mission, the `Verify` component can estimate the probability of satisfying the dependability requirements R for a given configuration.

Alternatively, the ground truth can be expressed as a formal specification of the system, modeled for instance as a network of Stochastic Hybrid Automaton (SHA) [13]. This formalism provides a mathematical framework to model CPSs that combines discrete control logic, continuous physical dynamics, and stochastic events—all of which are essential to capture the uncertainty and variability inherent in real-world operating environments. In this case, verification can be performed through Statistical Model Checking (SMC) [12], enabling systematic exploration of relevant behavioral traces under the constraints expressed by $\bar{v}'$.

The `Verify` component is employed at two distinct stages within the execution of the self-improving layer. In the first stage, it is used to detect drifts between the surrogate model and the ground truth. Specifically, the surrogate model provides an estimated probability $g(\bar{v}')$—the value that originally led the `Plan` component to select $\bar{v}'$ as the target configuration. Then, the same configuration is evaluated using the ground truth model, obtaining $\hat{f}(\bar{v}')$. If the two models disagree on whether $S[\bar{v}'] \models R$ holds, a drift is detected, indicating an ineffective adaptation (e.g., yellow circle in Fig. 3a).

[2]Examples include mainstream simulation platforms such as CARLA [15] and CoppeliaSim [29] which provide flexible APIs for simulations in realistic virtual settings.

In the second stage, once a drift has been detected, the `Verify` component is invoked again to label new data points generated by the `Augment` component described below.

4.3.2 Augment. The `Augment` component acquires additional knowledge about the semantic space, with particular focus on the region surrounding a detected drift. When the `Verify` component identifies a drift at configuration $\bar{v}'$, it signals that the corresponding area of the semantic space is likely underrepresented or poorly characterized in the surrogate model's training data, leading to unreliable predictions and ineffective adaptation actions. The goal of `Augment` is to strategically explore and sample this region to generate new, informative data points that help refine the surrogate model and improve its local fidelity (e.g., gray dots in Fig. 3c).

The augmentation process follows three conceptual steps:

- *neighborhood definition*: determine the region of interest around the drift point $\bar{v}'$ in the semantic space, typically within a configurable radius r using a *heterogeneous distance metric* [38] on the feature vector (usually including a mix of numerical, ordinal, and categorical values);
- *sample generation*: apply a sampling strategy to select new candidate configurations that are likely to provide informative feedback to the surrogate model; and
- *data preparation*: collect the generated points into a new set of samples $X' = x'_1, \ldots, x'_{m'} \subseteq \mathcal{X}^n$, where each x'_j represents a candidate configuration to be labeled by the ground truth model through the `Verify` component.

The choice of sampling strategy influences both the diversity and the informativeness of the generated data points. In the current implementation of the framework, three alternative augmentation techniques can be selected through the `Configuration` component.

Random Sampling. Random Sampling (RS) in the neighborhood is the simplest augmentation technique embedded in VAI. It generates N new candidate configurations by sampling uniformly within the local neighborhood of the drift point $\bar{v}'$ within the radius r. RS is computationally inexpensive and requires no assumptions about the underlying data distribution. However, its effectiveness is limited in high-dimensional or heterogeneous semantic spaces, where uniform sampling may produce many redundant or uninformative points. It also serves as a useful baseline for comparing more sophisticated augmentation strategies.

Kernel Density Estimation. Kernel Density Estimation (KDE) provides a more informed, probabilistic approach to sampling. Given the subset of configurations in the original dataset $\mathcal{D}$ that are close to $\bar{v}'$, KDE estimates the local probability density function of the semantic space around the drift. The new N samples are then drawn from this estimated distribution, effectively replicating the most likely patterns of behavior observed in the proximity of the drift—that is, combinations of feature values that frequently co-occur in past observations and are statistically representative of realistic operating conditions near $\bar{v}'$. This enables a targeted exploration of relevant areas, increasing the likelihood of obtaining meaningful data for retraining. While computationally more demanding than RS, KDE offers a better way to focus augmentation efforts on plausible operating conditions.

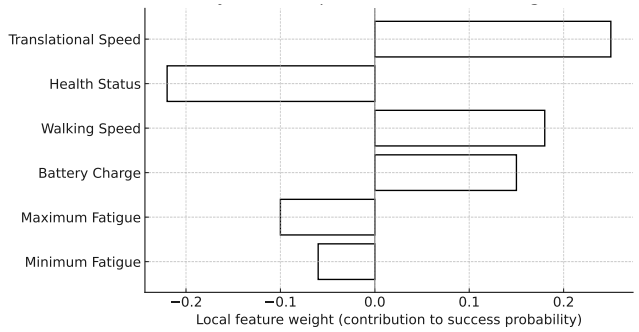


Figure 4: Example of LIME explanation.

eXplanation-driven Augmentation. The technique eXplanation-driven Augmentation (XA) leverages interpretability to guide the sampling process. Specifically, it uses LIME (see Sec. 3) to analyze the surrogate model's local behavior around the drift configuration $\bar{v}'$. LIME approximates the surrogate model with a locally interpretable linear model, identifying the features that most strongly influence the surrogate's prediction in that neighborhood. Figure 4 illustrates a LIME explanation in the context of our HMT illustrative example. In the plot, features are ranked by their relative importance to the predicted outcome, and each feature is associated with a weight indicating its positive or negative contribution. These feature attributions are then used to guide the sampling strategy: features with higher influence (e.g., translational speed in Fig. 4) are perturbed more extensively to explore the most critical regions of the semantic space, whereas less influential features (minimum fatigue in Fig. 4) are varied within narrower ranges, proportionally scaled to the magnitude of their weights. The approach yields N new samples through a semantically meaningful exploration of the area responsible for the model's inaccuracy.

4.3.3 Improve. The newly generated set X' is then passed to the `Improve` component, which maps each new sample x'_j to the corresponding output label y'_j. Recall that the purpose of this phase is to repair the surrogate model in the region where it was previously found to be inaccurate. Since the self-improving routine operates asynchronously with respect to the main MAPE-K loop, we assume it is not constrained by real-time requirements. Thus, each label $y'_j = \hat{f}(x'_j)$ is obtained by using again the `Verify` component (leveraging the ground truth model), ensuring accurate estimation of the probability that the system configuration x'_j satisfies the dependability requirement R. This process produces an augmented dataset $\mathcal{D}' = (x'_j, \hat{f}(x'_j))_{j=1}^{m'}$, which is then merged with the original training data to form $\mathcal{D} \cup \mathcal{D}'$ represented in Fig. 3c.

The resulting augmented dataset $\mathcal{D} \cup \mathcal{D}'$ is used to retrain the surrogate model, yielding an updated predictor g', ready to be deployed in the managing system's `Knowledge` component.

5 Evaluation

This section describes the empirical evaluation of VAI. We answer the following research questions:

RQ1: What is the effectiveness of VAI in terms of the relative error of the adaptation decisions?
RQ2: What is the effectiveness of VAI in terms of achievement of adaptation goals?
RQ3: What is the cost overhead of VAI?

To answer RQ1 and RQ2, we compare the results obtained by using different knowledge augmentation methods embedded into VAI. We answer RQ1 by comparing the relative error of the adaptation decisions before and after automated knowledge augmentation, while we answer RQ2 by comparing the ability of achieving the adaptation goals (i.e., satisfaction of dependability requirements) before and after augmentation. To answer RQ3, we quantify the cost (in terms of execution time) required by the knowledge augmentation methods embedded into VAI.

5.1 Design of the Evaluation

5.1.1 Evaluation subjects. We designed and carried out an experimental campaign to evaluate VAI using the same budget in terms of number of adaptation requests and the same benchmark: an open-source CPS in the area of HMT [5]. We selected an existing benchmark with nontrivial complexity, taking into account various factors, including the complexity of the scenario and the number of variables in the semantic space, which encompass both observable and controllable features (see Table 1). The selected benchmark represents the complete version of our illustrative example in Sec. 2.

We conduct our experiments using an existing platform designed to simulate and verify the stochastic behavior of systems. Specifically, we employ Uppaal SMC [12] to perform multiple simulations based on a formal specification of the human–machine teaming dynamics, expressed as an SHA network [1]. This specification defines a set of varying factors within the semantic space, enabling the model checker to assess the satisfaction of given dependability requirements under different value assignments of these factors. To ensure meaningful verification results, we constrain the parameter intervals in the specification to exclude trivial cases (i.e., scenarios where the requirements are always satisfied or always violated). The adaptation goal is defined as the satisfaction of a dependability requirement, which states "*the human-machine teaming mission must succeed with probability higher than* 0.9."

We consider two variants of the benchmark listed in Table 2, denoted as $subject_1$ and $subject_2$. In both cases, the system integrates a regression model within the shared knowledge of the MAPE-K loop, which steers adaptation decisions. The regressor is trained using data generated by the simulator component. Specifically, the training set consists of 500 data points (x, y), where each x is a configuration of the semantic space, and $y = \hat{f}(x)$ is the probability of success of the HMT mission estimated by the simulator. As discussed in Sec. 4, the simulator is further employed for automated knowledge augmentation whenever an adaptation attempt *fails*—that is, when the adaptation goal is not achieved.

The two subjects differ in ground truth model. $Subject_1$ serves as an exploratory case, using an alternative regressor that better approximates the system's behavior under the scenario simulated by the statistical model checker. For this case, we employ a larger

Table 2: Evaluation subjects.

subject	benchmark	adaptation goal	knowledge model	ground truth
$subject_1$	HMT [23]	P(succes) >0.9	regressor |training set| = 500	regressor |training set| = 1000
$subject_2$	HMT [23]	P(succes) >0.9	regressor |training set| = 500	model checker

Table 3: Statistical tests for RQ1.

subject	A	B	p-value	$\hat{A}_{AB}$
$subject_1$	KDE	RS	5.83e-02	0.46
	XA	RS	1.14e-01	0.47
	XA	KDE	5.91e-01	0.50
$subject_2$	KDE	RS	2.79e-01	0.48
	XA	RS	7.89e-09	0.38
	XA	KDE	3.92e-06	0.40

training set of 1000 data points (twice the size of the model embedded in the knowledge). In contrast, $subject_2$ directly relies on the statistical model checker as its simulation ground truth.

Further details on the statistical model checker, the modeled factors, and the applied constraints are provided in the supporting material (see Sec. 7).

5.1.2 Methods under comparison. We compare alternative knowledge augmentation methods embedded into VAI: XA, KDE, and RS. For each subject, we execute multiple adaptation requests and measure the ability to meet the adaptation goals. We then identify all failed adaptation attempts and apply the corresponding repair process guided by each alternative method. Finally, we re-execute the adaptation requests to quantify the improvement achieved relative to the baseline (no-repair) condition.

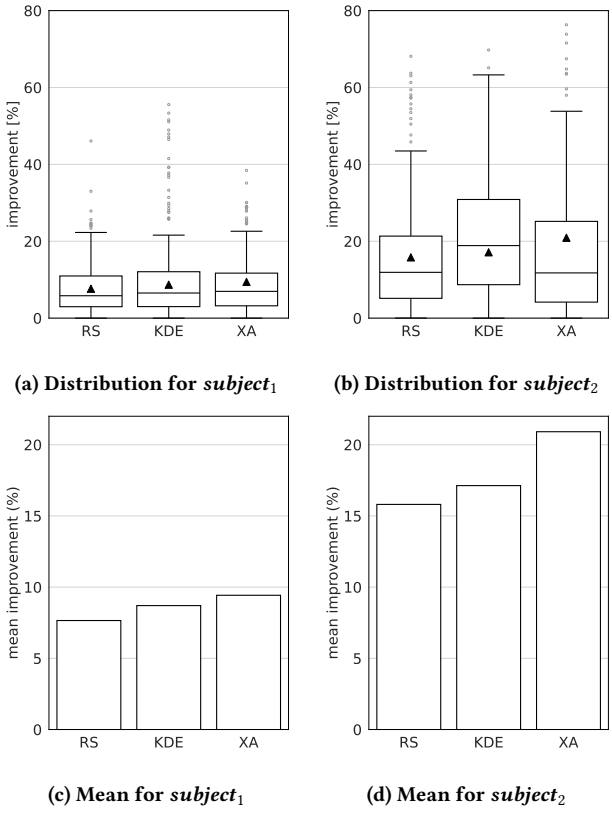
We use the same setting for all methods under comparison. Specifically, we use a sample size of $N = 50$ points to augment the knowledge during the repair process. Preliminary evaluations showed that this setting is sufficient for the relative error to plateau.

5.1.3 Statistical tests. To reduce the risk of obtaining results by chance, we account for randomness in all the approaches by repeating the repair process 50 times—starting from 50 unsuccessful adaptations out of 100 adaptation requests in total. According to the guideline introduced by Arcuri & Briand [2], we apply the non-parametric Mann–Whitney U test [24] to evaluate the statistical significance of the results, using a standard significance level of $\alpha = 0.05$. We also measure Vargha and Delaney's $\hat{A}_{AB}$ [33] to compute the effect size of the difference between A and B. We adopt the following standard classification: effect size $\hat{A}_{AB}$ ($= 1 - \hat{A}_{BA}$) is small, medium, and large when its value is greater than or equal to 0.56, 0.64, and 0.71, respectively.

5.1.4 Evaluation testbed. All experiments have been executed on a desktop machine running Ubuntu 22.04.1 LTS, equipped with Intel Xeon Silver 4114 CPU at 2.20GHz (16 cores) and 32GB RAM.

5.2 Evaluation Results

5.2.1 Relative Error (RQ1). To address RQ1, we execute VAI using the alternative knowledge augmentation methods (XA, KDE, and RS). We compare the outcomes of the adaptation requests



(a) Distribution for $subject_1$ **(b) Distribution for $subject_2$**

(c) Mean for $subject_1$ **(d) Mean for $subject_2$**

Figure 5: Relative error improvement (the higher, the better).

against the selected baseline condition—adaptation without any repair procedure—henceforth referred to as *no-repair*. The effectiveness of VAI relative to the no-repair baseline is quantified using the *relative error*, defined as the difference between the estimated probability of mission success and the actual probability determined by the ground truth. For each subject ($subject_1$ and $subject_2$), we execute 100 adaptation requests, each representing a full run of the MAPE-K loop initiated from points in the semantic space where the target dependability requirement is not satisfied. For each run, we measure the relative error. Whenever an adaptation is ineffective (i.e., the adaptation goal is not achieved), we invoke the repair process and subsequently re-execute the initial 100 adaptation requests to measure the relative error after enhancing the adaptation capability using the alternative knowledge augmentation methods. This procedure allows us to quantify the *improvement* in adaptation accuracy by computing the relative percentage reduction in error with respect to no-repair. The relative improvement is defined as:

$$\Delta_{\text{rel}} = \frac{E_{\text{before}} - E_{\text{after}}}{E_{\text{before}}} \cdot 100\% \tag{3}$$

where E_{before} denotes the relative error before the repair process (no-repair condition), and E_{after} represents the relative error after applying the respective knowledge augmentation method.

Results. Figure 5 shows the relative error improvement measured for all subjects and all augmentation methods embedded into VAI.

Specifically, Fig. 5a and Fig. 5b show the distribution of the relative error improvement for $subject_1$ and $subject_2$, respectively, while Fig. 5c and Fig. 5d show the mean relative error improvement for $subject_1$ and $subject_2$, respectively. Table 3 reports the results of the statistical tests comparing the distributions of relative error improvement across the evaluated methods.

VAI consistently reduces the relative error for both subjects. For $subject_1$, the improvement distributions are concentrated between approximately 5% (lower quartile, Q1) and 10% (upper quartile, Q3). According to Table 3, no statistically significant differences are detected among the distributions. However, Fig. 5c indicates that the mean improvement achieved by XA (9.43%) is higher than that of KDE (8.70%) and RS (7.65%). For $subject_2$, a similar trend is observed, with generally higher relative error improvements compared to $subject_1$. As shown in Table 3, the differences between XA and RS, as well as between XA and KDE, are statistically significant, although the corresponding effect sizes are negligible. Figure 5c further illustrates that XA achieves the highest mean improvement (20.91%), followed by KDE (17.13%) and RS (15.81%).

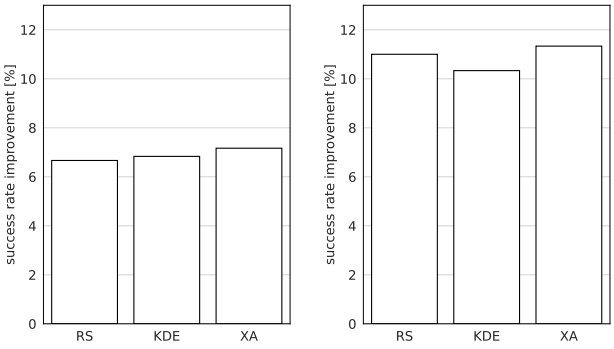
Generally, the higher improvement levels observed for $subject_2$ can be explained by the fact that $subject_1$ relies on a surrogate ground truth (see Table 2), which approximates the actual behavior and thus limits the precision of the feedback available for learning and repair. In contrast, $subject_2$ employs the actual ground truth provided by the statistical model checker, enabling more accurate evaluation and more effective refinement of the knowledge.

RQ1 summary. Across both subjects, VAI consistently reduces the relative error. XA achieves the highest mean improvement—9.43% for $subject_1$ and 20.91% for $subject_2$—with statistically significant differences in the latter. The greater improvement in $subject_2$ results from its use of the actual ground truth rather than a surrogate approximation.

5.2.2 Achievement of adaptation goals (RQ2). To address RQ2, we evaluate the impact of the alternative knowledge augmentation methods (XA, KDE, and RS) on the success rate of adaptation within VAI. As in RQ1, we compare the outcomes of the adaptation requests against the baseline condition—adaptation without any repair procedure. In this case, the effectiveness of VAI is quantified using the *success rate*, defined as the proportion of adaptation requests that successfully achieve the adaptation goal (i.e., the target dependability requirement). For each subject ($subject_1$ and $subject_2$), we execute 100 adaptation requests, each corresponding to a full execution of the MAPE-K loop initiated from points in the semantic space where the requirement is initially unsatisfied. Whenever an adaptation attempt fails, the repair process is invoked using one of the alternative knowledge augmentation methods, after which all 100 adaptation requests are re-executed. We then measure the improvement in success rate compared to the no-repair condition, computed as the percentage increase:

$$\Delta_{\text{scs}} = (S_{\text{after}} - S_{\text{before}}) \cdot 100\% \quad (4)$$

where S_{before} denotes the success rate before the repair process (no-repair), and S_{after} represents the success rate after applying the knowledge augmentation method.



(a) Improvement for $subject_1$ **(b) Improvement for $subject_2$**

Figure 6: Success rate improvement (the higher, the better).

Results. Figure 6 shows the success rate improvement measured for all augmentation methods and subjects: $subject_1$ in Fig. 6a and $subject_2$ in Fig. 6b.

Overall, the results are consistent with the findings reported for RQ1. For both subjects, VAI consistently increases the success rate of adaptation decisions, primarily due to the higher accuracy (i.e., reduced relative error) achieved after the repair process. Lower relative error levels generally correspond to higher improvements in success rate. On average, the success rate improvement is 6.89% for $subject_1$ and 10.88% for $subject_2$. Among the evaluated methods, XA yields the highest improvement in both subjects. For $subject_1$, XA achieves an improvement of 7.17%, followed by KDE (6.83%) and RS (6.67%). For $subject_2$, XA again performs best (11.33%), followed by KDE (10.33%) and RS (10.0%).

RQ2 summary. VAI consistently increases the success rate of adaptation decisions across both subjects, with average improvements of 6.89% for $subject_1$ and 10.88% for $subject_2$. XA achieves the highest gains in both cases, confirming that greater accuracy after repair (lower relative error) generally translates into higher adaptation success.

5.2.3 Cost (RQ3). To address RQ3, we assess the computational cost of the alternative knowledge augmentation methods (XA, KDE, and RS) integrated into VAI. We adopt the same experimental setup used for RQ1 and RQ2, and measure cost in terms of wall-clock execution time of the augmentation phase. Specifically, we measure the time required by each run of the repair process, excluding the time of the verification component, which derives the actual labels from the ground truth model (see Sec. 4). This approach isolates the intrinsic cost of each augmentation method, independent of the specific verification mechanism employed—such as a high-fidelity simulator or an accurate predictive model.

Figure 7 presents the distribution of the augmentation cost, expressed as execution time in log scale, for each method (RS, KDE, and XA). We observe that XA yields the highest cost, with execution times on the order of 1s, primarily due to the additional computation required for explanation generation and reasoning during augmentation. KDE incurs moderate cost, with execution

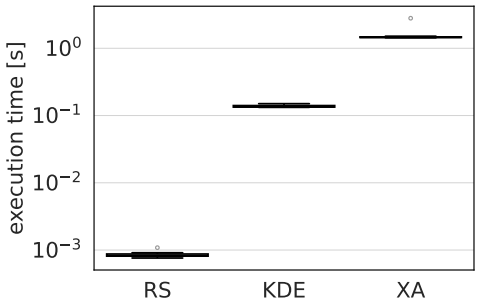


Figure 7: Cost of the sampling methods.

times around 0.1s, reflecting the overhead of density estimation and sample selection. RS is by far the most computationally efficient method, with execution times on the order of 0.001s, as it relies solely on random sampling in the neighborhood of the ineffective adaptation actions without additional processing.

Overall, the cost of all augmentation methods is negligible compared to the time required by the verification step, which involves substantially more computationally demanding operations. In our experimental campaign, this component relies on a regressor for $subject_1$ (the "simplified" exploratory case) and on the statistical model checker for $subject_2$ (the complete case). In the former, individual predictions execute in approximately 0.015s on average (standard deviation 0.0002s), while in the latter, the statistical model checker requires an average of 16.45s (standard deviation 14.61s). These results confirm that the computational cost of the augmentation methods is indeed negligible relative to the verification process.

RQ3 summary. The computational cost of the knowledge augmentation methods is minimal compared to the verification step. Among the evaluated methods, XA is the most expensive ($\sim$1s per run), followed by KDE ($\sim$0.1s) and RS ($\sim$0.001s). All augmentation costs remain negligible relative to the verification process, which dominates total execution time.

5.3 Threats to Validity

To mitigate internal validity threats, we used a controlled experimental setup where all knowledge augmentation methods (XA, KDE, and RS) are executed under identical configurations and computational budgets. We reduce the possibility of obtaining results by chance, mitigating randomness inherent to sampling and simulation by repeating the repair process 50 times (considering 100 adaptation requests in total) and by applying non-parametric statistical tests (Mann–Whitney U test) following established guidelines [2]. Still, some residual variability in results cannot be completely excluded, as stochastic simulations and learning-based components may introduce non-determinism

In our experimental campaign, we adopt two complementary indicators—relative error and success rate—to measure the effectiveness of adaptation, and wall-clock execution time to measure computational cost. These metrics quantify the accuracy and efficiency of the repair process, aligning with the research questions.

We mitigate external validity threats by leveraging an open-source human–machine teaming benchmark with nontrivial stochastic behavior, using two variants with different ground truths to cover both exploratory and complete cases. While this benchmark captures realistic adaptation dynamics, further studies on additional domains (e.g., autonomous driving, smart manufacturing) would increase the generalization of our findings. Our evaluation employs a specific configuration of the MAPE-K loop and model-based verification; alternative architectures or adaptation logics may yield different quantitative outcomes.

6 Related Work

Recent contributions have emphasized the use of interpretable and explainable models to increase trust and understandability of adaptation decisions. Bencomo et al. [4, 19] discuss how explainability can be embedded within software engineering processes to enhance the transparency of autonomic decisions. More recently, explainable, model-agnostic ML components have been incorporated into MAPE-K loops to improve adaptation efficiency in critical domains such as autonomous vehicles [27].

Another research line extends the MAPE-K loop with an additional upper layer for meta-adaptation and continual learning. Gheibi & Weyns [16] propose a lifelong self-adaptive architecture that continuously manages knowledge acquisition and model revision to deal with new emerging tasks, such as concept shift in input data. Similarly, Klos et al. [18] introduce an upper layer with evaluation, learning, and verification components that autonomously refine adaptation logic by learning new adaptation rules. Architectural guidelines for systematically integrating such self-improving capabilities are also envisioned by Chhetri et al. [11]. The authors propose a reference architecture for self-improving autonomic systems and a set of desirable features. Frameworks such as ALM [30] and HAFLOOP [41] embed recursive or hierarchical feedback loops to dynamically refine adaptation strategies. Weyns et al. [36] formalize these ideas in the context of self-evolving systems, where self- and context-awareness drive the continuous evolution of goals and architectures through evolutionary learning.

More recent work seeks to achieve antifragility [32], enabling systems to improve by learning from disruptive or adverse events. Grassi et al. [17] present a reference model that triggers and exploits learning through experimentation with system replicas. Other frameworks combine active and reactive learning strategies to achieve self-improvement at runtime [31]. Complementary approaches rely on search-based and lifelong learning techniques to accelerate adaptation under dynamic workloads. For example, Ye et al. [39] integrate lifelong planning with knowledge distillation to reuse beneficial past states, while search-based techniques are used to update configuration plans on the fly in highly configurable systems [40]. Chen et al. [10] propose a proactive self-adaptation framework that improves system reliability by fusing context predictions using evidence theory and by planning adaptations through an extended stochastic model predictive control approach.

VAI operationalizes self-improvement within the managing system itself by closing the loop between failure detection, explainable

diagnosis, and targeted repair—transforming explainability mechanisms from a passive descriptive aid into an active mechanism for continuous adaptation refinement.

7 Conclusion

We introduce VAI, a novel framework that augments MAPE-K with an asynchronous self-improving layer. By verifying executed adaptations against a ground truth model, explaining drifts, and augmenting knowledge with targeted samples, VAI incrementally repairs the surrogate models that drive adaptation.

Our empirical evaluation using two subjects in a human–machine teaming scenario shows that VAI increases the accuracy and dependability of adaptation. We compare alternative augmentation strategies—RS, KDE, and XA—under two different ground truth models. Results demonstrated that VAI consistently reduces the relative error of adaptation decisions and improves the success rate of achieving adaptation goals. Among the tested methods, XA achieved the best overall effectiveness while maintaining negligible computational cost compared to verification.

We plan to explore other interpretable augmentation strategies (e.g., SHAP, counterfactuals), integrate active learning for sample selection, and evaluate across additional domains with runtime assurance for safe online repair.

Data Availability

The replication package of our experiments, including the implementation of VAI and the instructions to set up and run the experiments is publicly available on Zenodo [3].

Acknowledgments

This work was partially supported by the Helmholtz Association (HGF) with the KiKIT project and the Grant 46.23 (Engineering Secure Systems) and by the German Research Foundation (DFG) - SFB 1608 - 501798263.

References

[1] Rajeev Alur, Costas Courcoubetis, Nicolas Halbwachs, Thomas A Henzinger, Pei-Hsin Ho, Xavier Nicollin, Alfredo Olivero, Joseph Sifakis, and Sergio Yovine. 1995. The algorithmic analysis of hybrid systems. *TCS* 138, 1 (1995), 3–34.

[2] Andrea Arcuri and Lionel Briand. 2011. A Practical Guide for Using Statistical Tests to Assess Randomized Algorithms in Software Engineering. In *Proceedings of the 33rd International Conference on Software Engineering* (Waikiki, Honolulu, HI, USA) *(ICSE '11)*. Association for Computing Machinery, New York, NY, USA, 1–10. https://_doi.org/10.1145/1985793.1985795

[3] Anonymous Author(s). 2025. Replication Package: Verify, Augment, Improve: Self-Adaptation Repair via Automated Knowledge Augmentation from Mistakes. https://doi.org/10.5281/zenodo.17459651. [Online; accessed Oct-2025].

[4] Nelly Bencomo, Kristopher Welsh, Pete Sawyer, and Jon Whittle. 2012. Self-Explanation in Adaptive Systems. In *IEEE Intl. Conf. on Engineering of Complex Computer Systems*. IEEE Computer Society, 157–166.

[5] Marcello M Bersani, Matteo Camilli, Livia Lestingi, Raffaela Mirandola, and Matteo Rossi. 2023. Explainable Human-Machine Teaming using Model Checking and Interpretable Machine Learning. In *Intl. Conf. on Formal Methods in Software Engineering*. IEEE, 18–28.

[6] Matteo Camilli and Raffaela Mirandola. 2025. Parametric Falsification of Many Probabilistic Requirements Under Flakiness . In *2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE)*. IEEE Computer Society, Los Alamitos, CA, USA, 178–190. doi:10.1109/ICSE55347.2025.00237

[7] Matteo Camilli, Raffaela Mirandola, and Patrizia Scandurra. 2023. Enforcing Resilience in Cyber-physical Systems via Equilibrium Verification at Runtime. *ACM Trans. Auton. Adapt. Syst.* 18, 3, Article 12 (Sept. 2023), 32 pages. doi:10.1145/3584364

[8] Matteo Camilli, Raffaela Mirandola, and Patrizia Scandurra. 2023. XSA: EXplainable Self-Adaptation. In *Intl. Conf. on Automated Software Engineering (ASE'22)*. ACM, Article 189, 5 pages.

[9] Matteo Camilli, Raffaela Mirandola, and Patrizia Scandurra. 2025. Many-objective Self-adaptation under Model Uncertainty. *ACM Trans. Auton. Adapt. Syst.* 20, 2, Article 12 (June 2025), 28 pages. doi:10.1145/3719349

[10] Zhengyin Chen, Jialong Li, Nianyu Li, and Wenpin Jiao. 2024. Reliable proactive adaptation via prediction fusion and extended stochastic model predictive control. *J. Syst. Softw.* 217, C (Nov. 2024), 14 pages. doi:10.1016/j.jss.2024.112166

[11] Mohan Baruwal Chhetri, Anton V. Uzunov, Bao Quoc Vo, Surya Nepal, and Ryszard Kowalczyk. 2019. Self-Improving Autonomic Systems for Antifragile Cyber Defence: Challenges and Opportunities. In *2019 IEEE International Conference on Autonomic Computing, ICAC 2019, Umeå, Sweden, June 16-20, 2019*. IEEE, 18–23. doi:10.1109/ICAC.2019.00013

[12] Alexandre David, Kim G Larsen, Axel Legay, Marius Mikučionis, and Danny Bøgsted Poulsen. 2015. Uppaal SMC tutorial. *STTT* 17, 4 (2015), 397–415.

[13] Alexandre David, Kim G Larsen, Axel Legay, Marius Mikučionis, Danny Bøgsted Poulsen, Jonas Van Vliet, and Zheng Wang. 2011. Statistical model checking for networks of priced timed automata. In *FORMATS*. Springer, 80–96.

[14] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6, 2 (2002), 182–197. doi:10.1109/4235.996017

[15] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. 2017. CARLA: An Open Urban Driving Simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*. 1–16.

[16] Omid Gheibi and Danny Weyns. 2024. Dealing with Drift of Adaptation Spaces in Learning-based Self-Adaptive Systems Using Lifelong Self-Adaptation. *ACM Trans. Auton. Adapt. Syst.* 19, 1 (2024), 5:1–5:57. doi:10.1145/3636428

[17] Vincenzo Grassi, Raffaela Mirandola, and Diego Perez-Palacin. 2024. A conceptual and architectural characterization of antifragile systems. *J. Syst. Softw.* 213 (2024), 112051. doi:10.1016/J.JSS.2024.112051

[18] Verena Klös, Thomas Göthel, and Sabine Glesner. 2018. Comprehensible and dependable self-learning self-adaptive systems. *Journal of Systems Architecture* 85-86 (2018), 28–42. doi:10.1016/j.sysarc.2018.03.004

[19] Börge Kordts, Jan Patrick Kopetz, and Andreas Schrader. 2021. A Framework for Self-Explaining Systems in the Context of Intensive Care. In *ACSOS*. IEEE, 138–144.

[20] Sotiris B Kotsiantis, Ioannis Zaharakis, P Pintelas, et al. 2007. Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering* 160, 1 (2007), 3–24.

[21] Werner Kritzinger, Matthias Karner, Georg Traar, Jan Henjes, and Wilfried Sihn. 2018. Digital Twin in manufacturing: A categorical literature review and classification. *IFAC-PapersOnLine* 51, 11 (2018), 1016–1022. doi:10.1016/j.ifacol.2018.08.474 16th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2018.

[22] Livia Lestingi, Marcello M. Bersani, Matteo Camilli, Raffaela Mirandola, Matteo Rossi, and Patrizia Scandurra. 2026. Proactive self-adaptation and assurance of explainable Human–Machine Teaming. *Journal of Systems and Software* 232 (2026), 112657. doi:10.1016/j.jss.2025.112657

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

[24] Henry B Mann and Donald R Whitney. 1947. On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics* (1947), 50–60.

[25] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. 2012. *Foundations of Machine Learning*. The MIT Press.

[26] Christoph Molnar. 2022. *Interpretable Machine Learning* (2 ed.). https://christophm.github.io/interpretable-ml-book

[27] Francesco Renato Negri, Niccolo Nicolosi, Matteo Camilli, and Raffaela Mirandola. 2024. Explanation-driven Self-adaptation using Model-agnostic Interpretable Machine Learning. In *Proceedings of the 19th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2024, Lisbon, Portugal, April 15-16, 2024*, Luciano Baresi, Xiaoxing Ma, and Liliana Pasquale (Eds.). ACM, 189–199. doi:10.1145/3643915.3644085

[28] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*. 1135–1144.

[29] Eric Rohmer, Surya P. N. Singh, and Marc Freese. 2013. V-REP: A versatile and scalable robot simulation framework. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 1321–1326. doi:10.1109/IROS.2013.6696520

[30] Felix Maximilian Roth, Christian Krupitzer, and Christian Becker. 2015. Runtime Evolution of the Adaptation Logic in Self-Adaptive Systems. In *2015 IEEE International Conference on Autonomic Computing*. 141–142. doi:10.1109/ICAC.2015.20

[31] Vincenzo Scotti, Diego Perez-Palacin, Valerio Brauzi, Vincenzo Grassi, and Raffaela Mirandola. 2025. Antifragility via Online Learning and Monitoring: an IoT Case Study. doi:10.5445/IR/1000182721

[32] Nassim Nicholas Taleb. 2012. *Antifragile: Things That Gain from Disorder*. Random House, New York.

[33] András Vargha and Harold D. Delaney. 2000. A Critique and Improvement of the "CL" Common Language Effect Size Statistics of McGraw and Wong. *Journal of Educational and Behavioral Statistics* 25, 2 (2000), 101–132. http://www.jstor.org/stable/1165329

[34] Zaitian Wang, Pengfei Wang, Kunpeng Liu, Pengyang Wang, Yanjie Fu, Chang-Tien Lu, Charu C. Aggarwal, Jian Pei, and Yuanchun Zhou. 2025. A Comprehensive Survey on Data Augmentation. arXiv:2405.09591 [cs.LG] https://arxiv.org/abs/2405.09591

[35] Danny Weyns. 2020. *An Introduction to Self-adaptive Systems: A Contemporary Software Engineering Perspective*. John Wiley.

[36] Danny Weyns, Thomas Bäck, René Vidal, Xin Yao, and Ahmed Nabil Belbachir. 2022. The vision of self-evolving computing systems. *J. Integr. Des. Process. Sci.* 26, 3-4 (2022), 351–367. doi:10.3233/JID-220003

[37] Danny Weyns, Radu Calinescu, Raffaela Mirandola, Kenji Tei, and et al. 2023. Towards a Research Agenda for Understanding and ManagingUncertainty in Self-Adaptive Systems. *ACM SIGSOFT Softw. Eng. Notes* 48, 4 (2023), 20–36. doi:10.1145/3617946.3617951

[38] D. Randall Wilson and Tony R. Martinez. 1997. Improved heterogeneous distance functions. *J. Artif. Int. Res.* 6, 1 (Jan. 1997), 1–34.

[39] Yulong Ye, Tao Chen, and Miqing Li. 2025. Distilled Lifelong Self-Adaptation for Configurable Systems. arXiv:2501.00840 [cs.SE] https://arxiv.org/abs/2501.00840

[40] Yulong Ye, Tao Chen, and Miqing Li. 2025. Distilled Lifelong Self-Adaptation for Configurable Systems. *CoRR* abs/2501.00840 (2025). arXiv:2501.00840 doi:10.48550/ARXIV.2501.00840

[41] Edith Zavala, Xavier Franch, Jordi Marco, and Christian Berger. 2020. HAFLoop: An Architecture for Supporting Highly Adaptive Feedback Loops in Self-Adaptive Systems. *Future Gener. Comput. Syst.* 105, C (apr 2020), 607–630. doi:10.1016/j.future.2019.12.026

[42] Man Zhang, Bran Selic, Shaukat Ali, Tao Yue, Oscar Okariz, and Roland Norgren. 2016. Understanding Uncertainty in Cyber-Physical Systems: A Conceptual Model.