Detecting Dependability Failures in Healthcare Scenarios via Digital Shadows

Bruno Guindani*, Matteo Camilli*, Livia Lestingi*, Marcello Maria Bersani*
*Dipartimento di Elettronica, Informazione e Bioingegneria
Politecnico di Milano, Via Ponzio 34/5, 20133 Milan, Italy
Email: name.surname@polimi.it

***Abstract*—In healthcare systems, practitioners are responsible for making decisions when a patient's health, or even life, are at stake. Real-time data-driven modeling, analysis, and prediction approaches, such as the Digital Shadow (DS) paradigm, inform and support decision-makers in such critical situations. We introduce GENGAR, a DS-based methodology to identify critical scenarios in a patient-device-physician (PDP) triad with human agents and cyber-physical devices interacting under uncertainty. The proposed solution relies on automata-based modeling and formal analysis techniques to predict and inform the practitioner of critical contingencies that may compromise patient safety, enhancing the system's dependability. In particular, it leverages automata learning to infer and evolve a realistic patient model from clinical logs. GENGAR then exploits mutational and search-based fuzzing to generate scenarios and detect failure cases, i.e., those violating predefined dependability requirements. Failure scenarios are then filtered using qualitative criteria on clinical plausibility, yielding up to 60% realistic cases.**

***Index Terms*—Human Digital Shadow, Statistical Model Checking, Cyber-Physical Systems, Automata Learning**

I. Introduction

Hospitalized patient care often involves interactions with medical devices monitored by healthcare personnel. This context highlights the existence of a category of Cyber-Physical Systems (CPSs), known as Medical CPSs, requiring operational security standards and specialized design [18]. The coexistence of human agents and cyber-physical devices and the interdependence of their actions makes the patient-device-physician (PDP) triad a critical system. Complexity stems from internal components and the uncertainty introduced by human actions in response to system or environmental events, which are influenced by individual expertise and judgment.

Understanding the role of human activity and identifying scenarios that could compromise patient health are essential for raising awareness among medical personnel, designing safer systems, and mitigating error-prone conditions. One effective method to achieve this goal involves examining significant scenarios to identify PDP failures, their underlying causes, and the contextual factors involved. A report by the US Institute of Medicine [35], outlining guiding principles for the design of safe health care systems, asserts that a safe system should "adopt a proactive approach that allows for examining processes of care for threats to safety" and highlights the need for healthcare organizations to implement mechanisms that facilitate "learning from errors". Clinical Decision Support System software [39] provides clinicians with knowledge and patient-specific information to enhance care at various stages of the clinical process, to reduce errors and adverse events.

This work presents a methodology called GENGAR[1], centered on the Digital Shadow (DS) paradigm [1] and aimed at identifying critical contingencies in PDP triads and informing medical personnel about the underlying contributing factors. The DS architectural paradigm underpins the development and maintenance of complex systems requiring accurate and timely monitoring and control actions. A DS is an infrastructure enabling a one-way digital representation of a physical system. Data is collected from the real world and transferred to the digital model, which has no feedback or direct influence on the physical system. The DS may include tools for retrospective analysis, diagnostics, and monitoring. To the best of our knowledge, DS of the PDP triad was first proposed by Bersani et al. [7], [8], whose contribution remains visionary and lacks implementation. In this respect, the following challenges emerge: *i)* developing an accurate DS of a PDP triad accounting for sources of uncertainty; *ii)* expressing *dependability requirements* for the PDP triad (e.g., reachability of patient stability); *iii)* pre-emptively identifying scenarios in which the PDP triad under analysis violates such requirements to inform medical personnel on potential deterioration contingencies. GENGAR addresses the first challenge by exploiting the Stochastic Hybrid Automata (SHA) formalism [16]. The model must capture the basic functionality of medical equipment, particularly related to the alarms revealing critical values of certain clinical metrics. The model of the physician interacting with the medical device incorporates domain knowledge represented by routine medical procedures and a *stochastic* characterization of the uncertainty inherent to their actions (e.g., the presence of decision alternatives and delays in implementing procedures).

Given the cross-cutting nature of the target DS, the GENGAR modeling phase partially relies on domain knowledge derived from expert interviews. For the elements of the DS that are inherently subject to uncertainty (e.g., unexpected variations of the patient's health condition), the methodology employs a data-driven automata learning technique called L^*_{SHA} [27], [28] to obtain the patient model from a finite set of real logs (related to a set of known physiological metrics). The so-obtained model represents a family of PDP triads, while a

[1]Guided Exploration of iNterfaces and Guidelines Adhering to Requirements.

specific clinical scenario corresponds to a concrete instance of the model combined with the representation of a particular disease onset (e.g., an asthma attack).

The SHA formalism is amenable to the rigorous specification of dependability requirements, which can be verified for a given PDP instance using Statistical Model Checking (SMC) [26]. The methodology then explores the model space, including possible behavioral variations, using fuzzing techniques [31]. Starting from a reference model built from domain knowledge, several variants (or mutants) are automatically generated to find instances that violate the requirements. These instances encode critical scenarios and therefore constitute contingencies that medical staff must anticipate and prevent to maintain the required dependability properties.

The methodology is validated on a selected PDP exemplar featuring a *patient* suffering from lung disease, connected to a pulmonary ventilator (the *device*) and under the care of an intensivist (the *physician*). Medical knowledge, obtained through interviews with a domain expert, guides the construction of the physician-device interaction model and the elicitation of dependability requirements, while data for training the model are generated by a simulator of the patient's physiology. The performed experimental campaign exploits 10 hours of cumulative acquisition time (accounting for both interviews and simulations). The failures identified by GENGAR are qualitatively evaluated by a domain expert to filter out unrealistic situations, resulting in up to 60% expert-validated failures.

Our contributions can be summarized as follows:

- GENGAR, a novel methodology that integrates automata-based modeling of medical procedures, automata learning, and model space exploration to uncover dependability failures in Intensive Care Unit (ICU) scenarios;
- an empirical evaluation of the cost-effectiveness of GENGAR considering a realistic PDP evaluation subject;
- a publicly available replication package, including sources and instructions to replicate our experiments.

The paper is structured as follows: Section II outlines preliminary concepts underlying the work; Section III presents the PDP exemplar serving as running example; Section IV introduces the developed methodology; Section V presents experimental results; Section VI surveys related work; and Section VII concludes.

II. Preliminaries

A. Stochastic Hybrid Automata

Hybrid Automata (HA) extend finite-state automata by adding real-valued variables that evolve continuously over time according to flow conditions defined by Ordinary Differential Equations (ODEs) [2]. This expressiveness enables the modeling of systems exhibiting complex, non-linear dynamics. An HA is a tuple $\mathcal{H} = (\Sigma, L, X, F, E, I, P)$ where: *i)* Σ is a set of actions, partitioned into inputs (Σ_i) and outputs (Σ_o); *ii)* L is a set of locations, each representing a distinct operational mode of an agent or a physical phenomenon; *iii)* X is a set of real-valued variables governing system dynamics and $G(X)$ is the set of constraints on variables of X (e.g., $x < 10$); *iv)* $F = \{f_\ell : \ell \in L\}$ defines flow conditions, where each f_ℓ describes the time evolution of variables of X in location ℓ via a system of ODEs $\dot{X} = f_\ell(X)$; *v)* $E \subseteq L \times G(X) \times \Sigma \times 2^X \times L$ is a finite set of edges between two locations of L, including a constraint from $G(X)$ called *guard*, an event in Σ, and an assignment to variables of X (e.g., $x = 0$) called *resets*; *vi)* $I : L \to G(X)$ assigns *invariants*, i.e., constraints on X, to locations; *vii)* $P : L \to \mathbb{R}^+$ assigns exit rates to locations. In HA, a state is characterized by a tuple $(\ell, u) \in L \times \mathbb{R}^X$ and can evolve through two types of transitions. *Discrete transitions* induce location changes through the execution of an enabled edge of E and possibly the reset of a subset of variables of X (an outgoing edge from ℓ is enabled when its guard is satisfied by values $u \in \mathbb{R}^X$). *Temporal transitions* solely update variable values as time elapses according to the laws in F. A *run* of an HA is a sequence of transitions that represents a possible evolution of a physical phenomenon over time, driven by the execution of actions associated with discrete transitions and time elapsing. The sequence of actions corresponding to the discrete transitions in a run is a *trace*.

Stochastic Hybrid Automata (SHA) extend HA with stochastic features [16]. SHA incorporate stochastic processes, allowing representation of uncertainties in physical phenomena. If parameter θ in a flow condition is treated as a random variable, such that $\dot{X} = f(X, \theta)$, with $\theta \sim \mathcal{D}$ and $\mathcal{D}$ is a probability distribution, then X follows a stochastic process.

A SHA network is composed of multiple SHA that synchronize over actions of Σ, ensuring that all participating automata execute a transition simultaneously when an event occurs. Specifically, an SHA can trigger synchronization by taking an edge with label $\sigma \in \Sigma_\mathrm{o}$, while others such that $\sigma \in \Sigma_\mathrm{i}$ react in the same time instant. In this work, PDP triads are modeled using networks of SHA. Specifically, the model consists of two SHA: one representing the physician-device interactions, derived from domain knowledge, and the other capturing patient dynamics, learned from data.

B. $\mathrm{L}^*_{\mathrm{SHA}}$ *for SHA learning*

Automata learning was developed within the formal languages area to infer the minimal representation of a regular language. L^*, a well-established learning algorithm, targets Deterministic Finite-state Automata [3]. In L^*, learning occurs in rounds through the interaction between a *learner* and a *teacher*. The learner maintains the hypothesis automaton and refines it by submitting queries to the teacher. The teacher (i.e., the omniscient oracle) answers two types of queries: *membership* queries (whether a sequence of events complies with the set of rules under learning known by the teacher); *equivalence* queries (whether there is a counterexample to the hypothesis proposed by the learner). If no counterexample exists, the hypothesis is the minimal correct representation of the regular language and the learning terminates.

In $\mathrm{L}^*_{\mathrm{SHA}}$(the extension of L^* targeting SHA [27], [28]) the teacher is not omniscient about the behavior of the system under learning but relies on data collected on the field. Therefore, counterexamples are limited to the available dataset. A *signal* for a measurable (physical or logical) quantity is a sequence of pairs including timestamps and values $(t_0, s_0), (t_1, s_1), \ldots,$ $t_i, s_i \in \mathbb{R}$, such that $0 \leq t_i \leq t_{i+1}$ for all the adjacent pairs of positions in the sequence. Besides the training dataset, $\mathrm{L}^*_{\mathrm{SHA}}$ requires as input: *i*) set Σ of the actions corresponding to real-world events, whose correlation with changes in the dynamics of variables in X_{F} is to be learned from data; *ii*) set X_{F} of real-valued variables whose time dynamics are to be inferred and constrained by flow conditions; and *iii*) set X_{M} of real-valued variables used to mine the events from collected data. Each variable in X_{F} and X_{M} is associated with a signal in the dataset. A user-defined *labeling* partial function $\mathcal{L}_{X_{\mathrm{M}}}$ enables mapping signals to traces. Specifically, $\mathcal{L}_{X_{\mathrm{M}}}(t) = \sigma \in \Sigma$ is an action representing the event that occurred at t, where timestamp t is included in signals associated with variables in X_{M}. Terms *event* and *action* are used interchangeably.

With reference to the SHA tuple, $\mathrm{L}^*_{\mathrm{SHA}}$ learns: locations L; edges E; flow conditions F describing the dynamics of the variables in X_{F}; and distributions $\mathcal{D}$ for the ODEs' random parameters. Besides being amenable to formal verification, the learned SHA serves as a predictor for variables in X_{F}, as a result of sequences of events in Σ.

C. Fuzzing

Fuzzing [31] is an automated testing technique that systematically generates inputs. Rather than relying on handcrafted test cases, fuzzing aims to discover defects by exercising a wide range of program behaviors, often guided by runtime feedback (e.g., code coverage). Inputs are typically generated from scratch or derived from existing examples to expose unexpected behavior, such as assertion failures. Mutational fuzzing [11] operates by applying transformations to a set of valid inputs (seeds) to produce new test cases. This approach relies on the assumption that small changes to valid inputs can trigger new program behavior without entirely compromising validity. Feedback mechanisms, such as tracking code coverage or estimated failure likelihood, help prioritize mutations, enabling the fuzzer to incrementally explore unseen behaviors. Search-based fuzzing (or testing) [10] extends mutational fuzzing by incorporating optimization strategies that guide input generation toward specific goals. These strategies often treat test case generation as an optimization problem that evaluates candidate inputs based on a fitness function, typically related to execution depth, branch coverage, or proximity to requirements violation. Techniques such as evolution strategy or genetic algorithms [17] allow the fuzzer to focus resources on input regions that are more likely to uncover failures.

III. Running Example: Respiratory Intensive Care

Our scenarios of interest take place in ICU hospital departments. The selected triad (outlined in Fig. 1) features a patient with lung disease admitted to the ICU, the medical personnel responsible for their care (e.g., an intensivist), and the mechanical ventilator (the device) operated by the staff to provide life support. The intensivist can change four parameters impacting the ventilator's operations: the fraction of inspired oxygen ($\mathsf{FIOX} \in [0, 1]$), positive end expiratory pressure ($\mathsf{PEEP} \in [5, 25]$ cmH2O), the forced respiration rate ($\mathsf{RERA} \in [6, 18]\mathrm{min}^{-1}$), and the forced tidal volume ($\mathsf{TVOL} \in [300, 500]$ mL[2]) [9]. Activating or deactivating the ventilator and increasing or decreasing its parameters impacts the patient's respiratory conditions [36]. The intensivist decides to take any of these actions by monitoring the patient's relevant vital signs: carbon dioxide CD, heart rate HR, oxygen saturation OS, respiration rate RR, and tidal volume TV. Specifically, when they detect a change in the vital signs that requires medical attention, they apply a specific mitigation action by manipulating ventilator settings, adhering to standard medical procedures [36]. According to Fig. 1, the intensivist has complete control over the ventilator but cannot directly observe events impacting the patient's health. For example, they may infer that an asthma attack is taking place by observing specific changes in the patient's vital signs.

This PDP triad constitutes a complex CPS with human agents interacting with cyber-physical devices, where the former exercise judgment and free will in operating the latter. The exemplar features inherent sources of uncertainty in these interactions due to the unpredictability of human behavior (specifically, the physician's actions) and events that can occur in the environment impacting the patient's vital signs (e.g., asthma attacks). These challenges underscore the need for DS paradigms that enable accurate modeling of all agents in the PDP triad. Furthermore, detecting and analysing critical scenarios that capture a deterioration in the practitioner's effectiveness through the DS can support medical staff in preventing health-threatening contingencies.

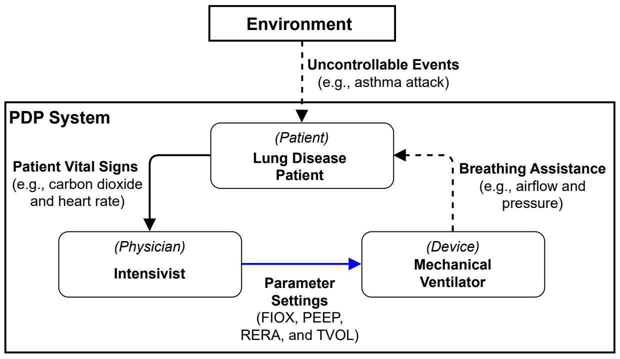


Fig. 1: PDP triad used as a running example showing interactions between the agents and the environment: an arrow from agent a to b represents an event that a triggers directed to b. Controllable events are in blue, uncontrollable but observable events are in solid black, while uncontrollable and not directly observable events are represented by dashed arrows.

[2]TVOL values depend on the patient's physiological characteristics like body mass [36]. The listed range reflects the values used in our experiments.

IV. Methodology

GENGAR is a semi-automated methodology to build DSs of PDP triads through domain knowledge and data collected on the field, and analyze them against dependability requirements. In particular, it can be exploited to identify potentially critical scenarios (the *failures*) representing a deterioration of the physician's effectiveness that can jeopardize the patient's health status. GENGAR envisages two phases, as outlined in Fig. 2, and requires the cooperation of actors with a medical background (i.e., the *domain experts*) and technical expertise (i.e., *software engineers*). The identified set of failures is then returned to the *stakeholders* of the PDP system. Stakeholders include professionals involved in both the clinical and organizational aspects of care delivery, such as clinicians and department directors, as well as those responsible for training medical personnel or monitoring their performance [19].

A. Model Definition

The model definition phase takes as input both field-collected data and domain knowledge, producing a parametric SHA that accurately reflects the time-dependent behavior of a family of PDP triads. This phase comprises two main activities: Design and Learning. The Design activity constructs models for the physician and the device based on domain expertise, while the Learning activity derives the patient model from empirical data. The Data Model Definition and Data Pre-processing activities must be completed beforehand to feed well-formed inputs to the learning. Finally, Model Post-processing refines the generated models to ensure compatibility with the verification procedures of the subsequent phase.

a) Data Model Definition: The objective of this activity is to define the inputs required for the learning process, which, as described in Section II-A, include the set of events Σ, the sets of physical variables used for event detection (X_{M}) and for modeling flow conditions (X_{F}), and the labeling function $\mathcal{L}_{X_{\mathrm{M}}}$. The selection of events and physical variables must be guided by domain expertise, as well as by the capabilities and operational characteristics of the available medical equipment. Among the relevant variables, some should ideally be continuously monitored and require accurate estimation—these are best suited for modeling flow conditions (X_{F}). Others are significant especially when they are representative of particular conditions of the patient or the environment in general, such as, for example, the exceeding of critical thresholds, which often set off alarms on the equipment—these are used for event identification (X_{M}). The labeling function $\mathcal{L}_{X_{\mathrm{M}}}$ maps specific conditions on the variables in X_{M} to the events in Σ, thereby formally defining the conditions under which real-world events can be identified from the sampled data. As in other aspects of the modeling process, domain knowledge plays a critical role in identifying the relevant events and establishing their associated variables and threshold-based activation criteria.

Example IV.1. In our running example (see Section III), events either capture a noticeable variation in the patient's vital sign or the intensivist changing ventilator settings. Therefore, these quantities all populate set $X_{\mathrm{M}} = V \cup P$, where $V = \{\mathsf{CD}, \mathsf{HR}, \mathsf{OS}, \mathsf{RR}, \mathsf{TV}\}$ contains the patient's vitals and $P = \{\mathsf{FIOX}, \mathsf{PEEP}, \mathsf{RERA}, \mathsf{TVOL}\}$ contains the ventilator's parameters. Each patient vital $v_i, i = 1, \ldots, 5$ is associated with a safe operating range $R_i = [v_i^{\min}, v_i^{\max}]$. An event occurs when the value of vital i at time t_k (noted as $v_i^{t_k}$) exceeds its safe range, signaling a deviation from normal physiological conditions, or returns within such range, signaling a potential recovery or stabilization. The ventilator parameters $p_j, j = 1, \ldots, 4$ trigger events when they are manually increased or decreased by the intensivist controlling them. Finally, events on and off correspond to the activation and deactivation of the mechanical ventilation device, respectively. While the device is inactive, all ventilator parameters are assumed to be set to zero. We define the partial labeling function $\mathcal{L}_{X_{\mathrm{M}}}$ as follows:

$$\mathcal{L}_{X_{\mathrm{M}}}(t_k) = \begin{cases} (\mathsf{V}_i)^{\mathsf{high}} & \text{if } v_i^{t_k} > v_i^{\max} \wedge v_i^{t_{k-1}} \leq v_i^{\max} \\ (\mathsf{V}_i)^{\mathsf{low}} & \text{if } v_i^{t_k} < v_i^{\min} \wedge v_i^{t_{k-1}} \geq v_i^{\min} \\ (\mathsf{V}_i)^{\mathsf{ok}} & \text{if } v_i^{t_k} \in R_i \wedge v_i^{t_{k-1}} \notin R_i \\ (\mathsf{P}_j)^{\mathsf{up}} & \text{if } p_j^{t_k} > p_j^{t_{k-1}} \wedge p_j^{t_k}, p_j^{t_{k-1}} \neq 0 \\ (\mathsf{P}_j)^{\mathsf{down}} & \text{if } p_j^{t_k} < p_j^{t_{k-1}} \wedge p_j^{t_k}, p_j^{t_{k-1}} \neq 0 \\ \mathsf{on} & \text{if } p_j^{t_k} > 0 \wedge p_j^{t_{k-1}} = 0 \ \forall j \\ \mathsf{off} & \text{if } p_j^{t_k} = 0 \wedge p_j^{t_{k-1}} > 0 \ \forall j \end{cases}$$

$$\text{for } \mathsf{V}_i \in V, \quad \mathsf{P}_j \in P, \quad k = 1, 2, \ldots$$

We choose $X_{\mathrm{F}} = \{\mathsf{TV}\}$ as the variable modeled through flow conditions, as even slight changes in its value are critical for determining the patient's ability to breathe.

b) Data Pre-processing: Pre-processing is often necessary to transform raw inputs into time series suitable for the learning algorithm. This step includes smoothing noisy signals, resampling data to obtain uniform time intervals, or extracting key features such as peak values or rate of change to capture relevant physiological dynamics more effectively.

Example IV.2. We apply smoothing to the CD (carbon dioxide) signal, which shows a sinusoidal trend. Specifically, we replace the raw signal with a time series that captures the most recent peak value at each time step. This transformation results in a smoother representation that aligns with our threshold-based event definitions more effectively.

c) Design: The components of the PDP system derived from the Design activity formalize established clinical practices for managing a patient's evolving health conditions, encoding medical expertise effectively. This activity produces models for both the physician M_{Ph} and the ventilation device M_{D}. Domain experts specify the set of interventions they perform, the events that trigger them, and the contextual conditions required for their execution. They also define the core activities involved in patient care. These elements are then translated into an SHA, where interventions correspond to actions on transitions and activities to locations, within the models of the physician and device. Transition guards define the contextual conditions for executing an action. When multiple actions are possible in response to a single event, M_{Ph} incorporates probabilistic transitions governed by predefined

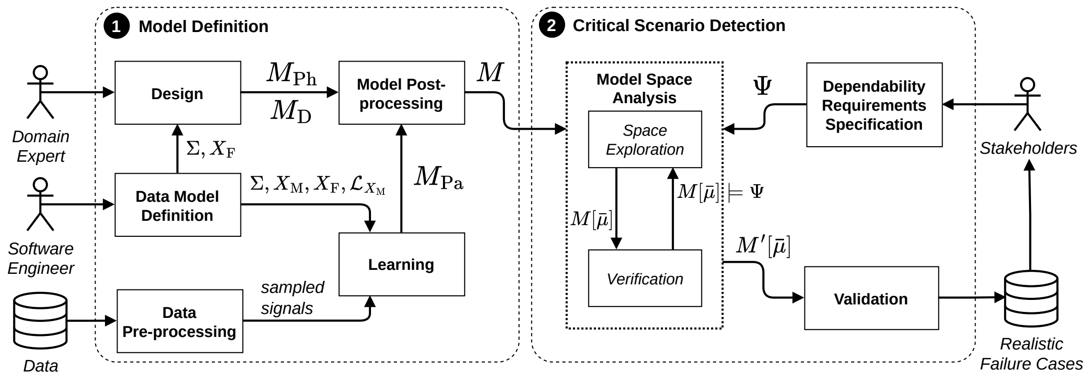


Fig. 2: Overview of proposed methodology, highlighting the two *phases* (dashed boxes), the data flow between phase *activities* (solid boxes), and the actors involved in activities requiring human input.

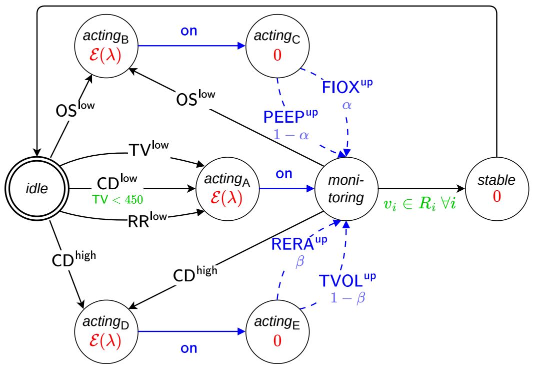

acting$_{\mathrm{B}}$ $\mathcal{E}(\lambda)$
on
acting$_{\mathrm{C}}$ 0
FIOX$^{\mathrm{up}}$ α
PEEP$^{\mathrm{up}}$ $1-\alpha$
OS$^{\mathrm{low}}$
OS$^{\mathrm{low}}$
TV$^{\mathrm{low}}$
CD$^{\mathrm{low}}$
TV < 450
RR$^{\mathrm{low}}$
idle
acting$_{\mathrm{A}}$ $\mathcal{E}(\lambda)$
on
moni-toring
$v_i \in R_i \, \forall i$
stable 0
CD$^{\mathrm{high}}$
CD$^{\mathrm{high}}$
RERA$^{\mathrm{up}}$ β
TVOL$^{\mathrm{up}}$ $1-\beta$
acting$_{\mathrm{D}}$ $\mathcal{E}(\lambda)$
on
acting$_{\mathrm{E}}$ 0

Fig. 3: Example of physician-device SHA.

probability weights. This enables the physician-device model to capture the inherent variability in clinical decision-making.

SHA are parameterized to reflect the degrees of freedom involved in the intensivist's clinical decision-making. The software engineer, assisted by the domain expert, may define a set of model parameters that govern key behavioral aspects of the triad, to encapsulate elements of human judgment and free will. Specific parameter instantiations may result in adverse or unsafe scenarios for the patient; therefore, they are fundamental to the Critical Scenario Detection phase.

Example IV.3. The SHA in Fig. 3, derived from domain expertise through expert elicitation, models the behavior of an intensivist who, upon observing specific clinical events, responds by activating the ventilator and adjusting its settings accordingly. Blue arrows and events represent physician-controlled transitions, with stochastic actions highlighted by dashed lines and probability weights. Other events have black labels and arrows. Guard conditions on transitions are in green and the distribution of leaving times from locations are in red.

As an example, the physician activates the ventilator (action on) when a decrease is detected in any of the vital signs TV, CD, or RR, provided the ventilator is not already active. Similarly, in response to either low oxygen saturation (OS) or elevated carbon dioxide levels (CD), the physician activates the ventilator and adjusts one of its parameters. In the case of low OS, the physician probabilistically increases either FIOX or PEEP, with weights $(\alpha, 1-\alpha)$; for high CD, they choose between increasing RERA or TVOL, with probabilities $(\beta, 1-\beta)$. The transition labeled with CD$^{\mathrm{low}}$ is constrained by guard TV < 450 to model the fact that the physician responds to a CD$^{\mathrm{low}}$ alarm only when the patient's tidal volume is lower than 450 mL. When all vital signs v_i return to their respective safe ranges R_i, the patient is considered stabilized, prompting a transition to the *stable* state, after which the physician returns to the initial *idle* state.

The SHA captures that the physician detects event alarms with a delay, reflecting realistic scenarios in which the physician is not immediately available to respond to the emergency. Delays are modeled as an exponentially distributed random variable with rate parameter λ. Other actions are executed instantaneously, i.e., with zero delay. Certain locations in the SHA, such as *idle* and *monitoring*, represent observation states. As these locations lack physician-controlled outgoing transitions, any exit from them depends entirely on external events generated by the patient model; therefore, no delay distribution is associated with them. In this example, model parameters are collectively denoted as $\mu = (\alpha, \beta, \lambda)$, where $\alpha \in [0,1]$, $\beta \in [0,1]$, and $\lambda \in [1/15, 1]$ seconds [3].

d) Learning: Patient health conditions in ICUs are inherently complex, particularly because respiratory diseases cannot be directly observed. Instead, intensivists must rely on indirect indicators, i.e., quantitative metrics obtained through monitoring equipment, which motivates the use of learning algorithms to construct the patient's model. The Learning activity builds on the components defined during model definition: event specifications, physical variables, and the labeling function. It also incorporates raw signals collected from the ICU, which may originate from various sources, including medical devices, hospital information systems, and manual entries by healthcare

[3] α and β are probabilities, while the range for λ reflects realistic average response time in ICU, as informed by domain expertise.

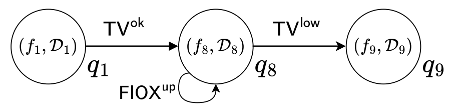

$(f_1, \mathcal{D}_1)$
q_1
$\mathsf{TV}^{\mathsf{ok}}$
$(f_8, \mathcal{D}_8)$
q_8
$\mathsf{FIOX}^{\mathsf{up}}$
$\mathsf{TV}^{\mathsf{low}}$
$(f_9, \mathcal{D}_9)$
q_9

Fig. 4: Detail of learned patient SHA.

professionals. The output of this activity is the patient model M_{Pa}, the third element in the PDP triad.

GENGAR adopts the $\mathrm{L}^*_{\mathrm{SHA}}$ algorithm to derive an SHA-based model from data. This algorithm infers both the locations and transitions of the patient SHA, assigning to each location a flow condition for physical variables in set X_{F}, along with a parametric distribution. These elements characterize each location as a distinct physiological state of the patient, as learned from clinical data. Transitions represent events that drive the patient from one physiological state to another.

Example IV.4. Fig. 4 shows a portion of the patient SHA model learned via $\mathrm{L}^*_{\mathrm{SHA}}$. The flow condition f_i and stochastic parameter distribution $\mathcal{D}_i$ jointly govern the evolution of variable $\mathsf{TV} \in X_{\mathrm{F}}$ while the system resides in location q_i. From location q_1, the occurrence of the $\mathsf{TV}^{\mathsf{ok}}$ event, indicating that tidal volume has returned to safe levels, leads the patient to location q_8. From there, event $\mathsf{TV}^{\mathsf{low}}$ transitions the patient to location q_9. The self-loop labeled by $\mathsf{FIOX}^{\mathsf{up}}$ indicates that the physician's action of increasing FIOX has no observable effect on the patient's state while in q_8.

e) Post-processing: The final activity of model definition integrates the individual models of the PDP components—M_{Ph} (physician), M_{D} (device), and M_{Pa} (patient)—into a unified model representing the model of the whole triad. This composite model is then refined to ensure compatibility with the requirements of the subsequent verification activity, resulting in model M, the final output of this phase.

Example IV.5. The learned patient model M_{Pa} is inherently incomplete, meaning that not all locations include outgoing transitions for every event in Σ. Unlike manually designed models, which can be constructed to explicitly handle all possible event transitions from each state, a learned model typically captures only the behaviors observed in the training data. Consequently, certain events may be absent from some locations if they were not encountered during data collection.

To ensure the model is well-defined for all events in every state and thus suitable for formal verification, we introduce a sink location representing unmodeled or unsupported behavioral aspects, for which the model lacks sufficient data to make reliable analysis. For every location in M_{Pa} that lacks an outgoing transition for some event $e \in \Sigma$, we add a transition labeled e that leads to the sink location.

B. Critical Scenario Detection

In this second phase, the initial Dependability Requirements Specification activity formalizes the target properties of the SHA, drawing from input from domain stakeholders. These formalized requirements are then passed to the Model Space Analysis activity, which takes as input a parametric SHA and investigates how variations of the model influence the satisfaction of dependability requirements. This activity consists of an iterative process involving Space Exploration techniques and formal Verification to identify critical scenarios, i.e., instances where the model behavior may violate the specified requirements. Identified scenarios are filtered during the Validation activity, which excludes those deemed unrealistic in actual ICU settings, based on plausibility criteria defined by domain stakeholders. The output is a validated set of plausible high-risk scenarios that violate the defined requirements.

a) Dependability Requirements Specification: Software engineers collaborate with stakeholders to identify and formalize the properties that the system must satisfy to ensure patient safety and treatment reliability. Through this elicitation process, domain experts contribute expectations and operational constraints, which are then distilled into verifiable properties. The targeted language should enable the specification of properties whose stochastic interpretation over SHA models supports formal analysis of uncertainty, hybrid dynamics, and temporal constraints within PDP triads. As a result, a set Ψ of formal properties is defined, each of which can be evaluated, given a specific instance of the SHA, as either satisfied or violated, with an associated probability. This probabilistic interpretation enables the identification of scenarios in which dependability may degrade under uncertainty, thereby supporting a risk-aware verification process.

Example IV.6. We specify three dependability properties, evaluated over a fixed time horizon:

1) *Reachability of patient stability*: reaching a designated location modeling the stabilization of patient vitals;
2) *Duration in non-breathing state*: TV (tidal volume) remains below a set threshold for a specified duration;
3) *Persistence of critical health condition*: remaining in a state with multiple vital signs outside their safe ranges for a sustained period.

These properties are expressed in Metric Interval Temporal Logic (MITL) [16] formulae for which a probabilistic semantics interpreted over SHA enables reasoning about uncertainties in PDP triads. Accordingly, the probability that the first formula holds in the system is assessed to fall below a stakeholder-defined threshold, while the probabilities that the second and third formulae hold are evaluated to exceed their respective thresholds. We adopt an approach in which properties are not considered violated for traces that reach the sink location, as the latter represents unmodeled behaviors and implies that the system's state is unknown. Different interpretations, such as treating these behaviors as failures, are possible by modifying the definition of MITL properties.

b) Model Space Analysis: As anticipated above, the PDP model is parameterized by a set of variables μ, representing the degrees of freedom in the behavior of both the physician and device models. In this activity, automated techniques systematically explore the PDP model space to identify diverse and behaviorally significant configurations violating the dependability requirements, i.e., critical scenarios. Model space analysis involves an iterative process between a Space Exploration step and a Verification step. The first selects a PDP model $M[\bar{\mu}]$ by applying mutations to a known model, whereas the second evaluates the probability of $M[\bar{\mu}]$ respecting the formulated dependability requirements Ψ. These probabilities are then provided back to the exploration step in a feedback loop. In particular, we evaluate the probability that models satisfy Ψ evaluated using Statistical Model Checking (SMC) [26]. If the computed probability of any undesirable behavior exceeds a stakeholder-defined threshold, the corresponding model yields a *failure*. Throughout the model space exploration activity, failure-inducing PDPs models are collected to identify critical behaviors of the physician-device system that may compromise patient safety. These failures reveal specific combinations of clinical decisions and system responses that warrant further investigation, either as indicators of potential safety risks or as opportunities for system improvement.

GENGAR adopts two alternative exploration strategies: mutational fuzzing and a search-based technique. Both strategies affect model parameters or alter structural elements in different ways, and are guided by the likelihood of a model violating requirements measured during the Verification activity. *Mutational fuzzing* applies domain-specific mutation operators to introduce behavioral variations into the SHA, iteratively creating new *mutants*. These operators are designed to preserve model validity while inducing meaningful variability. In mutational fuzzing, an archive is maintained to store mutants of the physician-device SHA. A new mutant is added to the archive if it exceeds the current maximum likelihood of violation for any defined property. This archive serves as a pool of promising candidates from which new mutants are randomly selected for further mutation. By focusing on high-risk mutants, the algorithm steers exploration toward underexplored areas of the model space, increasing the chance of uncovering critical behaviors. The *search-based* strategy formulates exploration of the model space as a multi-objective optimization problem, where each objective corresponds to the minimization of the probability of satisfying a specific dependability property. We adopt the Non-dominated Sorting Genetic Algorithm II (NSGA-II) [17] to solve the optimization problem. This strategy leverages the same domain-specific mutation operators used in fuzzing, ensuring consistency in how behavioral variations are introduced. These two techniques produce a collection of physician-device mutants, each representing a distinct instantiation of model M analyzed through verification and possibly labeled as a failure.

Example IV.7. In our running example, we apply two mutation operators to generate mutants. The first operator perturbs a randomly selected component of the parameter vector μ by either multiplying or dividing it by a fixed mutation factor, and then adds a small amount of random noise. In the SHA shown in Fig. 3, this mutation alters the intensivist's reaction times or their probabilistic preferences when choosing between actions. The second operator removes a randomly selected transition from the SHA, modeling a situation where the physician no longer responds to certain events. Space exploration using these operators helps identify which physician actions are critical for satisfying requirements and assess the sensitivity of outcomes to changes in reaction time.

We take as example two variations of the physician-device model: one in which the topmost on transition is missing, and another where the exponential rate representing the physician's reaction time is set to $\lambda = 1/15$. The first variation represents a simplified medical procedure in which the intensivist does not take any action upon observing the event $\mathsf{OS}^{\mathsf{low}}$, while the second models a slow-reacting intensivist, potentially due to real-life complications in the hospital department. We compute the probability of requirement satisfaction for both SHA using SMC. According to the results, both models fail to meet one or more of the requirements described in Example IV.6, and are therefore classified as failure-inducing scenarios.

c) Validation: PDP mutants that yield a failure but represent unrealistic or clinically irrelevant scenarios are filtered out. While some of these cases may technically pose risks to patient safety, they are considered either uninteresting for further analysis or unlikely to occur in real-world settings, based on stakeholder judgment. For example, a case where a physician takes an excessively long time to respond to an emergency may be deemed unrealistic, given the high availability of personnel in typical ICU environments. Other scenarios might involve implausible or nonsensical clinical actions that no trained physician would reasonably perform. Because such scenarios cannot always be excluded a priori, they must be explicitly reviewed by clinical experts to assess their validity and relevance.

Example IV.8. We rely on the expertise of a domain specialist to define criteria for classifying identified scenarios as unrealistic. This process involves a guided review of the SHA components shown in Fig. 3, during which the expert identifies transitions considered essential for any clinically reasonable medical procedure. Based on this review, we derive a set of structural and parametric constraints that serve as exclusion criteria. These constraints enable the automated filtering of scenarios that would not plausibly occur in real-world ICU settings. For example, among the two SHA presented in Example IV.7, the first is excluded because it lacks an appropriate physician response to the detection of $\mathsf{OS}^{\mathsf{low}}$, a behavior the domain expert deemed implausible.

The alignment between the digital model (i.e., the PDP triad model) and the physical object (i.e., the physical PDP triad) required by the DS paradigm can be achieved by: *i*) acquiring the patient's vital sign signals in real time; *ii*) applying labeling function $\mathcal{L}_{X_{\mathrm{M}}}$, which produces a trace τ representing the events recorded in the physical environment; and *iii*) simulating τ on the PDP model. The location reached in the SHA by sequentially firing the events from τ represents the DS of the current state of the physical system.

Data collection lies outside the two core phases of GENGAR (see Figure 2), as we assume it is handled by the underlying infrastructure. Our work focuses on the layer that consumes the data produced by the DS infrastructure to learn models and identify critical dependability scenarios.

V. Empirical Evaluation

This section reports on the experimental campaign[4] validating the accuracy and scalability of the presented methodology. We answer the following research questions:

RQ1: How accurate is the patient model resulting from the GENGAR learning phase?

RQ2: What is the cost of the learning phase?

RQ3: How effective is GENGAR in detecting realistic failure scenarios?

RQ4: What is the cost of the failure detection phase?

A. Design of the evaluation

a) Evaluation subjects: Training data was collected through the BREATHE simulator [14], which supports a wide range of scenarios, including adverse events and health complications that affect the patient with varying degrees of severity. It also provides real-time monitoring of the patient's vital signs and interactive control of ventilator parameters. BREATHE is based on Kitware's Pulse [12], an open-source physiology engine that allows ventilator-assisted medical simulation.

Data was collected with the involvement of a physician with four years of experience in intensive care, who executed 8 simulated clinical scenarios using BREATHE, totaling 45 minutes of simulation time. The scenarios included 10 health complications along with corresponding ICU interventions and patient responses. The intensivist was presented with real-time patient vitals and responded according to their clinical expertise, manipulating ventilation settings in line with established medical practices. They also contributed to the definition of the running example described in Sections III and IV, including suggesting relevant physiological signals, describing medical procedures to encode into SHA, and defining validation criteria to distinguish realistic scenarios from unrealistic ones.

b) Methods under comparison: We compare the patient model accuracy against a baseline composed of two mainstream regressors to answer RQ1. The first is a *dual-input* Neural Network (NN) [21]: one branch encodes a simulation's trace of events (i.e., of elements $e \in \Sigma$) as categorical variables using an embedding layer followed by an LSTM layer [24], while the other branch processes the TV values corresponding to the events using a convolutional layer followed by a pooling layer. The outputs of both branches are concatenated and passed through a dense layer with dropout to mitigate overfitting. The final output is a single value representing the predicted TV after the last event in the trace. The second model is an *XGBoost* regressor [13] with a similar two-branch structure: the combined input vector contains both an encoding of the event sequence and the corresponding TV values.

We examine the effectiveness of alternative PDP model space exploration techniques to address RQ3. Specifically, we compare our proposed approaches—*mutational fuzzing* and a *search-based* method leveraging NSGA-II—with a *random search* baseline, which uniformly draws random mutations affecting both parameters and structural elements.

[4]Replication package found at https://doi.org/10.5281/zenodo.15391501.

TABLE I: Results of statistical tests for regression.

	p-val	$\hat{A}_{12}$
$\mathrm{L}^*_{\mathrm{SHA}}$ vs Neural Network	3.97e-01	S
$\mathrm{L}^*_{\mathrm{SHA}}$ vs XGBoost	9.35e-03	L
Neural Network vs XGBoost	2.27e-02	L

c) Statistical tests: To mitigate the risk of obtaining results by chance, we account for randomness by testing 20 different scenarios (RQ1) or by running each exploration strategy 20 times (RQ3) under a fixed computational budget, generating a total of 500 mutants of the patient model. Preliminary evaluations indicate that this budget is sufficient to reach a performance plateau during space exploration. According to the guideline introduced by Arcuri & Briand [4], we apply the non-parametric Mann–Whitney U tests [30] to evaluate the statistical significance of the results. We also compute Vargha and Delaney's $\hat{A}_{12}$ score [40] to quantify the effect size of the difference between two samples, providing a measure of stochastic dominance. We adopt the following standard classification: effect size $\hat{A}_{12}$ $(= 1-\hat{A}_{21})$ is small (S), medium (M), and large (L) when its value is greater than or equal to 0.56, 0.64, and 0.71, respectively.

d) Evaluation testbed: We conducted our experiments on an Ubuntu 24.10 machine with commodity hardware: an 8-core Intel Core i5 processor at 4.2GHz and 16 GB RAM.

B. Evaluation results

RQ1 (model accuracy): The target metric used to evaluate the accuracy of the model learned via $\mathrm{L}^*_{\mathrm{SHA}}$ is the variable governed by flow conditions, namely $\mathsf{TV} \in X_{\mathrm{F}}$. We design a test set comprising 20 scenarios featuring health complications of varying types and severity. Each scenario is modeled using the learned patient SHA, combined with the complete physician-device SHA of Fig. 3 and a sequence of events representing the onset of a specific health complication and randomly selected physician actions. Each scenario is converted into an UPPAAL model [6], from which we extract the predicted TV value after the final event in the trace. We also simulate the same scenario in BREATHE to record the final TV value as the ground truth.

We compare the accuracy of the $\mathrm{L}^*_{\mathrm{SHA}}$ model and the baseline models across two prediction tasks: *regression* and *classification*. For *regression*, we compute each model's test-set Mean Absolute Percentage Error (MAPE) with respect to the ground truth. Fig. 5 displays the distribution of MAPE for each model as a boxplot, while Table I reports the p-value of the Mann-Whitney test and the Vargha-Delaney effect size ($\hat{A}_{12}$). The median MAPEs are 0.184 for $\mathrm{L}^*_{\mathrm{SHA}}$, 0.240 for the dual-input NN, and 0.832 for XGBoost. For the *classification* task, we discretize the regression outputs into three categories $(low, ok, high)$ based on their relation to TV's safe operating range, as defined in Example IV.1. We then evaluate the models' classification accuracy, defined as the proportion of correctly classified samples. The resulting accuracy scores are 90% for $\mathrm{L}^*_{\mathrm{SHA}}$, 40% for the NN, and 30% for XGBoost.

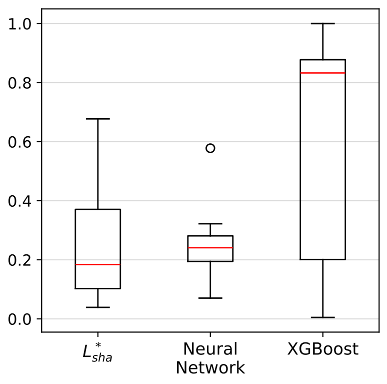
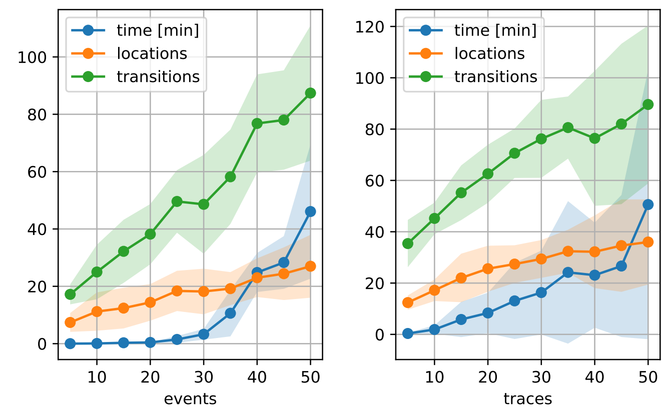


Fig. 5: MAPE values for each regression model.



(a) Maximum trace length. (b) Number of training traces.

Fig. 6: L^*_{SHA} scalability results with increasing trace length (a) and volume of training dataset (b).

The L^*_{SHA} model shows significantly better performance than XGBoost in both the regression and classification tasks. It also achieves a slightly smaller median MAPE compared to the NN model; however, the corresponding statistical tests do not allow rejection of the null hypothesis that both models have the same MAPE distribution. Nevertheless, L^*_{SHA} attains substantially higher classification accuracy, more than double that of the NN, suggesting a deeper understanding of the underlying phenomenon being modeled.

RQ1 summary. The SHA learned by L^*_{SHA} outperforms both baseline models, the dual-input Neural Network (NN) and XGBoost. This is particularly evident in classification accuracy, where L^*_{SHA} achieves results that are $2\times$ to $3\times$ higher. For regression tasks, L^*_{SHA} delivers performance comparable to the NN and demonstrates a $4\times$ median error reduction compared to XGBoost.

RQ2 (cost of learning): We measure the cost in terms of execution time of L^*_{SHA} during the learning process. For this purpose, we generate up to 50 random sets of signals, each corresponding to a clinical scenario. Traces of each scenario contain a variable number of events, up to a maximum of 50. We conduct the learning phase using different collections of traces as a training set. We repeat each experiment 5 times and measure both the execution time and the number of structural elements in the learned SHA. These structural elements serve as an indicator of the complexity of the underlying behavior.

Fig. 6a shows the average execution time and the number of learned locations and transitions when training with 10 traces and varying the number of events per trace from 5 to 50. Fig. 6b presents the same metrics, this time by varying the number of traces from 5 to 50, while keeping the number of events per trace fixed at 20. The shaded areas around the average lines represent the standard deviations across the 5 repeats.

The plots show that L^*_{SHA}'s training overhead scales linearly with the number of traces and superlinearly (with a quadratic trend) with the number of events per trace. Training time reaches up to one hour for the largest training datasets used, although substantial variance is observed in these cases. A linear correlation is also evident between the training data size, meaning both the number of events and of traces, and the number of learned structural elements (both locations and transitions). The number of learned transitions increases more substantially with the number of events per trace and also correlates with particularly long execution times. In contrast, the number of learned locations grows only slightly with both the number of traces and events per trace, and appears largely independent of training time. These findings suggest that L^*_{SHA} is well-suited for learning models in settings where the number of distinct behavioral episodes (i.e., traces) grows, rather than their internal complexity (i.e., events per trace).

RQ2 summary. The training time of L^*_{SHA} increases with the complexity of the underlying phenomenon, particularly with the number of events and transitions, following a polynomial pattern. In contrast, it grows more linearly with the number of learned locations and traces.

RQ3 (effectiveness of failure detection): We measure the effectiveness by counting failure-inducing scenarios during PDP space exploration, also assessing their realism. We apply the mutation operators described in Example IV.7, namely parameter perturbation with mutation factor $k = 1.5$ (but never exceeding the parameter's bounds) and transition removal, starting from the complete physician-device model depicted in Fig. 3 at the beginning of the search procedure. We run all methods with the same budget (500 generated mutants), distributed across 50 generations with 10 individuals each in the search-based approach. For each technique, we count the number of failures for each requirement and the number of such failures representing realistic scenarios using the criteria defined by the domain expert, as explained in Example IV.8.

Fig. 7 shows the distribution of the *total* number failures and *realistic* failures only, identified by each technique over 20 repeats. Table II shows statistical significance and effect size calculated for all pairwise comparisons. Mutational fuzzing identifies a proportion of failure scenarios ranging between 30% and 80% across different requirements, with only a

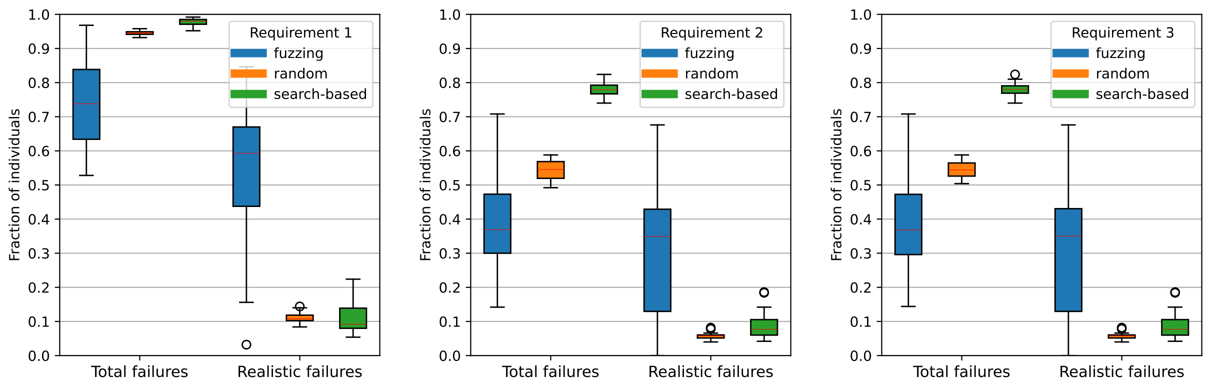


Fig. 7: Boxplots of fraction of failures, both total and realistic, for each requirement and exploration technique.

slight reduction when focusing exclusively on realistic failures. Specifically, fuzzing detects failures in approximately 60% of cases (median value) for Requirement 1 and 35% for Requirements 2 and 3. In contrast, the search-based approach results in failures in at least 75% of the cases, though the proportion of mutants deemed realistic is much smaller, around 10%. In this sense, the performance of the search-based approach is comparable to random search, although statistical tests indicate that the outcome distributions differ in all cases except one (realistic failures for Requirement 1). Overall, results suggest that while fuzzing uncovers fewer failures in total, it identifies a significantly higher number of realistic failures.

RQ3 summary. Fuzzing outperforms both the search-based approach and random search, identifying the highest number of realistic failures: from 35% to 60% (median) of all explored scenarios, depending on the specific requirement. In contrast, the search-based approach and random search identified only about 5% to 10% as realistic failures.

RQ4 (cost of failure detection): We maintain the same setting of RQ3, but we assess the cost by measuring the execution time of the experiments. Specifically, we record the duration of the PDP space exploration for each of the methods under comparison. The majority of this time is spent performing SMC using UPPAAL, while the overhead associated with generating mutants is mostly negligible. Fig. 8 shows the distribution of execution times across the 20 repeats. Fuzzing takes the longest to complete, typically requiring between 3 and 5 minutes ($\sim$ 0.5 seconds to generate and verify a mutant). The search-based approach is the most time-efficient, usually completing in just over 1 minute, whereas random search requires about 2 minutes. This difference can be attributed to the search-based method's focus on minimizing requirement satisfaction probabilities, which often results in simpler models, typically with fewer transitions, that concentrate around global optima in the parameter space. These models are both more likely to violate the requirements and significantly faster to evaluate. However, their simplicity may lead to unrealistic behavior, which could explain the large gap between total and realistic scenarios in RQ3.

TABLE II: Results of exploration tests.

Total failures	Requirement 1		Requirement 2		Requirement 3	
	p-val	$\hat{A}_{12}$	p-val	$\hat{A}_{12}$	p-val	$\hat{A}_{12}$
fuzzing vs random	1.17e-06	L	3.10e-05	L	2.29e-05	L
fuzzing vs search-based	1.28e-07	L	6.77e-08	L	6.71e-08	L
random vs search-based	2.41e-07	L	6.75e-08	L	6.70e-08	L
Realistic failures	**Requirement 1**		**Requirement 2**		**Requirement 3**	
	p-val	$\hat{A}_{12}$	p-val	$\hat{A}_{12}$	p-val	$\hat{A}_{12}$
fuzzing vs random	1.19e-06	L	1.22e-03	L	1.22e-03	L
fuzzing vs search-based	2.35e-06	L	3.04e-03	L	3.04e-03	L
random vs search-based	1.98e-01	S	5.87e-03	L	6.86e-03	L

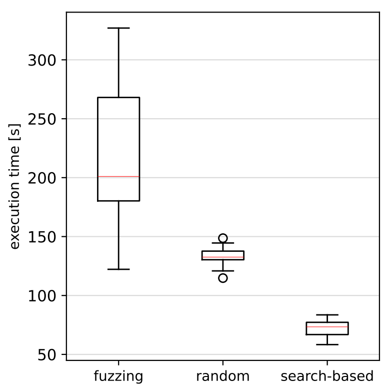


Fig. 8: Boxplot for execution times of exploration techniques.

RQ4 summary. All techniques considered incur an overhead of $\sim$ 0.5s per mutant generation and verification. The search-based approach is the most efficient, requiring roughly half the execution time of random search and one-third that of mutational fuzzing.

C. Threats to Validity

Relying on a single evaluation subject poses a potential threat to external validity. To mitigate this limitation, we incorporated clinical data during the learning phase to construct a portion of the DS, thereby reflecting actual patient behavior and increasing its relevance to practical settings. Moreover, the validity of the resulting DS heavily depends on domain expertise. This work relied on validation from one domain expert with experience in intensive care. These efforts ensure that both the learned and designed components are clinically meaningful and match real-world practice. Incorporating multiple experts and reporting inter-rater agreement would strengthen clinical relevance and reduce potential subjectivity. However, this aspect does not affect the core findings related to the effectiveness of GENGAR in identifying critical scenarios.

Concerning MITL requirements considered in our experiments, we carefully controlled the intervals defining probabilistic thresholds to prevent trivial or uninformative violations. This fine-grained tuning enhances internal validity by ensuring that observed outcomes reflect the effectiveness of our approach, rather than being influenced by requirements that are inherently unsatisfiable or easily violated.

Further studies with more complex study subjects would increase the generalizability of our results. Scaling beyond three requirements may require the use of alternative optimization techniques such as multi-objective optimization algorithms.

VI. Related Work

The literature often uses the terms Digital Twin (DT) and DS interchangeably, although the distinction between both architectural paradigms has become clearer over time. In this section, we adopt the terminology used in the reviewed papers.

A DT-oriented architectural solution to formalizing the PDP triad is introduced by Bersani et al. [7], [8]. The authors introduce a DT-based framework emphasizing trust, defined by the alignment between physical and DTs and the validation of performance properties under human and environmental uncertainties [7], a high-level reference architecture, and reliability estimation mechanisms [8]. However, the work is at a conceptual level and lacks empirical validation.

The DT concept, initially introduced by Grieves [22], has evolved significantly, particularly in manufacturing. A survey by Abanda et al. [1] compares ten DT definitions from 2012 to 2022, illustrating this evolution. As human involvement in industrial processes increased, DTs expanded to model human aspects, giving rise to the concept of the Human Digital Twin (HDT). Gaffinet et al. [20] comprehensively review ten surveys on HDTs, showing their growth across scientific domains. Miller et al. [33] define HDT as a digital representation of a human using mechanistic and statistical models, covering attributes like cognition, personality, and behavior. Lin et al. [29] survey HDT technologies and present a generic architecture, emphasizing human activity and social behavior modeling but omitting task-based workflows like medical procedures, which our model incorporates. Asad et al. [5] survey Human-Centric DTs, noting that while these DTs address ergonomics and health, they generally neglect human cognitive processes and decision-making. Wang et al. [41] similarly note the overemphasis on physical assets at the expense of human modeling. They present a layered HDT architecture, within which our SHA-based patient model aligns. However, our use of SMC introduces a unique analysis capability absent from their taxonomy. Li et al. [23] propose a framework for ergonomic analysis using DTs, focusing on body movement and cognition. Their "Digital Engine" includes simulation and analysis components, a structure that maps to our SHA-based models and SMC-driven dependability analysis. Sahal et al. [38] propose a Personal DT to support decision-making, although high-level and abstract. They list healthcare system requirements including clinical decision support and analysis. Our system can be seen as a concrete realization of these goals.

Katsoulakis et al. [25] survey DTs in healthcare, noting a predominant focus on patient care and organ-specific modeling for personalized medicine. One identified application area—hospital management and care coordination—includes hospital workflows and staff modeling, aligning with our approach. However, none of the surveyed works integrate automata-based models or software testing-inspired analysis techniques, marking our work as a novel contribution. Roopa and Venugopal [37] focus on DTs in Cyber-Physical Healthcare Systems (DT-CPHS), identifying operational efficiency as a key requirement. Our system contributes to this goal by detecting critical scenarios in intensive care units caused by physician misbehavior, thereby enhancing care delivery. Croatti et al. [15] explore DT integration with Multi-Agent Systems for smart healthcare environments, modeling heterogeneous entities and supporting trauma care. Their DT serves as a personal assistant for documentation and procedure tracking but lacks sophisticated analytics or decision-making logic. Mariani et al. [32] extend this work conceptually, discussing simulation and prediction capabilities without concrete implementation. While aligned in representing the PDP triad through DTs, our approach offers a more grounded and operational realization of these concepts. Mohamed et al. [34] propose a DT framework for healthcare systems engineering, identifying system components like patients, staff, and facilities. However, physicians and healthcare processes are treated more as administrative units than knowledge-driven clinical agents. Our work distinguishes itself by capturing both the behavioral dynamics and medical expertise of physicians within the DT.

VII. Conclusion

We introduce GENGAR, a novel methodology for detecting dependability threats in critical PDP systems running in healthcare scenarios. By integrating automata learning with model space exploration and verification, GENGAR enables the identification of realistic, failure-inducing scenarios in intensive care settings. Our experimental results show that GENGAR outperforms mainstream regression models in predictive accuracy, and also uncovers critical yet plausible failure-inducing scenarios according to domain experts. Particularly, mutational fuzzing in model space exploration yields a significantly higher number of realistic failures than competing techniques.

We plan to carry out a systematic analysis of failure patterns to aid in developing reusable mitigation strategies. We also plan to expand the modeling framework to capture more granular aspects of human-machine interaction, including cognitive load or team dynamics, to enhance its fidelity. Finally, inference frameworks, such as Bayesian inference, may further strengthen the robustness and adaptability of the DSs.

Acknowledgments

This work was partially supported by the PRIN project SAFEST (award No: 20224AJBLJ) funded by the Italian Ministry of Education, Universities, and Research (MIUR).

References

[1] F. Henry Abanda, N. Jian, Selorm Adukpo, Valerian Vanessa Tuhaise, and Marcelline Blanche Manjia. Digital twin for product versus project lifecycles' development in manufacturing and construction industries. *J. Intell. Manuf.*, 36(2):801–831, 2025.

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

[3] Dana Angluin. Learning regular sets from queries and counterexamples. *Inf. Comput.*, 75(2):87–106, 1987.

[4] Andrea Arcuri and Lionel Briand. A practical guide for using statistical tests to assess randomized algorithms in software engineering. In *Intl. Conf. on Software Engineering*, page 1–10. ACM, 2011.

[5] Usman Asad, Madeeha Khan, Azfar Khalid, and Waqas Akbar Lughmani. Human-centric digital twins in industry: A comprehensive review of enabling technologies and implementation strategies. *Sensors*, 2023.

[6] Gerd Behrmann, Alexandre David, and Kim G Larsen. A tutorial on uppaal. *Formal methods for the design of real-time systems*, pages 200–236, 2004.

[7] Marcello M. Bersani, Chiara Braghin, Vittorio Cortellessa, Angelo Gargantini, Vincenzo Grassi, F. Lo Presti, Raffaela Mirandola, Alfonso Pierantonio, Elvinia Riccobene, and Patrizia Scandurra. Towards trust-preserving continuous co-evolution of digital twins. In *Intl. Conf. on Software Architecture Companion*, pages 96–99. IEEE, 2022.

[8] Marcello M. Bersani, Chiara Braghin, Angelo Gargantini, Raffaela Mirandola, Elvinia Riccobene, and Patrizia Scandurra. Engineering of trust analysis-driven digital twins for a medical device. In *Software Architecture. ECSA*, pages 467–482. Springer, 2022.

[9] Andrew D Bersten and Jonathan Handy. *Oh's Intensive Care Manual E-Book*. Elsevier Health Sciences, 2013.

[10] Marcel Böhme, Van-Thuan Pham, Manh-Dung Nguyen, and Abhik Roychoudhury. Directed greybox fuzzing. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, page 2329–2344, New York, NY, USA, 2017. ACM.

[11] Marcel Böhme, Van-Thuan Pham, and Abhik Roychoudhury. Coverage-based greybox fuzzing as markov chain. In *ACM SIGSAC Conf. on Computer and Communications Security*, page 1032–1043. ACM, 2016.

[12] Aaron Bray, Jeffrey B. Webb, Andinet Enquobahrie, Jared Vicory, Jerry Heneghan, Robert Hubal, Stephanie TerMaath, Philip Asare, and Rachel B. Clipp. Pulse Physiology Engine: an Open-Source Software Platform for Computational Modeling of Human Medical Simulation. *SN Comprehensive Clinical Medicine*, 1(5):362–377, 2019.

[13] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *KDD Proc.*, pages 785–794, 2016.

[14] Alessandro Colombo, Gionatha Pirola, and Angelo Gargantini. BREATHE - Biomedical Respiratory Engine for Advanced Training and Human Evaluation. https://github.com/GionathaPirola/BREATHE, 2025.

[15] A. Croatti, M. Gabellini, S. Montagna, and A. Ricci. On the integration of agents and digital twins in healthcare. *Journal of Medical Systems*, 44, 2020.

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

[17] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[18] Dheeraj Kumar Dhaked and Valayapathy Lakshmi Narayanan. *Recent Trends in Medical Cyber-Physical System—A Brief Survey*, pages 39–54. Springer Nature Singapore, Singapore, 2024.

[19] Myron D Fottler, John D Blair, Carlton J Whitehead, Michael D Laus, and Grant T Savage. Assessing key stakeholders: who matters to hospitals and why? *Journal of Healthcare Management*, 1989.

[20] Ben Gaffinet, Jana Al Haj Ali, Yannick Naudet, and Hervé Panetto. Human digital twins: A systematic literature review and concept disambiguation for industry 5.0. *Computers in Industry*, 166:104230, 2025.

[21] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT Press Cambridge, 2016.

[22] Michael W. Grieves. Product lifecycle management: the new paradigm for enterprises. *Intl. Jnl. of Product Development*, 2(1/2):71–84, 2005.

[23] Qiqi He, Li Li, Dai Li, Tao Peng, Xiangying Zhang, Yincheng Cai, Xujun Zhang, and Renzhong Tang. From digital human modeling to human digital twin: Framework and perspectives in human factors. *Chinese Journal of Mechanical Engineering*, 37:1–14, 2024.

[24] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[25] E. Katsoulakis, Q. Wang, H. Wu, L. Shahriyari, R. Fletcher, J. Liu, L. E. K. Achenie, H. Liu, P. Jackson, Y. Xiao, T. Syeda-Mahmood, R. Tuli, and J. Deng. Digital twins for health: a scoping review. *NPJ Digital Medicine*, 7, 2024.

[26] Axel Legay, Benoît Delahaye, and Saddek Bensalem. Statistical model checking: An overview. In *RV10 Proc.*, pages 122–135. Springer, 2010.

[27] Livia Lestingi, Marcello M Bersani, and Matteo Rossi. Model-driven development of service robot applications dealing with uncertain human behavior. *IEEE Intelligent Systems*, 37(6):48–56, 2022.

[28] Livia Lestingi, Nicla Frigerio, Marcello M Bersani, Andrea Matta, and Matteo Rossi. Data-driven energy modeling of machining centers through automata learning. *IEEE Transactions on Automation Science and Engineering*, 2024.

[29] Yujia Lin, Liming Chen, Aftab Ali, Christopher Nugent, Ian Cleland, Rongyang Li, Jianguo Ding, and Huansheng Ning. Human digital twin: a survey. *Journal of Cloud Computing*, 13, 2024.

[30] H. B. Mann and D. R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *The Annals of Mathematical Statistics*, 18(1):50–60, 1947.

[31] Valentin J.M. Manès, HyungSeok Han, Choongwoo Han, Sang Kil Cha, Manuel Egele, Edward J. Schwartz, and Maverick Woo. The art, science, and engineering of fuzzing: A survey. *IEEE Transactions on Software Engineering*, 47(11):2312–2331, 2021.

[32] Stefano Mariani, Marco Picone, and Alessandro Ricci. Agents and digital twins for the engineering of cyber-physical systems: opportunities, and challenges. *Annals of Mathematics and Artificial Intelligence*, 92:1–22, 2023.

[33] Michael Miller and Emily Spatz. A unified view of a human digital twin. *Human-Intelligent Systems Integration*, 4:23–33, 2022.

[34] Nader Mohamed, Jameela Al-Jaroodi, Imad Jawhar, and Nader Kesserwan. Leveraging digital twins for healthcare systems engineering. *IEEE Access*, 11:69841–69853, 2023.

[35] Institute of Medicine (US) Committee on Quality of Health Care in America. *To Err is Human: Building a Safer Health System*. National Academies Press (US), Washington, DC, 2000.

[36] William Owens. *The ventilator book*. First Draught Press, 2018.

[37] M. S. Roopa and K. R. Venugopal. Digital twins for cyber-physical healthcare systems: Architecture, requirements, systematic analysis, and future prospects. *IEEE Access*, 13:44963–44996, 2025.

[38] Radhya Sahal, Saeed H. Alsamhi, and Kenneth N. Brown. Personal digital twin: A close look into the present and a step towards the future of personalised healthcare industry. *Sensors*, 22(15), 2022.

[39] R. Sutton, D. Pincock, D. Baumgart, D. Sadowski, R. Fedorak, and K. Kroeker. An overview of clinical decision support systems: benefits, risks, and strategies for success. *npj Digital Medicine*, 3(1):17, 2020.

[40] András Vargha and Harold D. Delaney. A critique and improvement of the "cl" common language effect size statistics of mcgraw and wong. *Journal of Educational and Behavioral Statistics*, 25(2):101–132, 2000.

[41] Baicun Wang, Huiying Zhou, Xingyu Li, Geng Yang, Pai Zheng, Ci Song, Yixiu Yuan, Thorsten Wuest, Huayong Yang, and Lihui Wang. Human digital twin in the context of industry 5.0. *Robot. Comput.-Integr. Manuf.*, 85(C), 2024.