Software and Systems Modeling (2024) 23:473–502
https://doi.org/10.1007/s10270-023-01125-1

SPECIAL SECTION PAPER

Analyzing the impact of human errors on interactive service robotic scenarios via formal verification

Livia Lestingi[1] · Andrea Manglaviti[1] · Davide Marinaro[1] · Luca Marinello[1] · Mehrnoosh Askarpour[2] · Marcello M. Bersani[1] · Matteo Rossi[3]

Received: 17 June 2022 / Revised: 22 March 2023 / Accepted: 25 July 2023 / Published online: 17 August 2023

Abstract

Developing robotic applications with human–robot interaction for the service sector raises a plethora of challenges. In these settings, human behavior is essentially unconstrained as they can stray from the plan in numerous ways, constituting a critical source of uncertainty for the outcome of the robotic mission. Application designers require accessible and reliable frameworks to address this issue at an early development stage. We present a model-driven framework for developing interactive service robotic scenarios, allowing designers to model the interactive scenario, estimate its outcome, deploy the application, and smoothly reconfigure it. This article extends the framework compared to previous works by introducing an analysis of the impact of human errors on the mission's outcome. The core of the framework is a formal model of the agents at play—the humans and the robots—and the robotic mission under analysis, which is subject to statistical model checking to estimate the mission's outcome. The formal model incorporates a formalization of different human erroneous behaviors' phenotypes, whose likelihood can be tuned while configuring the scenario. Through scenarios inspired by the healthcare setting, the evaluation highlights how different configurations of erroneous behavior impact the verification results and guide the designer toward the mission design that best suits their needs.

Keywords Human–robot interaction · Human errors · Service robotics · Formal verification · Formal modeling · Stochastic Hybrid Automata · Statistical model checking

1 Introduction

The latest technological advances are rapidly transforming the service sector, and the professional figures that populate it face an equally significant evolution [21]. Service robots are complex machines capable of sophisticated motion, manipulation, and interaction skills. These devices are equipped with cutting-edge sensors and actuators, allowing them to perceive several aspects of their environment and elaborate data to perform tasks efficiently [20]. Therefore, service robots are increasingly widespread in healthcare, personal care, domestic assistance, retail, and entertainment settings. State-of-the-art applications involve robots relieving employees from the most mundane jobs in industrial settings. However, human actions in everyday life are not tied to a specific workflow but are virtually unconstrained; therefore, developing such robotic applications comes with considerable challenges.

While remarkable progress has been made with individual robotic skills such as manipulation and sensing, **software engineering** practices bringing these skills together into fully fledged applications are still lacking [22]. The practitioner in charge of *designing* the robotic application must account for all the sources of uncertainty affecting the specific environment at an early development stage, which calls for accessible and reliable tools [44]. We address this issue through a **model-driven** framework for designing and developing interactive service robotic applications extended in this paper compared to previous publications [37–39]. Specifically, within the broad domain of human–robot *interactions*,

Communicated by Antonio Cerone and Frank de Boer.

✉ Livia Lestingi
livia.lestingi@polimi.it

[1] Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Via Ponzio 34/5, 20133 Milan, Italy

[2] Department of Computing and Software, McMaster University, Hamilton, Canada

[3] Dipartimento di Meccanica, Politecnico di Milano, Via Privata Giuseppe La Masa 1, 20156 Milan, Italy

the work focuses on tasks that require physical coordination between the involved actors, including the possibility of direct contact. (Spoken interactions or manual guidance tasks are, thus, out of scope.) The framework targets scenarios with mobile robots interacting with one or multiple humans in a fixed layout (i.e., points of interest and fixed obstacles, such as walls and furniture, are known beforehand). Hence, applications featuring industrial manipulators or exoskeletons (although the latter are considered *personal care* robots by standard ISO 13482 [30]) are out of the scope of our work.

The framework clusters frequent human–robot interaction (HRI) contingencies from real-world scenarios into six *patterns* [37, 39] (e.g., human *follower* or *leader*) presented in detail in Table 5. A pattern with a destination (e.g., the location where the robot must accompany the human) constitutes a *service* that a human may request to interact with the robot. A sequence of services that the robot has to provide to one (or multiple) human subjects constitutes the robotic *mission*. When all services in the sequence are complete, the framework considers the mission finished with *success*.

The framework allows designers to **configure** the scenario under analysis, precisely the characteristics of the agents at play—the humans and the robots—the layout, and the robotic mission. The designer configures the scenario through a domain-specific language (DSL) [40] from which a formal model of the system is automatically generated, specifically a network of **Stochastic Hybrid Automata** (SHA). The SHA network is subject to **statistical model checking** (SMC) to estimate the value of key indicators about the mission, primarily the probability of success. The formal model captures human **physiological** and **behavioral** aspects to obtain accurate and insightful estimations, specifically through a model of physical fatigue and a stochastic approximation of haphazard decision-making.

Given the variability of service settings, it is fundamental for the formal analysis to take into account unexpected human actions straying from the planned mission, referred to as human *errors*. To this end, in this paper, we extend the development framework with a formal model of **erroneous human behaviors** and the possibility of analyzing their impact on the outcome of the mission. The human behavioral model is enriched with a taxonomy of errors well-established in the human–computer interaction analysis field [27]. The practitioner designs a scenario featuring one or multiple robotic missions based on the needs of the facility. They can then adjust the likelihood of each error by applying different behavioral profiles (e.g., *inexperienced* or *inattentive*) to human subjects in the scenario and estimate the probability of success with the defined configuration through the framework. By examining results obtained with different configurations (e.g., different combinations of behavioral profiles), the practitioner determines which erroneous behaviors have the most considerable impact on the mission and modify its design suitably to compensate for the issues that they have identified.

In more detail, the contributions presented in this paper with respect to [37–39] are:

1. We introduce SHA add-ons capturing erroneous behavior mapped to the service robotics setting;
2. SHA modeling human–robot interaction patterns presented in [37, 39] are refined with the developed formalization of erroneous behaviors;
3. Formal analysis, based on SMC, is extended through erroneous behavior profiles.

We demonstrate the features of the extended formal model through case studies inspired by the healthcare setting. The model-driven framework supports developers from the early design stage to deployment and reconfiguration in a flexible and accessible manner. Specifically, the comparison between human behavioral profiles showcases how erroneous behaviors enrich the analysis and provide deeper insights into the mission the practitioner is designing.

The paper is structured as follows: Sect. 2 provides an overview of the model-driven framework and the role that erroneous behaviors play in the analysis; Sect. 3 outlines the theoretical background underlying the work; Sect. 4 introduces the formal models of erroneous behaviors and the extended human–robot interaction patterns; Sect. 5 presents the evaluation results; Sect. 6 surveys related works in the literature; and Sect. 7 concludes.

2 Model-driven framework

As mentioned in Sect. 1, this article builds upon a model-driven framework for the design and development of service robotic missions, where interaction with human subjects is a core element. The model-driven nature of the framework allows for a smooth design and reconfiguration process of the robotic mission without requiring a solid background in robot programming or formal modeling techniques. Target users of the framework (also referred to as *practitioners* or *designers*) are experts of the service domain in charge of managing the logistic workflow of a facility (e.g., a hospital). These professional figures likely lack solid technical expertise in formal modeling techniques and, therefore, require accessible and flexible tools [22].

The development framework consists of three phases (labeled accordingly in Fig. 1):

PH1: the **scenario configuration** phase, in which the practitioner designs the interactive scenario and computes figures of merit about its outcome;

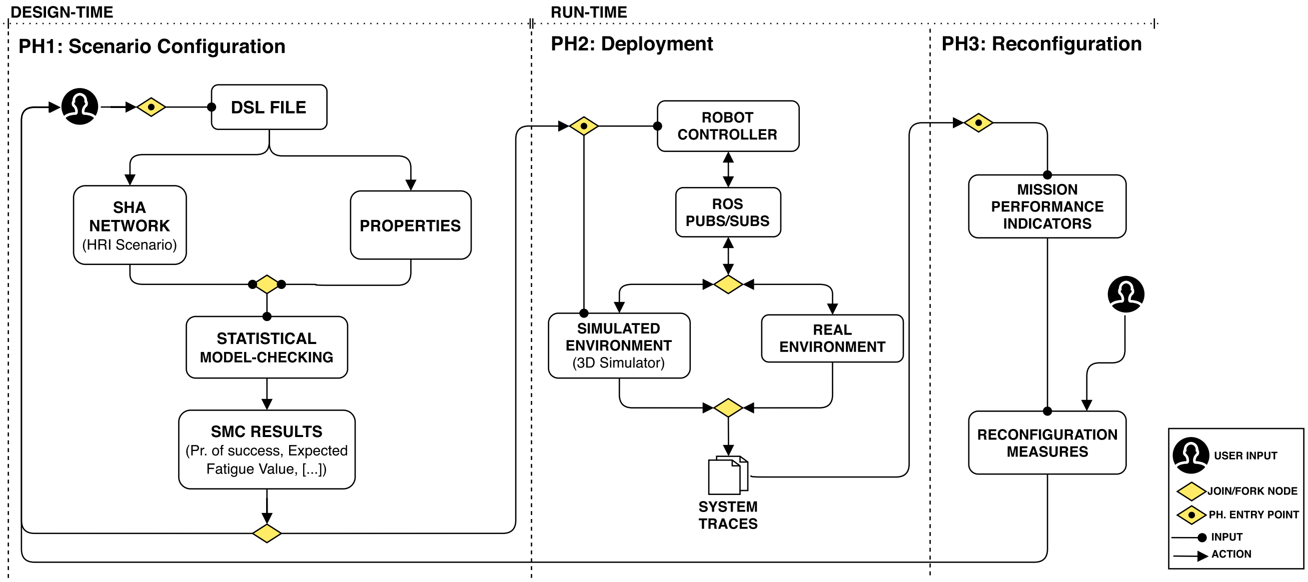

DESIGN-TIME
RUN-TIME
PH1: Scenario Configuration
PH2: Deployment
PH3: Reconfiguration
DSL FILE
SHA NETWORK (HRI Scenario)
PROPERTIES
STATISTICAL MODEL-CHECKING
SMC RESULTS (Pr. of success, Expected Fatigue Value, [...])
ROBOT CONTROLLER
ROS PUBS/SUBS
SIMULATED ENVIRONMENT (3D Simulator)
REAL ENVIRONMENT
SYSTEM TRACES
MISSION PERFORMANCE INDICATORS
RECONFIGURATION MEASURES
USER INPUT
JOIN/FORK NODE
PH. ENTRY POINT
INPUT
ACTION

Fig. 1 Workflow of the model-driven development framework, divided into three macro-phases: design-time analysis, deployment, and reconfiguration. Each box represents an operational step. Dotted yellow diamonds represent the entry point of each phase, whereas normal diamonds represent join/fork nodes. User symbols indicate manually performed operations or user decisions (color figure online)

PH2: the application **deployment** phase, in which the designed mission is deployed in a fully virtual or hybrid setting (adhering to the digital-twin paradigm);

PH3: the **reconfiguration** phase, in which the practitioner exploits the observations collected during deployment and the results of the design-time analysis to modify the scenario, if necessary.

Phase **PH1** takes place at design time, while both **PH2** and **PH3** take place at **runtime** since reconfiguration requires system traces as input.

To keep the framework accessible to target users, the entry point of the design-time phase (**PH1**) is the configuration of the scenario through a custom domain-specific language (DSL) [40]. The DSL is a lightweight textual notation to specify the main features of the scenario under scrutiny. The set of customizable features is presented in [40] and summarized in the following. The elements that can be defined through the DSL are the **floor layout**, the available **robots** and their characteristics, the involved **humans** and their characteristics, and the **mission**. The floor layout (i.e., the operational environment) is modeled as a two-dimensional plane where Cartesian coordinates of significant points (e.g., walls and doors) need to be specified. As for robots, designers specify the initial charge and the robots' commercial model, determining maximum speed and acceleration, rotational speed, and minimum voltage level to power the motors.

The core of the framework is a formal model capturing both behavioral and physiological aspects of the involved human subjects. For each human, it is possible to specify how susceptible to physical fatigue they are, referred to as the *fatigue profile*. We aggregate subjects by age (young/elderly) and state of health (healthy/sick) to identify four main fatigue profiles, determining the rate at which they fatigue and recover. The formalization of human behavior also captures the possibility that humans stray from the plan of the mission and perform actions out of their free will, constituting a significant source of uncertainty. This aspect is considerably extended in this paper and constitutes its main focus as we introduce a broader range of erroneous behaviors, whose likelihood can be tuned during the formal analysis to determine their impact on the mission's outcome.

As the framework targets interactive applications within the service domain, we recall that the robotic mission consists of the sequence of *services* the robot is asked to provide and the mission ends in success when all services are complete. The mission *fails* if the robot gets fully discharged or at least one of the humans reaches the maximum fatigue value. Note that an ongoing mission which has not been completed yet since it requires more time (i.e., the two conditions for failure are not satisfied) is not considered a failure. Human service requests impact the system's evolution while interacting with the robot and the condition that must be verified for the service to be complete. We have identified frequent interaction situations from real-world scenarios and grouped them into **patterns** [37, 39] described in Table 5. Therefore, scenarios within the scope described in Sect. 1 must meet two requirements to be eligible for analysis through the framework:

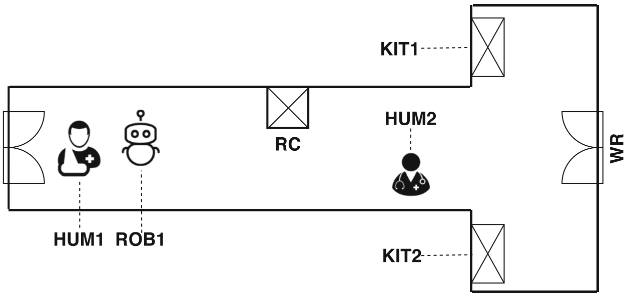


Fig. 2 Setup of the example scenario

1. as previously mentioned, the layout of the operational workspace must be known;
2. the robotic mission must be expressible as a sequence of the available patterns.

Example 1 Figure 2 shows the setup of an example scenario. The layout is a T-shaped corridor with four points of interest, i.e., significant locations within the layout: the robot's recharge station (**RC**), two cupboards containing medical kits (**KIT1** and **KIT2**), and the door leading to the waiting room (**WR**). There are three agents in the scenario: two humans (**HUM1** and **HUM2**) and one robot (**ROB1**). The designer assesses two alternative mission plans: the first mission features **ROB1** *leading* **HUM1** to the waiting room, then *delivering* **KIT2** to **HUM2**. The second mission features **ROB1** *following* **HUM2** to fetch **KIT1**, then *leading* **HUM1** to the waiting room.

This article extends the SHA modeling the human participating in each interaction pattern with suitable subsets of erroneous behaviors applicable to the specific pattern (i.e., not all interaction patterns are equally susceptible to all erroneous behaviors).

The DSL file is processed to customize the SHA network and the properties to be verified. Properties involve the estimation of domain-specific metrics, specifically the probability of occurrence of mission success, the maximum fatigue level reached by human subjects, and the residual battery charge of the robot. Given the stochastic nature of the formal model, properties are checked through SMC: Both the formal model and verification experiments are managed through the Uppaal tool [15] and its extension to SMC [14]. When verification ends, the designer examines the results of the SMC experiments, such as the estimated probability of success. If the results are not satisfactory, the designer modifies the scenario through the DSL and iterates the design-time analysis; otherwise, the analysis can switch to the runtime phase.

As shown in Fig. 1, the approach supports the application's **deployment** in a real or simulated environment (phase **PH2**). Simulation in a realistic virtual environment allows practitioners to perform hundreds of runs without any effort by real patients or healthcare professionals. In both cases, each element of the SHA network is translated into executable code. The mapping function between automata and deployment units (described in detail in [38]) ensures that two corresponding entities display corresponding behavior in response to the same events. The agents (i.e., the robot and humans) and the robot controller (referred to as the *orchestrator*) communicate through a ROS-based middleware layer [47]. The orchestrator receives data about the status of the system from sensors over ROS topics, checks it against a set of policies, and sends commands to agents in the same fashion. When communicating with the robot, instructions sent by the orchestrator actuate the motors for motion control. When communicating with the humans (e.g., doctors and patients in healthcare settings), the orchestrator issues suggestions on the action to perform that can be relayed through a wearable device. Data collected at runtime are processed to extract the values of relevant indicators (e.g., the *observed* success rate or the average patient fatigue level) that the designer examines. Should these figures of merit be deemed unsatisfactory, the scenario can be **reconfigured** (e.g., by changing the order in which humans are served or swapping the robot with another one in the fleet) to iterate the procedure.

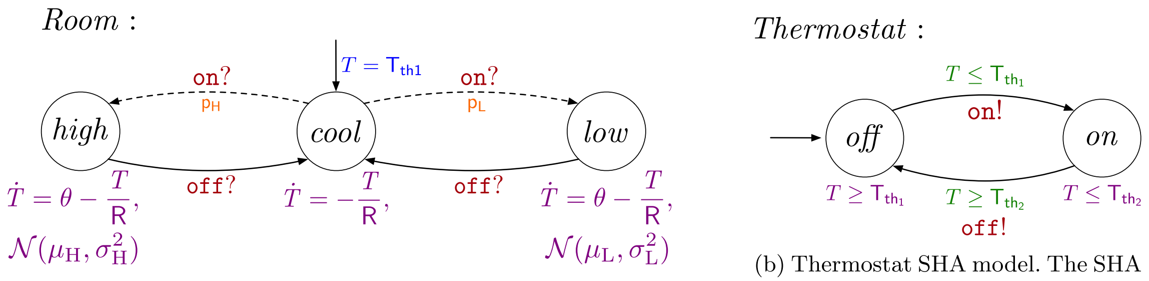
The framework is structured to have as many automated tasks as possible for the sake of accessibility to designers. As a matter of fact, their manual intervention is limited to the scenario configuration through the DSL (including potential iterative reconfigurations) and the examination of the formal analysis results. (The verification experiment is, instead, performed automatically.) For example, they are in charge of assessing whether a success probability of approximately 75% for the mission is sufficient given the facility's policies or calls for a reconfiguration.

3 Background

This section recaps the fundamental concepts underlying our work. Firstly, we introduce the formalism used to model HRI scenarios and the verification technique our framework exploits to estimate the robotic mission's outcome and further relevant indicators of the mission's performance. Secondly, we recap the principles underlying our modeling approach and the recurring features of the SHA network introduced in previous works to keep the article self-contained.

3.1 Stochastic Hybrid Automata and statistical model checking

The formalism underlying our work is Stochastic Hybrid Automata (SHA). We define SHA in the following and illus-


Room :
high
cool
low
on?
$\mathsf{p_H}$
on?
$\mathsf{p_L}$
$T = \mathsf{T_{th1}}$
off?
off?
$\dot{T} = \theta - \frac{T}{\mathsf{R}}$,
$\mathcal{N}(\mu_{\mathrm{H}}, \sigma_{\mathrm{H}}^2)$
$\dot{T} = -\frac{T}{\mathsf{R}}$,
$\dot{T} = \theta - \frac{T}{\mathsf{R}}$,
$\mathcal{N}(\mu_{\mathrm{L}}, \sigma_{\mathrm{L}}^2)$
Thermostat :
off
on
$T \leq \mathsf{T_{th_1}}$
on!
$T \geq \mathsf{T_{th_1}}$
$T \geq \mathsf{T_{th_2}}$
off!
$T \leq \mathsf{T_{th_2}}$

(a) Room SHA model. The SHA receives events from the thermostat to start heating at different rates (locations *high* and *low*) or cooling naturally (location *cool*).

(b) Thermostat SHA model. The SHA is in charge of triggering the events that cause the room to start (event `on`) or stop heating (`off`).

Fig. 3 Example of SHA network. Dashed arrows model probabilistic transitions with weights (in orange) $\mathsf{p_H}$ and $\mathsf{p_L}$, and solid arrows represent transitions with weight 1. Flow conditions, probability distributions, and exponential rates are in purple, channels in red, and guard conditions in green (color figure online)

trate their features through a running example inspired by [14, Section 4].

Example 2 The example captures a system composed of a room, whose model is shown in Fig. 3a, and the thermostat controlling its temperature, shown in Fig. 3b. The thermostat can be either *on*, which makes the room warmer, or *off*, thus, letting the room temperature decrease naturally. As soon as the temperature decreases below a threshold $\mathsf{T_{th_1}}$ (resp., exceeds a threshold $\mathsf{T_{th_2}}$), the thermostat fires event `on` (resp., `off`). The room temperature is modeled by variable T, which grows according to differential equation $\dot{T} = \theta - \frac{T}{\mathsf{R}}$ when the thermostat is on and decreases according to $\dot{T} = -\frac{T}{\mathsf{R}}$ when it is off, where R is a constant and θ is a randomly distributed parameter. When event `on` fires, the room may start heating at a high rate or a low rate (e.g., if a window is open): The choice is modeled probabilistically and governed by probability weights $\mathsf{p_H}$ and $\mathsf{p_L}$, respectively. Parameter θ is a realization of distribution $\mathcal{N}(\mu_{\mathrm{H}}, \sigma_{\mathrm{H}}^2)$[1] when the room is heating at a high rate and $\mathcal{N}(\mu_{\mathrm{L}}, \sigma_{\mathrm{L}}^2)$ in the opposite case. Throughout the paper, we express that a random parameter θ is a realization of random variable Θ governed by distribution $\mathcal{N}(\mu, \sigma)$ through notation $\theta \sim \mathcal{N}(\mu, \sigma)$.

SHA are defined in the following [2, 5, 15]. Let W be a set of symbols; we indicate with $\Gamma(W)$ the set of conjunctions of linear constraints on elements of W. We indicate with $\Xi(W)$ the set of updates on elements of W. An *update* in $\Xi(W)$ (for example, $w' = w + 2$) is an arithmetical constraint where free variables are elements of W (e.g., $w \in W$) and of its primed version W' (e.g., $w' \in W'$).

Definition 1 A Stochastic Hybrid Automaton is a tuple $\langle L, W, \mathcal{F}, \mathcal{D}, \mathcal{I}, C, \mathcal{E}, \mu, \mathcal{P}, l_{\mathrm{ini}} \rangle$, where:

1. L is the set of locations and $l_{\mathrm{ini}} \in L$ is the initial location;
2. W is the set of real-valued variables of which clocks $X \subseteq W$, dense-counter variables $V_{\mathrm{dc}} \subseteq W$, and constants $K \subseteq W$ are subsets;
3. $\mathcal{F} : L \to \{(\mathbb{R}_+ \cup (\mathbb{R}_+ \times \mathbb{R})) \to \mathbb{R}^W\}$ is the function assigning a set of flow conditions to each location;
4. $\mathcal{D} : L \rightharpoonup \{\mathbb{R} \to [0, 1]\}$ is the *partial* function assigning a probability distribution from $\{\mathbb{R} \to [0, 1]\}$ to locations which feature flow conditions with two parameters;
5. $\mathcal{I} : L \to \Gamma(W)$ is the function assigning a (possibly empty) set of invariants to each location;
6. C is the set of channels, including the internal action ϵ;
7. $\mathcal{E} \subset L \times C_{!?} \times \Gamma(W) \times \wp(\Xi(W)) \times L$ is the set of edges, where $C_{!?} = \{\mathtt{c!} \mid \mathtt{c} \in C\} \cup \{\mathtt{c?} \mid \mathtt{c} \in C\}$ is the set of *events* involving channels in C. Given an edge $(l, c, \gamma, \xi, l') \in \mathcal{E}$, l (resp. l') is the outgoing (resp. ingoing) location, c is the *edge event*, γ is the *edge condition* and ξ is the *edge update*. For each $l \in L$, $\mathcal{E}(l) \subseteq C_{!?} \times \Gamma(W) \times \wp(\Xi(W)) \times L$ is the set of edges outgoing from l (for each $(c, \gamma, \xi, l') \in \mathcal{E}(l)$, then, $(l, c, \gamma, \xi, l') \in \mathcal{E}$ holds and vice-versa);
8. $\mu : (L \times \mathbb{R}^W) \to \{\mathbb{R}_+ \to [0, 1]\}$ is the function assigning a probability distribution from $\{\mathbb{R}_+ \to [0, 1]\}$ to each *configuration* of the SHA; a configuration is a pair (l, v_{var}) constituted by a location $l \in L$ and a valuation $v_{\mathrm{var}} \in \mathbb{R}^W$;
9. $\mathcal{P} : L \rightharpoonup \{(C_{!?} \times \Gamma(W) \times \wp(\Xi(W)) \times L) \rightharpoonup [0, 1]\}$ is the *partial* function assigning a discrete probability distribution from $\{(C_{!?} \times \Gamma(W) \times \wp(\Xi(W)) \times L) \rightharpoonup [0, 1]\}$ to locations such that, for each $l \in L$, $\mathcal{P}(l)$ is defined if, and only if, $\mathcal{E}(l)$ is non-empty; also, the domain of the distribution is $\mathcal{E}(l)$ ($\sum_{\substack{\alpha=(\mathtt{c!}, \gamma, \xi, l') \in \mathcal{E}(l) \\ \mathtt{c} \in C}} \mathcal{P}(l)(\alpha) = 1$ holds).

In SHA, real-valued variables (i.e., a generalization of clocks) evolve in time according to expressions referred to as *flow conditions* [2]. The flow conditions constraining the evolution over time of variables in W are defined through sets of ordinary differential equations (ODEs). This feature makes SHA a fitting formalism to model systems with complex

[1] Throughout the paper, notation $\mathcal{N}(\mu, \sigma^2)$ is used to indicate *Normal* distributions.

dynamics, as it is possible to model through flow conditions, for example, laws of physics or biochemical processes. ODEs constraining clocks (for which $\dot{x} = 1$ holds for all $x \in X$), dense-counter variables, and constants (where $\dot{v} = 0$ holds for all $v \in V_{\mathrm{dc}} \cup K$) are special cases of flow conditions.

If a randomly distributed parameter θ is an independent term for a flow condition $f \in \mathcal{F}(l)$ on location $l \in L$—i.e., $f = f(\cdot, \theta)$ holds—then f is interpreted as a **stochastic process** [24]. We limit the analysis to flow conditions depending on *at most one* random parameter, as per Definition 1, which is the case for human–robot interaction models within the scope of our work. For example, in the SHA shown in Fig. 3a, the room temperature is modeled by real-valued variable $T \in W$. When the temperature is decreasing, it is constrained by flow condition $\dot{T}(t) = -T(t)/\mathrm{R}$, where function $\dot{T}(t) \in \mathcal{F}(cool)$ depends on time only and has solutions in $\mathbb{R}$. When the temperature is increasing, it evolves according to flow condition $\dot{T}(t, \theta) = \theta - T(t, \theta)/\mathrm{R}$, depending both on time and random parameter θ. The domain of $\dot{T}(t, \theta)$ is, thus, $\mathbb{R}_+ \times \mathbb{R}$, and its solutions belong to $\mathbb{R}$.

The formal model is developed through the Uppaal tool [36]. Update instructions in $\Xi(W)$ are either deterministic or stochastic assignments (the latter if stochastic parameters are involved). In the latter case, upon entering a location $l \in L$ such that $\mathcal{D}(l) \neq \perp$ holds, with Uppaal, it is possible to code update instructions on the incoming edge that generate a realization of distribution $\mathcal{D}(l)$ (e.g., $\mathcal{N}(\mu_{\mathrm{H}}, \sigma_{\mathrm{H}}^2)$ in Fig. 3a) and assign it to the stochastic parameter (e.g., θ in Fig. 3a) [14].

In SHA, probability measures are associated with time delays or transition outputs [15]. In Uppaal, probability distributions over delays are either *uniform* or *exponential* [14]. If an edge can fire with a *bounded* delay (i.e., within a finite range of values), function μ assigns a uniformly distributed probability to all delays within such range. If an edge can fire with an *unbounded* delay, the probability is assigned according to an exponential distribution. Probabilities over transition outputs are assigned as weights $\mathcal{P}(l, \mathtt{c}, \gamma, l') \in [0, 1]$ on edges, which determine how likely the system is to evolve in a certain direction. For example, $\mathsf{p_L}$ and $\mathsf{p_H}$ in Fig. 3 determine the probability of taking each of the two edges from *idle*.

Complex systems with multiple entities are modeled as a combination of SHA, forming a **network**. Synchronization among different automata inside a network occurs through the channels of set C [36]. Given a channel $\mathtt{c} \in C$ and two complementary edges labeled by `c!` (the *sender*) and `c?` (the *receiver*), triggering an event through channel `c` causes both edges to fire simultaneously. In Fig. 3b, the thermostat triggers an event through channels `on!` and `off!` to start or stop heating the room. The room automaton receives the triggered event through labels `on?` and `off?`, which make the corresponding edges fire.

SHA are eligible for statistical model checking (SMC) [1]. SMC requires a model M with stochastic features (the SHA network) and a property ψ expressed, in our case, in metric interval temporal logic (MITL) [3]. SMC exploits a finite set of *runs* obtained through simulations of the formal model to calculate the probability of ψ holding within a specific time bound. Specifically, the model-checker processes each run to verify whether ψ holds for that specific run, thus collecting a set of Bernoulli trial outcomes. The probability of ψ holding for M corresponds to the value of expression $\mathbb{P}_M(\psi)$ [14], calculated by applying statistical techniques to the collected Bernoulli trials. Therefore, unlike traditional model checking and probabilistic model checking (that analytically computes the probability of a property holding [34]), SMC does not entail an exhaustive exploration of the state space. In our framework, property ψ is of the form $\diamond_{\leq \tau}\mathsf{ap}$, where $\diamond$ is the "eventually" operator, $\mathsf{ap} \in \mathrm{AP}$ is an atomic proposition, and τ is an integer time bound. Since we do not compare the value of $\mathbb{P}_M(\psi)$ against a threshold $\vartheta \in [0, 1]$, the SMC experiment returns confidence interval $[\mathrm{p_{min}}, \mathrm{p_{max}}]$ for the probability of property ψ holding for M.

SHA network M is subject to SMC to estimate the probability of success of the interactive scenario under analysis (corresponding to expression $\mathbb{P}_{\mathrm{M}}(\diamond_{\leq \tau}\mathsf{scs})$, where Boolean variable scs becomes true when the mission ends in success). To determine when the generated set of runs is sufficient to conclude the experiment, Uppaal checks the length of the interval $[p - \epsilon, p + \epsilon]$ to which the *real* success probability belongs (where $p = \mathrm{p_{min}} + (\mathrm{p_{max}} - \mathrm{p_{min}})/2$ holds), which is calculated according to the Clopper–Pearson method [13]. Uppaal stops generating new traces when $\epsilon \leq \epsilon_{\mathrm{th}}$ holds, where ϵ_{th} is a user-specified parameter indicating the maximum desired estimation error. The smaller the ϵ_{th}, the more accurate the estimation must be; thus, more traces are required.

3.2 HRI scenarios modeling principles

We illustrate the fundamental modeling principles underlying the whole SHA network, which features automata modeling the humans, the robot and its battery, and the orchestrator. Specifically, we recap the features of SHA modeling human behavior, which is the primary focus of this article. Interested readers can refer to [38] for a detailed description of the SHA modeling the robot and its battery, respectively, whereas the orchestrator is thoroughly described in [37, 39].

The SHA network models the agents' behavior based on their current *operational state* (e.g., the human resting or walking). Every operational state of the agent corresponds to a *location* in L. At the current stage of development, operational states are *fixed* for each agent and, in case of human subjects, for each interaction pattern. In all interaction patterns, SHA modeling humans differentiate between operational states based on how fatigue evolves (i.e., whether individuals are recovering or not) and how humans are interacting with the robot (e.g., they are leading the action or waiting for a robot's action). Operational states specific to each interaction pattern are recapped in Sect. 4.2.

SHA captures the evolution of relevant quantitative attributes of the real system, such as human fatigue and battery level of charge. Each physical attribute corresponds to a real-valued variable in set $W \setminus \{X \cup V_{\mathrm{dc}} \cup K\}$ of their modeling automata and flow conditions $\mathcal{F}(l)$, associated with a location l, reproduce the set of ODEs constraining the evolution of real-valued variables in that specific operating state.

The orchestrator controls the robots and instructs humans based on sensor readings. Dense counters (set V_{dc}) are the discrete equivalents of real-valued variables. Dense counters are periodically updated every $\mathsf{T}_{\mathsf{poll}} \in K$ time units (where $\mathsf{T}_{\mathsf{poll}}$ corresponds to the refresh period of the specific sensor) through update instructions in $\Xi(W)$ that are compatible with the ODEs modeling the dynamics of the physical attributes in every location. Every SHA in the network uses a clock $t_{\mathrm{upd}} \in X$ to measure the time elapsed between two consecutive measurements and trigger an update. Sensors share data with the orchestrator through ROS publisher nodes. Therefore, when $t_{\mathrm{upd}} = \mathsf{T}_{\mathsf{poll}}$ holds for an automaton $\mathcal{A}$, hence when time $\mathsf{T}_{\mathsf{poll}}$ has elapsed since the last measurement, $\mathcal{A}$ switches to a *committed* location. A committed location is equivalent to an ordinary location with invariant $t \leq 0$ and all incoming edges with update instruction $t = 0$ for some $t \in X$: Therefore, time cannot elapse while in these locations [36]. In that location, the dense counters modeling the latest sensor readings are immediately notified to the orchestrator by firing an event over a dedicated channel that triggers the publishing routine. (The corresponding modeling pattern is described in detail in [38, Section IV.2].)

SHA modeling human behavior feature a model of fatigue. Incorporating the fatigue model into the automata leads to formal analysis results that also account for at least one indicator of the humans' physical status. As a matter of fact, especially in healthcare settings, subjects may be in critical physical conditions that may significantly impact the duration of the mission and, thus, its probability of success. Human fatigue is a complex phenomenon driven by several factors: Our work focuses on muscular fatigue due to physical strain. We exploit the fatigue and recovery model proposed by Konz [23, 33], described by Eq. 1. Human action undergoes alternate fatigue and recovery cycles, and each cycle is associated with an index i uniquely identified, given time t, by function $j : \mathbb{R}_{+} \to \mathbb{N}$. (Thus, $i = j(t)$ holds.) We indicate as t_i the timestamp at which cycle i ends. During both fatigue and recovery, fatigue $F(t)$ depends on the residual value from the previous cycle ended at time t_{i-1}. Parameters λ_i and ρ_i are the fatigue and recovery rates for cycle i.

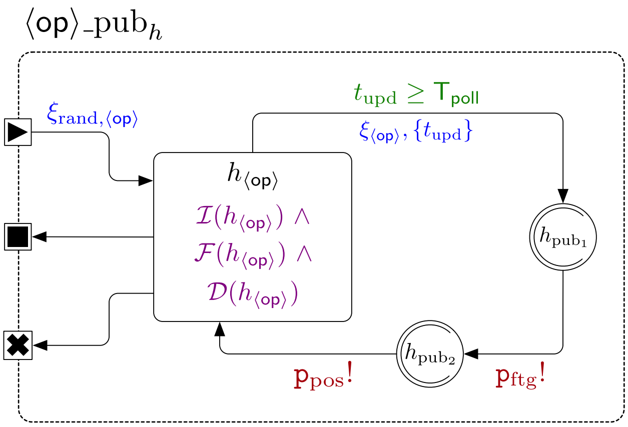

$\langle \mathsf{op} \rangle$_pub$_h$
$\xi_{\mathrm{rand},\langle \mathsf{op} \rangle}$
$t_{\mathrm{upd}} \geq \mathsf{T}_{\mathsf{poll}}$
$\xi_{\langle \mathsf{op} \rangle}, \{t_{\mathrm{upd}}\}$
$h_{\langle \mathsf{op} \rangle}$
$\mathcal{I}(h_{\langle \mathsf{op} \rangle}) \wedge$
$\mathcal{F}(h_{\langle \mathsf{op} \rangle}) \wedge$
$\mathcal{D}(h_{\langle \mathsf{op} \rangle})$
h_{pub_1}
h_{pub_2}
$\mathsf{p}_{\mathsf{pos}}!$
$\mathsf{p}_{\mathsf{ftg}}!$

Fig. 4 $\langle \mathsf{op} \rangle$_pub$_h$ modeling pattern [38]. Color-coding is the same as Fig. 3. Entry and exit ports are marked by ▶, ■, and × symbols (color figure online)

$$F(t) = \begin{cases} 1 - (1 - F(\mathsf{t}_{i-1})) \cdot e^{-\lambda_i (t - \mathsf{t}_{i-1})} & \text{(fatigue)} \\ F(\mathsf{t}_{i-1}) \cdot e^{-\rho_i (t - \mathsf{t}_{i-1})} & \text{(recovery)} \end{cases} \tag{1}$$

Full recovery occurs when $F(t) = 0$ holds, whereas condition $F(t) = 1$ models the case in which the muscle has reached the maximum level of *endurance*. Experiments run on a pool of subjects have shown how a **Normal** distribution is a good fit to capture the variability of rates in the fatigue model [42]. Therefore, we assume that each λ_i (resp., ρ_i) is a sample of distribution $\mathcal{N}(\mu_\lambda, \sigma_\lambda^2)$ (resp., $\mathcal{N}(\mu_\rho, \sigma_\rho^2)$) whose mean and variance depend on the specific subject's characteristics.

All SHA modeling humans feature real-valued variable $F \in W$, capturing physical fatigue, and a dense counter $\mathsf{f} \in V_{\mathrm{dc}}$ capturing the digital counterpart of F. Besides physical fatigue, for each human, suitable sensors also periodically (with frequency $\frac{1}{\mathsf{T}_{\mathsf{poll}}}$) refresh their position within the building. Dense counters model the position $\mathsf{h}_{\mathsf{pos}_\mathsf{x}}$ and $\mathsf{h}_{\mathsf{pos}_\mathsf{y}}$ capturing a pair of Cartesian coordinates. Fatigue and position are periodically updated for the entire duration of the interaction between the human and the robot.

The portion of SHA—referred to as $\langle \mathsf{op} \rangle$_pub$_h$ and shown in Fig. 4—modeling the update of periodic sensors readings is present in all human states. Exit and entry points are represented with "ports." Specifically, the edges marking the start, stop, and failure of an $\langle \mathsf{op} \rangle$_pub$_h$ instance are marked by symbols ▶, ■, ×, respectively. We remark that ports are not part of the formalism but merely a visualization expedient for the edges entering and leaving the subautomaton. All instances of $\langle \mathsf{op} \rangle$_pub$_h$ feature ordinary location $h_{\langle \mathsf{op} \rangle}$ and committed locations h_{pub_1}, h_{pub_2}. Significant variables from this modeling pattern are summed up in Table 1.

Table 1 Significant variables from the $\langle\mathsf{op}\rangle$_pub$_h$ modeling pattern

Name	Description
$\langle\mathsf{op}\rangle$	Operational state label
$\mathsf{T}_{\mathsf{poll}}$	Sensor data sharing period
t_{upd}	Clock measuring the time elapsed between two consecutive sensor data readings
F	Real-valued variable modeling human fatigue
λ_i	Fatigue rate for cycle i
ρ_i	Recovery rate for cycle i

Location $h_{\langle\mathsf{op}\rangle}$ represents the specific human operational state (e.g., *walking* or *standing*). Therefore, it is endowed with suitable flow conditions $\mathcal{F}(h_{\langle\mathsf{op}\rangle})$ constraining the evolution of real-valued variable F, corresponding to the derivative of Eq. 1 (recovery) (shown in Eq. 2) if $h_{\langle\mathsf{op}\rangle}$ is a recovery state or Eq. 1 (fatigue) (shown in Eq. 3), otherwise.

$$\dot{F} = f(\rho, t) = -F(\mathsf{t}_{i-1})\rho e^{-\rho(t-\mathsf{t}_{i-1})} \tag{2}$$

$$\dot{F} = g(\lambda, t) = F(\mathsf{t}_{i-1})\lambda e^{-\lambda(t-\mathsf{t}_{i-1})} \tag{3}$$

Location $h_{\langle\mathsf{op}\rangle}$ features probability distribution $\mathcal{D}(h_{\langle\mathsf{op}\rangle})$ describing the random parameter (either ρ_i or λ_i) in the corresponding agent's state. Upon entering an $\langle\mathsf{op}\rangle$_pub$_h$ instance, update $\xi_{\mathrm{rand},\langle\mathsf{op}\rangle}$ draws a new sample from $\mathcal{D}(h_{\langle\mathsf{op}\rangle})$ and assigns it to λ_i if $h_{\langle\mathsf{op}\rangle}$ is a fatigue state, otherwise to ρ_i.

In all instances of $\langle\mathsf{op}\rangle$_pub$_h$, invariant $\mathcal{I}(h_{\langle\mathsf{op}\rangle})$ includes condition $t_{\mathrm{upd}} \leq \mathsf{T}_{\mathsf{poll}}$, which, in conjunction with guard condition $t_{\mathrm{upd}} \geq \mathsf{T}_{\mathsf{poll}}$ on the edge to h_{pub_1}, ensures that the SHA switches to h_{pub_1} every $\mathsf{T}_{\mathsf{poll}}$ time instants (i.e., when a new sensor measurement is available). When time $\mathsf{T}_{\mathsf{poll}}$ has elapsed between two sensor readings, t_{upd} is reset, and dense counters representing fatigue and position are updated by $\xi_{\langle\mathsf{op}\rangle}$ (consistently with the corresponding flow condition) and shared with the orchestrator through channels $\mathtt{p}_{\mathrm{ftg}}$ and $\mathtt{p}_{\mathrm{pos}}$, respectively.

4 Formal modeling of human behavior in HRI scenarios

This section illustrates how we model human behavior extended with errors through SHA. First, we present the set of phenotypes of erroneous human behavior, how we have mapped the phenotypes to the service setting, and how we have modeled the mapped phenotypes as SHA add-ons. Second, we introduce the extensions of the SHA modeling human–robot interaction patterns presented in [37, 39] with the erroneous behavior models.

4.1 Phenotypes of erroneous human behavior

The issue of defining and categorizing manifestations of human *erroneous* behavior has been largely investigated in the field of human–computer interaction. Hollnagel's human error taxonomy [27] is among the best-established works in the field addressing how unexpected human behavior can cause the interaction with a machine to fail. Specifically, [27] focuses on the cases in which although the interaction plan is adequate, the performed actions stray from the plan: Such actions are referred to as *human errors*. Although the form and frequency of human errors cannot be fully predicted, they are bound to take place in a complex interactive system and tend to occur in patterns. More specifically, error patterns consist of a **phenotype**, i.e., the manifestation of the erroneous action (e.g., a user failing to press a button in time), and a **genotype**, i.e., the cognitive process that causes the erroneous action (e.g., forgetting the intention of pressing the button). As our work does not capture the cognitive sources of human actions but their observations, throughout the paper, we focus on phenotypes, while genotypes are investigated in [26].

The taxonomy proposed by Hollnagel features four macro-categories of human errors, referred to as *error modes* in Table 2: actions in the wrong place (i.e., the position of the action within the sequence is not correct), at the wrong time (i.e., the timing of the action is not as planned), of the wrong type (i.e., the action is not planned but does not disrupt the plan), not included in the current plan (i.e., the action is not in the planned sequence). An error mode groups one or multiple phenotypes, each capturing a deviation (i.e., the *error*) with respect to a *plan*, where a plan is intended as *a representation of both a goal […] and the possible actions required to achieve it* [48]. For each phenotype, Table 2 reports action sequences displaying the corresponding erroneous behavior with respect to a planned sequence of generic actions indicated with symbols [A, B, C, D, E]. When timing is necessary to characterize the phenotype, timestamps of the form t_i indicating the expected time of occurrence of an action are also reported. Notation $(\mathsf{t}_\mathsf{i}, \perp)$ indicates that *no* action takes place at time t_i contrary to what the plan envisages, while symbol $\downarrow$ indicates an unexpected termination of the sequence of actions.

The formal model in our framework captures human behavior while interacting with a robot, and the possible interactions are referred to as *patterns*. Since each pattern represents the service requested by the human to the robot, the *goal* of a pattern is to provide the service and conclude the interaction successfully. The condition determining whether a service is completed successfully is specific to the pattern and indicated in the following as $\gamma_{i,\mathrm{scs}} \in \Gamma(W)$, where $i \in [1, \mathsf{N}_\mathsf{h}]$ identifies a specific human (equivalently, a specific pattern, as each SHA modeling a human is an instance

Table 2 Phenotypes of human erroneous behavior identified by Hollnagel's taxonomy [27]

Error mode (action:)	Phenotype	Description	Action sequence
In the wrong place	Repetition	An action is a repetition of the previous contiguous one.	A, B, C,C, D, E
	Reversal	The next two actions in the expected sequence are reversed.	A, B,D, C, E
In the wrong place/at the wrong time	Omission	An action has been omitted from the sequence.	(t_1, A), (t_2, B), $(t_3, \perp)$, (t_4, D), (t_5, E)
At the wrong time	Delay	An action is performed, but not when required.	(t_1, A), (t_2, B), $(t_3, \perp)$, (t'_3, C), (t_4, D), (t_5, E)
	Premature Action	An action is performed, but not when expected.	(t_1, A), (t_2, B), (t'_3, C), $(t_3, \perp)$, (t_4, D), (t_5, E)
Of the wrong type	Replacement	An action is performed as a substitute of another, but the two are functionally equivalent (i.e., C is equivalent to C′).	A, B, C', D, E
Not included in current plan	Insertion	An action is performed that is not in the original plan, but does not disrupt it.	A, B, Y, C, D, E
	Intrusion	An action is performed that is not in the original plan and causes disruption. (The goal cannot be achieved.)	A, B, Y ↓

For each phenotype, the table reports the corresponding error mode, an informative description, and an example of an action sequence displaying the erroneous behavior (highlighted in RED) of a single pattern) and $\mathsf{N_h} \in K$ is the total number of humans involved in the scenario. The goal of a pattern i is then expressed through formula $\diamond_{\leq\tau}(\gamma_{i,\mathrm{scs}})$, where τ is the time bound of the SMC experiment, as explained in Sect. 3.

As described in Sect. 3.2, SHA modeling human behavior feature multiple instances of the $\langle\mathsf{op}\rangle\_\mathsf{pub}_h$ subautomaton, each capturing an operational state, such as walking or standing. Similarly, the SHA modeling the robot features multiple instances of the $\langle\mathsf{op}\rangle\_\mathtt{pub}_{\langle\mathsf{id}\rangle}$ subautomaton presented in [38, Section IV] modeling the operational state of the robot. To complete a service, the human and the robot perform a sequence of actions, i.e., the *plan* in our framework. The *occurrence* of an action in our modeling approach is captured through a *change* in the operational state (i.e., $\langle\mathsf{op}\rangle\_\mathsf{pub}_h$ instance) of the SHA modeling the human or a change of location in the SHA modeling the robot. Given a SHA involved in a plan and indicating its location at a given time t as l, an atomic element $(\mathsf{t_i}, l')$ of the plan such that $\mathsf{t_i} > t$ holds indicates that the SHA should switch from l to l' at time $\mathsf{t_i}$. On the other hand, element $(\mathsf{t_i}, \perp)$ indicates that the change of operational state planned at time $\mathsf{t_i}$ does not take place due to a human error, thus, the SHA modeling the human remains in its current state. For example, let us consider the Human-Follower pattern. The goal (expressed in informal terms) is that the human follows the robot to a certain location (not known beforehand by the human). To this end, the plan in informal terms is the following: *robot starts moving*, *human starts walking*, *robot stops moving*, *human stops walking*. The corresponding plan reporting the operational state change is shown in sequence (4). (The human's initial operational state is $\langle\mathsf{stand}\rangle\_\mathsf{pub}_h$, while the robot starts in r_{idle}.) Note that, to ease the distinction between changes related to the robot and to the human when presenting sequences, we report the full $\langle\mathsf{op}\rangle\_\mathsf{pub}_h$ label for humans and only the label of the ordinary location within the $\langle\mathsf{op}\rangle\_\mathtt{pub}_{\langle\mathsf{id}\rangle}$ subautomaton for the robot (i.e., r_{start} and r_{stop}).

$$\begin{aligned}&[(\mathsf{t}_1, r_{\mathrm{start}}), (\mathsf{t}_2, \langle\mathsf{walk}\rangle\_\mathsf{pub}_h),\\ &(\mathsf{t}_3, r_{\mathrm{stop}}), (\mathsf{t}_4, \langle\mathsf{stand}\rangle\_\mathsf{pub}_h)].\end{aligned} \tag{4}$$

In reality, human behavior in a service setting is barely constrained, and the decision-making process is highly susceptible to free will; therefore, numerous deviations from the plan are possible. In our framework, SHA modeling human behavior capture this aspect by embedding a formalization of human errors. In the following, we present the developed SHA add-ons modeling the phenotypes as summarized in Table 3, providing examples of the corresponding erroneous behaviors within the domain of our framework. Table 4 summarizes the significant variables from each add-on, whose role is explained in detail throughout the section. Specifically, for each add-on, we present its features and show the corresponding SHA portion modeling the standard (i.e., non-erroneous) behavior. We remark that we indicate as add-ons portions of SHA (thus, sub-tuples of the one defined in Definition 1) that can be flexibly incorporated into other SHA. In more detail, all add-ons consist of patterns of connections between multiple instances of the $\langle\mathsf{op}\rangle\_\mathsf{pub}_h$ component (see Fig. 4). Therefore, pre-existing SHA modeling human behavior and featuring a fixed set of $\langle\mathsf{op}\rangle\_\mathsf{pub}_h$ components (i.e., the operational states capturing the human's fatigue state) can be extended by incorporating add-ons. (A specific example is presented in detail in Sect. 4.2.1.) This feature increases the extensibility of the modeling approach for two reasons: If new add-ons are developed in the future, these can be easily plugged into existing human–robot interaction patterns; if new human–robot interaction patterns are developed in the future, these can be easily extended with the developed add-ons.

Table 3 Mapping between the developed add-ons and the phenotypes identified by Hollnagel [27]

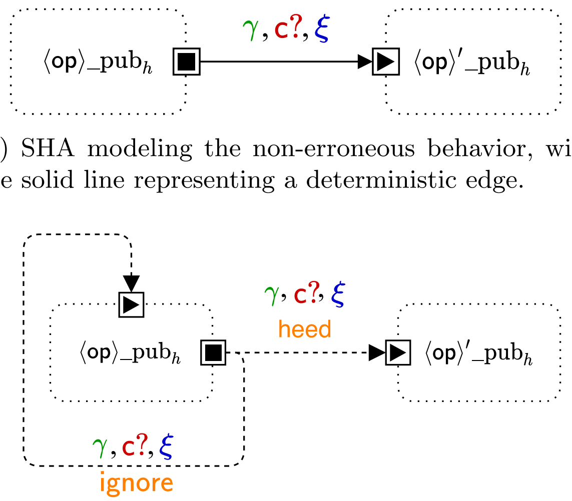
Add-On	Phenotype(s)
Heed/Ignore	Delay, →Intrusion
Free Will	Premature Action, Reversal, Insertion
Timer Expired	Omission
Safety Violation	Insertion, Intrusion, Premature Action
Critical Status	Intrusion

Given the *stochastic* nature of the employed formalism, it is not possible for the developed add-ons to be purely non-deterministic like the original phenotypes. However, a probabilistic formalization is necessary and described in more detail when introducing the various add-ons in the next sections. Indeed, quantifying the probability of human errors' manifestations is a long-standing approach in probabilistic Human Reliability Analysis techniques such as CREAM [28], HEART [56], THERP [53], and THEA [46].

As per Table 3, developed add-ons cover 6 out of 8 phenotypes from Hollnagel's taxonomy. The reason why Repetition and Replacement are not supported is due to the range of human actions currently covered by the modeling approach. In the first case, supported actions cannot physically or logically be repeated: For instance, the human cannot *start* walking twice in a row since they either stop and restart (counting as two separate actions) or simply keep walking. Concerning Replacement, the approach does not include sets of *functionally equivalent* actions (e.g., standing and sitting, or walking and running). Therefore, performing a substitute of the correct action is not supported. Introducing functionally equivalent action sets will be investigated in future work, as well as an extension of the available set of add-ons to model the Replacement error.

Table 4 Summary of the variables from the developed add-ons

Name	Add-On	Description
heed/ignore	Heed/Ignore	Probability weights for the human heeding/ignoring the robot's instructions
fw	Free Will	Variable governing the human's haphazard decisions
$\mathsf{FW}_{\mathsf{th}}$	Free Will	Threshold to trigger the human's haphazard decisions
$\mathsf{FW}_{\mathsf{max}}$	Free Will	Maximum fw value
φ	Timer Expired	Real-valued variable modeling the progress of the human task
TE	Timer Expired	Threshold to consider the time to complete the task expired
t_{exp}	Timer Expired	Clock measuring time elapsed since the start of the task
$\mathrm{h}_{\mathrm{pos}}$	Timer Expired	Human's coordinates within the layout
target	Timer Expired	Destination of the pattern
δ	Timer Expired	Allowance factor for the maximum duration of the task
$\mathrm{r}_{\mathrm{pos}}$	Safety Violation	Robot's coordinates within the layout
$\mathsf{d}_{\mathsf{crit}}$	Safety Violation	Maximum distance allowed between human and robot
λ_{safe}	Safety Violation	Exponential rate for the probability of the human leaving the safe state
p_{d}	Critical Status	Time-dependent factor of the probability of unexpected accidents
FS	Critical Status	Constant factor of the probability of unexpected accidents


$\gamma, \mathsf{c}?, \xi$
$\langle\mathsf{op}\rangle\_\mathrm{pub}_h$
$\langle\mathsf{op}\rangle'\_\mathrm{pub}_h$
$\gamma, \mathsf{c}?, \xi$
heed
$\langle\mathsf{op}\rangle\_\mathrm{pub}_h$
$\langle\mathsf{op}\rangle'\_\mathrm{pub}_h$
$\gamma, \mathsf{c}?, \xi$
ignore

(a) SHA modeling the non-erroneous behavior, with the solid line representing a deterministic edge.

(b) Heed/Ignore SHA add-on: as in Fig. 3, dashed lines represent probabilistic edges, with weights highlighted in orange.

Fig. 5 SHA showing the standard behavior, and the Heed/Ignore add-on. Color-coding is the same as Fig. 3 (color figure online)

4.1.1 Heed/Ignore Add-On

The Heed/Ignore SHA add-on formally models the situation in which the orchestrator issues an instruction for the human, and the human **ignores** it and protracts the action they were previously performing. We do not further investigate or formally model whether ignoring the instruction is intentional since, as previously discussed, the formal model captures the *manifestation* of the erroneous behavior and not its cognitive source.

The standard behavior is captured by the SHA in Fig. 5a. Subautomaton $\langle\mathsf{op}\rangle\_\mathrm{pub}_h$ represents the current state of the human (e.g., *standing* or *walking*), while $\langle\mathsf{op}\rangle'\_\mathrm{pub}_h$ represents the following state in the sequence. The switch from $\langle\mathsf{op}\rangle\_\mathrm{pub}_h$ to $\langle\mathsf{op}\rangle'\_\mathrm{pub}_h$ occurs through an edge, which fires when an instruction is *sent* through channel $\mathsf{c} \in C$. (Thus, the edge is labeled with c?.) Optionally, the edge may also be labeled with guard condition γ and update ξ.

Consider, for instance, the running example of the action sequence envisaged by the HumanFollower pattern, where the orchestrator fires an instruction through channel $\mathtt{cmd\_h}_{\mathrm{start}}$ to instruct the human to start walking and follow the robot. In this case, the human acts *erroneously* as they do not abide by the instruction, which is captured by the SHA add-on in Fig. 5b. The deterministic edge from $\langle\mathsf{op}\rangle\_\mathrm{pub}_h$ to $\langle\mathsf{op}\rangle'\_\mathrm{pub}_h$ in Fig. 5a is changed into two *probabilistic* edges with weights heed, ignore $\in K$, and the same labels γ, c?, and ξ as the original edge. The edge with weight heed reaches $\langle\mathsf{op}\rangle'\_\mathrm{pub}_h$, thus capturing the human following the instruction and changing their state when c? fires. The edge with weight ignore is a self-loop on $\langle\mathsf{op}\rangle\_\mathrm{pub}_h$, capturing the human ignoring the instruction and staying in the state modeled by $\langle\mathsf{op}\rangle\_\mathrm{pub}_h$ when c? fires. The observed behavior when introducing this add-on is that the human performs the required action *when* instructed by the orchestrator with probability $\mathsf{p} = \mathsf{heed}/(\mathsf{heed} + \mathsf{ignore})$ and does *not* perform the required action with probability $1 - \mathsf{p}$.

We remark that even if the SHA modeling the human takes the ignore edge, an event is still received through channel c due to label c?. Nevertheless, the behavior of the SHA network that is *effectively* observed is that the SHA modeling the human does *not* initiate the action semantically associated with channel c. When reporting examples of erroneous action sequences (also for upcoming add-ons), we recall that the orchestrator checks the state of the system and issues one or multiple instructions, if necessary, every $\mathsf{T_{int}} \in K$ time units. Therefore, referring to the HumanFollower running example, if the human erroneously behaves according to the Heed/Ignore add-on and no other error occurs throughout the pattern, the observed action sequence is shown in sequence (5), where $\mathsf{k} \in \mathbb{N}$ is the number of times the self-loop on $\langle \mathsf{op} \rangle\_\mathrm{pub}_h$ is taken in favor of the edge to $\langle \mathsf{op} \rangle'\_\mathrm{pub}_h$.

$$\begin{aligned}&[(\mathsf{t}_1, r_{\mathrm{start}}), (\mathsf{t}_2, \bot), (\mathsf{t}_2 + \mathsf{T_{int}}, \bot), \dots,\\ &(\mathsf{t}_2 + \mathsf{kT_{int}}, \bot), (\mathsf{t}_2 + (\mathsf{k}+1)\mathsf{T_{int}}, \langle \mathsf{walk} \rangle\_\mathrm{pub}_h),\\ &(\mathsf{t}_3, r_{\mathrm{stop}}), (\mathsf{t}_4, \langle \mathsf{stand} \rangle\_\mathrm{pub}_h)]\end{aligned} \tag{5}$$

With $\mathsf{k} = 1$, we obtain a Delay phenotype (see Table 2). All sequences observed with $\mathsf{k} > 1$ are an iteration of said phenotype, resulting in a *longer* delay. The probability of choosing the ignore edge k times is $(1-\mathsf{p})^{\mathsf{k}}$. It is possible—in theory—that the heed edge is *never* chosen in favor of the ignore self-loop: The probability of this happening (i.e., $(1-\mathsf{p})^{\mathsf{k}}$ with $\mathsf{k} \to \infty$) tends to 0 if $\mathsf{p} < 1$ holds—i.e., if ignore is greater than 0. The action sequence observed in this corner case (marked by symbol → in Table 3) is given in sequence (6).

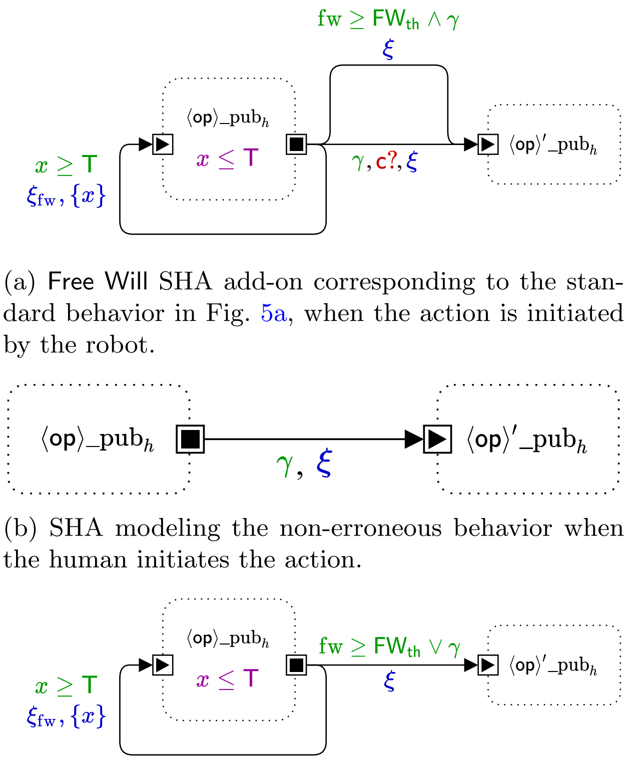
$$[(\mathsf{t}_1, r_{\mathrm{start}}), (\mathsf{t}_2, \bot), \dots] \tag{6}$$

Since the goal is never reached, this constitutes a special case of Intrusion with $Y = \bot$ (see Table 2).

4.1.2 Free Will Add-On

The Free Will SHA add-on, shown in Fig. 6, captures the situation in which the human performs an action **independently** of the orchestrator's instructions (if the robot initiates the action) or when the system does **not** meet the pre-conditions for the actions (if the human initiates the action). In both cases, the manifestation of this erroneous behavior depends on dense counter fw that approximates the free will phenomenon through a random distribution [11] and whose underlying mechanism is explained below.

If the robot initiates the action, the planned behavior is shown in Fig. 5a and described in Sect. 4.1.1: The orchestrator instructs the human to perform the next required action through channel c, triggering them to switch to $\langle \mathsf{op} \rangle'\_\mathrm{pub}_h$. The SHA add-on modeling the erroneous behavior (shown in Fig. 6a) features an *additional* edge between $\langle \mathsf{op} \rangle\_\mathrm{pub}_h$ and $\langle \mathsf{op} \rangle'\_\mathrm{pub}_h$ with update ξ, no channel label, and whose guard is a conjunction between the original guard γ and condition $\mathrm{fw} \geq \mathsf{FW_{th}}$ (explained in detail below). The purpose of the self-loop $\langle \mathsf{op} \rangle\_\mathrm{pub}_h$, for both cases, is also explained below contextually to the update of variable fw. The firing of this edge represents the human erroneously starting the action represented by channel c when the orchestrator has issued no instruction.



$\mathrm{fw} \geq \mathsf{FW_{th}} \wedge \gamma$
ξ
$\langle \mathsf{op} \rangle\_\mathrm{pub}_h$
$x \leq \mathsf{T}$
$x \geq \mathsf{T}$
$\xi_{\mathrm{fw}}, \{x\}$
$\gamma, \mathsf{c}?, \xi$
$\langle \mathsf{op} \rangle'\_\mathrm{pub}_h$
$\langle \mathsf{op} \rangle\_\mathrm{pub}_h$
γ, ξ
$\langle \mathsf{op} \rangle'\_\mathrm{pub}_h$
$\langle \mathsf{op} \rangle\_\mathrm{pub}_h$
$x \leq \mathsf{T}$
$\mathrm{fw} \geq \mathsf{FW_{th}} \vee \gamma$
ξ
$x \geq \mathsf{T}$
$\xi_{\mathrm{fw}}, \{x\}$
$\langle \mathsf{op} \rangle'\_\mathrm{pub}_h$

(a) Free Will SHA add-on corresponding to the standard behavior in Fig. 5a, when the action is initiated by the robot.

(b) SHA modeling the non-erroneous behavior when the human initiates the action.

(c) Free Will SHA add-on when the action is initiated by the human.

Fig. 6 Free Will SHA add-on, color-coded as in Fig. 3 (color figure online)

In the second case (i.e., the action initiated by the human), the standard behavior is shown in Fig. 6b. Subautomata $\langle \mathsf{op} \rangle\_\mathrm{pub}_h$ and $\langle \mathsf{op} \rangle'\_\mathrm{pub}_h$ represent the current and the next operational states. The switch between the two subautomata does not depend on a robot instruction (there is no channel label on the edge) but is entirely up to the human to perform the action when a certain condition γ holds. The modeling approach assumes that no SHA other than the orchestrator fire an event through a channel (thus, with label $\mathsf{c}!$) representing the start of an action. The reason is that, in the real system, the human cannot *actively* signal the start of each action for practicality. When the human initiates an action, the orchestrator infers from sensor measurements that such an event occurred (e.g., that the human started moving because their position changed). The erroneous behavior, shown in Fig. 6c, captures the human potentially performing the action even if the required pre-conditions (represented by γ) do not hold, due to guard $\mathrm{fw} \geq \mathsf{FW_{th}} \vee \gamma$.

The mechanism determining free will, i.e., how new values are assigned to dense counter fw, is stochastic. Let $x \in X$

be a clock of the SHA the add-on is applied to, subautomaton $\langle \mathsf{op}\rangle\_\mathrm{pub}_h$ is then endowed with invariant $x \leq \mathsf{T}$, where $\mathsf{T} \in K$ is a constant. Both in the Heed/Ignore and Free Will add-ons, subautomata $\langle \mathsf{op}\rangle\_\mathrm{pub}_h$ and $\langle \mathsf{op}\rangle'\_\mathrm{pub}_h$ may be endowed with further flow conditions, probability distributions, and invariants—i.e., $\mathcal{F}(h_{\langle \mathsf{op}\rangle})$, $\mathcal{D}(h_{\langle \mathsf{op}\rangle})$, $\mathcal{I}(h_{\langle \mathsf{op}\rangle})$, $\mathcal{F}(h_{\langle \mathsf{op}\rangle'})$, $\mathcal{D}(h_{\langle \mathsf{op}\rangle'})$, and $\mathcal{I}(h_{\langle \mathsf{op}\rangle'})$ can be non-empty. Nevertheless, since they do not directly impact the erroneous behavior like invariant $x \leq \mathsf{T}$, these labels are not shown in Fig. 5 nor Fig. 6 to ease the visualization of the add-ons' essential elements.

Subautomaton $\langle \mathsf{op}\rangle\_\mathrm{pub}_h$ features a self-loop with guard $x \leq \mathsf{T}$ and update ξ_{fw}. The joint presence of the invariant and the guard condition enforces update ξ_{fw} to be executed every T time units. Simultaneously, clock x is reset (indicated as $\{x\}$) to ensure that the invariant holds after the self-loop fires. Update ξ_{fw} assigns a new value to dense counter $\mathrm{fw} \in V_{\mathrm{dc}}$. Specifically, the update yields a new sample of Uniform distribution $\mathcal{U}_{[0,\mathsf{FW}_{\max})}$, where $\mathsf{FW}_{\max} \in K$ is a numerical constant. Guard $\mathrm{fw} \geq \mathsf{FW}_{\mathrm{th}}$ on the Free Will edge (in conjunction or disjunction with γ in Fig. 6a, c, respectively) ensures that the erroneous behavior occurs only if the last value drawn for variable fw belongs to range $[\mathsf{FW}_{\mathrm{th}}, \mathsf{FW}_{\max})$ where $\mathsf{FW}_{\mathrm{th}} \in K$ is a constant such that $\mathsf{FW}_{\mathrm{th}} \leq \mathsf{FW}_{\max}$ holds.

The HumanFollower pattern is eligible for the add-on in Fig. 6a since actions are initiated by the robot. While the intended plan is reported in sequence (4), if the Free Will edge fires (thus, the erroneous behavior occurs) at time $\mathsf{t}_2' < \mathsf{t}_2$, the observed actions are those shown in sequence (7).

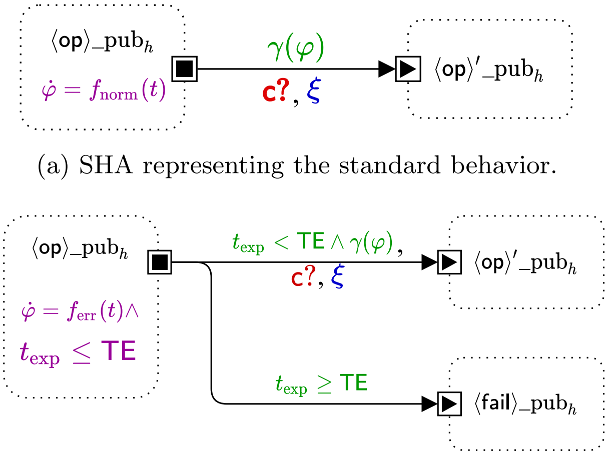
$$\begin{aligned}&[(\mathsf{t}_1, r_{\mathrm{start}}), (\mathsf{t}_2', \langle \mathsf{walk}\rangle\_\mathrm{pub}_h),\\ &(\mathsf{t}_2, \bot), (\mathsf{t}_3, r_{\mathrm{stop}}), (\mathsf{t}_4, \langle \mathsf{stand}\rangle\_\mathrm{pub}_h)]\end{aligned} \tag{7}$$

In this case, sequence (7) reports the switch to $\langle \mathsf{walk}\rangle\_\mathrm{pub}_h$ (i.e., the human starting to walk) at time t_2' out of free will even if no event is fired through channel $\mathtt{cmd\_h}_{\mathrm{start}}$. Therefore, in our framework, the Free Will add-on realizes the Premature Action phenotype if it involves the correct action according to the sequence (like starting to walk, in this example), but is not performed at the expected time. For example, the human may start walking during the $\mathsf{T}_{\mathrm{int}}$ time range in which the orchestrator is processing data.

A possible corner case of this erroneous behavior is obtained by decreasing t_2' to the point that $\mathsf{t}_2' < \mathsf{t}_1$ holds. In this case, the observed actions are given in sequence (8), whose timestamps are not shown as they are not necessary to identify the error.

$$[\langle \mathsf{walk}\rangle\_\mathrm{pub}_h, r_{\mathrm{start}}, r_{\mathrm{stop}}, \langle \mathsf{stand}\rangle\_\mathrm{pub}_h] \tag{8}$$

Therefore, the realized phenotype, in this case, is a Reversal. We remark that sequence (8) is feasible in our framework since the orchestrator still issues instruction $\mathtt{cmd\_h}_{\mathrm{start}}$ *after* $\mathtt{cmd\_r}_{\mathrm{start}}$, but since the human is already walking, no response to such instruction is observed on the human's side.



$\langle \mathsf{op}\rangle\_\mathrm{pub}_h$
$\dot{\varphi} = f_{\mathrm{norm}}(t)$
$\gamma(\varphi)$
c?, ξ
$\langle \mathsf{op}\rangle'\_\mathrm{pub}_h$
$\langle \mathsf{op}\rangle\_\mathrm{pub}_h$
$\dot{\varphi} = f_{\mathrm{err}}(t)\wedge$
$t_{\mathrm{exp}} \leq \mathsf{TE}$
$t_{\mathrm{exp}} < \mathsf{TE} \wedge \gamma(\varphi)$,
c?, ξ
$\langle \mathsf{op}\rangle'\_\mathrm{pub}_h$
$t_{\mathrm{exp}} \geq \mathsf{TE}$
$\langle \mathsf{fail}\rangle\_\mathrm{pub}_h$

(a) SHA representing the standard behavior.

(b) SHA representing the Timer Expired add-on, color-coded as in Fig. 3.

Fig. 7 SHA representing a standard behavior and its version with the Timer Expired add-on representing the erroneous behavior, both color-coded as in Fig. 3 (color figure online)

The third and final manifestation of the Free Will add-on is the human erroneously performing an action that is not envisaged by sequence (4), irrespectively of the time at which it is performed. For example, the human may abruptly stop walking while following the robot, resulting in sequence (9). (Timestamps are not reported.)

$$\begin{aligned}&[r_{\mathrm{start}}, \langle \mathsf{walk}\rangle\_\mathrm{pub}_h, \langle \mathsf{stand}\rangle\_\mathrm{pub}_h,\\ &\langle \mathsf{walk}\rangle\_\mathrm{pub}_h, r_{\mathrm{stop}}, \langle \mathsf{stand}\rangle\_\mathrm{pub}_h]\end{aligned} \tag{9}$$

This case realizes an Insertion phenotype since the human stopping is not expected but still allows the agents to reach the goal. In this situation, the orchestrator has to instruct the human to start walking again after the error occurs. Therefore, the additional $\mathtt{cmd\_h}_{\mathrm{start}}$ action is not expected, but it constitutes a *response* of the system to the error. (Thus, it is highlighted in blue and not in red.)

The add-on in Fig. 6c, which captures actions initiated by the human, realizes the same phenotypes, with the difference that the point of reference is not the firing of channel $\mathtt{c}$ but condition γ being verified based on the system's state.

4.1.3 Timer Expired Add-On

The Timer Expired SHA add-on captures the human extremely delaying the completion of a task whose progress they are in charge of, to the point of being considered non-responsive. An example is the HumanLeader pattern, which is the dual case of the HumanFollower, in that the human is in charge of leading the robot to a certain destination. Similarly, the

HumanApplicant pattern (described in detail in Sect. 4.2.1) features the human performing an action with the support of the robot, such as administering a treatment. In such cases, if the human performs impeccably, the action (i.e., walking or treating the patient) ends within a reasonable amount of time. However, unexpected time losses or the incumbency of an emergency may prevent the human from completing the task, leading the robot to consider them non-responsive and the service failed.

The standard behavior is shown in Fig. 7a. The SHA features two subautomata $\langle\mathsf{op}\rangle\_\mathsf{pub}_h$ and $\langle\mathsf{op}\rangle'\_\mathsf{pub}_h$ representing (as in Sects. 4.1.1, 4.1.2) the current and the next operational state envisaged by the plan. Real-valued variable $\varphi \in W$ models the progress of the human task while in $\langle\mathsf{op}\rangle\_\mathsf{pub}_h$. The evolution of φ with time (e.g., the distance to the destination decreasing) is constrained by flow condition $f_{\text{norm}}(t)$. The switch from $\langle\mathsf{op}\rangle\_\mathsf{pub}_h$ to $\langle\mathsf{op}\rangle'\_\mathsf{pub}_h$ is realized through a solid edge with guard $\gamma(\varphi)$, update ξ, and channel c: If the human initiates the switch, the channel is replaced by the internal action. Guard $\gamma(\varphi)$ is a condition on the value of φ evaluating to true when the task is complete.

The erroneous behavior modeled by the Timer Expired add-on, shown in Fig. 7b, captures the situation in which the progress of the task performed by the human is excessively delayed. The evolution of variable φ is, therefore, constrained by a different flow condition ($f_{\text{err}}(t)$ in Fig. 7b). While function $f_{\text{norm}}(t)$ models the human behaving normally, function $f_{\text{err}}(t)$ is such that condition $\gamma(\varphi)$ (capturing the completion of the task) may not be verified within a maximum time bound, corresponding to constant $\mathsf{TE} \in K$. A concrete example of how $f_{\text{err}}(t)$ is implemented is given in Sect. 4.2.1 when describing in detail the HumanApplicant pattern. Subautomaton $\langle\mathsf{op}\rangle\_\mathsf{pub}_h$ is further endowed with invariant $t_{\text{exp}} \leq \mathsf{TE}$, where $t_{\text{exp}} \in X$ is a clock that is reset upon entering $\langle\mathsf{op}\rangle\_\mathsf{pub}_h$. If time bound TE is exceeded, the SHA switches to subautomaton $\langle\mathsf{fail}\rangle\_\mathsf{pub}_h$. The edge from $\langle\mathsf{op}\rangle\_\mathsf{pub}_h$ to $\langle\mathsf{fail}\rangle\_\mathsf{pub}_h$ is labeled with guard condition $t_{\text{exp}} \geq \mathsf{TE}$, which, in conjunction with the invariant on $\langle\mathsf{op}\rangle\_\mathsf{pub}_h$, ensures that the transition occurs if and only if $t_{\text{exp}} = \mathsf{TE}$ holds.

For each pattern eligible for this add-on, time bound TE is estimated based on the characteristics of the human and the requested service. Equation 10 shows an example of how the value of TE is calculated in patterns requiring the human to move to a certain destination when initiating the movement is up to the human. In this case, variable φ corresponds to the distance between the human and the destination, and the task completion (i.e., condition $\gamma(\varphi)$) captures the distance being equal to 0. Function dist computes the distance between two points accounting for fixed obstacles (e.g., walls), $\mathsf{h}_{\text{pos}} \in W$ is the Cartesian coordinate pair representing the human's position within the layout, $\mathsf{target} \in K$ is the Cartesian coordinates pair representing the destination of the service, $\mathsf{v} \in K$ is the human's walking speed, and $\delta \in K$ is the *allowance factor*.

$$\mathsf{TE} = \frac{\mathsf{dist}(\mathsf{h}_{\text{pos}}(0), \mathsf{target})}{\mathsf{v}} \cdot (1 + \delta) \tag{10}$$

The ratio between the distance to be covered (thus, the distance between the human's starting position $\mathsf{h}_{\text{pos}}(0)$ and the destination) and the walking speed represents the *expected* duration of the service in ordinary conditions, whereas δ determines how much the expected duration can be exceeded for the human to be considered non-responsive. Therefore, the higher the value of δ, the lower the likelihood of this erroneous behavior.

Let us refer to a [HumanLeader, HumanFollower] service sequence to illustrate the manifestation of this erroneous behavior. The expected plan for the two services is shown in sequence (11), where the first 4 elements constitute the HumanLeader plan, while the last 4 constitute the HumanFollower plan.

$$\begin{aligned}&[(\mathsf{t}_1, \langle\mathsf{walk}\rangle\_\mathsf{pub}_h), (\mathsf{t}_2, r_{\text{start}}), (\mathsf{t}_3, \langle\mathsf{stand}\rangle\_\mathsf{pub}_h),\\ &(\mathsf{t}_4, r_{\text{stop}}), (\mathsf{t}_5, r_{\text{start}}), (\mathsf{t}_6, \langle\mathsf{walk}\rangle\_\mathsf{pub}_h),\\ &(\mathsf{t}_7, r_{\text{stop}}), (\mathsf{t}_8, \langle\mathsf{stand}\rangle\_\mathsf{pub}_h)]\end{aligned} \tag{11}$$

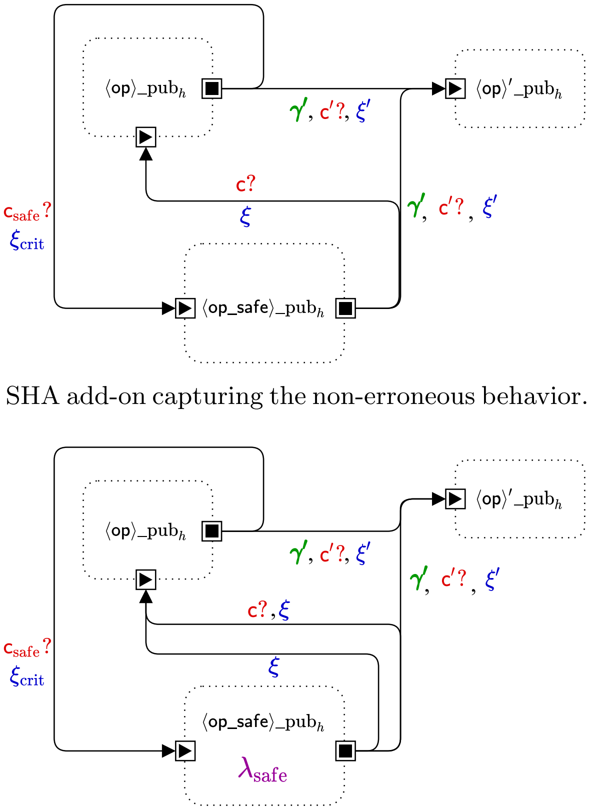
The erroneous behavior modeled by the Timer Expired add-on may occur, while the human is *leading* the robot to the destination (captured by parameter target, as in Sect. 4.1.3) and they fail to reach it within time TE. In this case, the system behaves as in sequence (12).

$$\begin{aligned}&[(\mathsf{t}_1, \langle\mathsf{walk}\rangle\_\mathsf{pub}_h), (\mathsf{t}_2, r_{\text{start}}), (\mathsf{t}_3, \bot), \ldots,\\ &(\mathsf{t}_3 + \mathsf{TE}, \bot), (\mathsf{t}'_4, r_{\text{stop}}), (\mathsf{t}_5, r_{\text{start}}), \ldots]\end{aligned} \tag{12}$$

We remark that, in sequence (12), t_3 is the expected duration of the task, and $\mathsf{t}'_4 \geq \mathsf{t}_3 + \mathsf{TE}$ holds since the robot stops serving the HumanLeader and starts serving the HumanFollower after the extra time allowed to complete the task has elapsed. Therefore, as per Table 2, this add-on realizes an Omission error phenotype. Note that, since the robot waits for the human to perform their task until time TE elapses, the switch to r_{stop} occurs at time $\mathsf{t}'_4 > \mathsf{t}_4$. However, this is not an error by itself but the system's response to the error made by the human. (Thus, it is highlighted in blue and not in red.)

4.1.4 Safety Violation Add-On

The Safety Violation add-on captures the human entering a critical situation (e.g., moving too close to the robot), possibly causing a **safety hazard** [55]. Unlike operators in industrial settings, people in service settings do not wear protective devices nor receive systematic and thorough training in working alongside robots. Enforcing safety measures


$\langle\text{op}\rangle\_\text{pub}_h$
$\langle\text{op}\rangle'\_\text{pub}_h$
γ', c′?, ξ'
γ', c′?, ξ'
c?
ξ
c_safe?
ξ_crit
$\langle\text{op\_safe}\rangle\_\text{pub}_h$
$\langle\text{op}\rangle\_\text{pub}_h$
$\langle\text{op}\rangle'\_\text{pub}_h$
γ', c′?, ξ'
γ', c′?, ξ'
c?,ξ
ξ
c_safe?
ξ_crit
$\langle\text{op\_safe}\rangle\_\text{pub}_h$
λ_safe

(a) SHA add-on capturing the non-erroneous behavior.

(b) SHA add-on capturing the erroneous behavior: exponential rate λ_safe is color-coded like an invariant.

Fig. 8 SHA depicting the standard behavior and the erroneous behavior captured by the Safety Violation add-on, color-coded as in Fig. 3 (color figure online)

may be necessary throughout the interaction to prevent undesirable events, such as collisions. This add-on introduces a formalization of the human and the robot operating under a safety measure and the human violating such measure out of error.

The standard behavior, shown in Fig. 8a, models the situation in which the human is required to switch to a *safe mode* under specific circumstances while performing an action. As in previous cases, the current state is modeled by subautomaton $\langle\text{op}\rangle\_\text{pub}_h$, whereas the subsequent state in the plan is subautomaton $\langle\text{op}\rangle'\_\text{pub}_h$. The switch to $\langle\text{op}\rangle'\_\text{pub}_h$ is enforced by the orchestrator by firing an event through channel c′? when condition γ' holds and, upon firing such event, update ξ' is executed. The condition determining whether the human should switch to the safety mode depends on the specific interaction pattern. For example, referring to the HumanFollower pattern, the condition raising safety concerns is the human getting too close to the robot, expressed as $\text{dist}(\text{h}_\text{pos}, \text{r}_\text{pos}) \leq \text{d}_\text{crit}$, where function dist and variable h_pos are as described in Sect. 4.1.3, $\text{r}_\text{pos} \in W$ is the Cartesian coordinate pair representing the robot's position, and $\text{d}_\text{crit} \in K$ is a system-wide constant representing the maximum distance allowed before a safety measure is enforced. Therefore, the human should stay in $\langle\text{op}\rangle\_\text{pub}_h$ only, while the distance from the robot is greater than d_crit.

As soon as the safety–critical condition holds, the human receives an orchestrator instruction over channel c_safe to switch to the safe mode, i.e., subautomaton $\langle\text{op\_safe}\rangle\_\text{pub}_h$, which captures the same operational state as $\langle\text{op}\rangle\_\text{pub}_h$ but with the safety measure enforced. Realistic examples of this contingency would be the human being instructed to take a few steps to avoid the moving robot or walking at a slower pace to avoid colliding with the robot. Upon switching to $\langle\text{op\_safe}\rangle\_\text{pub}_h$, SHA variables are updated through ξ_crit to reflect the safety measure being enforced (e.g., setting a lower value of the human's walking speed). If the safety–critical condition eventually holds, the human is instructed to switch back to $\langle\text{op}\rangle\_\text{pub}_h$ through channel c. Note that the operational state modeled by $\langle\text{op\_safe}\rangle\_\text{pub}_h$ represents the *same* state as $\langle\text{op}\rangle\_\text{pub}_h$ with different parameters (e.g., walking *at a slower pace*) and not a different functionally equivalent action. (Thus, it is not eligible for a Replacement phenotype.)

The corresponding add-on modeling the erroneous behavior is shown in Fig. 8b. In this case, the modeled human error consists of arbitrarily *leaving* $\langle\text{op\_crit}\rangle\_\text{pub}_h$ even if the safety–critical condition is still in place (e.g., the human resuming walking at a full pace even if they are still too close to the robot). The human arbitrarily *leaving* $\langle\text{op\_safe}\rangle\_\text{pub}_h$ is captured by an additional edge back to $\langle\text{op}\rangle\_\text{pub}_h$ without any channel, whose firing depends on exponential rate λ_safe added to subautomaton $\langle\text{op\_safe}\rangle\_\text{pub}_h$. The mechanism determining whether the SHA takes the new erroneous edge is stochastic rather than deterministic as in Fig. 8a. Specifically, the probability of leaving $\langle\text{op\_safe}\rangle\_\text{pub}_h$ t time units after entering it is $1 - \text{e}^{-\lambda_\text{safe} t}$. Therefore, the probability of switching back to $\langle\text{op\_safe}\rangle\_\text{pub}_h$ irrespective of the orchestrator's instructions increases with time. Note that the longer the human stays in $\langle\text{op\_crit}\rangle\_\text{pub}_h$, which models the safe mode, the safer it is for the system. However, the higher λ_safe is, the more likely the human is to leave $\langle\text{op\_safe}\rangle\_\text{pub}_h$ shortly after entering it when the safety–critical condition still holds.

The human erroneously not *entering* $\langle\text{op\_safe}\rangle\_\text{pub}_h$ upon receiving a message through c_safe would be covered by applying the Heed/Ignore add-on to the edge from $\langle\text{op}\rangle\_\text{pub}_h$ to $\langle\text{op\_crit}\rangle\_\text{pub}_h$.

This add-on gives rise to several phenotypes of erroneous behavior, thus different erroneous action sequences are illustrated in the following. Examples are provided taking as reference the HumanFollower pattern, specifically the sequence that envisages the orchestrator enforcing the safety measure, shown in sequence (13). Although the switch to $\langle\text{walk\_safe}\rangle\_\text{pub}_h$ is not envisaged by default by the Human-

Follower pattern (see sequence (4)), it is not considered an error but a desired effect of the orchestrator's policies and the realization of the standard behavior in Fig. 8a. Therefore, the corresponding elements in sequence (13) are highlighted in blue and not in red.

$$[r_{\text{start}}, \left(\langle \mathsf{walk}\rangle\_\mathrm{pub}_h, \langle \mathsf{walk\_safe}\rangle\_\mathrm{pub}_h\right)^{\mathsf{m}}, (\langle \mathsf{walk}\rangle\_\mathrm{pub}_h)^{\mathsf{n}}, r_{\text{stop}}, \langle \mathsf{stand}\rangle\_\mathrm{pub}_h] \tag{13}$$

Sequence (13) captures all the possible realizations of the standard behavior in Fig. 8a with $\mathsf{m} \geq 1$ and $\mathsf{n} \in \{0, 1\}$. In more detail, the human is instructed to enter the safe mode (i.e., walking at a slower pace) at least once. Afterward, the human may be instructed to switch between $\langle \mathsf{walk}\rangle\_\mathrm{pub}_h$ and $\langle \mathsf{walk\_safe}\rangle\_\mathrm{pub}_h$ $\mathsf{m} - 1$ times. Finally, in any case, the task can either terminate with the human in $\langle \mathsf{walk\_safe}\rangle\_\mathrm{pub}_h$ (if $\mathsf{n} = 0$ holds) or in $\langle \mathsf{walk}\rangle\_\mathrm{pub}_h$ (if $\mathsf{n} = 1$ holds).

When applied to sequence (13), the Safety Violation add-on realizes different phenotypes based on the value of n. If $\mathsf{n} = 0$ holds (thus, the human should conclude the task in $\langle \mathsf{walk\_safe}\rangle\_\mathrm{pub}_h$) and the erroneous behavior in Fig. 8b occurs, sequence (14) is observed, which captures the human unexpectedly resuming walking at full speed.

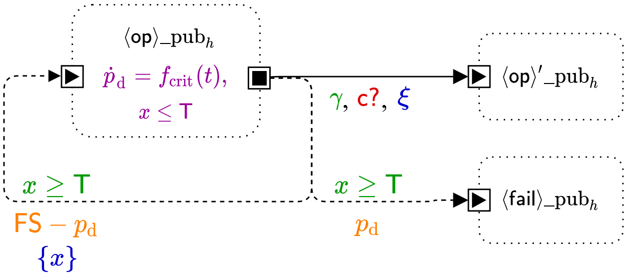
$$[r_{\text{start}}, \langle \mathsf{walk}\rangle\_\mathrm{pub}_h, \langle \mathsf{walk\_safe}\rangle\_\mathrm{pub}_h, \langle \mathsf{walk}\rangle\_\mathrm{pub}_h, r_{\text{stop}}, \langle \mathsf{stand}\rangle\_\mathrm{pub}_h] \tag{14}$$

Sequence (14) realizes two phenotypes from the same error mode (i.e., action not included in current plan): If the safety measure (even if active for a reduced amount of time) was successful in avoiding a hazard, the service can be completed successfully, resulting in an Insertion phenotype; otherwise (i.e., lifting the safety measure too early causes a hazard), the mission fails, effectively realizing an Intrusion phenotype. The same phenotypes are realized whether $\mathsf{m} = 1$ or $\mathsf{m} > 1$ hold.

If $\mathsf{n} = 1$ holds, it is sufficient to enforce the safety measure for a limited amount of time before all actions can resume in their normal mode. (The human concludes the action in $\langle \mathsf{walk}\rangle\_\mathrm{pub}_h$.) In this case, the possible erroneous behavior is the human resuming the normal mode *too early*, as shown in sequence (15), which realizes a Premature Action phenotype with $\mathsf{m} = 1$.

$$[(\mathsf{t}_1, \mathtt{cmd\_r}_{\text{start}}), (\mathsf{t}_2, \mathtt{cmd\_h}_{\text{start}}), (\mathsf{t}_3, \mathtt{cmd\_h}_{\text{safe\_start}}), (\mathsf{t}'_4, \mathtt{cmd\_h}_{\text{start}}), (\mathsf{t}_4, \bot), (\mathsf{t}_5, \mathtt{cmd\_r}_{\text{stop}}), \ldots] \tag{15}$$

If $\mathsf{m} > 1$ holds, the Premature Action phenotype can either refer to the last switch to $\langle \mathsf{walk}\rangle\_\mathrm{pub}_h$ (like in sequence (15)) or any intermediate one.



$\langle \mathsf{op}\rangle\_\mathrm{pub}_h$
$\dot{p}_{\mathrm{d}} = f_{\mathrm{crit}}(t),$
$x \leq \mathsf{T}$
γ, $\mathsf{c}?$, ξ
$\langle \mathsf{op}\rangle'\_\mathrm{pub}_h$
$x \geq \mathsf{T}$
$x \geq \mathsf{T}$
$\langle \mathsf{fail}\rangle\_\mathrm{pub}_h$
$\mathsf{FS} - p_{\mathrm{d}}$
$\{x\}$
p_{d}

Fig. 9 SHA representing the Critical Status add-on, color-coded as in Fig. 3 (color figure online)

4.1.5 Critical Status Add-On

Although our modeling framework applies to generic service settings, some of its features specifically target healthcare environments. In such cases, where people are often in pain or discomfort, robotic applications must safeguard humans' well-being and take into account *unexpected* (rather than purely erroneous) health-related accidents. This contingency is captured by the Critical Status add-on, capturing human subjects facing a sudden unexpected health issue (e.g., fainting) that requires immediate medical attention.

The standard behavior, in this case, may be captured by both Figs. 5a and 6b: In the following, we present the add-on as a variation of Fig. 5a. However, the same conclusions can be drawn on the SHA in Fig. 6b by replacing c with the internal action. The standard behavior envisages the human whose current state is modeled by subautomaton $\langle \mathsf{op}\rangle\_\mathrm{pub}_h$ and upcoming state by $\langle \mathsf{op}\rangle'\_\mathrm{pub}_h$. The switch from $\langle \mathsf{op}\rangle\_\mathrm{pub}_h$ to $\langle \mathsf{op}\rangle'\_\mathrm{pub}_h$ either depends on the orchestrator's instructions sent through channel c or the human's initiative. The edge is enabled when guard γ holds and, upon firing, causes update ξ to execute.

As per Sect. 3.2, our modeling approach features a model of physical fatigue that increases when the human is actively performing an action and decreases when they are resting. Human fatigue can only increase up to a maximum threshold (1 in our case, representing that 100% of muscle reservoir units have been activated) before the human is no longer able to move autonomously. The Critical Status add-on captures the possibility that the human faints or a similarly impairing accident occurs even if their current fatigue level is still below the maximum threshold. The SHA add-on representing this contingency is shown in Fig. 9: The additional location $\langle \mathsf{fail}\rangle\_\mathrm{pub}_h$ represents the deadlock reached by the SHA if the accident occurs, causing the failure of the mission.

As shown in Fig. 9, location $\langle \mathsf{op}\rangle\_\mathrm{pub}_h$ features invariant $x \leq \mathsf{T}$ as in Fig. 6, where $x \in X$ is a clock and $\mathsf{T} \in K$ is a constant. Compared to the standard behavior, the add-on has two additional edges leaving $\langle \mathsf{op}\rangle\_\mathrm{pub}_h$, a self-loop and the edge to $\langle \mathsf{fail}\rangle\_\mathrm{pub}_h$, both labeled with guard $x \geq \mathsf{T}$. Every T time

Table 5 Summary of human–robot interaction patterns with a description of the captured service

Pattern	Description
HumanFollower	The human *follows* the robot to a specific destination. The robot signals that the service has been completed when both the robot and the human are close to the destination
HumanLeader	The human *leads* the robot to a specific destination of which they know the precise location. The human claims the service is complete when both the human and the robot have reached their destination
HumanRecipient	The human has the robot *fetch* an item from a specific location and *deliver* it to the human. The robot determines the service has been provided when the item has been successfully picked up
HumanCompetitor	The human and the robot *compete* to fetch a critical resource (for example, a medical kit during an emergency). Both agents move to the location of the resource to reach it as quickly as possible. The competition ends when either of the agents reaches the target location
HumanRescuer	The robot requires human *intervention* to complete a *task*, such as pressing a button to call the elevator or opening a closed door. In this case, the robot will emit audible or visible signals to notify its need for human support. The human moves to the robot's current location, performs the required action, and concludes the interaction
HumanApplicant	The human requires the robot's *support* in performing a *task* that implies timely or close-contact interaction, such as feeding a patient or administering medication. The robot approaches the human's location, then the action requiring synchronization starts

units, there is a certain probability that the accident occurs, and the mission fails (the SHA switches to $\langle\mathsf{fail}\rangle\_\mathrm{pub}_h$), or that the human remains in the same state (the self-loop on $\langle\mathsf{op}\rangle\_\mathrm{pub}_h$ fires). Unlike in the Heed/Ignore add-on, probability weights are not constant, but their value changes with time. (Thus, they are real-valued variables.) We indicate as $p_\mathrm{d} \in W$ the real-valued variable in question, whose derivative is constrained through a flow condition on $\langle\mathsf{op}\rangle\_\mathrm{pub}_h$. In our specific case, the add-on envisages that the higher the level of fatigue, the higher the probability of an accident occurring. Therefore, the flow condition constraining p_d is indicated as $f_\mathrm{crit}(t)$ in Fig. 9 for the sake of generality, but in our specific case, it is a customizable function of fatigue, modeled by real-valued variable $F \in W$. An example, featured by the SHA presented later in this section, is $f_\mathrm{crit}(t) = \mathsf{hs} \cdot \dot{F}(t)$, where $\mathsf{hs} \in K$ is a customizable parameter determining how rapidly the probability of an accident increases with fatigue. In general, the probability weight for the self-loop on $\langle\mathsf{op}\rangle\_\mathrm{pub}_h$ should evolve in time inversely with respect to fatigue. (The higher the fatigue level, the lower the probability that the human does *not* have an accident and stay in the same state.) A trivial example of expression determining the probability weight on the self-loop, also depicted in Fig. 9, is $\mathsf{FS} - p_\mathrm{d}$, where $\mathsf{FS} \in K$ is a constant such that $\mathsf{FS} \geq \sup(p_\mathrm{d})$ holds.

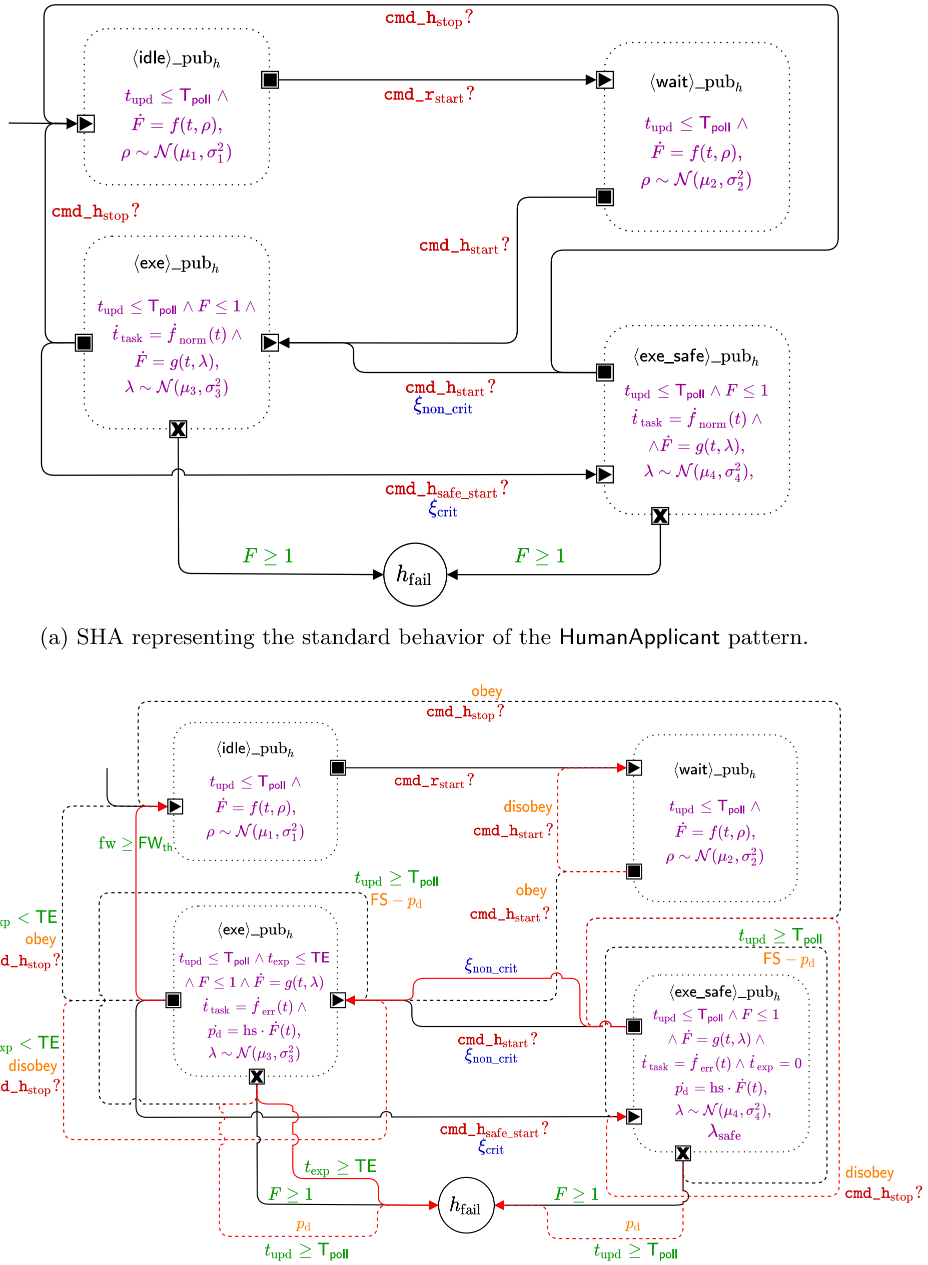
The action sequence observed if the behavior captured by this add-on occurs features, irrespective of the specific pattern, an unexpected action corresponding to the accident, this is not part of the original plan and prevents the human–robot pair from achieving the goal, effectively realizing an Intrusion phenotype.

4.2 Human–robot interaction patterns

As described in Sect. 2, in our framework, human behavior is modeled through different SHA based on the service they are requesting, and services correspond to **interaction patterns**. Each SHA modeling an interaction pattern captures how the human behaves—either autonomously or in response to a robot's action—to achieve the goal of the specific service. The standard behavior envisaged by each interaction pattern is described in detail in [37, 39] and briefly summarized in Table 5 to keep the paper self-contained. This article extends the six SHA modeling human agents through the previously presented erroneous behavior add-ons. Not all add-ons apply to all patterns since we rule out behaviors that are unfeasible or unrealistic. In the following, we present the extended SHA modeling the HumanApplicant pattern in detail as an example of how add-ons are applied to HRI patterns. We then outline how the other five patterns have been enriched through a similar procedure.

4.2.1 HumanApplicant pattern

As per Table 5, the HumanApplicant interaction pattern captures contingencies in which the human requests the robot's support to complete a task that requires working in a very close distance or sharp timely synchronization [39]. Exam-



$\mathtt{cmd\_h}_{\mathrm{stop}}$?
⟨idle⟩_pub$_h$
$t_{\mathrm{upd}} \leq \mathsf{T}_{\mathrm{poll}} \wedge$
$\dot{F} = f(t, \rho)$,
$\rho \sim \mathcal{N}(\mu_1, \sigma_1^2)$
$\mathtt{cmd\_r}_{\mathrm{start}}$?
⟨wait⟩_pub$_h$
$t_{\mathrm{upd}} \leq \mathsf{T}_{\mathrm{poll}} \wedge$
$\dot{F} = f(t, \rho)$,
$\rho \sim \mathcal{N}(\mu_2, \sigma_2^2)$
$\mathtt{cmd\_h}_{\mathrm{stop}}$?
$\mathtt{cmd\_h}_{\mathrm{start}}$?
⟨exe⟩_pub$_h$
$t_{\mathrm{upd}} \leq \mathsf{T}_{\mathrm{poll}} \wedge F \leq 1 \wedge$
$\dot{t}_{\mathrm{task}} = \dot{f}_{\mathrm{norm}}(t) \wedge$
$\dot{F} = g(t, \lambda)$,
$\lambda \sim \mathcal{N}(\mu_3, \sigma_3^2)$
$\mathtt{cmd\_h}_{\mathrm{start}}$?
$\xi_{\mathrm{non\_crit}}$
⟨exe_safe⟩_pub$_h$
$t_{\mathrm{upd}} \leq \mathsf{T}_{\mathrm{poll}} \wedge F \leq 1$
$\dot{t}_{\mathrm{task}} = \dot{f}_{\mathrm{norm}}(t) \wedge$
$\wedge \dot{F} = g(t, \lambda)$,
$\lambda \sim \mathcal{N}(\mu_4, \sigma_4^2)$,
$\mathtt{cmd\_h}_{\mathrm{safe\_start}}$?
ξ_{crit}
$F \geq 1$
h_{fail}
$F \geq 1$

(a) SHA representing the standard behavior of the HumanApplicant pattern.

obey
$\mathtt{cmd\_h}_{\mathrm{stop}}$?
⟨idle⟩_pub$_h$
$t_{\mathrm{upd}} \leq \mathsf{T}_{\mathrm{poll}} \wedge$
$\dot{F} = f(t, \rho)$,
$\rho \sim \mathcal{N}(\mu_1, \sigma_1^2)$
$\mathtt{cmd\_r}_{\mathrm{start}}$?
⟨wait⟩_pub$_h$
$t_{\mathrm{upd}} \leq \mathsf{T}_{\mathrm{poll}} \wedge$
$\dot{F} = f(t, \rho)$,
$\rho \sim \mathcal{N}(\mu_2, \sigma_2^2)$
disobey
$\mathtt{cmd\_h}_{\mathrm{start}}$?
$\mathrm{fw} \geq \mathsf{FW}_{\mathrm{th}}$
$t_{\mathrm{upd}} \geq \mathsf{T}_{\mathrm{poll}}$
$\mathsf{FS} - p_{\mathrm{d}}$
obey
$\mathtt{cmd\_h}_{\mathrm{start}}$?
$t_{\mathrm{exp}} < \mathsf{TE}$
obey
$\mathtt{cmd\_h}_{\mathrm{stop}}$?
⟨exe⟩_pub$_h$
$t_{\mathrm{upd}} \leq \mathsf{T}_{\mathrm{poll}} \wedge t_{\mathrm{exp}} \leq \mathsf{TE}$
$\wedge F \leq 1 \wedge \dot{F} = g(t, \lambda)$
$\dot{t}_{\mathrm{task}} = \dot{f}_{\mathrm{err}}(t) \wedge$
$\dot{p_{\mathrm{d}}} = \mathrm{hs} \cdot \dot{F}(t)$,
$\lambda \sim \mathcal{N}(\mu_3, \sigma_3^2)$
$\xi_{\mathrm{non\_crit}}$
$t_{\mathrm{upd}} \geq \mathsf{T}_{\mathrm{poll}}$
$\mathsf{FS} - p_{\mathrm{d}}$
⟨exe_safe⟩_pub$_h$
$t_{\mathrm{upd}} \leq \mathsf{T}_{\mathrm{poll}} \wedge F \leq 1$
$\wedge \dot{F} = g(t, \lambda) \wedge$
$\dot{t}_{\mathrm{task}} = \dot{f}_{\mathrm{err}}(t) \wedge \dot{t}_{\mathrm{exp}} = 0$
$\dot{p_{\mathrm{d}}} = \mathrm{hs} \cdot \dot{F}(t)$,
$\lambda \sim \mathcal{N}(\mu_4, \sigma_4^2)$,
λ_{safe}
$t_{\mathrm{exp}} < \mathsf{TE}$
disobey
$\mathtt{cmd\_h}_{\mathrm{stop}}$?
$\mathtt{cmd\_h}_{\mathrm{start}}$?
$\xi_{\mathrm{non\_crit}}$
$\mathtt{cmd\_h}_{\mathrm{safe\_start}}$?
ξ_{crit}
$t_{\mathrm{exp}} \geq \mathsf{TE}$
$F \geq 1$
h_{fail}
$F \geq 1$
disobey
$\mathtt{cmd\_h}_{\mathrm{stop}}$?
p_{d}
$t_{\mathrm{upd}} \geq \mathsf{T}_{\mathrm{poll}}$
p_{d}
$t_{\mathrm{upd}} \geq \mathsf{T}_{\mathrm{poll}}$

(b) SHA representing the HumanApplicant enriched with erroneous behavior add-ons. Color-coding is as in Fig. 3 except for edges capturing human errors, highlighted in red for visualization purposes.

Fig. 10 SHA modeling the HumanApplicant pattern (color figure online)

ple applications are robotic companions supporting a patient while feeding or healthcare professionals receiving the support of a service robot while administering medication. In the following, firstly, we recap the standard behavior of this pattern, shown in Fig. 10a; secondly, we describe the SHA (shown in Fig. 10b) extended with add-ons to incorporate erroneous behaviors.

The SHA modeling the HumanApplicant pattern features four instances of ⟨op⟩_pub$_h$ corresponding to the three phases of the service plus the operational state under critical conditions (⟨op_safe⟩_pub$_h$ in Fig. 8b). When the service starts, the human is idle and resting, captured by subautomaton ⟨idle⟩_pub$_h$. The robot starts moving to approach the human when the orchestrator fires an event through channel $\mathtt{cmd\_r}_{\mathrm{start}}$, causing the human to switch to ⟨wait⟩_pub$_h$, also a recovery state. The flow condition constraining F is, therefore, $f(t, \rho)$ (see Eq. 2) both in ⟨idle⟩_pub$_h$ and ⟨wait⟩_pub$_h$. Normal distributions $\mathcal{N}(\mu_1, \sigma_1^2)$ and $\mathcal{N}(\mu_2, \sigma_2^2)$ determine the values of rate ρ in $\langle\mathsf{idle}\rangle\_\mathrm{pub}_h$ and $\langle\mathsf{wait}\rangle\_\mathrm{pub}_h$, respectively.

When the robot has reached the human's position, the orchestrator instructs the human to start performing the required task by firing an event through $\mathtt{cmd\_h}_{\mathrm{start}}$, causing the human to switch to subautomaton $\langle\mathsf{exe}\rangle\_\mathrm{pub}_h$. The standard duration of the task is modeled by a parameter $\mathsf{T}_{\mathsf{task}} \in K$, while $\mathsf{dext} \in K$ represents the human's *dexterity*, i.e., the rate at which they perform the specific task. The real-valued variable capturing the progress of the task is $t_{\mathrm{task}} \in W$ (φ in Fig. 7a), which evolves in time according to flow condition $f_{\mathrm{norm}}(t) = \mathsf{dext} \cdot t$. In the ordinary case, the orchestrator instructs the human to stop by means of $\mathtt{cmd\_h}_{\mathrm{stop}}$ and switch back to $\langle\mathsf{idle}\rangle\_\mathrm{pub}_h$ when the human has spent $\frac{\mathsf{T}_{\mathsf{task}}}{\mathsf{dext}}$ time units working on the task [39]. The latter switch marks the completion of the service (thus contributing to the success of the mission).

If, while performing the task, the orchestrator determines that the human and the robot are in a critical situation (i.e., their distance is below a certain threshold or human fatigue is above a critical level), it instructs the human to proceed cautiously. The human receives this instruction through channel $\mathtt{cmd\_h}_{\mathrm{safe\_start}}$ and switches to subautomaton $\langle\mathsf{exe\_safe}\rangle\_\mathrm{pub}_h$. Both $\langle\mathsf{exe}\rangle\_\mathrm{pub}_h$ and $\langle\mathsf{exe\_safe}\rangle\_\mathrm{pub}_h$ subautomata are fatigue states, thus endowed with flow condition $g(t, \lambda)$ (see Eq. 3). Distributions $\mathcal{N}(\mu_3, \sigma_3^2)$ and $\mathcal{N}(\mu_4, \sigma_4^2)$ determine the values of fatigue rate λ. Since $\langle\mathsf{exe\_safe}\rangle\_\mathrm{pub}_h$ captures the human working at a slower pace to avoid exhaustion or bumping against the robot (enforced through update ξ_{crit}, which reduces the value of parameter dext), $\mu_4 < \mu_3$ holds. If the safety measure is successful, the orchestrator instructs the human to switch back to $\langle\mathsf{exe}\rangle\_\mathrm{pub}_h$ through channel $\mathtt{cmd\_h}_{\mathrm{start}}$, and update ξ_{non}_crit restores the normal value of dext. Otherwise, if the task is completed, while the human is in $\langle\mathsf{exe\_safe}\rangle\_\mathrm{pub}_h$, the orchestrator instructs it to switch back to $\langle\mathsf{idle}\rangle\_\mathrm{pub}_h$ through channel $\mathtt{cmd\_h}_{\mathrm{stop}}$.

Finally, deadlock location h_{fail} is reachable by the two fatigue states upon reaching the maximum endurable level of fatigue. (Guard $F \geq 1$ holds.)

The edges modeling erroneous behaviors are highlighted in red in Fig. 10b, and the applied add-ons are individually described in the following.

Heed/Ignore *Add-On*

Initially, the human may *delay* the start of the action and not respond to the $\mathtt{cmd\_h}_{\mathrm{start}}$ command. Therefore, the edge from $\langle\mathsf{wait}\rangle\_\mathrm{pub}_h$ to $\langle\mathsf{exe}\rangle\_\mathrm{pub}_h$ is expanded into a Heed/Ignore add-on. Similarly, both edges from $\langle\mathsf{exe}\rangle\_\mathrm{pub}_h$ and $\langle\mathsf{exe\_safe}\rangle\_\mathrm{pub}_h$ to $\langle\mathsf{idle}\rangle\_\mathrm{pub}_h$, marking the end of the action through channel $\mathtt{cmd\_h}_{\mathrm{stop}}$, might be erroneously ignored by the human, and are thus expanded into a Heed/Ignore add-on.

Free Will *Add-On*

While in $\langle\mathsf{exe}\rangle\_\mathrm{pub}_h$, the human may erroneously pause the task before it is complete (thus, before $\mathtt{cmd\_h}_{\mathrm{stop}}$ fires). Subautomata $\langle\mathsf{exe}\rangle\_\mathrm{pub}_h$ and $\langle\mathsf{idle}\rangle\_\mathrm{pub}_h$ are connected by an additional edge implementing the Free Will add-on, which fires when $\mathrm{fw} \geq \mathsf{FW}_{\mathsf{th}}$ holds. Dense counter fw is updated every $\mathsf{T}_{\mathsf{poll}}$ time instants by $\xi_{\langle\mathsf{exe}\rangle}$ (an instance of $\xi\langle\mathsf{op}\rangle$ in Fig. 4, embedded into the $\langle\mathsf{exe}\rangle\_\mathrm{pub}_h$ subautomaton) as described in Sect. 4.1.2.

Timer Expired *Add-On*

While performing the task, the human might erroneously waste time and delay the completion of the action to the point that the robot considers them no longer responsive, as envisaged by the Timer Expired add-on. Therefore, variable t_{task}, capturing the progress of the task, is constrained by a different flow condition, i.e., $f_{\mathrm{err}}(t)$ shown in Eq. 16.

$$f_{\mathrm{err}}(t) = (\mathsf{dext} \cdot (\mathrm{rand}(0, \mathsf{T}_{\mathsf{max}}) \geq \mathsf{T}_{\mathsf{th}})) \cdot t \tag{16}$$

A stochastic mechanism governs the progress of the task as, for each time instant, function rand draws a new sample from Uniform distribution $\mathcal{U}_{[0,\mathsf{T}_{\mathsf{max}})}$ and increases the value of t_{task} if the sample is greater than a customizable threshold T_{th} [39]. Therefore, the randomized evolution of variable t_{task} may lead to an unacceptable delay in completing the task.

Since, in this case, the human is not walking toward a target but performing a task for a certain amount of time, time bound TE to deem the human non-responsive depends on the expected duration of the task (parameter $\mathsf{T}_{\mathsf{task}}$) and the rate at which the human performs it (dext) as per Eq. 17, where δ is the allowance factor as illustrated in Sect. 4.1.3.

$$\mathsf{TE} = \frac{\mathsf{T}_{\mathsf{task}}}{\mathsf{dext}} \cdot (1 + \delta) \tag{17}$$

Subautomaton $\langle\mathsf{exe}\rangle\_\mathrm{pub}_h$ is endowed with invariant $t_{\mathrm{exp}} \leq \mathsf{TE}$, where $t_{\mathrm{exp}} \in X$ is a clock (as in Fig. 7b). As in Fig. 7b, the edge connecting subautomaton $\langle\mathsf{exe}\rangle\_\mathrm{pub}_h$ to deadlock location h_{fail} with guard condition $t_{\mathrm{exp}} \geq \mathsf{TE}$ fires as soon as time TE elapses.

On the other hand, although the evolution of t_{task} is constrained by Eq. 16 also while in $\langle\mathsf{exe\_safe}\rangle\_\mathrm{pub}_h$, this operational state is not extended through the Timer Expired add-on since working at a slower pace is implied by the safety measure. Therefore, the time the human spends in $\langle\mathsf{exe\_safe}\rangle\_\mathrm{pub}_h$ does not count toward upper bound TE ($\langle\mathsf{exe\_safe}\rangle\_\mathrm{pub}_h$ is endowed with flow condition $\dot{t}_{\mathrm{exp}} = 0$).

Safety Violation *Add-On*

If the orchestrator finds that the human and the robot are in a safety–critical situation, it will instruct the human to switch to $\langle\mathsf{exe\_safe}\rangle\_\mathrm{pub}_h$, corresponding to $\langle\mathsf{op\_safe}\rangle\_\mathrm{pub}_h$ in Fig. 8a. As envisaged by the Safety Violation add-on, the human may erroneously resume working at a normal pace, potentially causing a safety hazard. Therefore, while the orchestrator instructs the human to resume normal operations through channel $\mathtt{cmd\_h}_{\mathrm{start}}$, an additional edge without labels except update $\xi_{\mathrm{non\_crit}}$ restoring the standard value of dext connects $\langle\mathsf{exe\_safe}\rangle\_\mathrm{pub}_h$ to $\langle\mathsf{exe}\rangle\_\mathrm{pub}_h$. Subautomaton $\langle\mathsf{exe\_safe}\rangle\_\mathrm{pub}_h$ is endowed with parameter λ_{safe} representing the rate at which the probability of erroneously switching back to $\langle\mathsf{exe}\rangle\_\mathrm{pub}_h$ increases with time.

Critical Status *Add-On*

Since this pattern applies to patients and healthcare professionals who may find themselves in stressful situations or undiagnosed conditions, the modeled human subject is susceptible to accidents. Therefore, both $\langle\mathsf{exe}\rangle\_\mathrm{pub}_h$ and $\langle\mathsf{exe\_safe}\rangle\_\mathrm{pub}_h$ are extended through the Critical Status add-on. Specifically, at time t, where t is a multiple of $\mathsf{T}_{\mathsf{poll}}$ (T in Fig. 9), the probability of an accident occurring is p_{d}, where $p_{\mathrm{d}} \in W$ is a real-valued variable. If this occurs, the SHA switches to h_{fail}. Otherwise, the probability of the SHA remaining in the same operational state depends on weight $\mathsf{FS} - p_{\mathrm{d}}$, where $\mathsf{FS} \in K$ is a constant such that $\mathsf{FS} \geq \sup(p_{\mathrm{d}})$ holds. The probability of an accident occurring increases with the level of fatigue, as implied by flow condition $\dot{p_{\mathrm{d}}} = \mathsf{hs} \cdot \dot{F(t)}$, where $\mathsf{hs} \in K$ is a numerical constant.

4.2.2 HumanFollower pattern

The HumanFollower pattern envisages the human following the robot to a particular destination. Operational states (corresponding to as many $\langle\mathsf{op}\rangle\_\mathrm{pub}_h$ instances) capture the human *standing* (thus, recovering) and *walking*. Upon receiving the instruction from the robot to start or stop walking (thus, either in the *standing* and *walking* states), the Heed/Ignore add-on introduces the possibility that the human ignores it. Similarly, the human may start or stop walking irrespective of the robot's instructions through the Free Will add-on. Since the Follower pattern does not envisage any motion initiated by the human, the Timer Expired add-on does not apply. On the other hand, it is feasible for the human to walk too close to the robot, leading to the enforcement of a safety measure (i.e., the human walking slower). Moving in critical conditions is captured by a third $\langle\mathsf{op}\rangle\_\mathrm{pub}_h$ instance, which is subject to the Safety Violation add-on. Finally, to capture the possibility of unexpected accidents, all three operational states are extended through the Critical Status add-on: While walking (either normally or at a slower pace), probability p_{d} increases, while it decreases when the human is resting.

4.2.3 HumanLeader pattern

The HumanLeader pattern captures the mirrored situation compared to the Follower, featuring the human leading the robot to a certain destination. This SHA features the same $\langle\mathsf{op}\rangle\_\mathrm{pub}_h$ instances as the Follower (*walking* and *standing*), although the human, rather than the robot, initiates the actions of starting to walk and stopping. Therefore, all edges between the two subautomata are extended through the Free Will add-on. The robot only sends an instruction when the human's fatigue level rises to a critical threshold, advising them to stop and recover. This edge is extended through the Heed/Ignore add-on, as the human may erroneously ignore or miss the robot's suggestion. Since this pattern captures a human freely operating on the floor, they can get caught up in alternative tasks causing them to excessively delay the *walking* phase, which is captured by the Timer Expired add-on. As with the Follower pattern, the Safety Violation add-on captures the situation in which the human erroneously starts walking at full pace while in a critical situation. Finally, the Critical Status add-on captures the possibility of unexpected accidents.

4.2.4 HumanRecipient pattern

The HumanRecipient pattern captures fetch-and-delivery tasks where the robot retrieves a required object and delivers it back to the human. The standard behavior, in this case, features two operational states, i.e., the human *waiting* for the robot to retrieve the object and *interacting* with the robot to collect the item. The latter action is performed upon the robot's instruction and is thus extended through the Heed/Ignore add-on. To capture the possibility that the human might move while waiting for the robot, the SHA features an additional operational state capturing the human walking. The switch from the idle operational state, which is entirely up to the human, occurs through the Free Will add-on. Since the pickup action is supposedly almost instantaneous, the HumanRecipient pattern is not eligible for the Timer Expired add-on. Finally, the Critical Status captures the possibility of unexpected accidents, while the Safety Violation add-on the possibility that the human might ignore a safety measure. However, the limited duration of the interactive phase leads to a reduced impact of this error on the HumanRecipient pattern.

4.2.5 HumanCompetitor pattern

The HumanCompetitor pattern captures the human and the robot racing toward a specific location and is, thus, the only non-cooperative pattern. The standard operational states are the human *moving* to the requested location, then either *waiting* for the robot to return to its original position (if the robot wins the competition) or *return* to their initial position themselves (if the human wins the competition). Starting to walk and stopping are actions initiated by the human, both extended through the Free Will add-on, capturing the

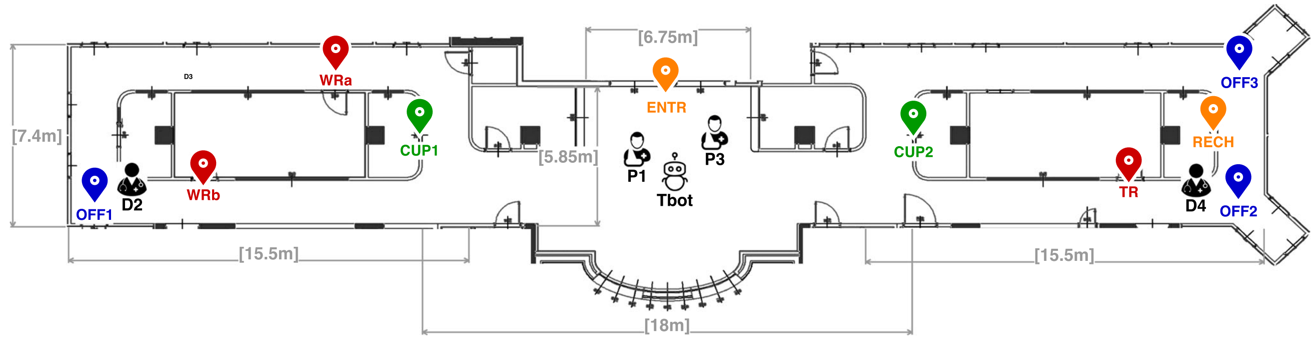


Fig. 11 Floor layout used for scenarios treat1patient and treat2patients. Agents (P1, P3, D2, D4, and Tbot) are represented in their starting positions. Location symbols mark the position of points of interest (POIs): Entrance and robot's recharge station are in orange, waiting room and treatment room doors are in red, cupboards are in green, and doctors' offices are in blue. Wall lengths (in meters) are also reported (color figure online)

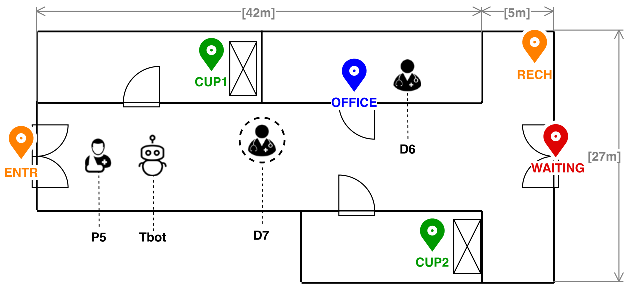


Fig. 12 Floor layout for scenario wardEmergency, color-coded as Fig. 11. Since D7's starting position is randomized, the displayed location is purely representative (color figure online)

possibility that the human may get distracted or waste time while trying to reach the item's location. Erroneous add-ons applied to the HumanCompetitor pattern do not *directly* impact the outcome of the mission, but they result in a higher chance of the robot winning the competition. Therefore, errors that occur amidst a HumanCompetitor service increase the impact on the overall mission outcome of errors that may arise amidst services provided by the robot if it wins the competition.

4.2.6 HumanRescuer pattern

The HumanRescuer pattern captures the mirrored situation compared to the HumanApplicant, i.e., the robot requiring the human's support in performing a task (such as opening a door or placing an item on the robot's tray). The standard behavior features three phases, modeled by as many $\langle \mathsf{op} \rangle\_\mathsf{pub}_h$ instances: the human in *idle* state, the human *walking* toward the robot after noticing the signal requesting support, and the human *performing* the task requested by the robot. Given the similar structure, this SHA is extended through the same add-ons as the HumanApplicant pattern described in Sect. 4.2.1. Deciding to help the robot and move to its location is an action initiated by the human, extended through the Free Will add-on. It is the robot, instead, that instructs the human to begin the task when they are sufficiently close: The *walking* operational state is, thus, eligible for the Heed/Ignore add-on. The human may be distracted by concurrent tasks before assisting the robot; thus, the Timer Expired add-on imposes an upper bound on the time they take to reach the robot from their initial location. As in the Applicant pattern, a critical situation may occur that requires safety measures to be enforced, while the human is performing the required task, which is captured by an additional $\langle \mathsf{op} \rangle\_\mathsf{pub}_h$ instance, subject to the Safety Violation add-on. Finally, as in previous patterns, the Critical Status add-oncaptures the possibility of unexpected accidents, while the human supports the robot.

5 Evaluation of human errors' impact

This section reports on a set of formal verification experiments that were carried out to assess the added value of the erroneous behavior add-ons to the model-driven development framework. Three scenarios, inspired by the healthcare setting and featuring three different robotic missions (i.e., sequences of services), have been developed through the model-driven framework presented in Sect. 2. We perform design-time analysis with the formal model devoid of erroneous behavior add-ons and, subsequently, with the extended SHA modeling human behavior presented in Sect. 4.2. The comparative analysis allows us to observe how different human errors impact different robotic missions and how introducing this aspect into the mission's design process can guide the practitioner toward forward-looking management choices.

Table 6 Scenarios used for the validation phase (abbreviation, detailed description, and sequence of services constituting the mission)

Scenario	Description	Mission
treat1patient	The robot (Tbot) serves a patient-doctor pair (P1/D2, respectively). The robot meets the patient by the entrance (ENTR) and *leads* them to the waiting room (WRb) to wait for the doctor to visit them. The robot *follows* the doctor to CUP1 where they fetch required tools, and *follows* them back (carrying the tools) to the treatment room (TR) where the patient will receive the treatment. Finally, the robot returns to WRb and *escorts* the patient to TR, where the doctor is waiting	P1 Follower, D2 Leader, D2 Leader, P1 Follower
treat2patients	The robot (Tbot) serves two patient-doctor pairs (P1/D2 and P3/D4). The robot meets P1 by the entrance (ENTR) and *leads* them to the waiting room (WRa), then it performs the same task for P3 *leading* them from the entrance to WRb. The robot fetches the first required medical kit from CUP1 and *delivers* it to D2 at OFF1. The robot then serves D4 by *following* them to CUP2 and back to their office (OFF3) while carrying the kit. Finally, the robot *leads* P1 to OFF1 and P3 to OFF3 as both doctors are ready to visit them	P1 Follower, P3 Follower, D2 Recipient, D4 Leader, D4 Leader, P1 Follower, P3 Follower
wardEmergency	The robot (Tbot) serves a doctor patient pair (P5/D6), while a second doctor (D7) is active on the same floor. The robot *escorts* P5 to the waiting room. Then it *competes* with D6 for a resource in CUP1. If the robot wins the competition (referred to as **PLAN a**), it requires D6's *help* in opening the office door and then *delivers* them the fetched item in OFFICE. If D7 wins (referred to as **PLAN b**), D6 *leads* the robot to CUP2 to fetch the required item and has the robot carry it back to the office. Irrespective of the competition outcome, when D6 is ready to treat the patient, the robot *escorts* P5 from the waiting room to the office and then *assists* D6 in administering the medication	P5 Follower, D7 Competitor, **PLAN a:** D6 Rescuer D6 Recipient **PLAN b:** D6 Leader D6 Leader P5 Follower D6 Applicant

5.1 Evaluation scenarios

The developed scenarios capture a service robot assisting Doctor/Patient pairs (one or multiple) and are hereinafter referred to as treat1patient, treat2patients, and wardEmergency. Scenarios are designed to capture realistic robotic missions featuring the complete set of services, highlight the flexibility of the overall framework, and test the impact of erroneous behavior models in a wide range of situations.

Scenarios treat1patient and treat2patients are set in the floor layout in Fig. 11. Figure 11 shows the planimetry of the third floor of Building 22 of Politecnico di Milano, whose areas are featured in the two scenarios as three doctors' offices, a waiting room, and a treatment room. Figure 12 depicts the layout for wardEmergency, as presented in [39], featuring a T-shaped corridor with a waiting room, a doctor's office, and two rooms with cupboards containing medical kits. Table 6 summarizes the missions captured by each scenario. Although our framework supports multi-robot teams [39], the three scenarios feature only one robot, indicated as Tbot, since this paper focuses on human behavior modeling. In all three scenarios, the employed service robot is a TurtleBot3 Waffle Pi,[2] with an initial charge of 90% to ensure that it is sufficiently charged to complete each mission. Patients are identified as P1, P3 in treat1patient and treat2patients, and P5 in wardEmergency, and exhibit critical fatigue pro-

Table 7 Developed erroneous behavior profiles

	Dis./ Ob.	Fr.W.	T.Exp.	S.Viol.	Cr.St.
Normal	O	O	O	O	O
Inattentive	CR	O	O	O	O
Focused	NS	O	O	O	O
Busy	O	CR	CR	O	O
Available	O	NS	NS	O	O
Inexperienced	O	O	O	CR	O
Experienced	O	O	O	NS	O
Critical	O	O	O	O	CR
Stable	O	O	O	O	NS

Each profile has an identifier and the associated likelihood of occurring for each add-on

O ordinary, *NS* not significant, *CR* critical

[2] Technical specification available at https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/.

files, specifically Young/Sick for P1 and P5 and Elderly/Sick for P3. Doctors are identified as D2, D4 in treat1patient and treat2patients, and D6, D7 in wardEmergency, and all exhibit less critical fatigue profiles than the patients, specifically Elderly/Healthy for D2, and Young/Healthy for D4, D6, and D7. We recall that fatigue profiles impact the rate at which humans fatigue and recover, i.e., the values of parameters λ and ρ in Eq. 3 and Eq. 2, respectively.

As per Fig. 1, scenarios are configured through the DSL[3], which is then automatically translated into the SHA network through the tool available at [41]. The generated formal model and set of queries are subject to SMC,[4] performed through Uppaal v.4.1.26 on a machine with 4 cores and 16GB of memory. Performance data are reported and discussed at the end of this section.

5.2 Evaluation results

The evaluation aims to illustrate—through the three example scenarios in Table 6—how the framework's design-time analysis is enriched by the introduction of erroneous behaviors. Each add-on features one or multiple parameters that can be tuned to calibrate the likelihood of the corresponding erroneous behavior. Such parameters are recapped in the following: (a) probability weights heed and ignore for the Heed/Ignore add-on; (b) thresholds $\mathsf{FW}_{\mathsf{th}}$ and $\mathsf{FW}_{\mathsf{max}}$ for the Free Will add-on (T equals constant $\mathsf{T}_{\mathsf{poll}}$ in our modeling approach); (c) allowance factor δ for the Timer Expired add-on, which determines the value of upper bound TE; (d) exponential rate λ_{safe} for the Safety Violation add-on; (e) rate hs and constant FS for the Critical Status add-on.

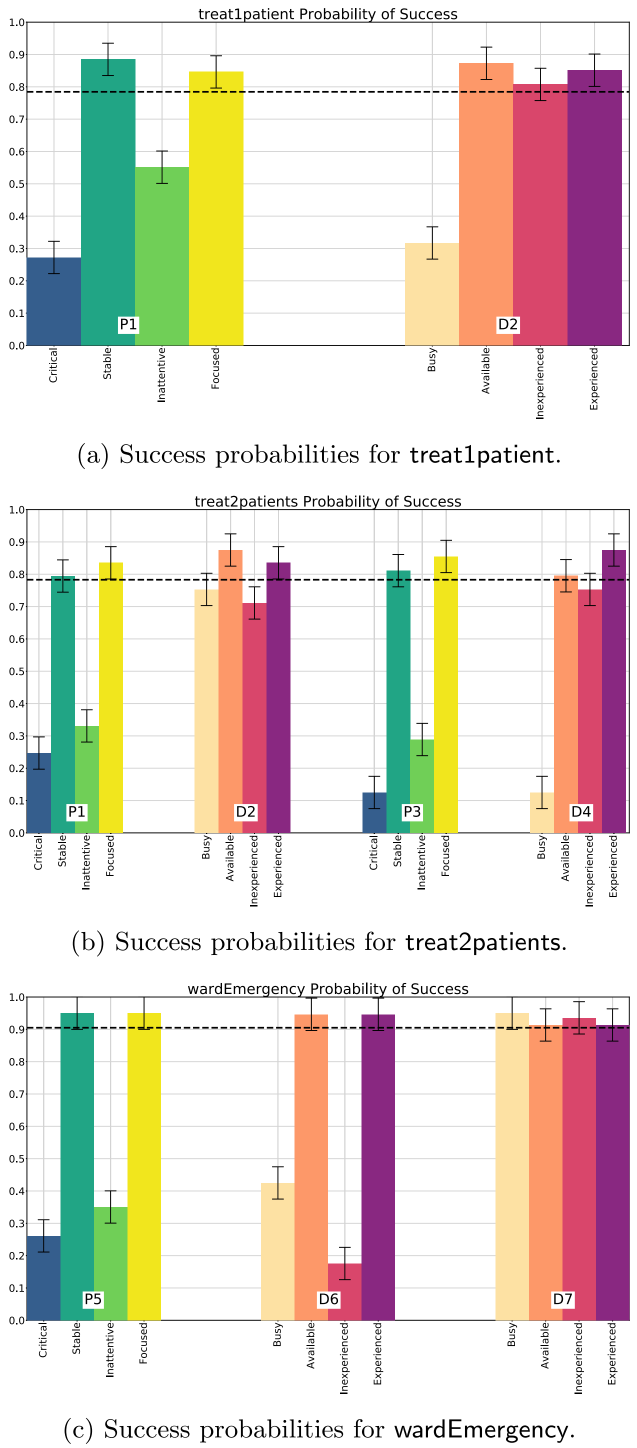
To exhaustively investigate the impact of each error on a specific scenario, it would be necessary to compute the probability of success for all possible values of such parameters within their respective domain. However, to keep the duration of a design-time analysis round within an acceptable range, we group possible values for each add-on into three macro-categories, similarly to control modes classification in the CREAM technique [32]. The identified levels are: ordinary (O), critical (CR), and not significant (NS) error likelihood. As with fatigue profiles, we have identified 9 profiles of erroneous behavior, each elevating (or dampening) the likelihood of one or multiple specific errors. Erroneous behavior profiles are summarized in Table 7. Given the acknowledged lack of empirical data for techniques analyzing human behavior such as Human Reliability Analysis (HRA) [18], the specific parameter values are not extracted from real datasets but arbitrarily chosen for this analysis. However, the purpose of this validation is not to assess the accuracy of SMC results against real empirical data but to illustrate how the introduction of erroneous behaviors impacts design-time results and supports the mission design process. Therefore, the lack of real data has limited consequences on the significance of the obtained results.

For the impact analysis of the three scenarios, we calculate the mission's success probability range (the y-axis in Fig. 13) by applying in turn a different erroneous behavior profile to each human subject (the x-axis in Fig. 13). For instance, the left-most bar of Fig. 13a reports the success probability range of treat1patient with the Critical profile applied to P1, while D2 is set to Normal. For each scenario, as explained in Sect. 3.1, we calculate the probability of success within a time bound τ through expression $\mathbb{P}(\diamond_{\leq\tau}\mathrm{scs})$. Verification is iterated by changing the erroneous behavior profile for one human subject at a time, while the value of τ remains unchanged. For the first iteration, all humans are assigned the Normal profile (see Table 7), representing the standard probability of success (the dashed horizontal lines in Fig. 13), also referred to as the *baseline*. We recall that, as explained in Sect. 3.1, SMC results are of the form $[\mathrm{p}-\epsilon, \mathrm{p}+\epsilon]$, representing the confidence interval to which the real success probability belongs. All SMC experiments have been performed with Uppaal's default statistical parameters, specifically the width of the estimated confidence interval (see Sect. 3.1) is set to $\epsilon = 0.05$. In Fig. 13, the height of each bar equals the value of p obtained for the corresponding experiment, while black lines represent the 2ϵ-wide confidence interval. Note that none of the baseline success probability estimations is exactly 100%. The first reason behind this result is that the Normal behavioral profile features an *average* likelihood for all errors (while these are made more or less prominent by the other profiles); therefore, errors have a non-null impact on the success probability also when calculating the baseline. Secondly, time bounds (parameter τ in each scenario) are chosen so that, should the success probability be calculated with an error-free model (i.e., without any add-on), it would equal the maximum value allowed by Uppaal with this set of parameters, which is $[0.95, 1]$. Even in this case, the SMC experiment does not yield *exactly* 100% for the probability of success because the result must be a confidence interval in any case. (Thus, it yields the feasible half of the confidence interval with $\mathrm{p} = 1$.)

Since human subjects have different roles (i.e., either professionals or patients), not all profiles realistically apply to every subject. Specifically, verification is performed with patients (subjects P1, P3, and P5) cycling between Critical, Stable, Inattentive and Focused profiles. This set of profiles represents the fact that patients are more susceptible than professionals to accidents (captured by the Critical Status add-on) and prone to ignore the robot's instructions either due to lack of familiarity with the technology or to inattention due to their condition and surrounding environment

[3] Source DSL files available at https://github.com/LesLivia/hri_dsl/tree/main/hridsl_sources/SoSym_cs.

[4] Models available at https://doi.org/10.5281/zenodo.7754156.



(a) Success probabilities for treat1patient.

(b) Success probabilities for treat2patients.

(c) Success probabilities for wardEmergency.

Fig. 13 Bar plots reporting the estimated probability of success ($[0-1]$) for the three scenarios. Each bar represents the estimation with a different erroneous behavior profile grouped by the human subject. Dashed lines represent the success probability estimated with all human subjects' profiles set to Normal (see Table 7) (color figure online)

(captured by the Heed/Ignore add-on). On the other hand, professionals (subjects D2, D4, D6, and D7) rotate between Busy, Available, Inexperienced, and Experienced profiles. In this case, healthcare professionals are more likely to act in a hectic environment, effectively pushing them to either rush through a task (i.e., one of the phenotypes captured by the Free Will add-on) or start working on different tasks than the one involving interaction with the robot, thus exceeding the maximum allowed time bound (captured by the Timer Expired add-on). Moreover, professionals with little experience working alongside a robot are more likely to erroneously step out of a safe operational state when a critical situation is still in place (captured by the Safety Violation add-on). Note that the described pairings between subjects and behavioral profiles are only conceived for the purposes of this analysis and do not reflect actual limitations of the approach. (All profiles, including combinations of them, are applicable to any human agent, irrespective of their role.)

Figure 13a displays the results for scenario treat1patient. With $\tau = 700\,\mathrm{s}$, in 5 cases out of 8 the probability of success is essentially unchanged (if not higher) compared to the one calculated in standard conditions, which is approximately 80%. Concerning human P1, these results are due to the nature of the profiles themselves: Both Stable and Focused are *positive* profiles, as they feature lower likelihood of erroneous behaviors than the Normal profile. The same conclusion can be drawn about the Available and Experienced profiles for D2. On the other hand, the probability of success is also unaffected by the Inexperienced profile. As a matter of fact, D2 *leads* the robot in treat1patient and, since their walking speed is higher than the robot's (a healthy human walks at about 1.4m/s, whereas the robot moves at a maximum speed of 0.26m/s) it is unlikely for the human to walk too closely to the robot and trigger the enforcement (and subsequent erroneous violation) of the safety measure.

As shown in Fig. 13a, scenario treat1patient is most affected by the Critical and Inattentive profiles for P1, and Busy for D2. As explained in Sect. 4.2.2, the HumanFollower interaction pattern, which P1 adheres to, is susceptible to both the Critical Status and Heed/Ignore add-ons (influenced by the Critical and Inattentive profiles), which cause the probability of success to drop to approximately 25% and 55%, respectively. To address this issue, the practitioner designing the mission may decide to adopt additional monitoring measures regarding the patient's health status or have them walk a shorter distance to reduce the impact of unexpected accidents. The result might also lead the designer toward solutions improving the robot's communication capabilities that increase the patient's attention level. Concerning D2, the Busy profiles causes a 50% drop in the success probability: To address this issue, a possible design choice is to assign the mission to a different employee with a clearer schedule.

Similar conclusions can be drawn about P1 and D4 in scenario treat2patients, whose results are reported in Fig. 13b. The estimated success probability with $\tau = 1500\,\mathrm{s}$ in standard conditions is approximately 80%. In this case, the

Critical Status add-on is more impactful for patient P3 compared to P1 due to the more critical fatigue profile, causing a steeper growth of probability weight p_d. The resulting success probability is slightly above 10% (compared to 25% for P1). These results can guide the practitioner in modifying the plan of the mission to reduce the physical burden on the two patients, especially P3: for example, by having the robot lead them, whenever possible, straight to the treatment room rather than to the waiting room first. On the other hand, since the Heed/Ignore add-on has no correlation with the evolution of fatigue, the Inattentive profile has a comparable impact on the mission's outcome when applied to P1 and P3. In this case, the same reconfiguration measures discussed for subject P1 in scenario treat1patient may be applied.

In scenario treat2patients, although the same set of behavioral profiles are applied to D2 and D4, the different interaction patterns they participate in (i.e., HumanRecipient and HumanLeader) are differently influenced by error phenotypes. More specifically, the Busy profile has a significantly larger impact when applied to D4 rather than D2. As described in Sect. 4.2.4, the HumanRecipient does not feature any instance of the Timer Expired add-on, whereas the Free Will add-on, which, like the Timer Expired add-on is made more prominent by the Busy profile, allows the subject to move, *while* the robot is fetching the required object. Therefore, the human erroneously moving causes the robot to adjust the target of the delivery task to the new human's position, which does not necessarily result in a delay of the completion of the service nor lowers the success probability within time bound τ. Indeed, given the starting position of D2 (also shown in Fig. 11), it is more likely for them to move *closer* to CUP1 (the required object's location) than farther, leading to only a slight decline of the success probability (approximately 75% compared to 80% in standard conditions). For subject D4, instead, since they also participate in a HumanLeader pattern like subject D2 in scenario treat1patient, the Busy profile has a very significant impact leading to a 70% drop in the success probability compared to ordinary conditions. The Inexperienced profile (which increases the likelihood of the Safety Violation add-on) has a comparably limited impact when applied both on D2 and D4. Concerning D4, the same conclusions drawn about the HumanLeader pattern for scenario treat1patient also apply in this case. As for the HumanRecipient pattern, D2 can enter the critical *interacting* operating state at most for the amount of time required to pick up the item from the robot. Consequently, the likelihood of erroneously ignoring the safety measure leading to a collision during the interaction is also limited. In conclusion, the practitioner does not need to consider specific design choices concerning D2, while D4 is affected by the same guidelines discussed for scenario treat1patient.

Estimated success probabilities for scenario wardEmergency are shown in Fig. 13c, all calculated with $\tau = 600$s. In standard conditions, the mission ends successfully with a 90% probability. Patient P5 exhibits a similar trend to that observed for P1 and P3 in scenarios treat1patient and treat2patients, as the Critical and Inattentive profiles cause a drop of the success probability of approximately 65% and 55%, respectively.

On the other hand, given the same set of behavioral profiles, the trend is different for D6 and D7 compared to subjects covering the role of professionals in previous scenarios. We recall that, given the presence of a HumanCompetitor pattern, in this case, the robotic mission features two alternative plans depending on whether the human or the robot wins the competition [39], both summarized in Table 6. If the robot loses, D6 is involved in two HumanLeader interaction patterns, whose dependency on different behavioral profiles has already been discussed for subjects D2 in treat1patient and D4 in treat2patients. Otherwise, D6 participates in a HumanRecipient, HumanRescuer sequence. The initial position of D7 is randomized to make the outcome of the competition unpredictable. As observed in scenario treat2patients, the HumanRecipient pattern (involving subject D2) is only slightly affected by both *negative* profiles, while the outcome of the HumanRescuer pattern (see Sect. 4.2.6) is impacted by the Free Will, Timer Expired, and Safety Violation add-ons. The impact of the Busy profile, which makes the first two add-ons more prominent, on D6, is an average between the drop it causes on **PLAN a** (the impact of profile Busy is low for the HumanRecipient and high for the HumanRescuer patterns) and **PLAN b** (the impact of profile Busy is low for both instances of the HumanLeader pattern), resulting in an overall approximate 45% drop compared to the baseline. Concerning the Inexperienced profile, both the HumanRescuer (featured by **PLAN a**) and HumanApplicant patterns (featured by both plans) are highly susceptible to the Safety Violation add-on, which can occur throughout the entire duration of the task they perform jointly and in close distance with the robot. Therefore, unlike in scenarios treat1patient and treat2patients, the Inexperienced profile leads to a larger success probability drop (more than 70%) than the Busy profile. After examining these results, the practitioner designing the robotic mission may either assign the mission to a more experienced employee or invest in thorough training of the personnel in charge of performing tasks alongside the robot.

Finally, confirming the modeling choices discussed in Sect. 4.2.5, the HumanCompetitor pattern, in which subject D7 participates, is the least influenced by erroneous behaviors. This trait of the pattern is reflected by the results in Fig. 13c, showing that the success probability does not significantly change with respect to the baseline, irrespective of the erroneous behavior profile assigned to D7. As a matter of fact, should D7 perform any erroneous action, this does not result in a failure of the service nor a delay of the overall mission, but rather it favors the *victory* of the robot in the competition. Therefore, the erroneous actions of D7 indirectly influence the outcome of the mission as they result in **PLAN a** being enacted more often than **PLAN b**, so erroneous behaviors with a larger impact on **PLAN a** also have a larger impact on the mission in its entirety.

As previously mentioned, we have selected a subset of behavioral profiles for each human subject to perform this impact analysis for the three scenarios, representing the most realistic contingencies. Consequently, we have performed four verification experiments (resulting in different success probability estimations) for each human subject plus the baseline, so 9 experiments for treat1patient, 17 for treat2patients, and 13 for wardEmergency. With the described parameter set, verification ends in 66.72 min for treat1patient, 133.95 min for treat2patients, and 104.64 min wardEmergency. (These durations refer to the *cumulative* time required to complete *all* experiments—thus the complete analysis of errors' impact—for each scenario.) Given the flexibility of the scenario configuration phase, the practitioner can modulate the number of experiments to be performed (thus, the duration of the design-time analysis round). Modulation is performed by selecting the combinations of behavioral profiles found to be more critical or more likely to be observed in their specific application.

5.3 Limitations of the analysis

The formal verification experiments highlight the versatility of the approach and its potential impact on the design process of robotic applications. Still, the analysis has some limitations. Mainly, the lack of data recording human actions (which in part originates from the technology not being widely deployed in practice) does not enable an assessment of the accuracy of the formal analysis with respect to the behavior of real subjects. Therefore, the latter remains an open research question which is left for future developments.

The analysis has been performed under the assumption that detected deviations from the expected behavior are due to actual human errors and not to imprecise sensor readings. However, should real data be incorporated into the analysis, potential sensor errors should also be taken into account.

Finally, there is scientific evidence of a correlation between human reliability and fatigue [25], of which the Critical Status add-on represents an initial investigation. The work should be further extended in this direction to modulate the probability of humans acting erroneously depending on the level of fatigue they experience. The issue is particularly relevant for safety–critical service sectors such as healthcare, in which long and stressful shifts are common.

6 Related works

Modeling human behavior is a long-standing issue in human–automation interaction analysis. With the advent of collaborative robotics, the issue has recently started to attract attention in its declination to human–*robot* interaction. Unforeseen human actions, especially those originating from errors, hugely impact the design of general human–machine interaction, especially with robots [7, 17]. In the field of human–robot interaction, most works investigate human errors as sources of safety *hazard* (e.g., leading to a collision with the robot), which is the core issue tackled by Human Reliability Analysis [19, 29] and probabilistic risk assessment techniques [28, 56]. Given the complexity of the human mind and the human decision-making process, a perfectly accurate, all-encompassing model of human behavior is not feasible. However, existing works in the literature, mainly originating from research on human cognition, propose mathematical models of human behavior within specific boundaries, for example, limited to decision-making in the workplace. These models fall into three main categories [6]: (1) **cognitive** models investigate the mental process leading to a certain decision; (2) **task-analytic** models capture human behavior as a hierarchy of actions; (3) **probabilistic** models refine the non-determinism of human behavior through probability distributions over actions.

Well-established cognitive models are Soar [35] and Adaptive Character of Thought (ACT-R) [4]. However, as discussed in Sect. 4, cognitive sources behind human behavior are out of the scope of our model-driven framework; therefore, cognitive models are not further investigated. Task-analytic models such as ConcurTaskTrees (CTT) [45], although recently expanded with a taxonomy of human errors [10], suffer from the drawback of being intrinsically case study-specific, thus hardly reusable. Probabilistic models are considered highly beneficial in designing cyber-physical systems where human factors are critical [16]. Some examples of probabilistic models are Boltzmann rationality [8], the LESS model [9], and Markovian models such as Partially Observable Markov Decision Processes (POMDPs) [50], and Bayesian Networks [54]. The main issue of probabilistic models is the lack of extensive and reusable datasets to train reliable probability distributions [18]. However, although our work does employ a probabilistic model of human behavior (i.e., SHA), it partially works around this issue by performing design-time analysis as a *function* of probabilistic parameters, as discussed in Sect. 5. Nevertheless, collecting real observations of human behavior while participating in the analyzed scenarios would still be necessary to provide compelling evidence of the formal model's accuracy.

Previous works propose a formalization of erroneous human behavior models for formal verification. Cerone et al. [12] propose a taxonomy of operator errors in human–

computer interaction formalized through the CSP process algebra and temporal logic. Shin et al. [51] present a formal model of human material handlers in manufacturing systems, depending on human tasks and errors modeled through part-state graphs. Rukšėnas et al. [49] present a verification framework for interactive systems with cognitive models of human errors under timing constraints based on the Goals, Operators, Methods, and Selection (GOMS) methodology [31]. Askarpour et al. [7] present an automated risk assessment technique for collaborative robotic applications, in which the eight phenotypes identified by Hollnagel are expressed through logic formulae and verified through the Zot formal verification tool.

As discussed in Sect. 1, the service sector has different demands than the industrial settings, especially in healthcare, since it is characterized by a higher degree of human task diversity and more significant sources of uncertainty [43, 52]. Therefore, given the different target domains and underlying formalisms, the results obtained in [7] cannot be directly embedded into our framework. However, the observations in [7] about the efficacy of Hollnagel's phenotypes constitute the foundation for the work presented in this article, which adapts phenotypes to behaviors observed in service settings and integrates them with a stochastic characterization.

7 Conclusion

This article extends a model-driven framework for developing interactive service robotic applications, presented in [37, 39], with a formalization of erroneous human actions. The extension is particularly valuable to address sources of uncertainty due to the significant presence of humans, especially in safety–critical domains. Designers can modulate the analysis to their needs and estimate the probability of success with different behavioral profiles for their subjects. Statistical techniques lead to a faster exploration of the resulting configuration space. Designers can thus perform a more exhaustive analysis of the impact of different configurations and reach more informed decisions on how to train and manage the personnel based on individual characteristics and level of expertise.

As discussed in Sect. 5.3, the main limitation of the work is the lack of empirical observations of real human behavior. The availability of such data would allow for an accurate estimation of the likelihood of each erroneous action. This work tackles the issue by performing the evaluation as a *function of* such parameters, specifically by sampling three levels of severity from their domain. Given the flexibility of the add-ons, discussed in Sect. 4.1, should accurate estimations become available, these can be easily incorporated into the formal model.

In the future, we plan on further investigating the issue of formal modeling human–robot interaction and human behavior in service settings, which can follow different directions. Firstly, it is possible to incorporate *complex* phenotypes of erroneous human behavior identified by Hollnagel's taxonomy since add-ons are currently limited to *simple* phenotypes. Moreover, as discussed in Sect. 4, the modeling approach will be expanded to cover alternative *functionally equivalent* actions (e.g., "walking" and "running") to support, as a consequence, the Repetition and Replacement phenotypes. Finally, a comparative analysis could be carried out between a model based on phenotypes (i.e., the one presented in this paper) and one based on genotypes, exploiting cognitive models of human decision-making existing in the literature, comparing the expressiveness, and should empirical data become available, how accurately they capture human actions.

Author Contributions LL contributed to conceptualization, investigation, methodology, supervision, and writing—original draft, and provided software; AM was involved in investigation and methodology and provided software; DM contributed to investigation and methodology and provided software; LM was involved in investigation and methodology and provided software; MA contributed to conceptualization, supervision, and writing—review and editing; MMB was involved in conceptualization, supervision, and writing—review and editing; MR contributed to conceptualization, supervision, and writing—review and editing.

Funding Open access funding provided by Politecnico di Milano within the CRUI-CARE Agreement. No funding was received to assist with the preparation of this manuscript.

Availability of data and materials The datasets generated and/or analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of interest The authors have no financial or proprietary interests in any material discussed in this article.

Ethical approval Not applicable. (This article does not contain any studies with human participants or animals performed by any of the authors.)

Consent to participate Not applicable. (This article does not contain any studies with human participants or animals performed by any of the authors.)

Consent for publication All authors have approved the manuscript and agree with its publication on Software and Systems Modeling (SoSyM).

Code availability The code developed during the current study is available from the corresponding author on reasonable request.

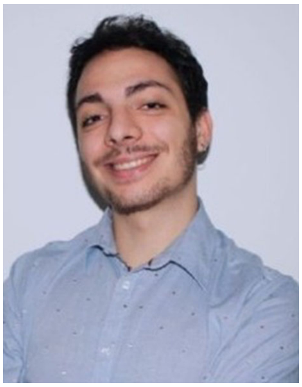
References

1. Agha, G., Palmskog, K.: A survey of statistical model checking. TOMACS **28**(1), 1–39 (2018)
2. Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T.A., Ho, P.H., Nicollin, X., Olivero, A., Sifakis, J., Yovine, S.: The algorithmic analysis of hybrid systems. TCS **138**(1), 3–34 (1995)
3. Alur, R., Feder, T., Henzinger, T.A.: The benefits of relaxing punctuality. J. ACM (JACM) **43**(1), 116–146 (1996)
4. Anderson, J.R.: ACT: a simple theory of complex cognition. Am. Psychol. **51**(4), 355 (1996)
5. Arenis, S.F., Vujinovic, M., Westphal, B.: On implementable timed automata. In: Formal Techniques for Distributed Objects, Components, and Systems, Volume 12136 of Lecture Notes in Computer Science, Valletta, Malta, pp. 78–95. Springer (2020)
6. Askarpour, M.: How to formally model human in collaborative robotics. In: Second Workshop on Formal Methods for Autonomous Systems (2020)
7. Askarpour, M., Mandrioli, D., Rossi, M., Vicentini, F.: Formal model of human erroneous behavior for safety analysis in collaborative robotics. Robot. Comput. Integr. Manuf. **57**, 465–476 (2019)
8. Baker, C.L., Tenenbaum, J., Saxe, R.R.: Goal inference as inverse planning. In: Proceedings of the Annual Meeting of the Cognitive Science Society, vol. 29 (29) (2007)
9. Bobu, A., Scobee, D.R., Fisac, J.F., Sastry, S.S., Dragan, A.D.: Less is more: rethinking probabilistic models of human behavior. In: International Conference on Human–Robot Interaction, pp. 429–437 (2020)
10. Bolton, M.L., Molinaro, K.A., Houser, A.M.: A formal method for assessing the impact of task-based erroneous human behavior on system safety. Reliab. Eng. Syst. Saf. **188**, 168–180 (2019)
11. Calude, C., Kroon, F., Poznanovic, N.: Free will is compatible with randomness. Philos. Inq. **4**(2), 37–52 (2016)
12. Cerone, A., Lindsay, P.A., Connelly, S.: Formal analysis of human–computer interaction using model-checking. In: International Conference on Software Engineering and Formal Methods, pp. 352–361. IEEE (2005)
13. Clopper, C.J., Pearson, E.S.: The use of confidence or fiducial limits illustrated in the case of the binomial. Biometrika **26**(4), 404–413 (1934)
14. David, A., Larsen, K.G., Legay, A., Mikučionis, M., Poulsen, D.B.: Uppaal SMC tutorial. STTT **17**(4), 397–415 (2015)
15. David, A., Larsen, K.G., Legay, A., Mikucionis, M., Poulsen, D.B., van Vliet, J., Wang, Z.: Statistical model checking for networks of priced timed automata. In: Formal Modeling and Analysis of Timed Systems, Volume 6919 of Lecture Notes in Computer Science, Aalborg, Denmark, pp. 80–96. Springer (2011)
16. De Felice, F., Zomparelli, F., Petrillo, A.: Functional human reliability analysis: a systems engineering perspective. In: CIISE, pp. 23–29 (2017)
17. Degani, A., Heymann, M.: Formal verification of human–automation interaction. Hum. Factors **44**(1), 28–43 (2002)
18. Di Pasquale, V., Iannone, R., Miranda, S., Riemma, S.: An overview of human reliability analysis techniques in manufacturing operations. Oper. Manag. **9**, 978–953 (2013)
19. Dougherty, E.M., Fragola, J.R.: Human Reliability Analysis. Wiley, New York (1988)
20. EU Robotics: Robotics Multi-Annual Roadmap (2020). https://eu-robotics.net/divi_overlay/roadmap/
21. Frey, C.B., Osborne, M.A.: The future of employment: How susceptible are jobs to computerisation? Technol. Forecast. Soc. Change **114**, 254–280 (2017)
22. García, S., Strüber, D., Brugali, D., Berger, T., Pelliccione, P.: Robotics software engineering: a perspective from the service robotics domain. In: ESEC/FSE, USA, pp. 593–604. ACM (2020)
23. Givi, Z., Jaber, M.Y., Neumann, W.P.: Modelling worker reliability with learning and fatigue. Appl. Math. Model. **39**(17), 5186–5199 (2015)
24. Grenander, U.: Stochastic processes and statistical inference. Ark. Mat. **1**(3), 195–277 (1950)
25. Griffith, C.D., Mahadevan, S.: Inclusion of fatigue effects in human reliability analysis. Reliab. Eng. Syst. Saf. **96**(11), 1437–1447 (2011)
26. Hollnagel, E.: The phenotype of erroneous actions: implications for HCI design. In: Human–Computer Interaction and Complex Systems, pp. 73–121 (1991)
27. Hollnagel, E.: The phenotype of erroneous actions. Int. J. Man Mach. Stud. **39**(1), 1–32 (1993)
28. Hollnagel, E.: Cognitive Reliability and Error Analysis Method (CREAM). Elsevier, Amsterdam (1998)
29. Hou, L.X., Liu, R., Liu, H.C., Jiang, S.: Two decades on human reliability analysis: a bibliometric analysis and literature review. Ann. Nucl. Energy **151**, 107969 (2021)
30. ISO 13482: Robots and robotic devices—safety requirements for personal care robots. ISO (2014)
31. John, B.E., Kieras, D.E.: The GOMS family of user interface analysis techniques: comparison and contrast. ACM Trans. Comput. Hum. Interact. **3**(4), 320–351 (1996)
32. Kim, M.C., Seong, P.H., Hollnagel, E.: A probabilistic approach for determining the control mode in CREAM. Reliab. Eng. Syst. Saf. **91**(2), 191–199 (2006)
33. Konz, S.: Work/rest: part II—the scientific basis (knowledge base) for the guide 1. EGPS **1**(401), 38 (2000)
34. Kwiatkowska, M.Z., Norman, G., Parker, D.: PRISM 4.0: verification of probabilistic real-time systems. In: Computer Aided Verification, Volume 6806 of Lecture Notes in Computer Science, Snowbird, UT, USA, pp. 585–591. Springer (2011)
35. Laird, J.E.: The Soar Cognitive Architecture. MIT Press, Cambridge (2019)
36. Larsen, K.G., Pettersson, P., Yi, W.: UPPAAL in a nutshell. Int. J. Softw. Tools Technol. Transf. **1**(1–2), 134–152 (1997)
37. Lestingi, L., Askarpour, M., Bersani, M.M., Rossi, M.: Formal verification of human–robot interaction in healthcare scenarios. In: International Conference on Software Engineering and Formal Methods, pp. 303–324. Springer (2020)
38. Lestingi, L., Askarpour, M., Bersani, M.M., Rossi, M.: A deployment framework for formally verified human–robot interactions. IEEE Access **9**, 136616–136635 (2021)
39. Lestingi, L., Romeo, G., Sbrolli, C., Scarmozzino, P., Bersani, M.M., Rossi, M.: Formal modeling and verification of multi-robot interactive scenarios in service settings. In: International Conference on Formal Methods in Software Engineering (2022)
40. Lestingi, L., Zerla, D., Bersani, M.M., Rossi, M.: Specification, stochastic modeling and analysis of interactive service robotic applications. Robot. Auton. Syst. **163**, 104387 (2023)
41. Lestingi, L.: HRI Design-Time Analysis (2020) https://github.com/LesLivia/hri_designtime

42. Liu, B., Ma, L., Chen, C., Zhang, Z.: Experimental validation of a subject-specific maximum endurance time model. Ergonomics **61**(6), 806–817 (2018)
43. Lyons, M., Adams, S., Woloshynowych, M., Vincent, C.: Human reliability analysis in healthcare: a review of techniques. Int. J. Risk Saf. Med. **16**(4), 223–237 (2004)
44. Miyazawa, A., Ribeiro, P., Li, W., Cavalcanti, A., Timmis, J., Woodcock, J.: RoboChart: modelling and verification of the functional behaviour of robotic applications. Softw. Syst. Model. **18**(5), 3097–3149 (2019)
45. Paternò, F., Mancini, C., Meniconi, S.: ConcurTaskTrees: a diagrammatic notation for specifying task models. In: Human–Computer Interaction, pp. 362–369. Springer (1997)
46. Pocock, S., Harrison, M., Wright, P., Johnson, P.: THEA: a technique for human error assessment early in design. In: International Conference on Human Computer Interaction, Newcastle University (2001)
47. Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: ROS: an open-source robot operating system. In: ICRA Workshop on Open Source Software, Volume 3, Kobe, Japan, p. 5. IEEE (2009)
48. Reason, J.: Actions not as planned: the price of automatisation. Asp. Conscious. **1**, 67–89 (1979)
49. Rukšėnas, R., Curzon, P., Blandford, A., Back, J.: Combining human error verification and timing analysis: a case study on an infusion pump. Formal Aspects Comput. **26**(5), 1033–1076 (2014)
50. Schmidt-Rohr, S.R., Losch, M., Dillmann, R.: Human and robot behavior modeling for probabilistic cognition of an autonomous service robot. In: International Symposium on Robot and Human Interactive Communication, pp. 635–640. IEEE (2008)
51. Shin, D., Wysk, R.A., Rothrock, L.: Formal model of human material-handling tasks for control of manufacturing systems. IEEE Trans. Syst. Man Cybern. Part A Syst. Hum. **36**(4), 685–696 (2006)
52. Sujan, M.A., Embrey, D., Huang, H.: On the application of human reliability analysis in healthcare: opportunities and challenges. Reliab. Eng. Syst. Saf. **194**, 106189 (2020)
53. Swain, A.D., Guttmann, H.E.: Handbook of human-reliability analysis with emphasis on nuclear power plant applications. Final report. Technical report, Sandia National Labs., Albuquerque, NM (USA) (1983)
54. Tenorth, M., De la Torre, F., Beetz, M.: Learning probability distributions over partially-ordered human everyday activities. In: International Conference on Robotics and Automation, pp. 4539–4544. IEEE (2013)
55. Vicentini, F., Askarpour, M., Rossi, M.G., Mandrioli, D.: Safety assessment of collaborative robotics through automated formal verification. IEEE Trans. Rob. **36**(1), 42–61 (2019)
56. Williams, J.: A data-based method for assessing and reducing human error to improve operational performance. In: Conference Record for 1988 IEEE Fourth Conference on Human Factors and Power Plants, pp. 436–450. IEEE (1988)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Livia Lestingi is a postdoctoral researcher at Politecnico di Milano. She earned a PhD in Information Technology from Politecnico di Milano in 2023. Her research interests include the analysis of human–robot interaction through formal methods and software engineering techniques for complex cyber-physical systems.



Andrea Manglaviti completed the EIT double MSc Degree in Human–Computer Interaction and Design from Politecnico di Milano and KTH in 2023. He specializes in software development and UI/UX and is currently working as a Software Engineer.

Davide Marinaro completed the EIT double MSc program in Data Science from Politecnico di Milano and KTH in 2023. He specializes in machine learning, distributed systems, and data mining for big data. He is currently working as a Data Analytics Engineer.

Luca Marinello completed the EIT double MSc program in Data Science from Politecnico di Milano and Université de Rennes 1 in 2022. He specializes in machine learning and is currently working as a Data Scientist and Software Engineer, with an interest in IoT Industry.

Mehrnoosh Askarpour is an adjunct assistant professor at McMaster University. Her current research interests include verification of safety–critical system properties and application of formal methods for safe robotics and autonomous vehicles.

Marcello M. Bersani is an associate professor at Politecnico di Milano. His research interests are mainly focused on formal methods, temporal logic, and verification.

Matteo Rossi is an associate professor at Politecnico di Milano. His research interests are in formal methods for safety–critical and real-time systems, architectures for real-time distributed systems, and transportation systems both from the point of view of their design, and of their application in urban mobility scenarios.