SPECIAL ISSUE TITLE
Editor: Name, xxxx@email

Model-Driven Development of Service Robot Applications Dealing with Uncertain Human Behavior

Livia Lestingi
Dipartimento di Elettronica, Informazione e Bioingegneria at Politecnico di Milano, Milan, Italy.

Marcello M. Bersani
Dipartimento di Elettronica, Informazione e Bioingegneria at Politecnico di Milano, Milan, Italy.

Matteo Rossi
Dipartimento di Meccanica at Politecnico di Milano, Milan, Italy.

***Abstract*—In the future, robots will interact with humans in highly variable and unpredictable settings, such as healthcare and home assistance. Frameworks are, therefore, fundamental to analyze and develop interactive robotic applications that can deal by design with the uncertainty of human behavioral and physiological features. Our framework is built upon formal modeling, verification, and learning techniques providing sound mathematical guarantees of human wellbeing preservation throughout the interaction and timely mission completion despite the uncertainties at play. The framework's workflow makes it accessible to professional figures without expertise in formal methods, and the high degree of automation minimizes the manual effort required throughout the toolchain. The development toolchain has been tested on use cases from the healthcare setting, in which human behavior is unconstrained, and subjects are often in critical mental or physical conditions.**

EMERGING TECHNOLOGIES are bound to transform the service sector in the upcoming years. In particular, robots are rapidly evolving from mere high-performing industrial equipment to sophisticated machines making decisions in delicate and uncertain situations. Tasks involving personal care or domestic assistance, once considered reserved for human workers, are now susceptible to robotization. Recent studies estimate that roles such as healthcare support workers and personal care aides have a 60-74% chance of being automated, and the percentage grows to 96% for receptionists and information clerks [1].

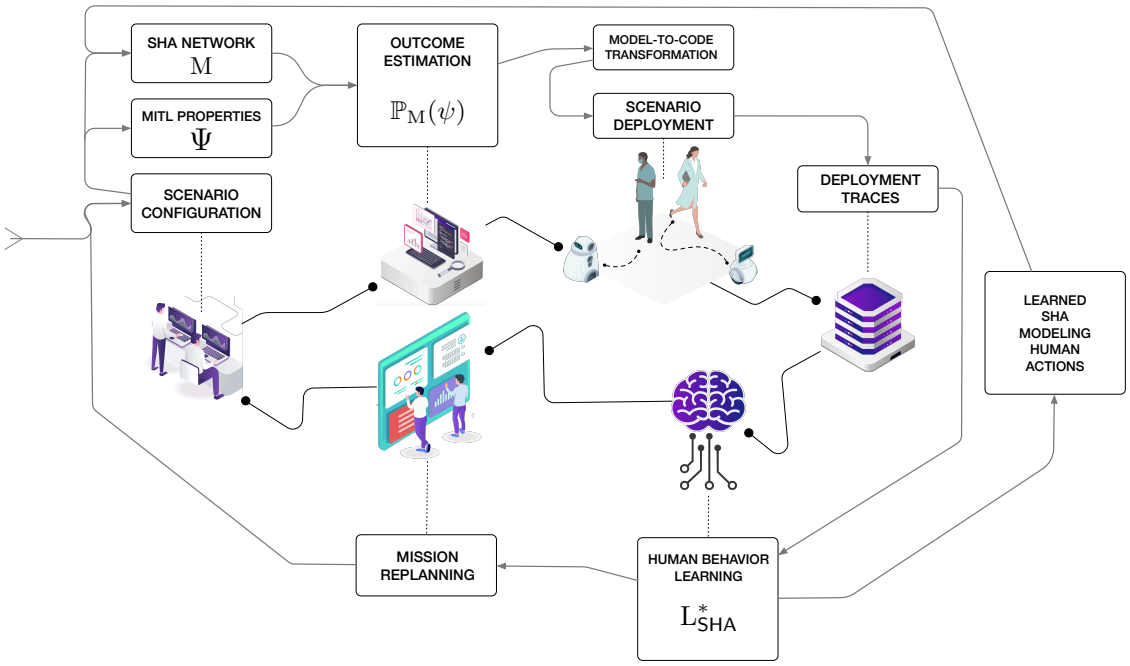

Figure 1. Diagram representing the workflow of the model-driven framework. The approach has a cyclical structure, starting with the configuration of the interactive scenario and, subsequently, the evaluation of its outcome. The scenario is deployed, and the collected deployment traces are used to learn a model of human actions. The updated model is used to reconfigure the robot mission, if necessary.

Robots are already being deployed as caregivers, performing tasks such as delivery services, human activity monitoring, and physical therapy. Nevertheless, they are far from fully integrated into service settings. For robots to successfully provide services to human customers, especially in healthcare environments, it is paramount that the robot's reasoning factors the recipient's mental and physical state. Service settings are demanding due to their high variability degree as human behavior is essentially **unconstrained**, unlike factories where human workers cycle through known sets of actions. Therefore, practitioners require tools to develop *reliable* robotic applications incorporating *flexible* models of human behavior that can evolve based on field observations [2]. This article presents a **model-driven framework** for the development of interactive robotic scenarios in service settings. The framework covers scenarios set in a known and fixed floorplan, featuring mobile robots (thus, industrial manipulator applications are out-of-scope) where human behavior is, instead, partially unknown. The framework, whose complete workflow is shown in Fig. 1, supports practitioners from the early design stage of the robotic application to the final deployment.

The framework exploits formal analysis to reliably estimate the probability of success of human-robot interaction scenarios while remaining accessible to designers without expertise in formal modeling. To this end, the formal model exploits a state-based formalism, also capturing the behavior of humans interacting with the robot. The formal model features a mathematical model of human **behavioral** and **physiological** aspects measured through suitable sensors (in our specific case, physical fatigue). Existing modeling and verification approaches dealing with the uncertainty of human behavior mainly adopt either a *game-based* or a *probabilistic* approach. Game-based approaches model human-robot interaction as an adversarial game, in which human moves are unpredictable, and the robot must be programmed with a proper reaction to any possible contingency [3], [4]. Probabilistic approaches primarily exploit formalisms such as Markov Decision Processes (MDPs) to model human behavior

and Probabilistic Model Checking to compute the probability that the human performs specific actions and the robot is able to complete the task [5]. The proposed approach, first introduced in [6], [7], is the first to combine a *stochastic* formalization of human decisions (thus, falling into the probabilistic category) with a *continuous-time* model of physiological variables with *non-linear* dynamics (specifically, Ordinary Differential Equations constraining muscular fatigue), featuring a model of the scenario as a network of **Stochastic Hybrid Automata** (SHA) [8], [9].

A novel **automata learning** algorithm, introduced in this article for the first time, allows for a seamless refinement of the human model based on data collected at runtime, dampening the sources of uncertainties that affect the design-time analysis. The algorithm, $\mathsf{L}^*_{\mathrm{SHA}}$, combines hypothesis testing with time series analysis to learn an SHA modeling human actions. The learned SHA captures humans' sequence of actions in a specific scenario, which is partially unknown when first planning the robotic mission. Learning about human actions is the subject of research lines such as behavior informatics [10] and interaction learning [11], and it could be tackled through machine learning techniques (e.g., deep neural networks). However, the output of these approaches is not strictly compatible with formal verification nor easily interpretable, in contrast to the framework's application domain requirements. Existing automata learning algorithms in the literature also target probabilistic models (specifically, MDPs) and learn the probability that, while in a specific state, the human will perform a specific action [12]. Other works target Hybrid Automata and exploit clustering techniques to identify when two signal segments have comparable dynamics (i.e., they capture the same state of the system) [13]. $\mathsf{L}^*_{\mathrm{SHA}}$ is the first automata learning algorithm combining the stochastic and hybrid aspects. We present a range of case studies from the healthcare setting illustrating the effectiveness of the framework and how the automata learning phase reduces the error of design-time results.

Design-Time Analysis

The entry point to the framework is the **configuration** of the interactive scenario under analysis. Through a custom, user-friendly textual

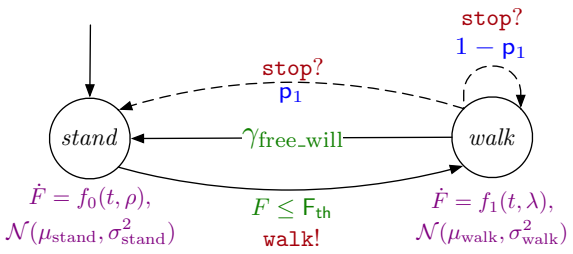

Figure 2. Example of SHA modeling human behavior. Flow conditions and probability distributions are in purple, actions in red, guard conditions in green, and probability weights in blue.

notation, the user defines: how many agents (i.e., the humans and the robots) are at play, their characteristics and how each human interacts with the robot. The framework covers a set of **patterns** capturing recurring human-robot interactions (e.g., *leader-follower* dynamics) and is open to extension thanks to the underlying formalism. User-defined parameters automatically customize the formal model capturing all the system entities: the mobile robot providing services, the humans receiving such services, and the layout of the building where they operate.

The formal model is an SHA network, representing the **agents**: robots and their batteries, humans, and *orchestrators*—the SHA managing the robotic mission by sending instructions to the agents. The network captures the **synchronization** between the agents as they interact through messages signaling the start or end of an *action*. Agents' behavior changes based on their current state: for instance, while resting, humans behave differently than while walking.

Fig. 2 shows an example SHA modeling human behavior in our scenarios. The human can either *stand* or *walk*, modeled by as many *locations*. The SHA captures the evolution of **muscular fatigue**, modeled by real-valued time-dependent variable F. Each location l is labeled with a *flow condition* (i.e., the differential equations constraining physical variables while in l) and the probability distribution of random parameters. In Fig. 2, each location is characterized by either flow $f_0(t, \rho)$ or flow $f_1(t, \lambda)$, where ρ and λ are the recovery and fatigue rates, which are normally distributed [14].

Agents switch from one location to another

when they are notified that an action x started (corresponding to label x?) or by actively signaling the start of an action x (label x!). For example, upon switching from $stand$ to $walk$, when sufficiently rested (i.e., when fatigue F is smaller than threshold $\mathsf{F_{th}}$, which guard $F \leq \mathsf{F_{th}}$ captures), the SHA modeling the human notifies the orchestrator through label walk!. Also, the human *receives* the instruction to *stop* walking from the orchestrator, captured by label stop?. The dashed edge back to $stand$ captures the possibility of humans ignoring instructions or making autonomous decisions out of **free will**. When the orchestrator issues instruction stop!, the human abides with probability $\mathsf{p_1}$ and ignores it (thus, the location does not change) with probability $1 - \mathsf{p_1}$. The SHA also captures through the solid edge transition the human switching back to $stand$ irrespectively of the orchestrator's instructions. This switch may fire when guard $\gamma_{\text{free\_will}}$ holds, whose Boolean value results from a coin toss approximating human free will [15]. The developed SHA are thoroughly described in [6].

Given their stochastic nature, SHA are eligible for Statistical Model Checking (SMC) [16]. Instead of exhaustively exploring the state space as in traditional model-checking, SMC applies statistical techniques to a set of samples (i.e., "traces") decreasing verification times significantly. Each trace is checked against the desired property, expressed in Metric Interval Temporal Logic (MITL). In the specific case of human-robot interaction, we estimate probability $\mathbb{P}_{\mathrm{M}}(\diamond_{\leq\tau}\mathsf{scs})$ of the mission *eventually* (operator $\diamond$) ending with success (Boolean variable scs) within time-bound τ given SHA network M, whose value is a range $[p-\varepsilon, p+\varepsilon]$. If a trace satisfies (resp., does not satisfy) the property, it is considered a *success* (resp., a *failure*): the set of traces is thus mapped to a sample set of a Bernoulli variable. SMC relies on a procedure that iteratively generates traces by simulating the SHA network through a Monte Carlo approach. For the procedure to terminate, a criterion to stop generating the traces must be in place. The decision revolves around range $p \pm \varepsilon$, to whom the *real* success rate belongs: the smaller the range, the more accurate the analysis needs to be and the more traces are required.

The framework exploits Uppaal to formally model the scenario and run SMC experiments [9]. Uppaal is a tool for modeling and verifying real-time systems, recently extended to cover stochastic timed systems and SMC. In Uppaal, $p \pm \varepsilon$ is calculated through the Clopper-Pearson method [17], and the tool stops generating traces, concluding the SMC experiment when ε is smaller than a user-defined parameter.

When the SHA model is ready, and SMC results are available, the designer assesses the resulting success rate range. If the probability of success is insufficient, the designer reconfigures the scenario and iterates the procedure to increase the chances of success. Possible reconfiguration measures include assigning the mission to a different robot, changing the order in which humans are served, or changing the destination of a service if no logical constraints forbid it. Otherwise, the design-time analysis is complete, and the application is ready for deployment.

Application Deployment

The framework features a **code generation** technique described in detail in [7], mapping each element of the SHA network to an element of a deployment unit. A *deployment unit* is an executable script replicating the behavior of one of the agents—robot or human—or of the orchestrator managing the execution of the mission. The key requirement for the deployment phase is that the model-to-code transformation is such that, at runtime, the behavior of the system matches the estimates obtained at design time.

The deployment approach allows for flexible interaction between the cyber and physical layers of the robotic application. Also, the deployment environment (i.e., the agents instrumented with suitable sensors and the layout) can be **entirely virtual** or **hybrid**. In the latter case, real robotic devices interact with virtual humans in the simulation scene in compliance with the **digital-twin** paradigm. Human avatars can be fully automated via simulator scripts or respond to manual commands sent by the designer to have realistic decision-making also in simulation. Since the framework's domain includes healthcare and home assistance settings, human subjects in the scenarios are often in critical physical or mental

conditions. The hybrid deployment setting allows analysts to validate the design-time results with hundreds of runs without placing a significant burden on real subjects, which would call into question the ethicality of the validation approach.

Human Behavior Learning

The model of human behavior, of which a snippet is shown in Fig. 2, is usually an underapproximation of real human behavior. In our framework, SHA modeling human actions discriminate between different locations based on how fatigue evolves. Therefore, only human behaviors that impact fatigue are relevant to the model (e.g., *speaking* or *eating* do not meet this criterion, but *walking* does). Complex human behaviors (besides standing and walking) for a specific scenario are typically only **partially known** when drafting the formal model the first time, possibly leading to design choices that do not fully safeguard the subjects' wellbeing. However, the eligible range of human actions is too broad to produce an all-encompassing SHA *manually*. Even if this were possible, it would result in an unreasonably complex model (thus, long verification times) since, given a specific scenario, not all states are feasible, or the probability of subjects performing specific actions may be negligible.

To overcome this issue, the framework exploits sensor logs and action sequences (referred to as *traces*) collected at runtime to **learn** a more accurate SHA. We have developed an **automata learning** algorithm, called $\mathsf{L}^*_{\mathrm{SHA}}$, that infers an SHA from traces featuring observations of human behavior throughout the interaction with the robot. $\mathsf{L}^*_{\mathrm{SHA}}$ extends the well-known algorithm L^* for Deterministic Finite Automata learning [18] to also learn *stochastic* and *hybrid* features. The resulting SHA modeling human behavior is *up-to-date* with the accumulated knowledge and is plugged back into the SHA network to repeat the design-time analysis.

$\mathsf{L}^*_{\mathrm{SHA}}$ relies on the interaction between a **learner** and a **teacher**. Learning occurs in rounds and, during each round, hypothesis SHA $\mathcal{A}_{\mathrm{hyp}}$ modeling the system under learning is progressively updated. The learner stores $\mathcal{A}_{\mathrm{hyp}}$ and iteratively refines it by submitting **queries** to the teacher. The teacher stores the knowledge accumulated about the system—i.e., the collected traces. When the learner submits a query, the teacher extracts information from the traces to reply. At the end of each round, $\mathcal{A}_{\mathrm{hyp}}$ must satisfy the following two properties:

a) there exists a location featuring a flow condition and probability distribution for all the fatigue dynamics observed in the traces analyzed up to that point ($\mathcal{A}_{\mathrm{hyp}}$ is **closed**);
b) there is no location with more than one outgoing edge labeled with the same action ($\mathcal{A}_{\mathrm{hyp}}$ is **consistent**).

Queries determine how the system behaves after a specific action sequence s occurs for each trace stored in the teacher. For SHA, the system's behavior depends on which flow conditions constrain physical variables (e.g., $f_0(t, \rho)$ in Fig. 2) and which probability distributions yield random parameter values (e.g., $\mathcal{N}(\mu_{\mathrm{stand}}, \sigma^2_{\mathrm{stand}})$ in Fig. 2).

$\mathsf{L}^*_{\mathrm{SHA}}$ features four types of queries. To identify the flow conditions, the learner submits **model-fitting** queries ($\mathsf{mf}(s)$). In the SHA modeling humans, flow conditions constrain the real-valued variable corresponding to fatigue. The teacher examines the collected fatigue signals and identifies the function that best fits them out of a pre-determined set through the Derivative Dynamic Time Warping approach [19].

The learner submits **hypothesis testing** ($\mathsf{ht}(s)$) queries to identify probability distributions. The teacher performs a Kolmogorov-Smirnov test [20] to determine whether the collected observations of the random parameters (in our case, the rest and fatigue rates ρ and λ) are samples of a known distribution with the required confidence level. Otherwise, a new distribution is identified. For example, when the learner first queries about the system behavior for a sequence ending with action "*sit*," the teacher will find that, in such location, fatigue decreases exponentially (thus, according to flow condition $f_0(t, \rho)$), but the resting rate samples are not statistically compatible with action "stand" modeled by means of $\mathcal{N}(\mu_{\mathrm{stand}}, \sigma^2_{\mathrm{stand}})$. Humans, indeed, recover more quickly while sitting than while standing: a new distribution $\mathcal{N}(\mu_{\mathrm{sit}}, \sigma^2_{\mathrm{sit}})$ is thus identified, resulting in a new SHA location.

$\mathsf{L}^*_{\mathrm{SHA}}$ is an **active** learning algorithm: if the teacher finds that the observations of action se-

quence s are not sufficient to answer query $\mathsf{ht}(s)$, it can *actively* ask the system for new traces by submitting a **knowledge refinement** query (ref). The method of generating traces depends on the system under scrutiny. In our case, when deploying the application, it is not feasible to enforce the occurrence of a specific action sequence programmatically. Therefore, $\mathsf{L}^*_{\mathrm{SHA}}$ is suspended until new traces featuring sequence s are available.

At the end of each learning round, the learner asks the teacher through a cex query if there is an action sequence for which sufficient observations are available and which leads to a system state different from the one envisaged by $\mathcal{A}_{\mathrm{hyp}}$. A trace satisfying this criterion is called a **counterexample**. Specifically, trace t is a counterexample if:

a) actions in t lead to a system's state exhibiting fatigue properties—flow and rate distribution—that do not match any location of the hypothesis automaton;
b) the addition of edges corresponding to the actions in t would make the current hypothesis automaton not *consistent*.

If the teacher finds a counterexample, the learner performs a new learning round to refine $\mathcal{A}_{\mathrm{hyp}}$. If the teacher cannot find a counterexample, $\mathcal{A}_{\mathrm{hyp}}$, i.e., the SHA learned up to that point, is consistent with the available traces, and $\mathsf{L}^*_{\mathrm{SHA}}$ terminates.

Healthcare Case Studies

We illustrate the effectiveness of the approach while developing robotic applications through six experimental scenarios from the healthcare setting. The experiments showcase how the framework supports the designer from an early design stage in analyzing and deploying a wide range of interactive robotic tasks with different categories of human subjects. We also discuss how the learning procedure based on data collected at runtime reduces the error of verification results when exploiting automata learned through a batch of deployment traces to estimate the outcome of new scenarios.

Experimental Setting

Scenarios take place in a floor layout made up of a hall (with an area of approximately $60\mathrm{m}^2$) connected to two side wings (approximately $100\mathrm{m}^2$ each), with doors leading to a waiting room, an examination room, cupboards containing medical kits, and doctors' offices.

Scenarios DP0 through DP2 involve a Doctor-Patient pair with a healthcare professional (e.g., a doctor or a nurse) and a patient, each requesting two services provided by one mobile robot. Scenario DP3 features two doctors and two patients, each requesting two services, served by one robot. The last scenario, MR (Multi-Robot), features two mobile robots serving three humans (two doctors competing for a resource and a patient). Patients (indicated as Px in Table 1) require the robot to lead them to the waiting room, while professionals (indicated as Dx in Table 1) require assistance in fetching and delivering resources or carrying tools while following them to the examination room. When the examination room is ready, the robot escorts the patient to be visited and assists the professional in administering the medication (i.e., by holding or handing over the required tools).

We assume that healthcare professionals are healthy, while patients exhibit a more critical fatigue profile (i.e., fatigue rate λ is higher than the doctors' by one order of magnitude). In MR, the initially active robot's charge level is insufficient to complete the mission, requiring the task to be handed over to another robot. Furthermore, scenario MR features two doctors competing for the same medical kit (for example, during an emergency). This situation highlights how the framework handles missions with alternative outcomes (i.e., scenarios MR1 and MR2) depending on who wins the competition and collects the item first.

Validation Process

To validate the framework, for each experimental scenario, we follow the workflow presented throughout the article. The application designer configures the scenario (i.e., the layout, the involved agents, humans and robots, and the tasks constituting the robotic mission), and the tool automatically generates SHA network M. For the first round of design-time analysis, the SHA network features the manually drafted version of automata modeling human behavior, of which Fig. 2 shows a snippet. Subsequently, SMC experiments are performed to estimate the probability of success of the robotic mission for

CS	τ	Success Probability ([%]) $\mathbb{P}_{\mathrm{M}}(\diamond_{\leq\tau}\mathsf{scs})$				Complexity $\lvert\mathrm{M}\rvert$ ([$\times 10^3$])		Verification Time [min]		HUM	Exp. Fatigue ([%]) $\mathrm{E}_{\leq\hat{\tau}}[\max(F_{\mathsf{HUM}})]$			
		DT-1	DT-2	DEPL	$\Delta\mathrm{E}_p(\tau)$	DT-1	DT-2	DT-1	DT-2		DT-1	DT-2	DEPL	$\Delta\mathrm{E}_f(\hat{\tau})$
DP0	340s	95.0 ± 5	83.5 ± 5	80.5	-14.3	175.9	267.5	6.74	14.4	P1	24.6 ± 4	39.2 ± 7	41.4	-35.2
	260s	56.5 ± 5	44.5 ± 5	41.3	-29.1			4.50	10.0	D1	5.4 ± 1	7.4 ± 1	6.8	-11.4
	180s	12.5 ± 5	9.48 ± 5	8.75	-34.5			2.47	6.11					
DP1	600s	95.0 ± 5	79.3 ± 5	74.0	-21.2	175.9	267.5	16.4	34.8	P1	20.4 ± 3	38.4 ± 7	31.1	-10.7
	500s	58.6 ± 5	46.7 ± 5	40.0	-29.8			9.09	20.9	D1	5.5 ± 1	6.2 ± 1	5.88	-1.0
	400s	6.5 ± 5	5.0 ± 5	5.0	-30.0			4.67	14.8					
DP2	580s	95.0 ± 5	75.3 ± 5	81.0	-10.2	179.3	291.5	14.8	32.0	P1	21.7 ± 1	41.0 ± 3	34.2	-16.7
	520s	60.2 ± 5	45.5 ± 5	49.0	-15.7			8.08	14.7	D1	7.6 ± 2	13.4 ± 2	10.9	-7.5
	460s	6.67 ± 5	11.0 ± 5	10.0	-23.3			4.75	8.34					
DP3	1500s	93.7 ± 5	74.1 ± 5	80.0	-9.8	193.4	397.0	31.6	92.1	P1	22.6 ± 4	32.7 ± 2	28.6	-13.2
	1400s	74.2 ± 5	55.6 ± 5	61.0	-12.8			24.9	70.9	D1	5.9 ± 1	4.4 ± 3	5.1	-2.0
	1300s	37.9 ± 5	25.2 ± 5	28.0	-25.4			20.9	54.3	P2	51.9 ± 3	62.9 ± 9	61.7	-13.9
	1200s	6.5 ± 5	5.0 ± 5	5.0	-30.0			18.5	37.6	D2	2.2 ± 1	2.8 ± 1	2.4	8.3
MR1	1200s	95.0 ± 5	85.1 ± 5	89.0	-2.4	371.7	626.8	26.0	89.9	P1	39.4 ± 3	48.3 ± 4	46.9	-12.9
	1000s	63.5 ± 5	45.5 ± 5	51.0	-13.7			16.3	51.6	D1	2.8 ± 1	2.9 ± 1	3.1	-3.2
	800s	27.9 ± 5	23.7 ± 5	22.0	-19.1			12.9	33.4	D2	1.2 ± 1	1.8 ± 1	1.5	6.9
	700s	6.89 ± 5	6.34 ± 5	6.0	-9.2			8.70	14.5					
MR2	1200s	91.2 ± 5	81.3 ± 5	80.0	-12.4	368.3	602.8	28.7	83.8	P1	35.7 ± 2	43.6 ± 2	40.4	-11.6
	1000s	58.6 ± 5	42.4 ± 5	47.0	-14.9			17.1	50.4	D1	4.0 ± 3	7.4 ± 3	5.8	-3.4
	800s	24.5 ± 5	18.7 ± 5	21.0	-5.7			13.7	37.5	D2	1.5 ± 1	1.8 ± 1	1.6	5.0
	700s	7.68 ± 5	6.81 ± 5	5.0	-17.4			7.46	12.9					

Table 1. Experimental results obtained from the case studies (CS). For each scenario, the complexity of the SHA network ($\lvert\mathrm{M}\rvert$) is the cumulative size of its SHA (one for each agent), which is a function of the number of locations, edges, and of the cardinality of variables' domains. When reporting success probability and expected fatigue estimations, the darker the cells the worse the metric is (success probability is low or fatigue level is high). When reporting error differences ($\Delta\mathrm{E}_p(\tau)$ and $\Delta\mathrm{E}_f(\hat{\tau})$), green cells indicate a reduction of the error whereas red cells indicate an increase.

decreasing values of time bound τ (expression $\mathbb{P}_{\mathrm{M}}(\diamond_{\leq\tau}\ \mathsf{scs})$). Success probability ranges are of the form $p \pm \varepsilon$: we recall that SMC ends when $\varepsilon \leq \varepsilon_{\mathrm{th}}$ holds, where $\varepsilon_{\mathrm{th}}$ equals 5% in our case. For all scenarios, we calculate through Uppaal the value of expression $\mathrm{E}_{\leq\hat{\tau}}[\max(F_{\mathsf{HUM}})]$, which is the maximum fatigue reached by each human (indicated as HUM) with the highest time bound considered for that scenario (indicated as $\hat{\tau}$, e.g., 340s in DP0). Fatigue estimations are also of the form $f \pm \varepsilon$; in this case, the calculation ends when Uppaal identifies the value of ε such that the expected maximum fatigue level falls within range $[f - \varepsilon, f + \varepsilon]$ with confidence higher than 95% (thus, the value of ε varies between different subjects). The results of the first round of design-time analysis for all scenarios are shown in Table 1 (columns DT-1).

As a second phase, we exploit the virtual deployment environment to gather a large amount of data without putting a strain on real subjects. Collected traces are fed to $\mathsf{L}^*_{\mathrm{SHA}}$ to learn human behavior. Indeed, through the simulation environment, it is possible to replicate a broader spectrum of human behaviors affecting the fatigue curve throughout the scenario. These actions range from running and carrying loads (which are daily tasks for healthcare professionals) to sitting or standing in an uncomfortable environment (i.e., with a very low or very high room temperature or a high degree of humidity, such as in field hospitals). We perform multiple simulations of scenario DP0 to collect a large batch of runtime observations (i.e., traces), serving as *training* data as human subjects perform a wider range of actions than the ones captured by the initial SHA. For this specific experiment, 5407 traces (with *automated* human avatars) have been collected, each lasting approximately $340s$. To obtain the same amount of data in a physical environment, running the application non-stop for 21 workdays would be necessary. With this pool of traces, $\mathsf{L}^*_{\mathrm{SHA}}$ terminates in approximately 37min returning an SHA modeling human behavior that has five times the number of locations than the original one. The updated SHA modeling human actions is then plugged into the SHA network to repeat the design-time analysis, for all scenarios, with the

refined model (columns DT-2 in Table 1).

The learned model is tested with scenarios different than DP0 (i.e., the source of training data). DP1, DP2, and DP3 are deployed in the hybrid setting with a real robotic device, while MR1 and MR2 are fully simulated. Runtime observations collected while deploying these five scenarios (collectively, 500 runs) serve as *testing* data. To decrease the simulation-to-reality gap of features subject to learning, fatigue signals are extracted from real physical trials and a real user controls the actions performed by subjects in the simulation. This also ensures that training and testing data capture human behavior in comparable conditions. Table 1 reports the success rate and maximum fatigue level observed at runtime (columns DEPL). Testing scenarios are newly verified with Uppaal with the refined model of human behavior to evaluate how the learning phase reduces the estimation error of SMC results, as explained in the following. Let $\overline{\mathrm{X}}_{\text{DT-}i}(\tau)$ be the average of metric X (the success rate or fatigue level) calculated in Uppaal with time bound τ for phase $i \in \{1, 2\}$. Let $\overline{\mathrm{X}}_{\text{DEPL}}(\tau)$ be the average value of metric X observed at *runtime* within time bound τ. Let $\mathrm{E}_{\mathrm{X},i}(\tau)$ be the error between estimation $\overline{\mathrm{X}}_{\text{DT-}i}(\tau)$ and $\overline{\mathrm{X}}_{\text{DEPL}}(\tau)$, calculated as in Eq.1.

$$\mathrm{E}_{\mathrm{X},i}(\tau) = \frac{|\overline{\mathrm{X}}_{\text{DT-}i}(\tau) - \overline{\mathrm{X}}_{\text{DEPL}}(\tau)|}{\overline{X}_{\text{DEPL}}(\tau)} \cdot 100 \quad (1)$$

We indicate as $\Delta\mathrm{E}_{\mathrm{X}}(\tau) = \mathrm{E}_{\mathrm{X},2}(\tau) - \mathrm{E}_{\mathrm{X},1}(\tau)$ the difference between the error with the model learned through $\mathsf{L}^*_{\mathrm{SHA}}$ and the initial model. When $\Delta\mathrm{E}_{\mathrm{X}}(\tau)$ is negative, SMC results obtained through the SHA network (partially) learned by $\mathsf{L}^*_{\mathrm{SHA}}$ are more accurate than the manually drafted network ($\mathrm{E}_{\mathrm{X},1}(\tau) > \mathrm{E}_{\mathrm{X},2}(\tau)$ holds). We remark that, since $\mathsf{L}^*_{\mathrm{SHA}}$ is the first automata learning algorithm targeting SHA, we can internally assess its impact on our design and development framework, but we cannot evaluate its performance against that of alternative algorithms.

Discussion

The source of the large estimation errors of DT-1 experiments is the wider range of actions that humans perform at runtime, which are not captured by the initial SHA network but learned through $\mathsf{L}^*_{\mathrm{SHA}}$ and taken into consideration for DT-2 experiments. To highlight the gap between known human actions before and after the learning phase, Fig. 3 shows how long, on average, the subjects from scenario DP0 spend in each state of the SHA modeling human behavior (e.g., *stand* and *walk* in Fig. 2). The upper bar plot shows the distributions for the initial behavior model, whereas the bottom bar plot shows the distributions for the SHA learned by $\mathsf{L}^*_{\mathrm{SHA}}$. Distributions are obtained by analyzing 1000 traces of the SHA network generated through Uppaal. The plot highlights how human behavior has a significantly higher degree of variability than the one accounted for by the initial model. More specifically, the refined model features locations with more critical fatigue rates distributions (e.g., walking in uncomfortable environments), which

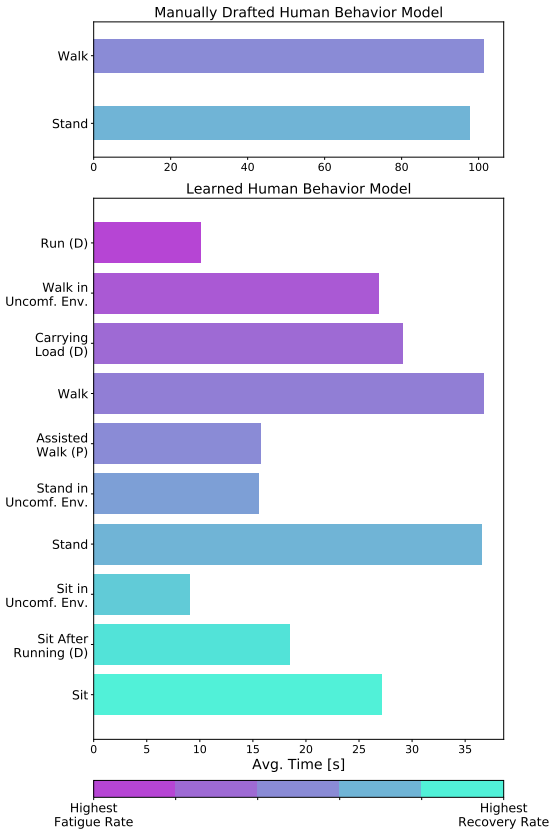

Figure 3. Average time ([s]) spent by the case study subjects in each state of the human behavior model state. States that are only feasible to a professional (resp. patient) are marked with a (D) (resp. (P)). The bars color code is shown at the bottom.

justify the gap between the two rounds of fatigue estimations.

When exploiting the model learned through $\mathsf{L}^*_{\mathrm{SHA}}$ to estimate fatigue for all subjects of scenarios other than DP0 (i.e., the source of traces used for learning), we obtain $\Delta \mathrm{E}_f(\hat{\tau}) < 0$ in 11 cases out of 14, indicating that the learning procedure results in a formal model that more accurately captures reality. The three subjects for which $\Delta \mathrm{E}_f(\hat{\tau}) \geq 0$ holds are professionals with non-critical fatigue profiles, indicating that the learned model may lead to an overestimation of the fatigue level for this category of subjects. We remark that, in both rounds, for these subjects the estimated fatigue levels are not critical ($< 10\%$), thus, small fluctuations do not overturn the result of the design-time analysis.

As shown in Table 1, DT-1 experiments establish that all scenarios can be completed successfully with probability greater than 90% with τ ranging from about 6min (for DP0) to 25min (for DP3). For decreasing values of τ, the success probability decreases as well as the robots and the humans do not have sufficient time to complete their tasks. Nevertheless, as per Table 1, the success rate observed at runtime (column DEPL) is lower than its initial estimation. Fatigue impacts the probability of success since the orchestrator (i.e., the robot's controller) instructs humans to stop and rest when their fatigue level exceeds a critical threshold, leading to a delay in the completion of the mission (thus, the success probability within a certain time bound decreases). The error reduction of fatigue estimation corresponds to an improvement in the success probability calculation. As for success probability, we obtain $\Delta \mathrm{E}_p(\tau) < 0$ for all scenarios and all time bounds.

Reducing the error has a price in terms of verification time. The time required to perform the estimations for phase DT-1, reported in Table 1, ranges from 2.47min to 31.6min, with approximately 400 runs required to conclude the SMC experiment. For phase DT-2, SMC experiments last from 6min to 92min in the worst case. The duration of a verification experiment depends on the duration of the robotic mission—indicated by τ—and the size of the SHA network (indicated as $|\mathrm{M}|$ in Table 1). In this experimental campaign, verification time increases more steeply with τ than with $|\mathrm{M}|$. As Table 1 highlights, smaller (in terms of $|\mathrm{M}|$), longer (in terms of τ) missions (e.g., DP3 with $\tau = 1500$s) take longer to verify than bigger, shorter ones (e.g., MR1 with $\tau = 1200$s).

Future Directions

The framework presented in this article shows promising results in developing interactive scenarios tackling the unpredictability of human behavior and physiological variability with an innovative approach. The approach can pave the way toward deploying assistive robotic applications that smoothly adapt to human needs, significantly promoting the diffusion of the service robotic technology.

As with any verification approach, increasing the scenario complexity inevitably increases verification times. We plan to extend $\mathsf{L}^*_{\mathrm{SHA}}$ to learn probability *weights* on edges (e.g., to learn that a subject might run *and* the likelihood of such action), which might help dampen the growth of verification time with larger human behavior models. The level of support provided to the designer can also be increased by automating the mission replanning phase, specifically by computing alternative mission plans without any user intervention.

REFERENCES

1. C. B. Frey and M. A. Osborne, "The future of employment: How susceptible are jobs to computerisation?" *Technological Forecasting and Social Change*, vol. 114, pp. 254–280, 2017.
2. S. García, D. Strüber, D. Brugali, T. Berger, and P. Pelliccione, "Robotics software engineering: A perspective from the service robotics domain," in *ESEC/FSE*, 2020, pp. 593–604.
3. H. Kress-Gazit, T. Wongpiromsarn, and U. Topcu, "Correct, reactive, high-level robot control," *IEEE Robotics & Automation Magazine*, vol. 18, no. 3, pp. 65–74, 2011.
4. M. M. Bersani, M. Soldo, C. Menghi, P. Pelliccione, and M. Rossi, "PuRSUE-from specification of robotic environments to synthesis of controllers," *Formal Aspects of Computing*, vol. 32, no. 2, pp. 187–227, 2020.
5. S. Junges, N. Jansen, J.-P. Katoen, and U. Topcu, "Probabilistic Model Checking for Complex Cognitive Tasks–A case study in human-robot interaction," *arXiv preprint arXiv:1610.09409*, 2016.

6. L. Lestingi, M. Askarpour, M. M. Bersani, and M. Rossi, “Formal verification of human-robot interaction in healthcare scenarios,” in *SEFM*. Springer, 2020, pp. 303–324.
7. ——, “A deployment framework for formally verified human-robot interactions,” *IEEE Access*, vol. 9, pp. 136 616–136 635, 2021.
8. R. Alur, C. Courcoubetis, T. A. Henzinger, and P.-H. Ho, “Hybrid Automata: An algorithmic approach to the specification and verification of Hybrid Systems,” in *Hybrid Systems*. Springer, 1992, pp. 209–229.
9. A. David, K. G. Larsen, A. Legay, M. Mikučionis, D. B. Poulsen, J. Van Vliet, and Z. Wang, “Statistical Model Checking for networks of Priced Timed Automata,” in *Intl. Conf. on Formal Modeling and Analisys of Timed Systems*. Springer, 2011, pp. 80–96.
10. L. Cao, T. Joachims, C. Wang, E. Gaussier, J. Li, Y. Ou, D. Luo, R. Zafarani, H. Liu, G. Xu, Z. Wu, G. Pasi, Y. Zhang, X. Yang, H. Zha, E. Serra, and V. Subrahmanian, “Behavior informatics: A new perspective,” *IEEE Intelligent Systems*, vol. 29, no. 4, pp. 62–80, 2014.
11. C. Wang, F. Giannotti, and L. Cao, “Learning complex couplings and interactions,” *IEEE Intelligent Systems*, vol. 36, no. 1, pp. 3–5, 2021.
12. C. Ghezzi, M. Pezzè, M. Sama, and G. Tamburrelli, “Mining behavior models from user-intensive web applications,” in *Intl. Conf. on Software Engineering*, 2014, pp. 277–287.
13. R. Medhat, S. Ramesh, B. Bonakdarpour, and S. Fischmeister, “A framework for mining hybrid automata from input/output traces,” in *EMSOFT*. IEEE, 2015, pp. 177–186.
14. B. Liu, L. Ma, C. Chen, and Z. Zhang, “Experimental validation of a subject-specific maximum endurance time model,” *Ergonomics*, vol. 61, no. 6, pp. 806–817, 2018.
15. C. Calude, F. Kroon, and N. Poznanovic, “Free will is compatible with randomness,” *Philosophical Inquiries*, vol. 4, no. 2, pp. 37–52, 2016.
16. G. Agha and K. Palmskog, “A survey of statistical model checking,” *TOMACS*, vol. 28, no. 1, pp. 1–39, 2018.
17. C. J. Clopper and E. S. Pearson, “The use of confidence or fiducial limits illustrated in the case of the binomial,” *Biometrika*, pp. 404–413, 1934.
18. D. Angluin, “Learning regular sets from queries and counterexamples,” *Information and computation*, vol. 75, no. 2, pp. 87–106, 1987.
19. E. J. Keogh and M. J. Pazzani, “Derivative dynamic time warping,” in *Intl. Conf. on Data Mining*. SIAM, 2001, pp. 1–11.
20. H. W. Lilliefors, “On the kolmogorov-smirnov test for normality with mean and variance unknown,” *Journal of the American statistical Association*, vol. 62, no. 318, pp. 399–402, 1967.

Livia Lestingi is a Ph.D. Candidate in Information Technology at Politecnico di Milano. Her research interests include the analysis of complex cyber-physical systems through formal methods and software engineering techniques for service robotics. Contact her at: livia.lestingi@polimi.it.

Marcello M. Bersani is a senior assistant professor at Politecnico di Milano. His research interests are mainly focused on Formal Methods, Temporal logic and Verification. Contact him at: marcellomaria.bersani@polimi.it.

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. Contact him at: matteo.rossi@polimi.it.