Formal Model of Human Erroneous Behavior for Safety Analysis in Collaborative Robotics[☆]

Mehrnoosh Askarpour[1,*] , Dino Mandrioli[1] ,
Matteo Rossi[1] , Federico Vicentini[2]

Abstract

Recent developments in manufacturing technologies, also known as Industry 4.0, seek to build Smart Factories where supply chains and production lines are equipped with a higher level of automation. However, this significant innovation does not entirely eliminate the need for the presence of human operators; on the contrary, it requires them to collaborate with robots and execute hybrid tasks. Thus, creating safe workspaces for human operators is crucial for the future of factories where humans and robots collaborate closely in common workspaces. The uncertainty of human behavior and, consequently, of the actual execution of workflows, pose significant challenges to the safety of collaborative applications. This paper extends our earlier work, a formal verification methodology to analyze the safety of collaborative robotics applications [1], with a rich non-deterministic formal model of operator behaviors that captures the hazardous situations resulting from human errors. The model allows safety engineers to refine their designs until all plausible erroneous behaviors are considered and mitigated. The solidity of the proposed approach is evaluated on a pair of real-life case studies.

Keywords: Formal Verification, Human modeling, Safety analysis, Human-Robot Collaboration

[☆] This paper is an extension of [2].

[*] Corresponding author

Email address: `Mehrnoosh.askarpour@polimi.it` (Mehrnoosh Askarpour)

[1] DEIB, Politecnico di Milano

[2] STIIMA, National Research Council of Italy

1. Introduction

Conventional industrial robotics use safeguarded spaces, which human operators are prevented to access during automatic mode. However, in human-robot collaboration (HRC) the close proximity between humans and robots necessitates a more comprehensive safety analysis due to human actions [51, 49]. The uncertainty of human behavior (e.g., on-the-fly decisions or errors),[3] together with other dynamic factors of HRC (e.g., mobile resources, reconfigurable environment, dynamic workflow of tasks [53]), make it difficult to predict and consider all possible hazardous situations that can arise during the execution of collaborative tasks at early stages of the design [48].

A comprehensive safety analysis should identify hazards automatically in order to overcome partial/wrong assumptions occasionally done in manual procedures. To this end, techniques based on formal models can be used, as they are capable of expressing human-robot interactions in a precise and computable way (e.g., through logic formulae), which allows for the application of algorithms for automated verification.

In earlier works, we introduced a methodology, called SAFER-HRC [1, 4, 5], which translates the informal and goal-oriented description of an HRC application into a logic model. The resulting model, of which fig. 1 gives an overview, captures: the layouts of workcells (L); the kinematics of robots (R); the features of operators (O); the interactions between robots and operators in tasks (T); hazardous situations (see ISO 10218-2 Annex A [11] for a list of hazards); their corresponding risk estimates (according to methodologies codified in ISO/TR 14121-2 [8]);[4] and the risk reduction process.

Among the many classes of hazards, in the model we consider—without loss of generality—mechanical hazards (those resulting in, e.g., crushing, shearing, trapping, impacting, stabbing) because contacts are the newly added situation in physical human-robot interaction, as opposed to conventional robotics. The formal model follows the classification of contact hazards reported in ISO/TS 15066 [43], with a simplification due to standardization needs: transient (`Tr`) and quasi-static (`Qs`) conditions. The first type of contact occurs by sudden, unconstrained, impact between the human and a part

[3] An error is an activity that does not achieve its goal [3].

[4] Methodologies of ISO/TR 14121-2 compute the risk of a hazard as a combination of its severity, frequency, probability and avoidability.

of the robot system, usually with the possibility to recoil from the impact. Contacts of the second type occur when the operator is entrapped between two solid surfaces, one being the robot system moving relatively slowly.

A risk reduction measure is activated when a hazard occurs; it can be of several different types [54], such as changes of the workcell or the workflow, or suspending the robot motion, and so on. Most notably, we give more relevance in the model to the two primary collaborative modes originally introduced in [11] and later detailed in ISO/TS 15066 [43]: power and force limitation (PFL) and speed and separation monitoring (SSM, see also [50]).

The logic language we use for our modeling purposes is called TRIO [9], which is capable of expressing the evolution over time of different phenomena.

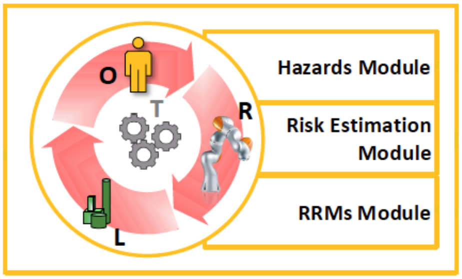


Figure 1: Overview of the formal model and its main components.

Then, an automated verification tool, called Zot [10], is used to explore the state-space of the model and to look for occurrences of hazards with a corresponding high risk. In particular, the model checker looks for any trace of the model which does not satisfy the following property.

Property 1. Always, in case of high risk, a risk reduction measure (RRM) should be applied to mitigate the risk (i.e., to reduce it below a safe level) within α time units.

This paper extends the previous work by enriching the human formal model, to capture human behavior in a more realistic way. In general machinery, and more specifically in collaborative applications, there are two types of observable human activities: (i) *intended*—or *normative*—behaviors refer to the regular use of the system as described by the instructions for use of the application; (ii) *unintended* or *erroneous* behaviors indicate any deviation from the instructions that could be caused by different reasons (e.g., misunderstanding the correct instructions, over/under estimating the functionality of the robot, being tired or absent-minded). Unintended human activities extend the domain of possible behaviors with potentially new hazardous situations that could be normally overlooked. Most of such unintended behaviors result from reasonably foreseeable misuse (see ISO 12100 [7]), which may be difficult to consider when devising hazards. Hence, creating a model that envisions human as more than just an operational element and replicates also human erroneous behavior is critical for a complete safety analysis.

Our previous work [1] viewed human operators from a purely operational perspective, which covered only intended behavior. This paper extends the previous human model by taking into account also erroneous behaviors. In particular, it proposes a formal model of the common human error phenotypes in manufacturing environments and checks it against property 1. We have introduced an early version of this model in [2]. In this paper we provide the full definition of human errors and provide details about their formalization. We have also improved the validation of the model with different case-studies and tackled the scalability issue with a bigger and more complex case study. Additionally, we switched from a 2D representation of the layout to a 3D model, which is more accurate in capturing different situations.

The paper is organized as follows: section 2 overviews the state of the art about human modeling; section 3 gives an introduction to TRIO and Zot; then, section 4 describes the formal model of human erroneous behaviors, and section 5 reports some evaluation results; finally, section 6 concludes and points to important and promising future research steps.

2. Related Work

The models of human observable behavior can be grouped into (i) human-device interface models, (ii) cognitive models, and (iii) task-analytic models [13]. The first group of models, which has been extensively studied in [14], replicate systems in which human and machine interact via interface tools and do not necessarily work in a common workspace. Therefore, human physical safety has never been an issue in these studies [15]. Physiograms [16] are examples of such models. The second group of models account for the cognitive motivations behind any human observable behavior [17]. SOAR [18] and ACT-R [19, 20] are two well-known cognitive models which are not amenable to automated formal verification due to the lack of a precise formal definition. Operator Choice Model (OCM) [21, 22] is another famous example which is suitable only for air traffic control applications. Task analytic models are the third group, in which the executing jobs are broken into a hierarchy of atomic actions. The activities of human operators are also modeled as atomic actions and then the order of those actions w.r.t. actions of the machine is described by if-then logical rules [13]. The best known examples are ConcurTaskTrees (CTT) [23] and their extensions [24], User Action [25] and Operator Function Model [26]. The latter two modeling approaches (cognitive and task-analytic) are able to express co-existence and collaboration of human and robot.

Given that human errors could be a significant source of physical hazards, human reliability analysis [27] has two important aspects: (i) identification of errors and (ii) quantification of their probability. The second item usually is a concern of probabilistic safety assessment techniques such as CREAM [28], HEART [29], THERP [30] and THEA [31]. There are also researches on combining human errors and system failures [32], or miscommunications between multiple human operators [33, 34] and their effect on safety.

Direct modeling of the erroneous behavior of human could facilitate the analysis of safety in human interactions with machines. Therefore, there is a vast literature on defining, formalizing and modeling human errors. Examples in this regard are CCT [35, 36], extended SAL model with empirical data [37], state machines [38] and extended cognitive models [39].

However, these models are usually very case-specific and are not suited to be re-used even for applications of the same type. Then, in the rest of this section we discuss works focusing on patterns and definitions that hold for human errors. Although there is no widely accepted and adopted classification of human errors, some notable references are reported here. [40] divides errors into three groups which are labeled as the following: (i) "location errors" occur when the operator executes an action out of its proper location; (ii) "orientation errors" happen when the operator does not achieve the intended objective of the job at one time (e.g., due to wrong equipment orientation); (iii) "operational errors" are unintended actions performed by humans during a task that delay or halt its execution.

[28] classifies erroneous human actions in eight simple phenotypes: (i) repetition of an action, (ii) reversing the order of actions, (iii) omission of actions, (iv) late actions, (v) early actions, (vi) replacement of an action by another, (vii) insertion of an additional action from elsewhere in the task, (viii) intrusion of an additional unrelated action. These simple phenotypes

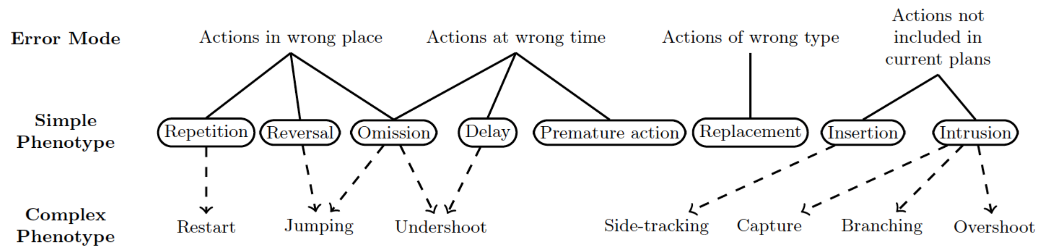


Figure 2: A taxonomy of erroneous actions phenotypes, taken from [41].

are then combined, as shown in fig. 2 (where "wrong place" refers to the action's temporal position in the execution sequence, not to a location in the layout), and create complex phenotypes.

[42] proposes four classes of errors that are pertinent for the assessment of physical safety: (i) slips and lapses, which are performance quality errors; (ii) cognitive errors, which are diagnostic and decision-making mistakes that occur due to a misunderstanding of the instructions by the human operator; (iii) errors of commission, which occur when the operator performs an incorrect or irrelevant activity; (iv) idiosyncratic errors, which depend on social variables and the current emotional state of the operator while performing a task/job.

Another error classification is provided in [3], where errors are categorized as negligible or serious rule violations (e.g., ignoring the instructions). The author studies errors at three levels: behavioral, contextual and conceptual, which answer the "what?", "where?" and "how?" questions, respectively, about an unintended situation.

3. Introducing TRIO and Zot

TRIO is a temporal logic language which assumes an underlying linear temporal structure and features a quantitative notion of time [9]. TRIO formulae are built on the usual first-order connectives, operators, and quantifiers, as well as a single basic modal operator, called Dist, that relates the current time, which is left implicit in the formula, to another time instant: given a time-dependent formula ϕ (i.e., a term representing a mapping from the time domain to truth values) and a (arithmetic) term t indicating a time distance (either positive or negative), formula $\mathsf{Dist}(\phi, t)$ specifies that ϕ holds at a time instant at a distance of exactly t time units from the current one.

While TRIO can exploit both discrete and dense time domains, here we assume the standard model of the nonnegative integers $\mathbb{N}$ as discrete time domain.

TRIO formulae are well-suited for expressing the temporal characteristics of a HRC system that could affect safety. For example, the intensity of a collision between human and robot depends on which of them arrives at the collision point first (e.g., the robot arrives first, and hence is still when the collision happens). These timing differences can be conveniently modeled in TRIO (rather than, say, LTL) by exploiting its derived temporal operators, which are defined from the basic Dist through propositional composition and first-order logic quantification. Table 1 defines some of the most significant ones, including all those used in this work.

As an example, the following formula formalizes in TRIO the informal property 1:

$$\mathsf{Alw}\left(\forall_{i,j,k}\left(\begin{array}{l} risk_{ijk} < 2 \,\vee \\ risk_{ijk} = 2 \Rightarrow \exists y(RRM^{y}_{ijk} \wedge \mathsf{WithinF}\left(risk_{ijk} < 2, \alpha\right)) \end{array}\right)\right) \tag{1}$$

where $risk_{ijk} \in \{0,1,2\}$ is the corresponding risk value for each hazard and subscripts i, j, k are indicators, respectively, of the involved robot system part, human body part (according to ISO 15066 [43] and ISO 7250 [12]), and location of the hazard. Finally, RRM^{y}_{ijk} describes a risk reduction measure, where y indicates its type (power and force limiting or separation monitoring)[5].

The satisfiability of TRIO formulae is in general undecidable. However, in this paper we consider a decidable subset of the language, that can be handled by automated tools, to build the system model and to express its properties.

Zot [10] is a bounded satisfiability checker for TRIO formulae [44]. We use Zot in this work to check the model of the system against desired safety properties. In case the property is not satisfied, Zot provides a counterexample witnessing a system execution that violates the property.

In this work we adopted the TRIO language and its supporting tool Zot not only because of our familiarity with them, but also because of the generality of the language, which can be applied in a natural way to different types

[5] The entire formal model of the two reported test-cases in this paper and an additional one can be found at `github.com/SAFER-HRC`.

Table 1: List of derived TRIO operators; ϕ, ψ denote propositions, and v is a variable and d is a constant value.

Operator	Definition	Meaning
$\mathsf{Futr}\,(\phi, d)$	$d > 0 \wedge \mathsf{Dist}\,(\phi, d)$	ϕ occurs exactly at d time units in the future
$\mathsf{Past}\,(\phi, d)$	$d > 0 \wedge \mathsf{Dist}\,(\phi, -d)$	ϕ occurred exactly at d time units in the past
$\mathsf{AlwF}\,(\phi)$	$\forall t(t > 0 \Rightarrow \mathsf{Dist}\,(\phi, t))$	ϕ holds always in the future
$\mathsf{Alw}\,(\phi)$	$\forall t(\mathsf{Dist}\,(\phi, t))$	ϕ always holds
$\mathsf{SomP}\,(\phi)$	$\exists t(t > 0 \wedge \mathsf{Dist}\,(\phi, -t))$	ϕ occurred sometimes in the past
$\mathsf{Lasted}\,(\phi, d)$	$\forall t(0 < t < d \Rightarrow \mathsf{Dist}\,(\phi, -t))$	ϕ held for the last d time units

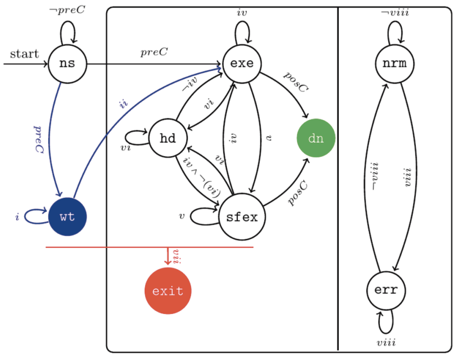
of applications, to describe undecidable models as well as decidable ones, and which can deal with different types of temporal domains[6]. For a comprehensive survey and comparison among various logic languages tailored towards modeling time-dependent phenomena see [45].

Furthermore, Zot exhibits comparable—even superior [46]—performance with respect to other verification tools such as NuSMV [47], which are reference tools for the Bounded Model Checking techniques adopted in this work.

4. Formal Model of human behavior

This section introduces a formal model that replicates common human errors in the manufacturing environment, driven from the operator's perception of the environment and his/her real-time decisions. Here, the cognitive-driven reasons of errors are treated as black boxes, while their consequences, which could lead to hazardous situations and affect human safety, are the center of focus. The proposed model generates execution traces which include both *functional* and *behavioral* human manifestations in the system. The model relies on a series of assumptions and remarks from our earlier works, which are stated below. Each action could be done either by a human or a robot, and during its execution can be in one of seven alternative states, as shown in the automaton-like representation in fig. 3.

[6] Together with MTL, TRIO is one the first temporal logic languages dealing with time in a metric way.



	Formula	Definition
i	$preC \wedge \neg \mathsf{opStarts}$	an operator action does not start before opStarts.
ii	$preC \wedge \mathsf{opStarts}$	operator actions start execution when pre conditions hold and human start to act.
iii	$\exists i, j, k, x(\mathsf{RRM}_{ijk,x})$	a RRM of type x, involving R_i,O_j and L_k, is active.
iv	$\neg(iii) \wedge \neg posC$	*posC* are not fulfilled yet and no RRM is active.
v	$(iii \wedge \neg(vi)) \wedge \neg posC$	*posC* are not fulfilled yet and some RRM is active.
vi	$\exists i, j, k(\mathsf{RRM}_{ijk,hold})$	a RRM of type hold is active.
vii	$\exists i, j, k(\mathsf{risk}_{ijk} > \tau)$	risk exceeds the threshold.
viii	$\mathsf{errTP}_x \vee \mathsf{errS}_x \vee \mathsf{errG}_x$	at least one error is detected.

Figure 3: Semantics of human and robot actions. For readability, some edges (numbered *i*-*viii*) are explained in the table above. For operator actions the blue part replaces the direct transition from `ns` to `exe`. To simplify the figure, red edges which mean that risk is high and execution must abort are shown with a single edge from the set of all the states except `dn` to `exit`.

- `ns` (not started): initial default state which holds until all pre-conditions become true.
- `exit`: given a safety property that the system should satisfy, such as the one in eq. (1), the task execution aborts upon its violation, which corresponds to the detection of hazardous situations with high risk during the lifetime of any of the actions.
- `wt` (waiting): defined only for operator actions, it mimics the time between when all pre-conditions are fulfilled and when the operator actually starts executing the action (*opStarts*).
- `exe` (executing): running state, which is triggered when all pre-conditions are satisfied.
- `sfex` (safe executing): extension of `exe` state with at least one active RRM in order to keep the risk level acceptable.
- `hd` (hold): exception state, entered upon an explicit suspension of execution due to a request from the operator. The state is used when the execution is momentarily paused, although some safety RRMs may be enabled.
- `dn` (done): regular termination state, which is triggered when all post-conditions are satisfied.

Each human action (a_x) has two corresponding attributes $opStarts_x$ and $opStops_x$, which correspond to human mental decisions about starting or stopping the execution of the action depending on what the operator may see, touch or feel. Instead of modeling each of these sensations separately, the decision itself is modeled directly in the model. $spatialC_x$ are spatial requirements of safe execution of action a_x (e.g., where is better for the operator to stand in). They are introduced so that the situations in which operator violates the spatial requirements of an action are detectable. As shown in fig. 3, an action could be in one of behavioral states $\{\texttt{err}, \texttt{nrm}\}$ in parallel with its executional states $\{\texttt{exe},\texttt{sfex},\texttt{hd},\texttt{exit}\}$.

To avoid problems such as complicating the model, producing many false positive situations or state-space explosion, we bound the number of possible human errors during the experiments. Once error types are explained, the manner of bounding them up to a certain value (explained in definition 9) will be easier to understand.

We categorize human errors in three main types:

1. Time-related errors ($errTP$), which occur when the operator does not follow the correct temporal ordering of actions. Eventually these errors lead to an instance of one of the phenotypes introduced in fig. 2.
2. Space-related errors ($errS$) happen when the operator is in the wrong locations or places instruments in wrong locations.
3. Goal-related errors ($errG$) arise when the operator is in the right spot to execute an action and starts the execution with no time-related error, but he/she does not follow instructions correctly and the action is performed poorly. This happens more frequently when the operator is not trained or skillful.

An error can appear due to two different layers of origins: first-order and second-order causes. Errors originate directly from first-order causes, which are the lack of at least one of the following factors: expertise (awareness + experience + trust in design), attention or vigor. First-order causes themselves originate from the state of the operator, who can be fatigued, inattentive, unaware of the instructions, rushing, etc.

The goal of this work is to enrich the list of detected hazards, and to this end considering only first-order causes is enough. Second-order causes, on the other hand, are less relevant during the process of identifying new instances of hazardous situations and assessing their corresponding risks. The relations between errors and their first- and second-order causes are shown in fig. 4.

The rest of this section illustrates how the formal model generates the situations that can arise because of human errors. In fact, formalized human errors should be included in the model to systematically propagate and verify them iteratively.

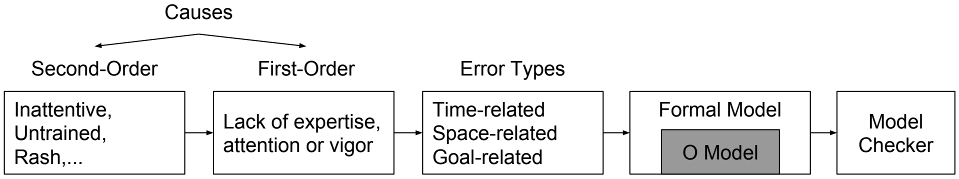

Causes
Second-Order
First-Order
Error Types
Inattentive, Untrained, Rash,...
Lack of expertise, attention or vigor
Time-related Space-related Goal-related
Formal Model
O Model
Model Checker

Figure 4: Types of errors that occur due to first-order causes related to the operator's state during execution. Expertise is awareness + experience + trust in design

Definition 1. Time-related errors happen when the operator manipulates the expected temporal order of actions and creates an unwanted situation.

$$\begin{aligned} errTP_x \Leftrightarrow & Repetition_x \vee Omission_x \vee Delayed_x \vee \\ & Early_{start,x} \vee Early_{end,x} \vee Intrusion_x \end{aligned} \tag{2}$$

The proposed model formalizes each phenotype introduced in [28] and listed in fig. 2 through the formulae below. However, there are a few phenotypes that have been merged with others. For example, when the nominal temporal position of an action in the execution work-flow of a task changes, it means that the action is being done earlier or later than planned. Thus, phenotype "replace" is implicit in "early" and "late" errors. The same thing is true for "reversal" and "insertion", meaning that reversing a_2 and a_1 (i.e., executing a_2 before a_1) also means doing a_2 early and a_1 late.

Definition 2. Repetition of action a_x occurs when it is currently done, but the operators starts again to execute it, or it was just finished and its post-conditions are satisfied, however the operator does not realize this and continues to execute it (e.g., over-screwing the workpiece). Consider for example a scenario in which the operator and robot should pin a number of workpieces on a pallet by screwdriving them with some fixtures. As a part of this task, the operator should prepare fixtures and then the robot should start moving towards the pallet. If the operator repeats the preparation for other jigs or continues to play with jigs after they are already in place, there could be a collision between the operator's hand and the robot end-effector.

$$Repetition_x \Leftrightarrow a_{x,agent} = \texttt{op} \wedge \begin{pmatrix} a_{x,sts} = \texttt{dn} \wedge opStarts_x \\ \vee \\ \mathsf{Past}\left(a_{x,sts} \neq \texttt{dn}, 1\right) \wedge posC_x \wedge \neg opStops_x \end{pmatrix} \tag{3}$$

Definition 3. An action a_x is **omitted** if the operator never executes it, thus predicate $opStarts_x$ is never true. Consequently, the status of a_x will always remain `ns` or `wt` in the future, which prevents the execution of robot actions whose pre-conditions require the termination of a_x.
We define a timeout equal to Δ time units starting from the fulfillment of pre-conditions and assume that the operator starts to execute her action within this bound. An action is considered omitted only if at some point it was ready for execution for at least Δ time units, but the operator refused to execute it. Otherwise it might be the case that the verification time was not long enough to cover its execution.

$$Omission_x \Leftrightarrow \begin{pmatrix} a_{x,agent} = \texttt{op} \wedge \mathsf{Alw}\left(\neg opStarts_x\right) \wedge \mathsf{SomP}\left(\mathsf{Lasted}\left(a_{x,sts} = \texttt{wt}, \Delta\right)\right) \\ \vee \\ a_{x,agent} = \texttt{ro} \wedge a_{x,sts} = (\texttt{ns}|\texttt{wt}) \end{pmatrix} \tag{4}$$

Definition 4. An action is **delayed** if the operator starts executing it some time after termination of Δ.

$$Delayed_x \Leftrightarrow a_{x,agent} = \texttt{op} \wedge opStarts_x \wedge \mathsf{SomP}\left(\mathsf{Lasted}\left(a_{x,sts} = \texttt{wt}, \Delta\right)\right) \tag{5}$$

Definition 5. The **early** error phenotype has two sides: $Early_{start,x}$ and $Early_{end,x}$. The first one happens when a_x is prematurely executed, meaning that its pre-conditions are not yet satisfied (it is still "not started") when the operator decides to execute it. In fact, the operator's act does not change the status of a_x, which remains "not started".

$$Early_{start,x} \Leftrightarrow a_{x,agent} = \texttt{op} \wedge \neg preC_x \wedge opStarts_x \tag{6}$$

The second one occurs when a_x prematurely ends, which means that the operator suddenly stops its execution, without completing the job.

$$Early_{end,x} \Leftrightarrow a_{x,agent} = \texttt{op} \wedge \neg posC_x \wedge opStops_x \tag{7}$$

Definition 6. Intrusion errors occur when the operator confuses the task to be executed with another one. More precisely, if $\mathcal{T}$ is the task being executed, $Intrusion_x$ holds when there is an action $a_x \in \mathcal{T}$ that is in the waiting, hold or (safe-)executing state, but the operator starts executing an action j that is not in $\mathcal{T}$:

$$a_{x,sts} \neq (\mathtt{ns}|\mathtt{dn}) \wedge \exists a_j \notin T : opStarts_j \tag{8}$$

Definition 7. Space-related errors occur due to movements of the operator which are not intended to be. An action is prone to space-related errors when the operator violates its spatial requirements during its execution (i.e., $spatialC_x$ is false while the action is executing), or if the action goes into a "hold" state without the operator having asked to stop the action (i.e., $opStops_x$ is false, which means that the holding state has been entered for reasons related to the behavior of the operator).
Examples of operator behaviors that lead to such situations are: leaving her required position or safe spot, getting closer to robot than the distance indicated in the instructions, approaching the robot when an alarm signals to stay away.

$$errS_x \Rightarrow \begin{pmatrix} \neg spatialC_x \wedge a_{x,sts} = (\mathtt{exe}|\mathtt{sfex}) \\ \quad \vee \\ a_{x,sts} = \mathtt{hd} \wedge (\mathsf{Past}(a_{x,sts} \neq \mathtt{hd} \wedge \neg opStops_x), 1) \end{pmatrix} \tag{9}$$

Definition 8. Goal-related errors happen when actions are not executed consistently with the instructions of the task. For example if the operator does not place the fixtures properly or does not tighten the part on the pallet while screw-driving. The presence of goal-related errors is represented by predicate $errG_x$, which is non-deterministically assigned values during the exploration of the system traces. Notice that, in practice, goal-related errors can be detected, for example, through the use of cameras installed in the work-cell; hence, predicate $errG_x$ can be seen as capturing the information provided by such cameras.

The model introduces constraints on the number of human errors during execution of a task. In fact, it is reasonable that an operator does not make too many errors during a single execution of an application; on the other hand, this number can be configured in the model, although the higher the number, the greater the complexity of the model and the required analysis time.

Definition 9. A counter is defined to count the number of occurrences of different errors:

$$\mathit{count} = \mathsf{past}\left(\mathit{count}, 1\right) + 1 \Leftrightarrow \exists x \in \mathcal{T} : \begin{pmatrix} errG_x \wedge \mathsf{Past}\left(\neg errG_x, 1\right) \\ \vee \\ errS_x \wedge \mathsf{Past}\left(\neg errS_x, 1\right) \\ \vee \\ errTP_x \wedge \mathsf{Past}\left(\neg errTP_x, 1\right) \end{pmatrix} \tag{10}$$

A constraint is introduced that imposes the counter not to exceed a threshold n. The threshold is configurable and could change with respect to different permutations of values for $\mathit{char} = \{\mathit{expertise},\ \mathit{attention},\ \mathit{vigor}\}$.[7]

$$\begin{aligned} &\forall x \in \mathit{char} : x = \texttt{high} \Rightarrow \mathit{count} = 0 \\ &\exists! x \in \mathit{char} : x < \texttt{high} \Rightarrow \mathit{count} \leqslant n_1 \\ &\exists x, y \in \mathit{char} : x, y < \texttt{high} \wedge x \neq y \Rightarrow \mathit{count} \leqslant n_2 \\ &\forall x \in \mathit{char} : x < \texttt{high} \Rightarrow \mathit{count} \leqslant n_3 \\ &\qquad 0 < n_1 < n_2 < n_3 \end{aligned} \tag{11}$$

Constraint $\mathsf{AlwF}\left(\mathit{count} \leqslant 4\right)$ is used in the reported experiments , which seems reasonable for an average operator, during execution of a task of approximately twenty actions. Similar constraints could be defined for less general counters such as count_{errTP}, which counts only the number of time-related errors.

In a similar vein, the model imposes a constraint on the number of $errG_x$, so that it cannot occur more often than every 5 time units.

$$\left(\exists a_{x_1} \in \mathcal{T} : errG_{x_1}\right) \Rightarrow \mathsf{Lasted}\left(\forall a_{x_2} \in \mathcal{T} : \neg errG_{x_2}, 5\right) \tag{12}$$

The addition of the formalization of these phenotypes to the model allows us to check whether there are hazardous situations that cannot be mitigated by currently introduced RRMs, and consequently if the base model (without formal cognitive model) failed to capture hazards that arise due to human errors.

[7] To make formulae shorter and easier to understand we use the notation $x \in \mathit{char}$ to predicate over user characteristics. For example, $(\forall x \in \mathit{char} : x = \texttt{high})$ in Formula (11) is short for $(\mathit{expertise} = \texttt{high} \wedge \mathit{attention} = \texttt{high} \wedge \mathit{vigor} = \texttt{high})$.

5. Evaluation

This section provides some evaluation results over two case studies. We first explain how the formal model is created and then report some snippets of the results of verifying the model with Zot. The full report of the experiments could be found in [5].

The robot system used in the experiments is a lightweight, PFL-capable manipulator[8] with an interchangeable end-effector (a force-limited collaborative gripper `gpr` or a screwdriver `sdr`), so to be able to switch jobs with the operator. In case of need, it is the operator who is responsible to change and mount the end-effector of the robot. The work-cell is equipped with cameras and sensors that detect the position of robot and operator. The operator has a three-dimensional gesture-detecting wearable device[9] armband on, and uses it to communicate with the robot, for example to issue commands or to receive feedback about the state of the system.

The experiments were carried out through the Zot plugins introduced in [46], running on a Linux desktop machine with a 3.4 GHz Intel® Core™ i7-4770 CPU and 16 GB RAM, of which only 231.56 MB have been used in the worst case. None of the reported experiments took more than 480 seconds, whilst the shortest one took 320 seconds. Zot is a bounded model checker, which means that it generates all possible traces of the model within a bounded horizon. These two experiments have been analyzed with bounds equal to 40 and 60 time units, respectively, which are enough for multiple iterations of the task.

5.1. First Case Study

In the first case study, we analyze a robot system which is installed in the layout depicted in fig. 5(a) and fig. 5(b). The robot picks workpieces from a bin and loads them on a CNC machine pallet, where they are manipulated in two phases.

The robot repetitively moves between L_1 and the pallet, shown in fig. 5(b), to bring and load workpieces on the pallet. Each workpiece should first be placed on point one (P_1) on the pallet and the operator should screwdrive it. When the first phase—which consists of carving the workpiece—is done, the operator should unscrew the workpiece and the robot should move it

[8] KUKA iiwa R800 (kuka-robotics.com).

[9] Myo armband (www.myo.com).

to point two (P_2) and prepare it for the second carving phase. While the second phase of carving is going on, the robot can go back to the bin to pick a new workpiece and bring it to P_1. After finishing both carving phases of a workpiece, the robot should deliver it to L_1. The operator should carry out an inspection activity for controlling the correctness of the execution.[10]

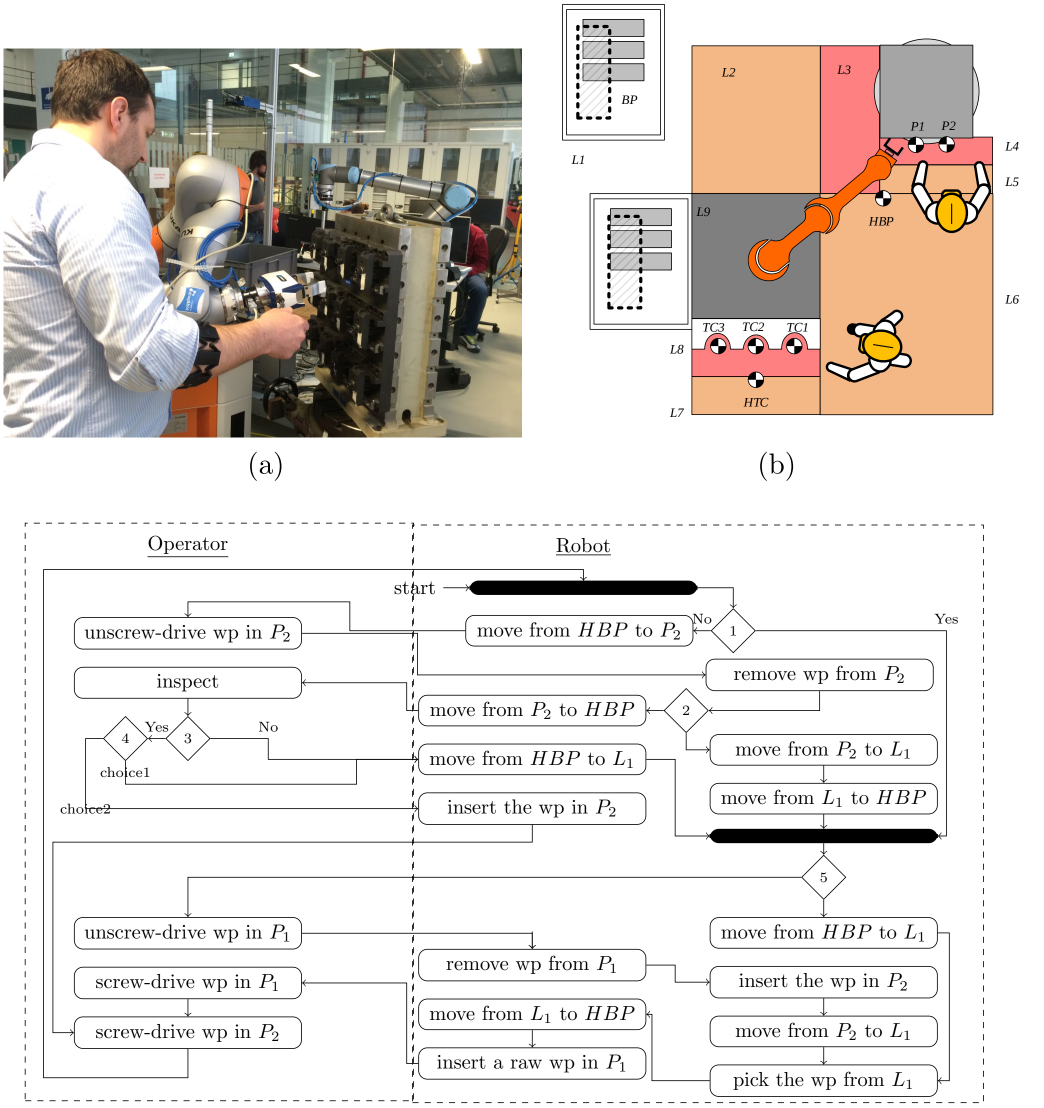
The manipulator robot is mounted on a platform which reaches up to human hips. The manipulator arm movements are limited to the areas above the platform, hence they never threaten the lower areas of the human body. Thus, we have studied hazards on three human body parts: Head (referring to head and neck), Waist (referring to the area between hips and neck) and Arm areas (referring to both arms and all fingers). Naturally, arms and fingers are the most mobile human parts, which change location more often than the others. The robot is discretized in three main parts (according to the KUKA structure): two arm links and the end-effector (gripper or screwdriver).

To carry out the verification, the general formal model for describing robots and humans [5] needs in this case to be initialized with one instance for each type of actor. Also, the layout is discretized into multiple parts according to the characteristics of the different sections of the workspace. The discretization is depicted in fig. 5(b), where the red (resp., orange) areas are where `Qs` (resp., `Tr`) hazards are more probable. In addition, we need to instantiate the generic formal model of the task and create the flow of actions for the task at hand. The overall activity diagram of this task is illustrated in fig. 5(c). Different execution traces are feasible according to the diagram. Operator and robot might execute different numbers of actions in different traces.

We fed the resulting model to Zot, which in turn provided a sequence of snapshots of the system at each time instant, and a report of all the hazards identified at each snapshot that also highlights the risky situations which do not have an effective mitigation. Here we report two interesting situations.

a. Location Error. Figure 6 shows a state of the model in which the operator is violating spatial requirements of safe execution of an action. Hence, a few instances of `Qs` hazards are detected by the tool. The operator is not supposed to wander around inside the area in-which the robot is frequently moving. If the operator does not follow the instructions to safely execute the task (e.g.,

[10] This model has been demonstrated in the EuroC project (euroc-project.eu) as part of a controlled setup demonstration and a real shopfloor run with the same robot system.



(a)

(b)

(c)

Figure 5: First Case-study: (a) collaborative pallet carving scenario; (b) top-view representation of the environment, where the areas with possible entrapment (resp., sudden contacts) are shown in red (resp., orange); (c) activity diagram including all possible execution variations. The conditions associated with choice nodes are the following: 1. Is P_2 empty? 2. Is robot moving? 3. Does any operation inspected to be done incorrectly? 4. Which execution alternation does the operator choose? 5. Is P_1 empty?

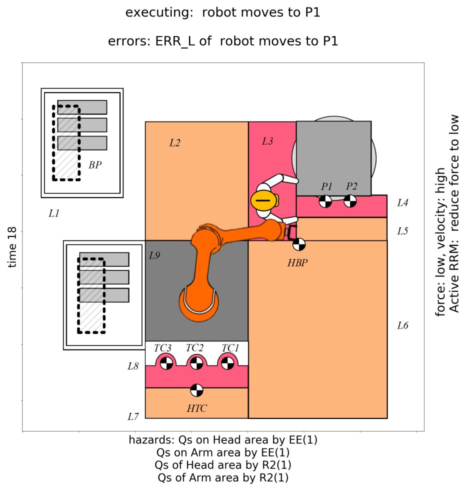
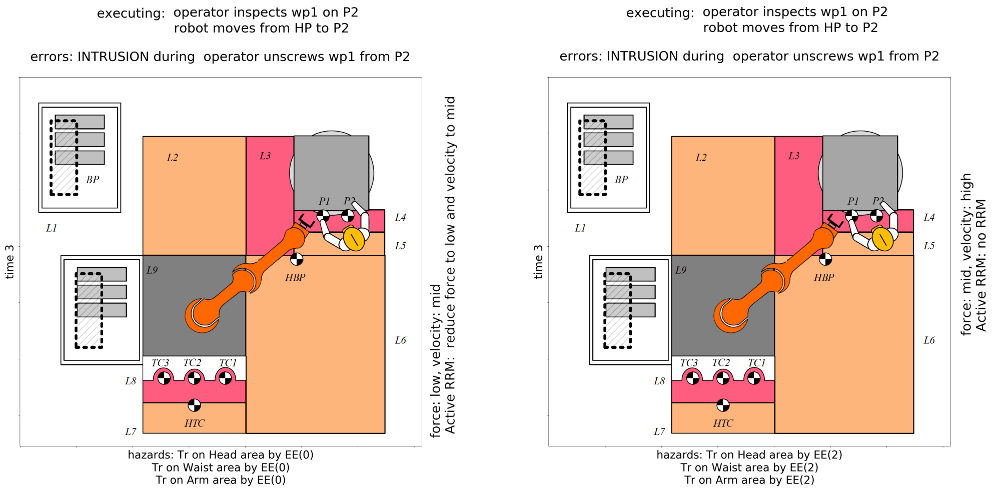

executing: robot moves to P1
errors: ERR_L of robot moves to P1
time 18
BP
L1
L2
L3
P1
P2
L4
L5
L9
HBP
L6
TC3
TC2
TC1
L8
HTC
L7
force: low, velocity: high
Active RRM: reduce force to low
hazards: Qs on Head area by EE(1)
Qs on Arm area by EE(1)
Qs of Head area by R2(1)
Qs of Arm area by R2(1)

Figure 6: The operator is in wrong place and is violating spatial safety requirements of the executing action.

violating $spatialC$), then the probability of hazards increases.

b. Intrusion Error. Figure 7 shows the same time unit in two different verification runs: one containing RRMs and the other one without RRMs. Both snapshots picture a time unit when the operator is executing an inspection at the wrong time. The operator should not execute an inspection while the robot is moving and getting close to the pallet. According to the definitions included in the formal model, such a situation would end in a `Tr` hazard, which is detected by the tool.
The critical difference between the time unit depicted in fig. 7a and the one in fig. 7b is the risk value. When RRMs are considered in the model, the risk value will not reach the undesired level of 2—the critical threshold that is captured by Formula (1)—since RRMs are invoked when the risk reaches level 1 and thus try to maintain—or reduce—the risk by reducing its severity factor by constraining the values of velocity and force.

Figure 7a shows that RRMs that keep the relative force less than or equal to `low` (meaning: `mid` or `low`) and the relative velocity less than or equal to `mid` (meaning: `mid`, `low` or `none`) are active in the model. The relative parameters (velocity and force) here pertain to the parts of the human body and of the robot that are involved in the definition of detected hazards.



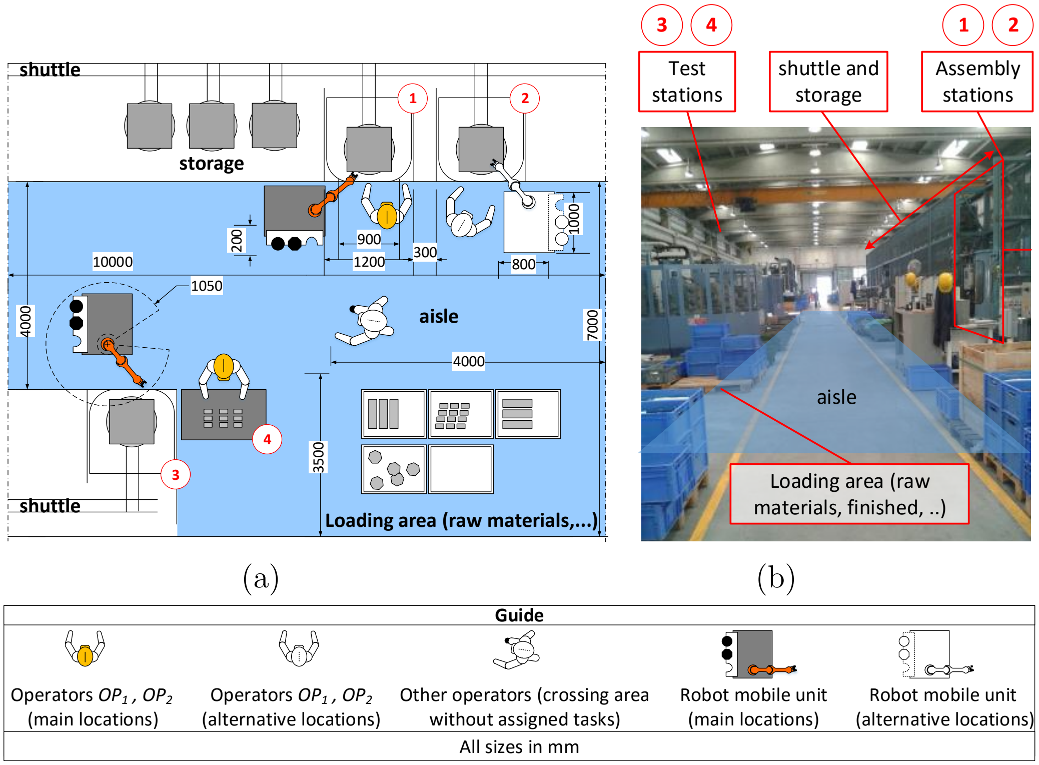
executing: operator inspects wp1 on P2
robot moves from HP to P2
errors: INTRUSION during operator unscrews wp1 from P2
time 3
BP
L1
L2
L3
L4
L5
L6
L7
L8
L9
P1
P2
HBP
TC3
TC2
TC1
HTC
force: low, velocity: mid
Active RRM: reduce force to low and velocity to mid
hazards: Tr on Head area by EE(0)
Tr on Waist area by EE(0)
Tr on Arm area by EE(0)
executing: operator inspects wp1 on P2
robot moves from HP to P2
errors: INTRUSION during operator unscrews wp1 from P2
time 3
BP
L1
L2
L3
L4
L5
L6
L7
L8
L9
P1
P2
HBP
TC3
TC2
TC1
HTC
force: mid, velocity: high
Active RRM: no RRM
hazards: Tr on Head area by EE(2)
Tr on Waist area by EE(2)
Tr on Arm area by EE(2)

Figure 7: An instance of an intrusion error.

5.2. Second Case Study

In this scenario, we have analyzed a mobile robot unit which autonomously relocates in the layout shown in fig. 8. This robot system is configured as a combination of a driverless truck[11] (i.e., AGV) and a manipulator, which mainly moves between three assembly stations—①, ② and ③—and a sensor-based inspection station ④, as shown in Figure 8(a). The robot unit can be manually relocated by operators around its predefined positions. The robot unit can travel and access the whole workspace (the blue area in Figure 8(b)), including a load/unload area for raw materials and finished parts. Two human operators (OP_1 and OP_2) are employed in the application. OP_1 is mostly present in stations ① and ②, while OP_2 works mainly in ③ or executes auxiliary manual tasks on the workbench in ④. Both operators can freely hold and resume their tasks, swap posts, or join one another in some area. The main robot-assisted intended tasks are: pallet assembly at

[11] The mobile unit is in the scope of EN 1515, ISO/DIS 3691-4 for safety requirements, and its suitable RRMs include speed limitation and contact prevention. On the other hand, mobile manipulations are not—to this date—in the scope of any standard, and their risk reduction requirements are application of force/velocity limitation for the combined system.


shuttle
storage
1
2
200
900
1200
300
1000
800
10000
1050
4000
aisle
7000
4000
3500
4
3
shuttle
Loading area (raw materials,...)
3
4
Test stations
shuttle and storage
1
2
Assembly stations
aisle
Loading area (raw materials, finished, ..)
Guide
Operators OP_1 , OP_2 (main locations)
Operators OP_1 , OP_2 (alternative locations)
Other operators (crossing area without assigned tasks)
Robot mobile unit (main locations)
Robot mobile unit (alternative locations)
All sizes in mm

(a) (b)

Figure 8: Second case study: (a) precise workcell depiction, (b) actual layout.

stations ① and ②, including bin-picking from a local storage carried by the mobile unit; pallet disassembly (reversal of assembly) at ① and ②, including bin-dumping; pallet inspection at station ③; lead-through programming of assembly, disassembly, and inspection tasks (trajectories, parameters, etc.) at stations ①, ② and ③; material handling on load/unload areas. Other manual tasks by OP_1 and OP_2 include manual loading of parts/boxes; (additional) visual inspection of pallet at stations ①, ② and ③; manual assembly/disassembly of pallet at stations ① and ②; manual measurements of parts at station ④; cleaning pallets at stations ① and ②; kitting of tools and parts at stations ①, ② and ③; general supervision (programming other tasks at HMI during operations, consulting production data at HMI, etc.) at stations ①, ② and ③. Note that *all* combinations of robot/manual task assignments are admitted (e.g., robot holds and OP_1 screw-drives jigs and vice versa, switching tasks on the fly, quitting a manual task and assigning

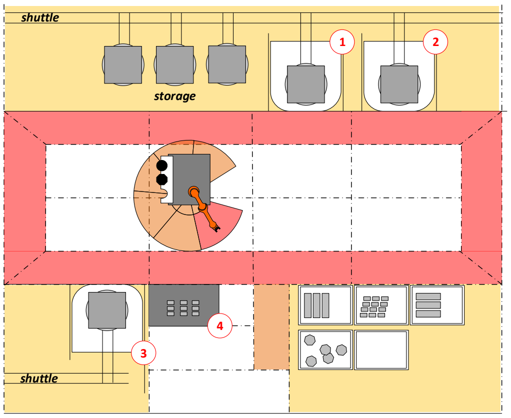


Figure 9: Discretization of the layout in the second case study. The areas with potential for entrapment (resp., sudden contact) are shown in red (resp., orange); areas that are unreachable by the robot are shown in yellow. At each time instant, areas around the robot arm dynamically change their type.

the robot to proceed autonomously). Frequently, robot base and operators move side-to-side across the central aisle, or other operators transit along the aisle because the target area is part of a larger plant and access to it is not restricted (fig. 8(b)).

Figure 9 shows that the workspace is discretized in 23 regions with different characteristics. In this case study, not all areas have static aspects. The robot is constantly moving around and the areas mostly situated on the aisle could have different characteristic from time to time. In order to resemble the human reasoning about spatial properties and construct a 3D model of the workspace, each region is modeled in three layers: lower, middle and upper sections. The mobile base of the robot is always allowed in the lower layer, while the manipulator arm could move in the middle and upper layers. In this case study, the robot unit is mobile, thus in addition to the previous three human body areas, we also study the hazards on the human Leg area (below the hip). Assuming a person standing still, the Leg area would be in the lower sections, while Waist and Arm areas are in middle sections, and the Head area is in upper sections.

Like the previous case study, the robot system is discretized in its main sections. Additionally, here there is a mobile base on which the two arm links and the end-effector are mounted.

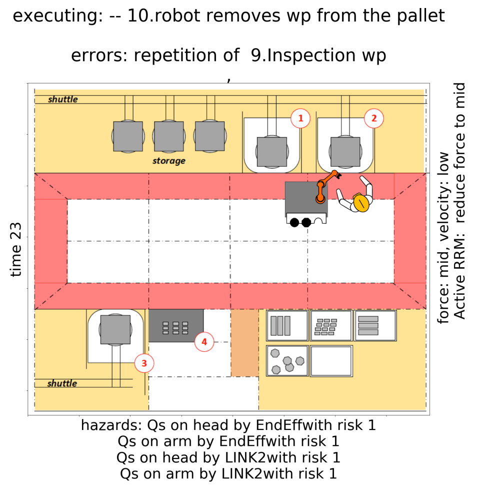

executing: -- 10.robot removes wp from the pallet
errors: repetition of 9.Inspection wp
shuttle
storage
1
2
time 23
4
3
shuttle
force: mid, velocity: low
Active RRM: reduce force to mid
hazards: Qs on head by EndEffwith risk 1
Qs on arm by EndEffwith risk 1
Qs on head by LINK2with risk 1
Qs on arm by LINK2with risk 1

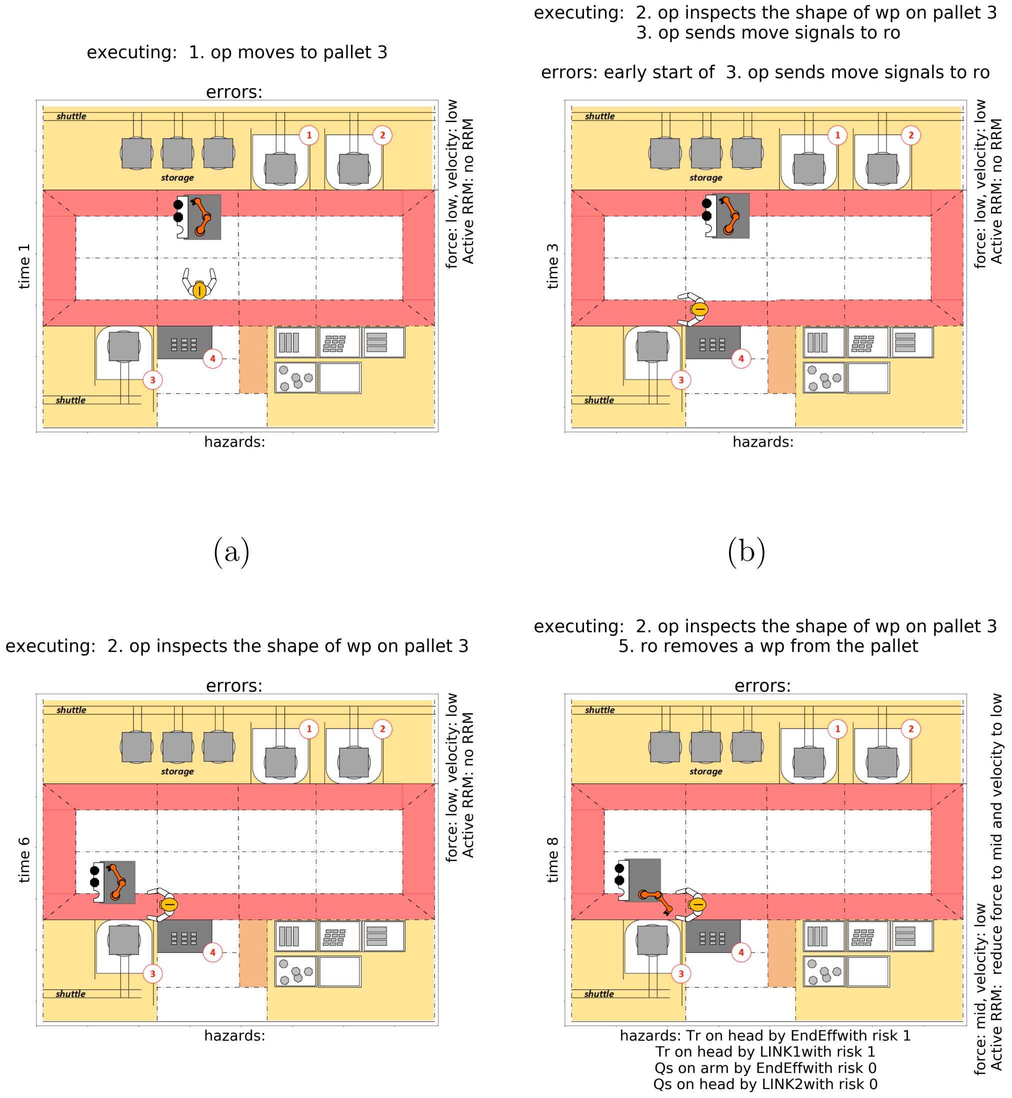
Figure 10: According to the instructions, the robot arm should retract after the inspection of the workpiece by the operator. However, the operator repeats the inspection while the robot arm is retracting and puts her head or arm in danger. However, The active RRM forbids any serious harm by tuning the value of retraction force.

a. Repetition. According to the instructions of the task, when the robot places a workpiece on one of the locations ①, ② or ③ the operator could execute manual operations or inspection. When the action of the operator is over, the robot could replace the workplace from the pallet and retract from it. Figure 10 shows a situation in which the operator mistakenly repeats her actions while the robot is starting to retract. Therefore, a `Qs` hazard occurs (e.g., the operator's head or arms could be entrapped between the robot and the tombstone, or between two links of the robot).

b. Sending a moving signal to the robot too early. Our human model not only is useful for detecting hazardous situations which occur because of human errors, but also enable the safety assessor to track down hazards and find out the reasons. Figure 11 highlights a situation in which the operator commits an error at time t= 3 (fig. 11b) which eventually causes a hazard to occur at time t=8 (fig. 11d).

6. Conclusions

This paper extended our previous work on the SAFER-HRC method, which employs automated formal verification techniques to comprehensively


executing: 1. op moves to pallet 3
errors:
time 1
force: low, velocity: low
Active RRM: no RRM
hazards:
executing: 2. op inspects the shape of wp on pallet 3
3. op sends move signals to ro
errors: early start of 3. op sends move signals to ro
time 3
force: low, velocity: low
Active RRM: no RRM
hazards:
executing: 2. op inspects the shape of wp on pallet 3
errors:
time 6
force: low, velocity: low
Active RRM: no RRM
hazards:
executing: 2. op inspects the shape of wp on pallet 3
5. ro removes a wp from the pallet
errors:
time 8
hazards: Tr on head by EndEffwith risk 1
Tr on head by LINK1with risk 1
Qs on arm by EndEffwith risk 0
Qs on head by LINK2with risk 0
force: mid, velocity: low
Active RRM: reduce force to mid and velocity to low
shuttle
storage
1
2
3
4

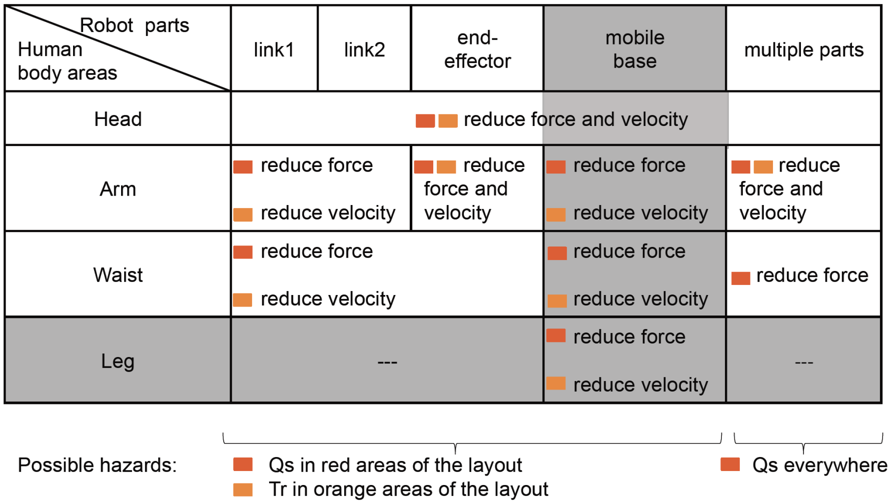
(a)

(b)

(c)

(d)

Figure 11: The operator (shortened as `op`) has stopped the robot in order to perform an inspection on pallet three. While s/he is moving there, s/he sends a resume signal to the robot (shortened as `ro`) before actually performing the inspection. Thus the robot starts to advance its arm and consequently a hazardous contact on human head or arms is probable. However, the presence of RRM tunes the robot's force and velocity so to maintain the risk of possible contacts low. For head hazards, a general limitation on the positioning of robot parts in the upper region is introduced as a constraint (e.g., implemented as a safe zone in the robot controller).



Human body areas \ Robot parts	link1	link2	end-effector	mobile base	multiple parts
Head	🟥🟧 reduce force and velocity				
Arm	🟥 reduce force 🟧 reduce velocity		🟥🟧 reduce force and velocity	🟥 reduce force 🟧 reduce velocity	🟥🟧 reduce force and velocity
Waist	🟥 reduce force 🟧 reduce velocity			🟥 reduce force 🟧 reduce velocity	🟥 reduce force
Leg	---			🟥 reduce force 🟧 reduce velocity	---

Possible hazards: 🟥 Qs in red areas of the layout
🟧 Tr in orange areas of the layout — 🟥 Qs everywhere

Table 2: This table lists the hazards that could occur in the two test-cases, between identified human body areas and parts of the robot system in the formal model. The table also introduces the required RRM(s) for each hazard.
`Tr` hazards happen in orange areas of fig. 5(b) and fig. 9 and activate velocity reduction. `Qs` hazards occur in orange areas where multiple robot parts entrap human parts, or in red areas, and they activate force reduction.

analyze the safety of HRC applications. In particular, in this work we introduced a formal model that replicates the most common human erroneous behaviors that can occur during the execution of manufacturing tasks; this allows for the detection (and correction) of hazardous situations that could be overlooked if a purely functional model of humans is used. We showed the effectiveness of the extended operator model by using it to study the safety of a pair of real-life applications. Table 2 depicts the list of all possible hazards for each case study and their corresponding RRMs. For example, the first row of the left table of table 2 is interpreted as follows: The orange areas of the layout shown in fig. 5b are susceptible to `Tr` hazards on the human Head area by either of the three parts of the robot system (i.e., the human head might be hit by the links or by the end-effector in orange areas). The red areas of the layout shown in fig. 5b are susceptible to `Qs` hazards on the human head by either of the three parts of the robot system (e.g., the human head could get entrapped between link1 and the pallet wall in L_3). Whenever multiple parts of the robot are in the same area (red or orange) with a human body part, a `Qs` hazard could occur (e.g., when the links of robot bend together in L_6 and suddenly the operator tries to straighten the bend of the robot arm, his/her hand might get entrapped between the two links). Depending on the value of the risks associated with each hazard, the RRMs (reductions of force and velocity) could be of different intensity (e.g., full stop, which typically reduces the risk to the minimum level).

Currently our model relies on a nondeterministic approach, which captures the possibility that operators behave erroneously, without quantifying what is the probability that this occurs. As future work, we plan to further extend the model to also include a notion of probability of errors, to better estimate the relevance of detected errors and the need to mitigate their effects from a safety point of view.

In addition, we have contacted three different teams to execute a risk analysis on a specific scenario (that we have provided) in order to carry out a comparison between the results of our tool and manual procedures.

Acknowledgments. This work was partially supported by Fondazione Cariplo and Regione Lombardia through the AUTOVAM project.

[1] M. Askarpour, D. Mandrioli, M. Rossi, F. Vicentini, SAFER-HRC: safety analysis through formal verification in human-robot collaboration, in: Computer Safety, Reliability, and Security (SAFECOMP), Vol. 9922 of LNCS, 2016, pp. 283–295.

[2] M. Askarpour, D. Mandrioli, M. Rossi, F. Vicentini, Modeling operator behavior in the safety analysis of collaborative robotic applications, in: Computer Safety, Reliability, and Security - 36th International Conference, SAFECOMP, Proceedings, 2017, pp. 89–104. doi:10.1007/978-3-319-66266-4_6.

[3] J. Reason, Human error, Cambridge university press, 1990.

[4] M. Askarpour, D. Mandrioli, M. Rossi, F. Vicentini, A human-in-the-loop perspective for safety assessment in robotic applications, in: Perspectives of System Informatics - 11th International Andrei P. Ershov Informatics Conference, PSI, Revised Selected Papers, 2017, pp. 12–27. doi:10.1007/978-3-319-74313-4_2.

[5] M. Askarpour, Safer-HRC: a methodology for Safety Assessment through Formal verification in human-robot collaboration, Doctoral dissertation, Polytechnic University of Milan (2018).

[6] ISO, ISO 10218-1:2011: Robots and robotic devices – Safety requirements for industrial robots – Part 1: Robots, International Organization for Standardization, Geneva, Switzerland, 2011.

[7] ISO, ISO 12100:2010: Safety of machinery – General principles for design – Risk assessment and risk reduction, International Organization for Standardization, Geneva, Switzerland, 2010.

[8] ISO, ISO/TR 14121-2:2012: Safety of machinery – Risk assessment – Part 2: Practical guidance and examples of methods, International Organization for Standardization, Geneva, Switzerland, 2012.

[9] C. A. Furia, D. Mandrioli, A. Morzenti, M. Rossi, Modeling Time in Computing, Monographs in Theoretical Computer Science. An EATCS Series, Springer, 2012.

[10] Zot: a bounded satisfiability checker, available from github.com/fm-polimi/zot.

[11] ISO, ISO 10218-2:2011: Robots and robotic devices – Safety requirements for industrial robots – Part 2: Robot systems and integration, International Organization for Standardization, Geneva, Switzerland, 2011.

[12] ISO, ISO /TR 7250-2:2010: Basic human body measurements for technological design – Part 2: Statistical summaries of body measurements from national populations, Geneva, Switzerland, 2010.

[13] M. L. Bolton, E. J. Bass, R. I. Siminiceanu, Generating phenotypical erroneous human behavior to evaluate human-automation interaction using model checking, Int. J. Hum.-Comput. Stud. 70 (11) (2012) 888–906. doi:10.1016/j.ijhcs.2012.05.010.

[14] M. L. Bolton, E. J. Bass, R. I. Siminiceanu, Using formal verification to evaluate human-automation interaction: A review, IEEE Trans. Systems, Man, and Cybernetics: Systems 43 (3) (2013) 488–503. doi:10.1109/TSMCA.2012.2210406.

[15] M. L. Bolton, E. J. Bass, Formally verifying human–automation interaction as part of a system model: limitations and tradeoffs, Innovations in Systems and Software Engineering 6 (3) (2010) 219–231. doi:10.1007/s11334-010-0129-9.

[16] A. J. Dix, M. Ghazali, S. Gill, J. Hare, D. Ramduny-Ellis, Physigrams: modelling devices for natural interaction, Formal Asp. Comput. 21 (6) (2009) 613–641. `doi:10.1007/s00165-008-0099-y`.

[17] P. Curzon, A. Blandford, From a formal user model to design rules, in: Interactive Systems. Design, Specification, and Verification, 9th International Workshop, DSV-IS 2002, Rostock Germany, June 12-14, 2002, 2002, pp. 1–15. `doi:10.1007/3-540-36235-5_1`.

[18] J. E. Laird, The Soar cognitive architecture, MIT Press, 2012.

[19] J. R. Anderson, ACT: A simple theory of complex cognition., American Psychologist 51.

[20] D. D. Salvucci, F. J. Lee, Simple cognitive modeling in a complex cognitive architecture, in: Proc. of CHI, 2003, pp. 265–272.

[21] A. Cerone, P. A. Lindsay, S. Connelly, Formal analysis of human-computer interaction using model-checking, in: Proc. of SEFM, 2005, pp. 352–362.

[22] P. A. Lindsay, S. Connelly, Modelling erroneous operator behaviours for an air-traffic control task, in: Proc. of Australasian UI Conf. (AUIC), 2002, pp. 43–54.

[23] Concur task trees (CTT), available from `w3.org`, accessed: 2017-07-06 (2012).

[24] F. Paternò, C. Mancini, S. Meniconi, Concurtasktrees: A diagrammatic notation for specifying task models, in: INTERACT, 1997.

[25] H. R. Hartson, A. C. Siochi, D. Hix, The uan: A user-oriented representation for direct manipulation interface designs, ACM Trans. Inf. Syst. 8 (3) (1990) 181–203. `doi:10.1145/98188.98191`.

[26] M. L. Bolton, R. I. Siminiceanu, E. J. Bass, A systematic approach to model checking human-automation interaction using task analytic models., IEEE Trans. Systems, Man, and Cybernetics, Part A 41 (5) (2011) 961–976.

[27] E. Dougherty, J. Fragola, Human reliability analysis, New York, NY; John Wiley and Sons Inc., 1988.

[28] E. Hollnagel, Cognitive reliability and error analysis method (CREAM), Elsevier, 1998.

[29] J. C. Williams, 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,, 1988, pp. 436–450. doi:10.1109/HFPP.1988.27540.

[30] A. D. Swain, H. E. Guttmann, Handbook of human-reliability analysis with emphasis on nuclear power plant applications. final report, Tech. rep., Sandia National Labs., Albuquerque, NM (USA) (1983).

[31] S. Pocock, M. D. Harrison, P. C. Wright, P. Johnson, THEA: A technique for human error assessment early in design, in: M. Hirose (Ed.), Human-Computer Interaction INTERACT '01: IFIP TC13 International Conference on Human-Computer Interaction, Tokyo, Japan, July 9-13, 2001, IOS Press, 2001, pp. 247–254.

[32] C. Martinie, P. A. Palanque, R. Fahssi, J. Blanquart, C. Fayollas, C. Seguin, Task model-based systematic analysis of both system failures and human errors, IEEE Trans. Human-Machine Systems 46 (2) (2016) 243–254. doi:10.1109/THMS.2014.2365956.

[33] D. Pan, M. L. Bolton, Properties for formally assessing the performance level of human-human collaborative procedures with miscommunications and erroneous human behavior, International Journal of Industrial Ergonomics (2016) –doi:https://doi.org/10.1016/j.ergon.2016.04.001.

[34] M. L. Bolton, Model checking human-human communication protocols using task models and miscommunication generation, J. Aerospace Inf. Sys. 12 (7) (2015) 476–489. doi:10.2514/1.I010276.

[35] S. Basnyat, P. Palanque, A task pattern approach to incorporate user deviation in task models, in: Proc. 1st ADVISES Young Researchers Workshop. Liege, Belgium, 2005.

[36] F. Paternò, C. Santoro, Preventing user errors by systematic analysis of deviations from the system task model, International Journal of Human-Computer Studies 56 (2) (2002) 225–245.

[37] P. Curzon, A. Blandford, Formally justifying user-centred design rules: a case study on post-completion errors, in: International Conference on Integrated Formal Methods, Springer, 2004, pp. 461–480.

[38] N. Kim, L. Rothrock, J. Joo, R. A. Wysk, An affordance-based formalism for modeling human-involvement in complex systems for prospective control, in: Proceedings of the 2010 Winter Simulation Conference, WSC 2010, Baltimore, Maryland, USA, 5-8 December 2010, 2010, pp. 811–823. doi:10.1109/WSC.2010.5679107.

[39] L. F. Cranor, A framework for reasoning about the human in the loop, in: Proc. of Usability, Psychology, and Security (UPSEC), 2008.

[40] D. Shin, R. A. Wysk, L. Rothrock, Formal model of human material-handling tasks for control of manufacturing systems, IEEE Trans. Systems, Man, and Cybernetics, Part A 36 (4) (2006) 685–696.

[41] R. E. Fields, Analysis of erroneous actions in the design of critical systems, Ph.D. thesis, University of York (2001).

[42] B. Kirwan, Human error identification techniques for risk assessment of high risk systems-part 1: review and evaluation of techniques, Applied Ergonomics 29 (3) (1998) 157 – 177. doi:https://doi.org/10.1016/S0003-6870(98)00010-6.

[43] ISO, ISO/TS 15066:2016: Robots and robotic devices – Collaborative robots, International Organization for Standardization, Geneva, Switzerland, 2016.

[44] M. Pradella, A. Morzenti, P. San Pietro, Bounded satisfiability checking of metric temporal logic specifications, ACM TOSEM 22 (3) (2013) 20:1–20:54.

[45] C. A. Furia, D. Mandrioli, A. Morzenti, M. Rossi, Modeling time in computing: A taxonomy and a comparative survey, ACM Comput. Surv. 42 (2) (2010) 6:1–6:59. doi:10.1145/1667062.1667063.

[46] L. Baresi, M. M. Pourhashem Kallehbasti, M. Rossi, Efficient scalable verification of LTL specifications, in: Proceedings of the 37th International Conference on Software Engineering, IEEE Press, 2015, pp. 711–721.

[47] A. Cimatti, E. M. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, A. Tacchella, Nusmv 2: An open-source tool for symbolic model checking, in: E. Brinksma, K. G. Larsen (Eds.), Computer Aided Verification, 14th International Conference, CAV 2002,Copenhagen, Denmark, July 27-31, 2002, Proceedings, Vol. 2404 of Lecture Notes in Computer Science, Springer, 2002, pp. 359–364. doi:10.1007/3-540-45657-0_29.

[48] B. Almannai, R. Greenough, J. Kay, A decision support tool based on QFD and FMEA for the selection of manufacturing automation technologies, Robotics and Computer-Integrated Manufacturing, Volume 24, Issue 4, ICMR2005: Third International Conference on Manufacturing Research, August 2008, Pages 501-507.

[49] Fabian Ranz, Titanilla Komenda, Gerhard Reisinger, Philipp Hold, Vera Hummel, Wilfried Sihn, A Morphology of Human Robot Collaboration Systems for Industrial Assembly, Procedia CIRP, Volume 72, 2018, Pages 99-104,

[50] Jeremy A. Marvel, Rick Norcross, Implementing speed and separation monitoring in collaborative robot workcells, Robotics and Computer-Integrated Manufacturing, Volume 44, 2017, Pages 144-155

[51] George Michalos, Sotiris Makris, Panagiota Tsarouchi, Toni Guasch, Dimitris Kontovrakis, George Chryssolouris, Design Considerations for Safe Human-robot Collaborative Workplaces, Procedia CIRP, Volume 37, 2015, Pages 248-253

[52] Ramy Meziane, Martin J.-D. Otis, Hassan Ezzaidi, Human-robot collaboration while sharing production activities in dynamic environment: SPADER system, Robotics and Computer-Integrated Manufacturing, Volume 48, 2017, Pages 243-253

[53] D. Mavrikios, V. Karabatsou, M. Pappas, G. Chryssolouris, An efficient approach to human motion modeling for the verification of human-centric product design and manufacturing in virtual environments, Robotics and Computer-Integrated Manufacturing, Volume 23, Issue 5, 2007, Pages 533-543

[54] Przemyslaw A. Lasota, Terrence Fong and Julie A. Shah ”A Survey of Methods for Safe Human-Robot Interaction”, Foundations and Trends in Robotics: Vol. 5: No. 4, 2017 Pages 261-349