Elicitation of Executable Safety Rules for Critical Autonomous Systems

Amina Mekki-Mokhtar*†, Jean-Paul Blanquart‡, Jérémie Guiochet*†, David Powell*† and Matthieu Roy*†

* LAAS-CNRS, 7 Avenue du colonel Roche, 31077 Toulouse, France
† Université de Toulouse, UPS, INSA, INP, ISAE ; IUT, UTM, LAAS, Toulouse, France
{prenom.nom@laas.fr}

‡ EADS Astrium, 31 rue des cosmonautes, 31402 Toulouse, France
{jean-paul.blanquart@astrium.eads.net}

***Abstract*—The progress of artificial intelligence techniques, particularly decisional mechanisms, has allowed reactive systems to become more autonomous. This allows new applications in domains such as service robotics in which failures can lead to human injury or death, or financial loss. To ensure safety of such systems, we propose in this paper a process, based on a HAZOP/UML risk analysis, to elicit safety rules that can be enforced on-line. We present a case study of safety rule elicitation for an assistive robot for strolling and discuss implementation of the safety rules in a practical safety monitor.**

***Keywords*-Safety, Dependability, Autonomous Critical Systems, Safety Rules, Safety Constraints, On-line Monitoring, HAZOP/UML.**

I. Introduction

Automated systems are now able to become more autonomous thanks to recent progress in artificial intelligence techniques, especially decisional mechanisms. Human intervention is often no longer required, allowing new applications in several critical domains, such as: automated transportation, space exploration, rescue and maintenance operations within irradiated areas, etc. These systems are often critical since their failure can lead to human injury or even death, or large financial losses. However, such systems are difficult to validate due to their high complexity and the fact that they operate within complex, variable and uncertain environments in which it is difficult to predict all possible system behaviors. Therefore, there has been considerable interest in making such systems safe [1], [2].

Safety can be defined in absolute terms as "the absence of catastrophic consequences on the user(s) and the environment" [3]. However, the notion of "zero risk" is utopian since systems can and do fail during execution. The IEC 61508 standard recognizes this and defines safety as "the absence of unacceptable risk" [4].

To obtain systems that fulfill their mission or avoid catastrophic failures, despite the presence of faults, a combination of several methods can be used [3], which are: *fault prevention*, *fault tolerance*, *fault removal* and *fault forecasting*. In the context of safety, a popular form of fault tolerance is *safety monitoring* through which the functional system is continuously monitored and forced to a safe state should some anomalous behavior be detected. Safety monitors appear in the literature under many different terms, such as: *safety manager* [5], *autonomous safety system* [6], *checker* [7], *guardian agent* [8], *safety bag* [9], or *diverse monitor* [4]. The latter IEC 61508 standard defines a diverse monitor as "*an external monitor, implemented on an independent computer to a different specification. This diverse monitor is solely concerned with ensuring that the main computer performs safe, not necessarily correct, actions. The diverse monitor continuously monitors the main computer. The diverse monitor prevents the system from entering an unsafe state. In addition, if it detects that the main computer is entering a potentially hazardous state, the system has to be brought back to a safe state either by the diverse monitor or the main computer.*"

A safety monitor observes the behavior of the functional system and aims to ensure that it respects a set of *safety rules*. There may be many such safety rules for autonomous systems since the latter can support multiple tasks, with each task potentially requiring a different set of safety rules to be enforced. Therefore, it is important to define a rigorous safety rule specification process, with clear concepts and notations. However, to the best of our knowledge, there has been little research on this issue. In this paper, we present a systematic process to generate the safety rules from a risk analysis of the monitored system.

The remainder of the paper is organized as follows: Section II presents some background material on safety critical autonomous systems and related work in safety monitoring. Section III presents definitions of terms used in our approach and the methodology developed to elicit the safety rules. In Section IV, we apply this process to a robotic rollator and finally, Section V presents conclusions and future work.

II. Background and related work

A. Safety critical autonomous systems and dependability

Autonomy has several descriptions in the literature; in its most generic sense, it is defined as "the ability to self-manage, to act or to govern without being controlled by others". This definition is insufficient for our purpose and a more suitable

definition of autonomy of robotic systems is found in [10] as *"the ability of integrated sensing, perceiving, analyzing, communicating, planning, decision-making, and acting/executing, to achieve its goals as assigned. The autonomy level is determined by the complexity of the missions that the system is able to perform, the degrees of difficulty of the environments within which the system can perform the missions, and the levels of operator interaction that are required to perform the missions"*. This definition focuses on two important aspects of autonomous systems: the first is the ability to make decisions, the second is the uncertain environment in which the system operates, portending unexpected behaviors that, if nothing is done, could cause catastrophic financial or human damage. The need to make decisions autonomously in an uncertain environment induces software that can be quite complex and error prone (e.g., through the use of heuristics). Moreover, autonomous software is difficult to verify exhaustively since the state space is potentially huge and cannot be defined in advance since some situations with which the system might be faced may not be known before system deployment. Indeed, from a dependability viewpoint, autonomous systems have to address both *endogenous* hazards arising from faults and defects in the autonomous system itself and *exogenous* hazards arising from the physical, logical and human environment of the autonomous system [11]. Here, we investigate a safety monitor approach for addressing these hazards.

B. Safety Monitors

Several works on safety monitors can be found in the literature [12], [5], [8], [13]. Here, we briefly analyze four exemplary applications of safety monitors:

- the autonomous safety system of the Ranger Robotic Satellite Servicer [6] developed by NASA and the University of Maryland to refuel, repair, and upgrade the International Space Station (ISS);
- the *safety bag* developed within Elektra (*LockTrac Electronic Interlocking System*) [9] by Alcatel Austria to ensure that railway safety regulations are respected despite hardware and residual design faults;
- the Request and Report Checker (R2C) developed within the LAAS architecture for autonomous systems [7] to enforce safety constraints between concurrent real-time activities;
- the DLR Co-Worker developed by the German Aerospace Center DLR [14], which implements various recovery strategies in the face of possible collisions between the robot and the human.

Here, we will analyze these examples regarding the processes through which safety rules are *elicited*, how they are *expressed* and how they are *checked on-line* (see Table I). To the best of our knowledge, there has been no work that specifically addresses the problem of definition and elicitation of safety rules or safety constraints despite the abundance of research relating to monitoring and verification of safety constraints. In most cases, the cited safety rules are defined on the basis of standards and/or the experience of domain experts. This can be problematic in terms of completeness. In Elektra, electronic systems were analyzed, in the early versions, using FMECA (Failure Mode, Effects and Criticality Analysis). This was dropped in the later versions, preferring the implementation of a *safety bag* to ensure safety. The safety rules have been elicited using standards with the help of railway domain experts.
Similarly, no systematic process for safety rule elicitation was followed for the DLR Co-Worker. However, conditions defined with respect to combination of observable variables trigger mode changes and consequently the enforcement of specific safety tasks (which can be considered as safety rules). These conditions for functional mode changes and the corresponding safety tasks were defined by robotic domain experts.
In Ranger, only three hazards, issued from a risk analysis, were addressed. Nevertheless, no additional information was given by the authors.

Regarding the expression of the safety rules in a formal language, there is no universal approach. The choice of the language depends on the level of expressiveness required. Considering the four examples, only Elektra and R2C use a dedicated rule expression language (respectively PAMELA and a first order logic). However, in these examples, expression of real time properties is not considered.

On-line checking is poorly documented except for R2C where the safety rules are automatically transformed into a decision tree that allows fast on-line checking. In DLR and Ranger, all safety properties are an integral part of the functional code and are not verified separately in an independent safety monitor. In Elektra, the the safety monitor (named safety bag) is independent from the functional system and consists of an expert system, i.e., a knowledge base that includes the safety rules and the current system state; and an inference engine which decides which rule needs to be applied and checks if a request (system input) must be blocked or committed. If the safety monitor and the functional system disagree about an output, this output is blocked and the system is put into a safe state (e.g., setting lights to red and blocking all the trains).
We conclude from our survey of related work that there is a need to define a systematic process for safety rule elicitation, and to express the rules in a formal language that can express real-time aspects and is amenable to on-line checking.

III. The proposed methodology

A. Baseline and concepts

In the literature surveyed in section II, various meanings are associated with terms such as: *safety constraints*, *safety rules*, *safety requirements*, *safety properties*, etc. Here, we propose some precise definitions of these terms in order to base the proposed methodology on a firm conceptual foundation.

There are several definitions of "safety requirements" in the literature. Medikonda et Panchumarthy [15] give the definition: *"[The] requirement [for a system] to be safe is that it does not cause or contribute to a violation of any of the system constraints on safe behavior"*. Another definition (in context

	Elektra	Ranger	R2C	DLR Co-Worker
Elicitation of constraints/rules	Railway domain experts, standards	Risk analysis	Robotic domain experts	Robotic domain experts
Expression of constraints/rules	PAMELA	Included in the code	Logical first order predicates	Included in the code
Verification of constraints/rules	On-line verification (Inference engine of the safety channel)	Code execution	ExoGen	Code execution

TABLE I
MEANS FOR ELICITATION, EXPRESSION AND VERIFICATION OF SAFETY RULES

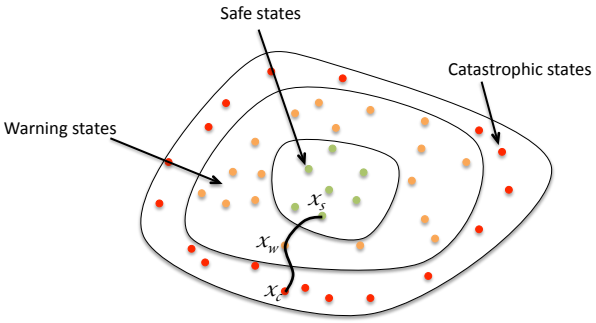

Fig. 2. Illustration of catastrophic, safe and warning states

of robotics) is given in the standard $IEC\ 9126-1$ [16]: *"Safety requirements are rules to ensure the safety of personnel associated with the use of a robotic system"*. These definitions are quite general, but it should be noted that the former definition is dependent on how the "system constraints on safe behavior" are defined. Here, we prefer to consider a safety requirement as a general but informal objective from which we can refine a set of formal safety invariants. We propose the following definition:

Definition 1. A **safety requirement** is a general high-level specification of what it means for a system to be safe. Example: *"the robot must not cause the patient to fall"*.

Firesmith *et al.* [17] define the notion of "safety constraints" that *"[specify] authorized system behaviors and component interactions, where a safety constraint specifies a specific safeguard"*. Medikonda and Panchumarthy [15] define a safety constraint in the following way: *"a hazard characterizes a system state that for safety reasons should not occur. If this is negated and some safety margins are included we get a safety constraint, i.e., a description of a property that the system should possess in order to be safe"*. These two definitions are inconsistent. The first definition implies that a safety constraint is a necessary condition (its violation means that a safeguard has failed) while the second definition considers it as a sufficient condition that, given the safety margin, might be temporarily violated. Indeed, the notion of a safety margin implicitly means that if such a constraint is violated, it is not too late to do something to prevent a catastrophe. Here, we follow the latter approach which we make explicit by means of four basic concepts: *safety action*, *safety condition*, *safety invariant*, and *safety trigger condition*.

Definition 2. A **safety condition** is a sufficient condition to avoid a catastrophic situation. Example: *"the robot is stopped and not used by a patient"*.

Definition 3. A **safety action** is an activity carried out explicitly to bring the system to a safe state. Example: *"apply emergency brake"*.

Definition 4. A **safety invariant** is a necessary safety condition, i.e., the violation of a safety invariant is intolerable in that it implies immediate harm and violation of a high-level safety requirement. Example: *"the robot speed shall not exceed 3 m/s"* (where 3 m/s is the speed beyond which harm is inevitable).

Definition 5. A **safety trigger condition** is a condition that, when asserted, triggers a safety action. Example: *"the robot speed is greater than 2 m/s"*.

We can then define a safety margin as follows:

Definition 6. A **safety margin** is the "distance" between a safety trigger condition and the negation of a safety invariant. Example: *in the examples above, the safety margin between the safety trigger condition ["the robot speed shall not exceed 3 m/s"] and the negation of the safety invariant ["the robot speed shall not exceed 2 m/s"] is equal to 1m/s.*

Finally, we define the notion of a *safety rule*. At the level of organizations, such rules are defined as regulations that must be respected by personnel, for example: "safety helmets must be worn at all times". Here, we use the term in a more operational sense. Based on the notions of safety trigger condition and safety action, we define a safety rule as follows:

Definition 7. A **safety rule** is a defined way of behaving in response to a hazardous situation. A safety rule can be operationalized as an if-then rule: Safety rule $\stackrel{\triangle}{=}$ if [safety trigger condition] then [safety action]. Example: *if the robot speed exceeds 2 m/s then apply emergency brake*.

Figure 2 provides an illustration of the main concepts in terms of a partition of the possible states of the monitored system into safe, warning and catastrophic states. A safety trigger condition must be asserted when the system passes from a safe state (e.g., x_s on figure 2) to a warning state (e.g., x_w).

B. Overview of the method

Figure 1 describes the steps of the proposed method of safety rule elicitation. The objective of this process is to elicit the safety rules that will be executed by the safety monitor.

If the system is in a warning state, then the safety monitor must trigger a safety action to bring the monitored system toward a safe state. The set of warning states specifies the

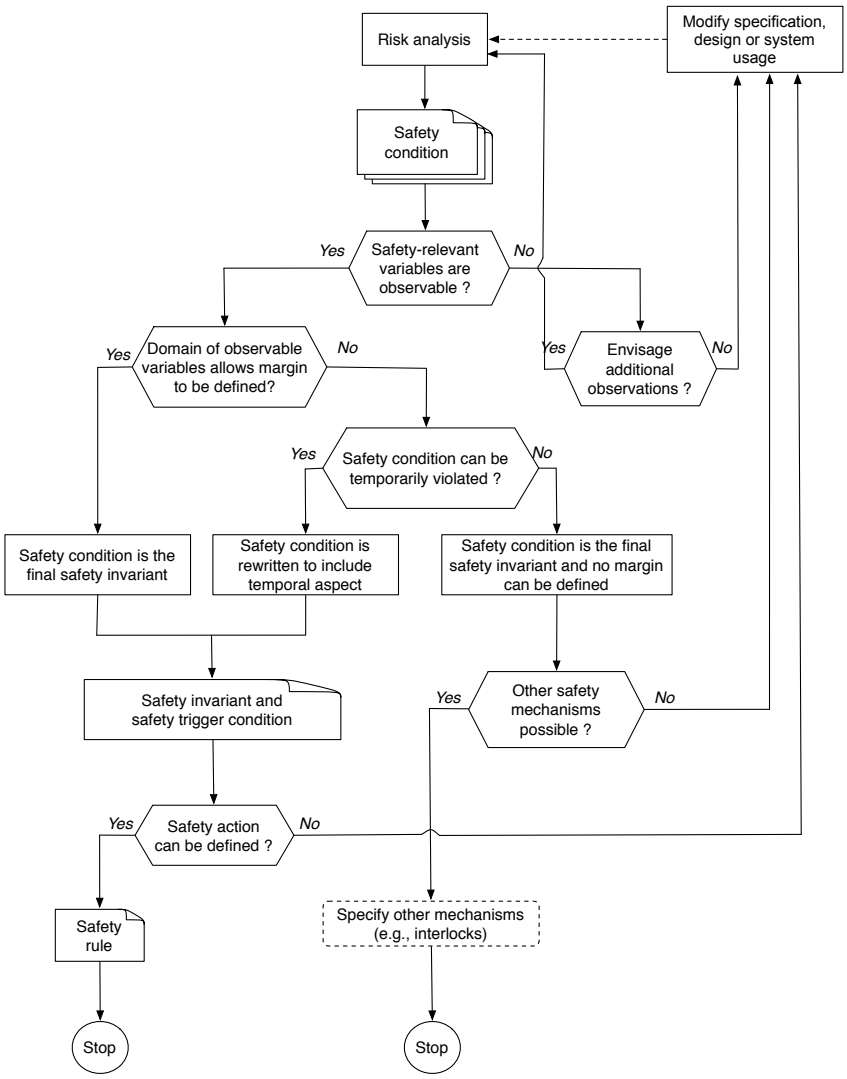

Fig. 1. Safety rule elicitation process

safety margin. If the safety monitor fails to bring the system back into a safe state, it may reach a catastrophic state (e.g., x_c).

Our goal is to define the thresholds separating warning states from catastrophic states, and safe state from warning state, corresponding respectively to safety invariants and safety trigger conditions.

Our starting point is a HAZOP/UML risk analysis [18]. This method allows the identification of major risks through the analysis of deviations of system usage scenarios described with UML (Unified Modeling Language) [19]. The method is systematic in that *guidewords* are applied to every *attribute* of each use case and each sequence diagram. Each such deviation is a possible erroneous system behavior that is further analyzed regarding potential causes, consequences and consequence severities (recorded in "deviation tables").

From these tables, a set of *safety conditions* are formulated. The next step is to analyze the nature of the "safety-relevant" variables contained in each safety invariant candidate. If the variables are not observable by the safety monitor, we need to envisage additional means of observation. If the variables are observable, we define safety margins for those variables whose admissible domains allow such margins to be defined. If a safety margin cannot be defined for a particular variable, we can attempt to specify a temporal margin by considering whether the safety condition can be temporarily violated. If a safety margin, and thus a safety trigger condition, can be defined, the final step is to define the actions that must be performed to bring the system toward a safe state. If no safety margin can be defined, the safety condition must be considered as an irreducible safety invariant that cannot be used to define safety rules executable by the safety monitor, but might be enforceable by some other mechanism.

When all safety conditions have been analyzed, the outputs of the proposed process are thus:

1) a set of safety rules resulting from safety invariants allowing margins to be defined;
2) a set of safety invariants not allowing margins to be defined, which can be handled by other safety mechanisms;
3) a set of safety invariants for which no enforcement mechanism can be defined, thus highlighting the need for further review of the specification, the design or the usage of the monitored system.

Hereafter, we will describe in detail each step of the process.

C. Identification of safety conditions from HAZOP/UML tables

1) HAZOP/UML based risk analysis: HAZOP (HAZard OPerability) [20] is a risk analysis method that allows the identification of major risks through the analysis of deviations of system design intention. Through the use of guidewords, it enables a systematic analysis. HAZOP/UML [18] is an adaptation of HAZOP for risk analysis based on system usage scenarios.

These scenarios are described with the common Unified Modeling Language (UML) [19] which is a standard for system description and easily understandable by non-experts. Furthermore, it is easy to model human-robot interactions with UML at the earliest stages of system development.

The method presented in [21] can be decomposed in two steps. First the system is represented in UML with use case diagrams and sequence diagrams. It provides a description for each use case, and represents a nominal scenario in a sequence diagram. Second, the HAZOP method is applied to each use case and each sequence diagram by applying the *guidewords* to the *attributes* of the diagram [18]. *Attributes* refer to preconditions, postconditions and invariants for a use case; and to predecessors and successors during the interaction, message timing, send and receive objects, guard conditions of the message and message parameters for a sequence diagram. The authors proposed an adaptation of the HAZOP guidewords for use cases and sequence diagrams. The result is a set of deviation tables which includes the identification of hazards and recommendations for requirements design and use of the system. Table II presents an extract of a deviation table. Each line in a deviation table corresponds to a single deviation. The columns of a deviation table include (among others):

- the definition of the deviation,
- a description of the effect of the deviation in the context of the considered use case,
- an analysis of the real-world effect of the considered deviation.

2) Identification of safety conditions: The safety rule elicitation process starts by the identification of *safety conditions* from the deviation tables. From each line of the deviation table in which the severity is different from *None*, i.e., $severity \in \{minor, moderate, serious, severe, critical, fatal\}$, an attempt is made to formulate safety conditions for each of the three columns: *Deviation*, *Use Case Effect*, *Real World Effect*. These safety conditions are typically the negation of the alteration defined in each column (see figure II). However, it should be noted that HAZOP/UML risk analysis is a systematic but informal process based on a human expertise, which can give rise to incoherent and unusable deviations from which it is not possible to specify a safety condition from any of the three columns.

For each use case and sequence diagram, we identify a set of safety conditions which are expressed using environmental quantities that we refer to as *safety-relevant* variables. Some of these variables may be *controllable* by the system (e.g, brakes, velocity setpoint,...). The environment variables collectively represent the state of the system's environment. Depending on whether or not appropriate perception mechanisms are available (laser detector, camera, etc), safety-relevant variables may or may not be *monitored* by the safety monitor. If a safety condition contains unobservable safety-relevant variables, additional observations can be envisaged and the safety condition rewritten accordingly. Otherwise, we must review the specification, the design or the usage of the monitored system.

D. Safety trigger condition and invariant definitions

Let $x \in X$ be the tuple of safety-relevant variables, $x =< x(1), x(2), ..., x(n) >$, such that X represents the set of discernible system states. A *safety invariant* is a true valuation of a function $SI : X \rightarrow \mathbb{B}$, i.e., $SI(x) = true$ [1].

With respect to a safety invariant $SI(x)$, the set of catastrophic states, X_{cata}, of the monitored system is thus:

$$X_{cata} = \{x \in X \mid \overline{SI(x)}\}$$

Given a safety invariant $SI(x)$ defining the set of catastrophic states, our aim is to refine the set $X \backslash X_{cata}$ into two disjoint sets: a set of safe states and a set of warning states. The safety trigger condition defined previously (Section III-A) is the condition that must be asserted when the system leaves a safe state to a warning state. Moreover, we require every path between a safe state and a catastrophic state must pass through at least one warning state (see Figure 2). We can then formally define the previous requirement using the safety condition. Let us consider two states x_1 and x_2.

We can define a path from x_s to x_c as a function π:

$$\pi \begin{cases} [0,1] \rightarrow X \\ 0 \mapsto x_s \\ 1 \mapsto x_c \end{cases}$$

such that π is a continuous function on continuous variables and monotonous on discrete variables (π follows transitions on discrete variables).

$\Pi'(x_s, x_c) = \{\pi |\ \pi$ is a path from x_s to $x_c\}$ is the set of all possible paths from x_s to x_c.

[1] We will also use $SI(x)$ and $\overline{SI(x)}$ to denote true and false valuations of $SI(x)$.

Project: MIRAS				**Use case description**				**Date:**	21/09/09	
HAZOP table number: UC01								**Prepared by:**	DMG	
Entity: UC01								**Revised by:**	JG	
				Use case name: Management of the strolling				**Approved by:**	AMM	
Element	**Guide word**	**Deviation**	**Use Case Effect**	**Real World Effect**	**Severity**	**Possible Causes**	**Integrity Level Requirements**	**New Safety Requirements**	**Remarks**	**Hazard Number**
The handles are at the right height (precondition)	No / none	The handles are not at the right height for strolling.	The patient strolls in a bad position.	Patient fall Health problems due to posture	serious	Failed software / hardware Incorrect robot mode Profile incorrect	SIL2 : SW/HW position detection of the patient	The profile must be validated by a human operator if it can not be validated automatically.		1,2
	Other than	**ref L1**								
		The handles are at the right height for strolling but the strolling does not begin.	The robot does not allow the strolling	Error in the sequences of execution / Patient confused / upset	None	Failed software / hardware Incorrect robot mode Profile incorrect	None	None		

TABLE II
EXTRACT FROM A HAZOP/UML ANALYSIS OF THE USE CASE "MANAGEMENT OF STROLLING".

We can now define a safety trigger condition, based on the previous requirement, as follows:

Constraint 1. A *safety trigger condition* is a function $STC : X \rightarrow \mathbb{B}$ that fulfills the following constraint:

$$\forall x_s, x_c, \overline{STC(x_s)}, \overline{SI(x_c)}, \tag{1}$$
$$\forall \pi \in \Pi(x_s, x_c) : \exists\ t,\ \ x_w = \pi(t),\ \ STC(x_w) \wedge SI(x_w)$$

With respect to a safety trigger condition $STC(x)$, the set of safe states of the monitored system is:

$$X_{safe} = \{x \in X \mid \overline{STC(x)}\}$$

and, with respect to safety invariant $SI(x)$, the corresponding set of warning states is:

$$X_{warning} = \{x \in X \mid (STC(x) \wedge SI(x))\} = (X \backslash X_{safe}) \backslash X_{cata}$$

The set of warning states corresponds to the safety margin between the assertion of the safety trigger condition and violation of the safety invariant. Those concepts are illustrated in a simplified representation in Figure 3.

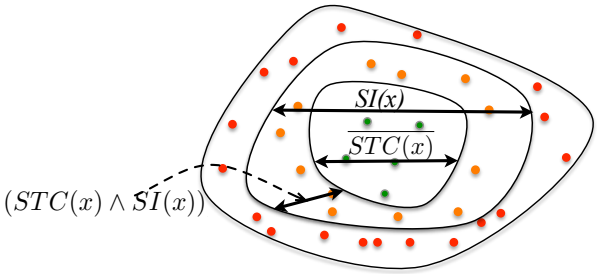

Fig. 3. The formal definition of the different system states

E. Safety actions

If a safety trigger condition is asserted, a safety action must be triggered to maintain the system in a safe state. The safety margin must be specified such that the system has time to react and return to a safe state before reaching a catastrophic state. The recovery action depends on the reaction strategy of the safety monitor. This strategy is defined by domain experts and depends particularly on the safety vs. availability objectives. It could happen that several safety trigger conditions are asserted at the same time, and trigger different safety actions that could be in conflict. We can determine the system states that could trigger several actions simultaneously, but it is the responsibility of domain experts to determine if these different actions are consistent or not (e.g., brake and accelerate at the same time). Formally specifying the safety trigger conditions can help in this respect since we can use solvers to check consistency and reachability of the resulting list of safety trigger conditions.

The safety actions triggered could vary depending on the dangerousness of the current warning state. For instance, we can set a time limit on the execution of the safety action. Once this time has elapsed, if the safety trigger condition is still asserted (i.e., the system has not yet recovered its nominal state), another more decisive safety action is triggered. This

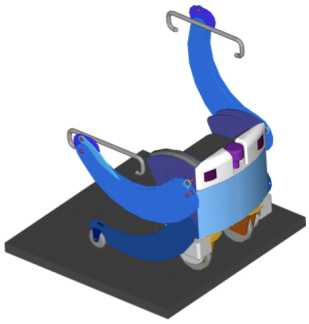

Fig. 4. The Multimodal Interactive Robot for Assistance in Strolling

strategy may be covered in our approach by defining several safety trigger conditions using time as an observable variable.

IV. Application to a Multimodal Interactive Robot

A. Target system

MIRAS (*Multimodal Interactive Robot for Assistance in Strolling*) is a collaborative research project involving ISIR (Institute of Intelligent Systems and Robotics), ROBOSOFT, LAAS-CNRS, and several French hospitals. This project aims to develop a semi-autonomous assistive robot for standing up and walking, designed to be used in elderly care centers by people suffering from gait and orientation problems (Figure 4). The objective is to let these patients become more autonomous, by helping them to stand and sit, and walk around. It also includes functionality to monitor the physical and physiological state of the patient, and an ability to autonomously move to the patient's position when summoned.

B. Elicitation of safety trigger condition and safety invariant

The functional behavior of the MIRAS robot was modeled using 16 use cases and 18 sequence diagrams. The HAZOP/UML risk analysis carried out on this model generated 395 deviations which in turn led to the identification of 15 high level dangers and 84 recommendations. The proposed methodology has currently been applied for now on 3 of the 16 use cases leading to the definition of 21 safety conditions, and 19 safety trigger conditions and invariants. Here, we will explain one safety rule elicitation process.

We consider the risk analysis of the use case *management of strolling*. From the first line of the deviation table illustrated in Table II, we can extract three safety conditions (see Figure 5), by negating the deviations obtained in the three concerned columns.

Let us consider the safety condition: *handles must be at the "right height" for strolling*; the "right height" is defined by experts as an interval $I = [h\_min,\ h\_max]$. Let the system state be defined by $x =< h\_handles, v >$, where $h\_handles \in \mathbb{R}^+$ is the height of the handles of the robotic

Deviation	Use Case Effect	Real World Effect
The handles are not at the right height for strolling	The patient strolls in a bad position	The patient falls or has problems due to posture

Safety conditions		
The handles must be at the right height for strolling	The patient must not stroll in a bad position	The patient must not fall or have problems due to posture

Fig. 5. Extraction of safety conditions from the HAZOP/UML table

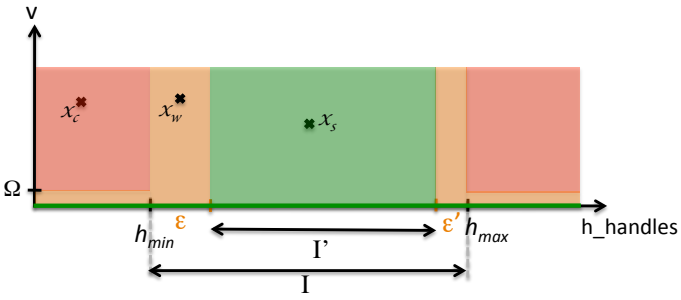

Fig. 6. Elicitation of safety margin on continuous variables (velocity and height of handles)

stroller and $v \in \mathbb{R}^+$ is the speed of the robot. Since these two variables are monitored in our system and allow margins to be defined (continuous variables); so we can consider the previous safety condition as the final safety invariant:

$$SI(< h\_handles, v >) = ((v < \Omega) \vee (h\_handles \in I))$$

where Ω is a (small) positive constant defining a maximum speed at which the robot can move while the handles are not at the right height. This safety invariant leads to a set of system states that can be split into a safe set and a warning set, as presented on Figure 6, with a safety trigger condition as follows:

$$STC(< h\_handles, v >) = ((v > 0) \wedge (h\_handles \notin I'))$$

with

$$I' = [h\_min + \epsilon, h\_max - \epsilon'],$$
$$\epsilon, \epsilon' \in \mathbb{R}^+, h\_min + \epsilon < h\_max - \epsilon'$$

This safety trigger condition can be checked on Figure 6 and fulfills the constraint (1) where $STC(y) \wedge SI(y) = (v \in [0, \Omega[\wedge h\_handles \notin I') \vee (h\_handles \in I \backslash I' \wedge v > 0)$ defines the set of warning states.

Alternative version of the previous example: We consider the same safety condition as in the previous example, i.e., that handles must be at the right height during strolling. However, in this example, we require that the robot be stationary when the handles are not at the right height. The corresponding safety invariant is then:

$$SI(< h\_handles, v >) = ((v = 0) \vee (h\_handles \in I))$$

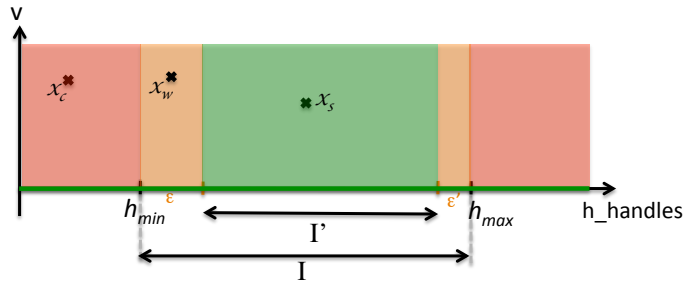

Fig. 7. Elicitation of safety margins on continuous variables (velocity and height of handles) - Alternative version

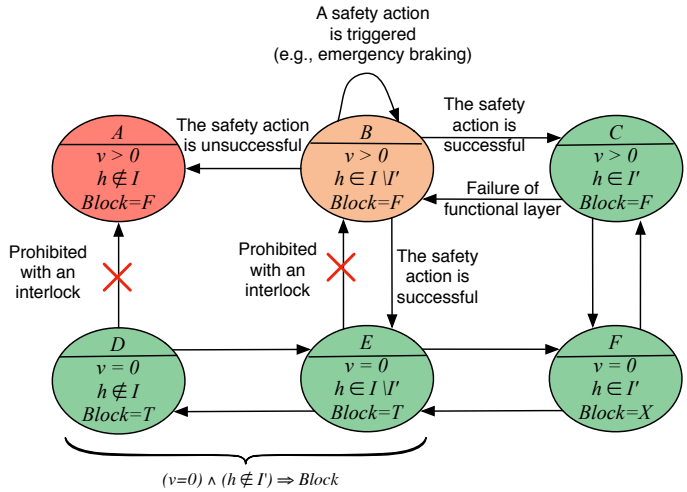

Fig. 8. State diagram illustrating all safety states including the interlock

Considering this invariant, the safe set is graphically presented on Figure 7 as the union of all states with $v = 0$ (x-axis) and sets with $h\_handles \in I$ (green and orange rectangles). The $Block$ variable will be used later. In this case, it is no longer possible to create a warning set between the set of safe states $\{< h\_handles, v > | v = 0 \wedge h \notin I\}$ and the set of catastrophic states $\{< h\_handles, v > | v > 0 \wedge h \notin I\}$. If the system is in the zone $\{< h\_handles, v > | v = 0 \wedge h \notin I\}$, increasing velocity will directly lead to a catastrophic state. Then, it is obvious that the safety trigger condition:

$$STC(< h\_handles, v >) = ((v > 0) \wedge (h\_handles \notin I')) \quad (2)$$

does not satisfy the constraint (1).

In this case, we claim that an interlock is needed in addition to safety monitoring. Consider the state diagram presented in Figure 8 which represents all possible states considering domains of Figure 7. One state is catastrophic (A), one is warning (B), and all the others are safe. In this diagram we illustrate that all transitions going directly from safe states to catastrophic states should be forbidden by means of an interlock subsystem. In our case, one solution is to prohibit transitions coming from states where $v = 0$, except when $h\_handles \in I'$. This can be done, for instance, by a mechanical interlock that blocks the wheels (thus maintaining $v = 0$) as long as $h\_handles \notin I'$.

We represent the state of the interlock by a boolean $Block$ (is true when activated, false when deactivated), and add it to the states as a new variable. In Figure 8, $Block = true$ for states D and E, $false$ for states A, B, C and either $true$ or $false$ for state F. In this case, the condition coming from the combination of states D and E, leads to the formula specifying the interlock : $v = 0 \wedge h \notin I' => Block$. Considering this interlock, the constraint (1) can be rewritten as:

$$\forall x_s x_c, \overline{STC(x_s)}, \overline{SI(x_c)}, \quad (3)$$
$$\forall \pi \in \Pi'(x_s, x_c) : \exists\ t,\ x_w = \pi(t),\ STC(x_w) \wedge SI(x_w)$$

where $\Pi'(x_s, x_c)$ is limited to paths allowed by the interlock ($\Pi' = \Pi \backslash \{paths\ \ forbidden\ \ by\ \ interlocking\}$). In this context, the previous safety trigger condition (2) satisfies the constraint (3).

Safety action and safety rule

In the previous examples (bad handle position while strolling), we have obtained the same safety trigger condition. If this safety trigger condition is asserted, we can engage the brakes on the axes of the handles and/or on the robot base since they are two controlled variables. Hence, the safety rule of the two examples could be expressed, as follows:

$$SR : (v > 0) \wedge (h\_handles \notin I')$$
$$\Rightarrow handles\_brake \wedge base\_brake$$

Note that the safety margins are specified to allow the safety monitor to trigger a safety action in order to bring the system toward a safe state. Hence, the defined safety margin must be sufficient to allow the safety monitor enough time to react before the monitored system reaches a catastrophic state. The sufficiency of the safety margin has to be decided by domain experts.

V. Conclusion and Future Work

Safety monitoring is a well-known safety assurance approach for critical systems. However, to the best of our knowledge, there has been no previous work on a systematic method for defining the safety rules that such safety monitors need to enforce. This paper proposes the foundations of such a method, based on the outputs of a systematic risk assessment process.

The contributions of the paper are the following. First, we have proposed precise definitions of concepts and formalisms to express safety conditions, safety trigger conditions, safety invariants and safety rules. Second, we have defined a systematic process for eliciting safety rules. This process starts with the identification of safety conditions by means of a HAZOP/UML risk analysis. We defined a corresponding formalism and a method for determining whether safety margins can be calculated and corresponding safety trigger conditions defined, or if safety interlocks are required.

Note that the completeness of the safety rules obtained by our method depends on the completeness of the deviations obtained from the risk analysis process (which depends on the thoroughness of the expert in charge of the risk analysis).

Several directions for future work can be identified. First, we need to define an automated method for checking that invariant/trigger condition pairs define valid safety margins (cf. constraint (1)) since the current manual procedure is tedious and will not scale to larger dimension tuples of safety-relevant variables. Second, for more complex safety rules, we plan to use a temporal logic such as TPTL (Timed Propositional Temporal Logic [22]) or timed automata. Third, we aim to grade warning states into different levels in order to be able to define nested safety rules based on the distance to a catastrophic state. Fourth, we plan to develop a prototype safety monitor implementing the generated safety rules and to experiment it on the rehabilitation robot. Finally, in the context of multi-functional systems such as service robots, we plan to investigate how safety rules can be grouped into safety modes [23] able to encompass different safety scenarios.

ACKNOWLEDGEMENTS

This work was partially supported by the MIRAS Research Project, funded under the TecSan (Technologies for Healthcare) program of the French National Research Agency (French ANR) and SAPHARI, Project funded under the 7th Framework Programme of the European Community.

REFERENCES

[1] R. Alami, A. Albu-Schaeffer, A. Bicchi, R. Bischoff, R. Chatila, A. De Luca, A. De Santis, G. Giralt, J. Guiochet, G. Hirzinger, F. Ingrand, V. Lippiello, R. Mattone, D. Powell, S. Sen, B. Siciliano, G. Tonietti, and L. Villani. Safe and dependable physical human-robot interaction in anthropic domains: State of the art and challenges. In *Workshop on Physical Human-Robot Interaction in Anthropic Domains, IEEE/RSJ International Conference on Intelligent Robots and Systems*, page 15, Bejing, China, 2006. IEEE.

[2] S. Haddadin, A. Albu-Schäffer, and G. Hirzinger. Requirements for safe robots: Measurements, analysis and new insights. *International Journal of Robotics Research*, 2009.

[3] A. Avižienis, J. C. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1:11–33, 2004.

[4] IEC 61508-3. Functional safety of electrical/electronic/programmable electronic safety-related systems - part 3: Software requirements. International Organization for Standardization and International Electrotechnical Commission, 2010.

[5] C. Pace and D. Seward. A safety integrated architecture for an autonomous safety excavator. In *International Symposium on Automation and Robotics in Construction*, 2000.

[6] S. Roderick, B. Roberts, E. Atkins, and D. Akin. The ranger robotic satellite servicer and its autonomous software-based safety system. *IEEE Intelligent Systems*, 19:12–19, 2004.

[7] F. Ingrand and F. Py. Online execution control checking for autonomous systems. In *Proceedings of the 7th International Conference on Intelligent Autonomous Systems*, Marina del Rey, California, USA, 2002.

[8] J. Fox. Designing Safety into Medical Decisions and Clinical Processes. In *International Conference on Computer Safety, Reliability and Security*, volume 1, 2001.

[9] P. Klein. The safety-bag expert system in the electronic railway interlocking system elektra. *Expert Systems with Applications*, 3:499 – 506, 1991.

[10] H. Huang, K. Pavek, B. Novak, J. Albus, and E. Messina. A framework for autonomy levels for unmanned systems (alfus). In *Proceedings of The AUVSI's Unmanned Systems North America*, june 2005.

[11] E. Baudin, J.P. Blanquart, J. Guiochet, and D. Powell. Independant safety systems for autonomy. Technical Report 07710, LAAS CNRS, 2007.

[12] J.P. Blanquart, S. Fleury, M. Hernerk, and C. Honvault. Software safety supervision on-board autonomous spacecraft. In *Proceedings of the 2nd European Congress Embedded Real Time Software*, 2004.

[13] Dennis Keith Peters. *Deriving real-time monitors from system requirements documentation*. PhD thesis, Hamilton, Ont., Canada, Canada, 2000. AAINQ66288.

[14] S. Haddadin, M. Suppa, S. Fuchs, T. Bodenmüller, A. Albu-Schäffer, and G. Hirzinger. Towards the robotic co-worker. In *International Symposium on Robotics Research*, 2009.

[15] B.S. Medikonda and S.R. Panchumarthy. An approach to modeling software safety in safety-critical systems. *Journal of Computer Science*, 5:311–322, 2009.

[16] ISO/IEC 9126. Software product evaluation - quality characteristics and guidelines for their use. International Organization for Standardization and International Electrotechnical Commission, 1991.

[17] D. Firesmith. Engineering safety requirements, safety constraints, and safety-critical requirements. In *Journal of Object technology*, 2004.

[18] D. Martin-Guillerez, J. Guiochet, D. Powell, and C. Zanon. A UML-based method for risk analysis of human-robot interactions. In *The Second International Workshop on Software Engineering for Resilient Systems, UK*. ACM, 2010.

[19] OMG. 2nd revised submission to OMG RFP ad/00-09-02 - Unified Modeling Language : Superstructure - version 2.0. Object Management Group, 2003.

[20] IEC61882. Hazard and operability studies (HAZOP studies): Application guide. International Electrotechnical Commission, 2001.

[21] J. Guiochet, D. Martin-Guillerez, and D. Powell. Experience with model-based user-centered risk assessment for service robots. In *Proceedings of the 2010 IEEE 12th International Symposium on High-Assurance Systems Engineering*, HASE '10, pages 104–113, Washington, DC, USA, 2010. IEEE Computer Society.

[22] R. Alur and T.A. Henzinger. A really temporal logic. *J. ACM*, 1994.

[23] J. Guiochet, D. Powell, E. Baudin, and J.P. Blanquart. Online safety monitoring using safety modes. In *6th IARP - IEEE/RAS - EURON Joint Workshop on Technical Challenges for Dependable Robots in Human Environments*, PASADENA, USA, 2008.