Surveillance en ligne de la sécurité basée sur les modes de sécurité

Jérémie Guiochet, David Powell, Étienne Baudin, Jean-Paul Blanquart

SURVEILLANCE EN LIGNE DE LA SÉCURITÉ BASÉE SUR LES MODES DE SÉCURITÉ
ONLINE SAFETY MONITORING USING SAFETY MODES

Jérémie Guiochet, David Powell and Étienne Baudin
Université de Toulouse
LAAS-CNRS
{name}@laas.fr

Jean-Paul Blanquart
EADS Astrium
Toulouse
jean-paul.blanquart@astrium.eads.net

Résumé

Les systèmes informatiques réalisent des tâches de plus en plus complexes, dont certaines peuvent mettre l'humain en danger. L'inévitable présence de fautes résiduelles de développement, de fautes physiques ou d'interaction activées pendant la vie opérationnelle, conduit à l'utilisation de systèmes indépendants de surveillance de la sécurité pour prévenir les défaillances catastrophiques. Nous présentons dans ce papier une approche et une formalisation du processus de détermination des règles de sécurité implémentées dans de tels dispositifs. Cette approche se base sur l'identification de modes de sécurité, dépendants des différentes tâches réalisées par le système sous surveillance. Chaque mode de sécurité est lié à un ou plusieurs modes fonctionnels, et est spécifié par un vecteur de permissivité, qui définit des domaines de variations autorisées de certaines variables physiques. L'ensemble des modes de sécurité est partiellement ordonné et représenté avec un graphe acyclique orienté. Ce graphe est ensuite utilisé pour construire un modèle spécifiant les transitions entre les modes de sécurité. Ceci fournit la base de l'implémentation du comportement du système de surveillance. Un cas d'étude a été réalisé sur un robot mobile équipé d'un bras manipulateur, travaillant en milieu humain au sein d'une usine.
Mots clés : Sûreté de fonctionnement, Mode de sécurité, Surveillance en ligne, Robotique

Summary

Computer systems have to carry out more and more complex tasks, including ones where humans can be endangered. Residual design faults in such systems, as well as the inevitability of physical faults and interaction faults during operation, motivate the use of safety monitors to prevent catastrophic failures. We present an approach and a formalization of the process for determining safety rules. It consists in identifying *safety modes*, according to the different tasks carried out by the monitored system. In practice, each safety mode is related to one or several functional modes and is specified by a *permissiveness vector* that defines the authorized domains of variation of key physical variables. The set of safety modes can be partially ordered according to their authorization vectors and can thus be represented as a directed acyclic graph. This graph is used to automatically build a model representing safety modes and their transitions, which can be implemented in an independent safety monitor. A case study has been carried out on a mobile manipulator robot, working in a factory alongside humans.
Keywords : Dependability, Safety mode, Online Monitoring, Robotics

1 Introduction

To carry out more and more complex tasks, computer systems are being given increasing authority and autonomy. This raises major dependability concerns, particularly for systems operating in the presence of humans such as robotic systems [1]. Despite the use of fault removal and prevention techniques, it is impossible to guarantee avoidance of all development faults or, of course, of physical faults and interaction faults arising during deployment. Thus, it is necessary to design systems capable of fulfilling their mission (reliability) and avoiding catastrophic failures (safety) despite the presence of faults [2]. Following this logic of accepting the inevitability of faults, we propose the use of an independent subsystem to carry out online verification of global safety properties in order to provide end-to-end protection against faults activated or occurring at run-time. In this paper, we will refer to the independent subsystem as the *safety monitor* [3], and to the functional system as the *monitored system*. Such safety monitors exist in many critical application domains: transportation [4], space [5], medical systems [6, 7], civil engineering [8], nuclear power plants [9], and multi-purpose robotics [10]. Many expressions have been used to denote such a safety monitor: *Monitoring and Safing Unit* [11], *Protection System* [12], *Safety Manager* [8], *Checker* [10, 13], *Safety Bag* [4] *Guardian Agent* [7]. The safety monitor obtains information about the state of the monitored system and the environment either by directly reading the values of key variables maintained by the monitored system or by using dedicated additional sensors. It decides whether the observed situation is safe or not according to a set of monitoring rules. If a hazardous situation is detected, the safety monitor triggers a forward recovery procedure to put the monitored system in a safe state.

Most of the literature on safety monitors focusses on system architecture and on how assertion-checking can be integrated into the architecture [14]. But none that we are aware of addresses the process leading to the identification, definition and expression of the safety assertions to be checked. Moreover, for the above cited systems, the same safety rules are checked continuously, without taking account the fact that systems can perform diverse tasks where the appropriate safety rules can change according to the tasks be carried out. This is particularly important with multi-functional robots, especially those including autonomous decisional capabilities.

The contribution presented in this paper aims to overcome these issues. We propose a structured approach based on the concept of *safety modes*. A systematic and formalized approach is proposed to improve the safety rule identification process, and to partially automate the production of safety rules. The notion of safety modes acknowledges the fact that systems can carry out tasks with different safety rules. For each functional mode of the monitored system, the safety monitor activates the corresponding safety mode, and checks a specific set of monitoring rules. An additional aspect of the proposed method is that it helps designers to specify reaction strategies that are more flexible than emergency stop, which leads to improved availability and efficiency. A preliminary case study has been carried out on a mobile manipulator robot developed for the PHRIENDS project[1]. In the considered scenario, the user can order the robot to pick up a specific object from a specific location and then to carry it to another location and to place it there, or give it to him. Considering these tasks, we assume a high-level of interaction between the robot and the human. First, they both work in the same work space, which is often avoided: robots are usually enclosed in a specific area or must follow a dedicated trajectory. Second, collaborative

[1]PHRIENDS (Physical Human Robot Interaction: Dependability and Safety) is a project supported by the European Community under the 6th Framework Program, http://www.phriends.eu.

work is possible: the robot is able to take an object from the hand of a human, and the user can physically stop the robot during a task by catching any part of the robot arm. The considered environments are workshops or factories. Examples coming from this case study are used throughout the paper to illustrate theoretical and formal aspects

In Section 2 of this paper, we formalize the concept of safety modes and define two types of safety rules: permissiveness rules and context rules. Section 3 discusses the management of safety modes and, in particular, the transitions between safety modes. Finally, Section 4 discusses the benefits of safety modes with respect to simplification of the specification process. The paper concludes by lessons learnt and future work.

2 Safety Modes

The objective of safety modes is to describe the dynamics of the safety monitor, identifying different sets of safety rules activated according to the current tasks of the system. We propose a formal notation for safety mode specification, and also a partial order relation between modes that facilitates the production of safety rules.

2.1 Background

In the literature, the term *mode* is mainly used to describe different configurations of a system, in which different control laws are applied [15]. In particular, modes allow discrete changes from one control law to another. Relatively few works link the notions of safety rules and functional modes of operation.

In [16], four functional modes of a manipulator robot arm are defined from the combinations of two discrete variables, each with two possible values: the maximum speed of the arm (slow or fast), and the choice of motion control (automatic or guided by a human). The control laws are different in each mode. Some transitions between modes require the introduction of an intermediate mode to respect safety criteria. In particular, the transitions from the automatic modes to the manual modes require an intermediate mode in which the arm is stopped. In [17], three modes are used to manage, within a *safety manager*, the in-flight testing of an experimental neural network for fly-by-wire flight control. The three modes are: *nominal* (with conventional flight control), *research* (neural network flight control), and *failure* (research mode with injected faults). When the pilot pushes a button in the nominal mode, the research mode is engaged if the necessary conditions (about flight parameters, hardware, software, communication buses) are fulfilled. If any of these conditions does not hold, the safety manager automatically returns to the nominal mode. From the research mode, the pilot can engage the failure mode. The monitoring rule set of this mode is a superset of the research mode rule set with two additional rules.

Those two approaches illustrate that in many systems, operational and safety modes are tightly linked. Our approach is similar in that it consists in defining some discrete modes, in which is applied a specific set of safety rules that aims to prevent the system from reaching an unsafe state. The main difference is in that we introduce safety modes as a first-class concept distinct from functional aspects and propose a general method for specifying them.

2.2 Definitions

We define a *safety mode* as a state of the safety monitor, in which a specific set of monitoring rules is applied. The safety monitor obtains information about the monitored system and its environmental context, determines the current safety mode, and checks if the conditions of a safe execution are fulfilled. Functional mode changes that do not impact safety do not lead to a change in safety mode. Hence, there is a one to many binary association between safety modes and functional modes.

For each safety mode and each transition between safety modes, the safety rules that are checked may be different. We distinguish two types of rules. The first type are *permissiveness rules*, that define the functional capabilities allowed for the current safety mode. This type of rule checks that some functional variables do not violate the domains of variation authorized in the current safety mode. For example, speed ranges [0, 1] $m.s^{-1}$ and [0, 2] $m.s^{-1}$ for a mobile robot are two possible authorized domains. However, permissiveness rules do not deal with context or environment. The second type of rules, *context rules*, are intended for monitoring the system with respect to hazardous situations, either related to the environmental conditions, or the state of critical resources.

2.3 Permissiveness rules

Safety modes are defined on the basis of safety-relevant functional variables that have ranges of authorized variations. We define the *authorization variable* A_f, associated to a functional variable f, to denote the *authorized domain* of variation of f. A_f may take on different values $A_f^{(i)}$ at different times, thus leading to a variable constraint on the functional variable f. A domain that can be authorized is called an *admissible domain*. Each admissible domain of f is a member of the powerset (set of all possible subsets) of the domain of f, noted $\mathcal{P}(dom(f))$.

Definition 1 *The **set of admissible domains** for a functional variable f is defined as $\mathcal{A}_f = \{A_f^{(1)}, ..., A_f^{(m)}\}$, where $\forall i, A_f^{(i)} \in \mathcal{P}(dom(f))$*

$\mathcal{A}_f$ is thus a family of sets over $dom(f)$. A property of a family of sets is that it is partially ordered under the inclusion relation, so the elements of $\mathcal{A}_f$ can be represented as a directed acyclic graph, which we call the *permissiveness graph*. Two nodes $A_f^{(i)}$, $A_f^{(j)}$ in the permissiveness graph are linked by a directed arc (from $A_f^{(i)}$ to $A_f^{(j)}$) if $A_f^{(i)}$ includes $A_f^{(j)}$ as a sub-domain. In this case, we say that $A_f^{(i)}$ is more *permissive* than $A_f^{(j)}$, since it allows a wider variation of safety-relevant functional variable f. Conversely, $A_f^{(j)}$ is said to be more *restrictive* than $A_f^{(i)}$.

Definition 2 *An admissible domain $A_f^{(i)}$ is more **permissive** (or, less **restrictive**) than $A_f^{(j)}$ if $A_f^{(i)} \supseteq A_f^{(j)}$.*

Paths on the permissiveness graph represent possible reaction strategies in the face of detected hazardous situations, under the premise that more restricted domains of safety-relevant functional variables are safer than more permissive ones. To allow feasible changes of a functional variable without any discontinuity, the permissiveness graph must be weakly connected. Thus, if two admissible domains are not ordered by the inclusion relation, they must either include, or be included by, another admissible domain.

Definition 3 *A set of admissible domains for f is said to be **feasible** if:*
$\forall i, j, \left(A_f^{(i)} \not\subseteq A_f^{(j)}\right) \wedge \left(A_f^{(i)} \not\supseteq A_f^{(j)}\right) \Rightarrow \exists k, \left(A_f^{(k)} \supseteq A_f^{(i)} \cup A_f^{(j)}\right) \vee \left(A_f^{(k)} \subseteq A_f^{(i)} \cap A_f^{(j)}\right)$

Let us consider a few simple examples. Let $BaseSpeed$ (type `real`) be the speed of the base of a mobile robot, such that $dom(BaseSpeed) \subseteq \mathbb{R}^+$. Let us consider three admissible domains (in this case, intervals) for $BaseSpeed$:

$$A_{BaseSpeed} = \begin{cases} A^{(1)}_{BaseSpeed} = [0,0] m.s^{-1} \\ A^{(2)}_{BaseSpeed} = [0,1] m.s^{-1} \\ A^{(3)}_{BaseSpeed} = [0,2] m.s^{-1} \end{cases}$$

The first possible value of the authorization variable $A_{BaseSpeed}$ defines a constraint of no movement ($BaseSpeed = 0$), whereas the other two define two different upper speed limits. In this case, the admissible domains form a totally-ordered set: $A^{(3)}_{BaseSpeed} \supset A^{(2)}_{BaseSpeed} \supset A^{(1)}_{BaseSpeed}$, ranging from the most permissive to the most restrictive constraint on $BaseSpeed$.

As a second example, consider a functional variable $GripperState$ (type `enum`) representing the state of a gripper at the tip of a robot arm, such that $dom(GripperState) = \{Open, Closed\}$. We can consider three admissible domains for $GripperState$:

$$A_{GripperState} = \begin{cases} A^{(1)}_{GripperState} = \{Open\} \\ A^{(2)}_{GripperState} = \{Closed\} \\ A^{(3)}_{GripperState} = \{Open, Closed\} \end{cases}$$

The first and second values of $A_{GripperState}$ denote obligatory positions of the gripper, whereas the third value indicates that both positions are authorized. Here, $\mathcal{A}_{GripperState}$ is only

partially-ordered with $A^{(3)}_{GripperState}$ representing the most permissive domain and $A^{(1)}_{GripperState}$ and $A^{(2)}_{GripperState}$ being alternative less permissive domains. Alternatively, authorized gripper actions might be designated by means of an authorization variable $A_{GripperTransition}$ pertaining to a functional variable $GripperTransition$ representing transition events between gripper states where a possible set of values for $A_{GripperTransition}$ might be:

$$A_{GripperTransition} = \begin{cases} A^{(1)}_{GripperTransition} = \emptyset \\ A^{(2)}_{GripperTransition} = \{open, close\} \end{cases}$$

In this third example, the first value of $A_{GripperTransition}$ forbids any change in state of the gripper, whereas the second value allows both possible transitions. The domain $A^{(2)}_{GripperTransition}$ is evidently more permissive than $A^{(1)}_{GripperTransition}$.

The three examples all lead to sets of admissible domains that can be represented as weakly-connected directed acyclic graphs, and are thus feasible sets in the sense of Definition 3.

The notion of a set of admissible domains for a single safety-relevant functional variable f can be generalized to consider *vectors* of n safety-relevant variables, $\vec{f} = (f_1, ..., f_n)$ such that $\mathcal{A}_{\vec{f}} = \{A^{(1)}_{\vec{f}}, ..., A^{(m)}_{\vec{f}}\}$ denotes a set of m admissible domains for $\vec{f}$. $\mathcal{A}_{\vec{f}}$ is a subset of the Cartesian product of the sets of admissible domains for each element of $\vec{f}$: $\mathcal{A}_{\vec{f}} \subseteq \mathcal{A}_{f_1} \times ... \times \mathcal{A}_{f_n}$. We define $\vec{F}$ to be the vector of *all* the safety-relevant variables of a given system S. Each admissible domain $A^{(i)}_{\vec{F}}$ defines a *safety mode* m_i of system S.

Definition 4 *A **safety mode** m_i of a system S with a vector of safety-relevant functional variables $\vec{F} = (f_1, ..., f_{|\vec{F}|})$ is defined by an affectation of authorized domains for each element of $\vec{F}$, $A^{(i)}_{\vec{F}} = (A^{(i)}_{f_1}, ..., A^{(i)}_{f_{|\vec{F}|}})$. We call $A^{(i)}_{\vec{F}}$ the **permissiveness vector** associated with mode m_i.*

Safety modes can be partially ordered by permissiveness by direct extension of Definition 2, applied to the modes' permissiveness vectors.

Definition 5 *A safety mode m_i is more **permissive** (or, less **restrictive**) than m_j if $A^{(i)}_{\vec{F}} \supseteq A^{(j)}_{\vec{F}}$, i.e., if $\forall k \in [1, |\vec{F}|], A^{(i)}_{f_k} \supseteq A^{(j)}_{f_k}$. We use the notation $m_i \succ m_j$ to denote the permissiveness relation between safety modes.*

The permissiveness vector of a given safety mode defines the domains that must be respected in that mode by the set of safety-relevant functional variables of the system. Formally, we express this as the *permissiveness rule* associated with the safety mode.

Definition 6 ***Permissiveness rule**: in safety mode m_i with associated permissiveness vector $A^{(i)}_{\vec{F}}$, the permissiveness rule is defined as the boolean function: $P(m_i) = (\forall k \in [1, |\vec{F}|], f_k \in A^{(i)}_{f_k})$*

The notion of a set of *feasible safety modes* follows by direct extension of Definition 3.

Definition 7 *A set of safety modes for system S is said to be **feasible** if:*
$\forall i, j, \left(A^{(i)}_{\vec{F}} \not\subseteq A^{(j)}_{\vec{F}}\right) \wedge \left(A^{(i)}_{\vec{F}} \not\supseteq A^{(j)}_{\vec{F}}\right) \Rightarrow \exists k, \left(A^{(k)}_{\vec{F}} \supseteq A^{(i)}_{\vec{F}} \cup A^{(j)}_{\vec{F}}\right) \vee \left(A^{(k)}_{\vec{F}} \subseteq A^{(i)}_{\vec{F}} \cap A^{(j)}_{\vec{F}}\right)$

An admissible set of safety modes is one in which it is possible for the system to change safety modes without necessarily falsifying a permissiveness rule, i.e., there are no discontinuities in the admissible domains of the system's safety-relevant functional variables. As an example, consider the mobile robot equipped with a gripper on a manipulator arm. We have identified four safety-relevant functional variables: $BaseSpeed$, $GripperTransition$ (defined as previously), $ArmStatus$ (type `enum`) where $dom(GripperTransition) = \{folded, unfolded\}$, and $ArmForce$ (type `real`) where $dom(BaseSpeed) = \mathbb{R}^+$. The identification of the safety-relevant functional variables is based on a preliminary risk analysis and UML sequence diagrams as in [18].

We wish to define the following six safety modes for this system:

- *FastMove*: the robot can move at maximum speed, assuming that no human beings are in its vicinity. While moving fast, its arm must be in the folded position.
- *SlowMove*: the robot can move at reduced speed, even if there are human beings in its vicinity.
- *FastWork*: the robot can use the full functionality of its manipulator arm, as long as its base is stationary and there are no human beings in its vicinity.
- *CollaborativeWork*: the robot can use its arm at reduced speed in the vicinity of or in collaboration with a human being, as long as its base is stationary.
- *MoveAndWork*: the robot can use its arm at reduced speed while moving, as long as no human beings are in its vicinity and any load it manipulates is not dangerous.
- *Stop*: the base and the arm of the robot are stationary; any load in the gripper must not be dropped.

Table 1 defines the permissiveness vectors over the four safety-relevant variables defined previously. Figure 1 shows the permissiveness graph that can be generated automatically from the set of safety modes defined in Table 1. It can be seen that the set of safety modes is *feasible* (the permissiveness oriented graph is weakly connected) and that safety mode $Stop$ is the most restrictive safety mode (it appears as a sink node on the graph).

2.4 Context rules

As previously presented, permissiveness rules do not include external conditions, and particularly hazardous situations induced by non-controllable variables. We introduce the notion of *context rules* to define external conditions that must be respected to ensure safety. For instance, the presence of a human in the robot's vicinity can be considered as a *context variable* that could be used in such a context rule. Context rules can be defined using the same formal notation as for permissiveness rules. For instance, let us consider the context variable $HumanDistance$, which is the distance between the robot and the closest human. This information is safety-relevant for instance in the $FastMove$ mode of the previous example where the robot can reach speeds that could cause injury in case of collision. Context rules for different safety modes can be defined in terms of different $HumanDistance$ ranges such as [0, $+\infty$], [0.5 , $+\infty$], and [2, $+\infty$] meters. For example, the context rule for the $FastMove$ mode in Figure 1 would be $C(FastMove) = (HumanDistance \in A^{(3)}_{HumanDistance})$, where $A^{(3)}_{HumanDistance} = [2, +\infty]$, i.e., we specify that in the safety mode $FastMove$ the distance to any humans cannot be under 2 meters. In this example, only one context variable is used in the rule, but in general, the context rule could include constraints on many other context variables (e.g., natural light intensity or cluttering level of environment, which both have a high impact on the robot's abilities to sense its environment and plan a safe trajectory).

Formally, we define $\vec{G} = (g_1, ..., g_{|\vec{G}|})$ to be the vector of safety-relevant context variables related to the environment of system S.

Definition 8 *The **set of admissible domains** for a context variable g_k is defined as $\mathcal{A}_{g_k} = \{A^{(1)}_{g_k}, ..., A^{(m)}_{g_k}\}$, where $\forall i, A^{(i)}_{g_k} \in \mathcal{P}(dom(g_k))$*

Definition 9 *A **context vector** for safety mode m_i is an affectation of authorized domains for each element of $\vec{G}$, $A^{(i)}_{\vec{G}} = (A^{(i)}_{g_1}, ..., A^{(i)}_{g_{|\vec{G}|}})$.*

Definition 10 ***Context rule**: in safety mode m_i, it must be the case that $C(m_i)$ is true, with $C(m_i) = (\forall k \in [1, |\vec{G}|], g_k \in A^{(i)}_{g_k})$*

As presented in Table 2, two safety-relevant context variables have been identified in our case study to detect hazardous situations. The first one is the safe distance with respect to the closest human, $HumanDistance$, which has currently been fixed at 2 meters in $FastMove$ safety mode, and 0.5 meters in $SlowMove$ safety mode. The second one is the hazardous nature of the load, which impacts the functional abilities of the robot (for instance, holding a hazardous load will constrain the mobile base and the arm to move slowly).

Table 1: Example of safety modes and associated permissiveness vectors

Safety modes	Safety related functional variables			
	BaseSpeed	ArmStatus	ArmForce	GripperTransition
FastMove	[0,2]	{folded}	[0,10]	$\{\emptyset\}$
SlowMove	[0,1]	{folded}	[0,10]	$\{\emptyset\}$
Stop	[0,0]	{folded}	[0,10]	$\{\emptyset\}$
CollaborativeWork	[0,0]	{folded,unfolded}	[0,50]	{open, close}
FastWork	[0,0]	{folded,unfolded}	[0,120]	{open, close}
Move&Work	[0,1]	{folded,unfolded}	[0,120]	{open, close}

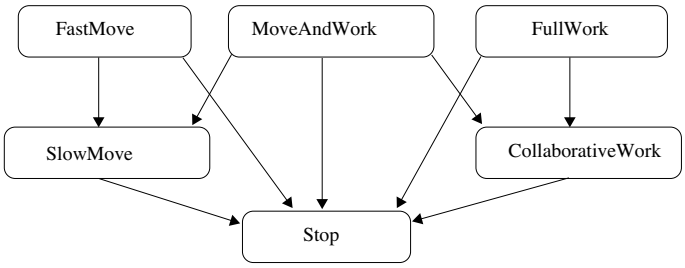

Figure 1: Graph of safety modes with permissiveness relation order (a → b: a is more permissive than b)

2.5 Safety mode automaton

Permissiveness and context rules are used to detect if the system is entering a hazardous state. These rules can be used both for checking conditions that must be maintained in a safety mode (*mode conditions*) and for checking conditions that must be fulfilled to allow transitions between safety modes (*guard conditions*). Mode conditions and guard conditions are obtained from the permissiveness and context vectors of the safety modes. Practically, the guard condition on a transition between safety modes m_1 and m_2 depends on the permissiveness relation between m_1 and m_2 (defined in Definition 5) which implies three types of transition (an example is presented in Figure 2):

1. To a more permissive safety mode: $m_2 \succ m_1$. The monitored system increases its functional abilities, but some contextual conditions have to be fulfilled (e.g., absence of humans). According to the permissiveness relation, $A^{(2)}_{\vec{F}} \supseteq A^{(1)}_{\vec{F}}$, thus, reaching m_2 from m_1, the $P(m_1)$ rule still remains $true$. In this case, only the $C(m_2)$ rule has to be checked for this transition (consider, for instance, a transition from $SlowMove$ to $FastMove$ in Figure 1).
2. To a less permissive safety mode: $m_2 \prec m_1$. As functional abilities decrease, the permissiveness rule of the targeted mode (m_2) must be checked. For example, if the monitored system has to enter the $Stop$ mode, it has to stop its base and arm before the acceptance of the transition by the safety monitor.
3. To an incomparable safety mode: $m_2 \not\succeq m_1$ and $m_2 \not\preceq m_1$. In this case, the guard condition is defined by identifying a path in the permissiveness graph, passing through less permissive intermediate modes. The resulting guard condition is a logical and of all guard conditions along the path to the final mode.

With these three types of transition, all the transition conditions can be determined. However, all the transitions are not functionally interesting. For that reason, some transitions may be manually specified as forbidden. In Figure 2, transitions to more or less permissive safety modes are represented by solid arrows, whereas transitions between incomparable safety modes are represented by dashed arrows. An illustration is the case of the transition from $CollaborativeWork$ to $SlowMove$ mode. Here, the intermediate mode $Stop$ is used to evaluate the final guard condition: $P(Stop) \wedge C(SlowMove)$. The transitions that are not represented are forbidden.

To ease readability, the request of the monitored system $changeMode(m_i)$ on each transition condition is not represented on the automaton of Figure 2. For example, using statechart notation, the transition from $Stop$ to $SlowMove$ should be annotated with: $changeMode(SlowMove)[P(SlowMove]$, i.e., a transition is activated on event $changeMode(SlowMove)$, guarded by the condition $P(SlowMove)$.

3 From Safety Modes to Safety Rules

Once safety modes, and permissiveness and context vectors have been specified, it is necessary to analyze how the safety monitor will react if the induced conditions are not fulfilled. Whereas previous sections can be applied in a generic way, this section is highly linked with the design of the system, because it requires knowledge about which reaction strategies are possible. However, we present in this section some guidelines to implement safety modes and safety rules.

3.1 Activation of the safety modes

Since the safety rules to be ensured by the safety monitor depend on the current safety mode, one fundamental issue remains the identification of the current safety mode. This can be done by the observation of the physical attributes of the monitored system to deduce the corresponding safety mode. This approach has a major disadvantage: in some cases the safety monitor may not be able to identify the current safety mode. This issue is studied in the diagnosis community, where graph algorithms are used to identify system state given a set of observation variables. Of course our method can integrate this approach, but we decided to first focus on the monitoring of safety rules. To simplify our method, we make some assumptions. First, we assume that the monitored system has been designed with a set of functional modes, linked with a many-to-one relation to the safety modes. Second, we have chosen to be notified by the monitored system about its safety mode change requests. This assumption avoids any ambiguity. The drawback of this solution is the required confidence given to the monitored system, which may send erroneous information and particularly wrong mode change requests. In that case, it should be demonstrated that in any case of mismatch, the safety monitor will put the system in a safe state. This may lead to a lower availability but guarantees a higher safety.

3.2 Mode condition violation

In every safety mode, the monitor should be able to detect if there is a violation of the corresponding mode conditions, of type P or C. When it is not possible for the system to ensure both, the system cannot remain in the current safety mode, so the safety monitor must force a transition towards a safe state, i.e., a less permissive safety mode. To do this, we propose the concept of a *fall-back mode*, in which actions are undertaken to reach conditions of a less permissive safety mode (for example, a fall-back mode might correspond to activation of emergency braking). A limit can be set on the time

Table 2: Case study safety modes and associated context vectors

Safety modes	Safety related context variables	
	HazardousLoad	HumanDistance
FastMove	{False}	[2, $+\infty$]
SlowMove	{False}	[0.5, $+\infty$]
Stop	{True,False}	[0, $+\infty$]
CollaborativeWork	{True,False}	[0, $+\infty$]
FastWork	{False}	[0, $+\infty$]
Move&Work	{False}	[0, $+\infty$]

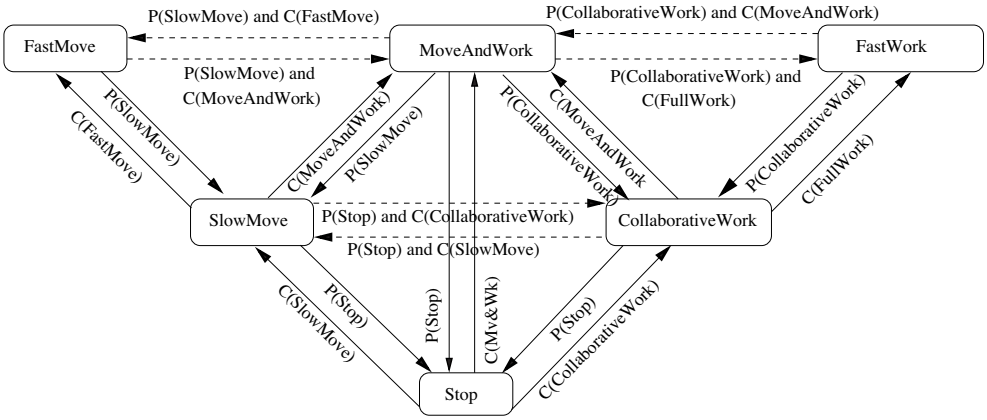

Figure 2: Generated safety modes automaton (only guard conditions are shown as transition labels)

spent in the fall-back mode. If that time limit is exceeded, the safety monitor attempts to force a transition towards an even less permissive mode, e.g., a $Stop$ mode. As a last resort, the safety monitor should put the monitored system in an ultimate fall-back mode, such as $EmergencyStop$, from which it may not be possible to recover, but which guarantees safety. Many levels of fall-back modes can be considered, but particular attention should be given to reaction time constraints (stopping the robot has to be fast in case of a hazardous situation).

The statechart in Figure 3 is an example. Only three safety modes are represented ($FastMove$, $SlowMove$, and $Stop$), and three fall-back modes are introduced ($ControlledMovement$, $ControlledStop$, $EmergencyStop$). As a first reaction, the safety monitor tries to control the speed. If it is not possible to reach the $SlowMove$ mode, the $ControlledStop$ mode is activated. As a last resort, $EmergencyStop$ is activated.

A protocol must be defined to allow communication between the monitored system and the safety monitor. Indeed, in case of activation of a fall-back mode, the actions initiated by the safety monitor must be taken into account by the monitored system. To ensure full independence between the safety monitor and the monitored system, actions such as $reduceSpeed()$ (it is not specified here how speed can be reduced) should be engaged by the safety monitor. However, the monitored system should be made aware of this action so as to take it into account in its future plans.

3.3 Transition condition violation

We now analyze potential violations of a guard condition. As before, three types of transition are identified: a transition to a more permissive mode, to a less permissive mode, and to a mode that is not comparable. Indeed, in case of a $changeMode()$ request, the guard condition can include a permissiveness rule (P rule), a context rule (C rule), or both, and the reaction of the safety monitor is described in a generic way.

3.3.1 Transition towards a more permissive safety mode (C rule)

A transition towards a more permissive mode is allowed if the environment has changed in such a way that it respects more restrictive contextual conditions (for instance there are no humans, no hazardous obstacles). If C rules are not fulfilled, the safety monitor should keep the system in the current mode, and reject the mode change request. This implies that the monitored system does not switch to its desired functional mode, and can integrate this rejection in its future plans. Again, as for mode conditions (previous section), a protocol needs to be defined for the monitored system to receive and react to mode change requests rejected by the safety monitor.

3.3.2 Transition towards a less permissive safety mode (P rule)

Switching to a less permissive mode is guarded by P rules, i.e., functional variables need to be restricted (for instance, speed has to be reduced) to satisfy the P rules before mode switching is allowed. If those conditions are not verified, this means that the monitored system wants to reach such a mode, but is unable to do so. Then, the safety monitor has to react to impose the conditions mandated by the more constrained environment. This is again done through the notion of fall-back modes, as described in Section 3.2. For example, consider the transition from $FastMove$ to $SlowMove$ on Figure 3. If the guard condition $P[SlowMove]$ is not fulfilled when the mode change is requested, the safety monitor can engage the $ControlledMovement$ fall-back mode previsously defined for handling the violation of the mode condition of $FastMove$. For example, considering the transition from $FastMove$ to $SlowMove$ on Figure 3. If the guard condition $P(SlowMove)$ is not fulfilled when the mode change is requested the safety monitor can engage the $ControlledMovement$ fall-back mode previously defined for handling a violation of the mode condition of $FastMove$. Only a subset of the whole diagram is given here for readability. Mode change requests are represented by $cm()$.

3.3.3 Transition towards a non comparable safety mode (P and C rules)

In the third case, a distinction should be made between P and C rules. Indeed, if a C rule is not verified, this means that the environment conditions do not fulfill the requirement, so the system should stay in its current mode (as previously presented). If a P rule is not satisfied, then the safety monitor will take over to force the monitored

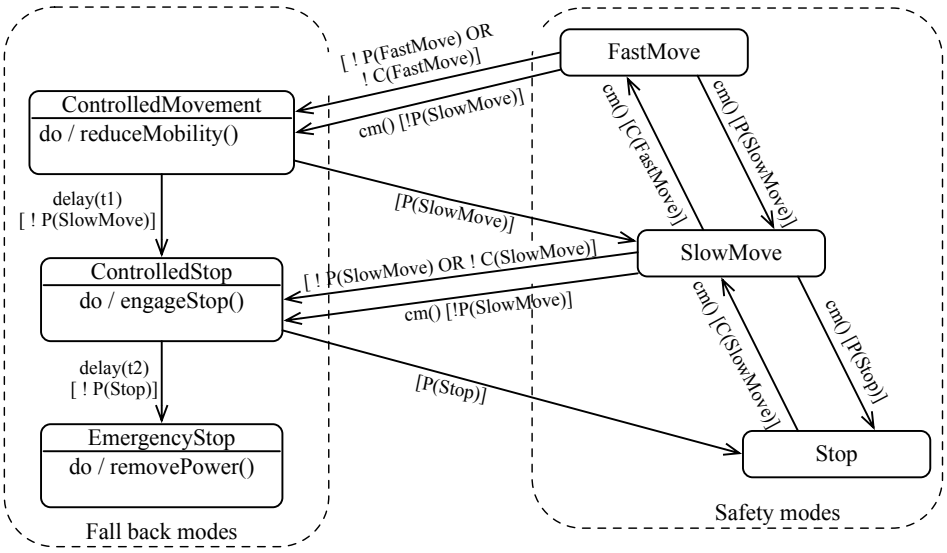

Figure 3: Partial safety mode automaton with fall-back modes. $cm(x)$: abbreviation of request to change mode to "x" ("x" omitted avoid cluttering the figure)

system through one or several fall-back modes until a more restrictive P rule is fulfilled.

4 Discussion

The application of the approach on the case study gives evidence to the applicability of the proposed formalism. The terminology and the notation have been systematically applied. An important point is the difference between functional and context variables that has been proposed. This segregation is fundamental for building a safety mode automaton with a partial order permissiveness relation based both on P and C conditions, and for determining transition conditions. This is a key point in our approach. Determination of safety modes, and functional and context variables, is a process that can be used to supplement risk analysis methods. In our case, we use sequence diagrams, and expert reviews to determine hazards. We are currently studying a systematic approach to link risk analysis to our safety mode approach.

Another point that needs to be further developed is the consistency of safety limits between the safety monitor and the functional system. Indeed, the safety monitor should trigger reactions in case of hazardous situations, but it should let the functional system react first. For instance, when a human appears in in the robot trajectory, the functional system should react, and only if the situation does not change, then the monitor has to react. This can be done defining different limits for human distance, for example, or by using a timeout to trigger mode switching.

Some limitations have been identified during the last step of the approach, which is the definition of the reactions of the safety monitor. First, we need to consider in more detail the perception capabilities of the safety monitor. Indeed, speed monitoring can be done through sensors, but detection that the system is about to open the gripper while that is forbidden cannot be done with a sensor. This means that the safety monitor should should be informed about internal requests of the system, which can decrease independence between the two channels. Second, if reactions from the monitor are performed (for instance, a forced change of control law, or forced stop of all movement), this should be made known to the functional system, to rebuild a plan for instance. Both of these current limitations point to the need for further work on the architecture level [14] and on the protocol between monitored system and safety monitor. This is also related to our proposal for the use of *fall-back modes*, which are transitional modes where actions are performed to impose less permissive conditions. Such modes depend on the observation and reaction means available to the monitor (which are constrained by the architecture and the inter-channel protocol).

Currently, a small number of safety modes has been considered, and simple authorization domains have been proposed. For instance, all the domains of continuous variables (speed, force, etc.) are totally ordered ($[0,0] \subset [0,1] \subset [0,2]$). This implies that it is possible to switch to a more permissive mode without checking any permissiveness condition (if speed is in $[0,1]$, it is also true that speed is in $[0,2]$). Nevertheless, we should consider that, sometimes, domains will not be totally ordered, and have for instance unordered (but overlapping) domains such as $[0,20]$ and $[15,30]$. In order to determine transition conditions, intermediate modes might be added, such as in this case, the interval $[15,20]$. Transition conditions are then more complex and are difficult to determine manually.

Finally, the safety monitor approach implies an extension of the system with additional possibilities to change the system state. New risks can thus be introduced, so particular attention needs to be paid to integrity of the monitor itself. Our study is based on the assumption that the monitor will not fail dangerously, but we also have to prove that there is not any dangerous inconsistency between the monitor and the monitored system.

5 Conclusion

We have presented in this paper a formal framework to facilitate the specification of safety rules used by an independent safety monitor. Our approach is based on the safety mode concept, which associates a specific rule set to each functional behavior of the monitored system. A graph representing the partial order between safety modes according to a permissiveness relation allows the automatic determination of transition conditions. In addition, each safety mode can be associated with a fall back mode aimed at enforcing safe execution conditions.

A case study was carried out on a mobile manipulator robot and furnished very interesting results. This did not reveal any inconsistencies, and confirmed that the formal framework is indeed useful. However, a real implementation has yet to be done. Moreover, some issues presented in Section 4 are still open and are the subject of ongoing work. The main issue to be clarified is the protocol between the safety monitor and the monitored system for observation, reaction and mode synchronization. In the near future, we aim to apply our approach to another robotic system and, in another domain, to an autonomous satellite.

Acknowledgment

This work was partially supported by Astrium Satellites France and by the PHRIENDS Specific Targeted Research Project, funded under

the 6th Framework Programme of the European Community under Contract IST-045359. The authors are solely responsible for its content. It does not represent the opinion of the European Community and the Community is not responsible for any use that might be made of the information contained therein.

References

[1] J. Guiochet, G. Motet, B. Tondu, and C. Baron, “Sécurité des systèmes de la robotique médicale,” *Techniques de l’ingénieur*, vol. SE2, no. Sécurité et gestion des risques, 2007.

[2] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, “Basic concepts and taxonomy of dependable and secure computing,” *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 11–33, 2004.

[3] S. Roderick, B. Roberts, E. Atkins, and D. Akin, “The ranger robotic satellite servicer and its autonomous software-based safety system,” *IEEE Intelligent Systems*, vol. 19, no. 5, pp. 12–19, 2004.

[4] P. Klein, “The safety-bag expert system in the electronic railway interlocking system Elektra,” *Expert Systems with Applications*, vol. 3, pp. 499–506, 1991.

[5] J. 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 (ERTS’04)*, 2004.

[6] K. Wika and J. Knight, “A safety kernel architecture,” University of Virginia - Department of Computer Science, Tech. Rep. CS-94-04, 1994.

[7] J. Fox and S. Das, *Safe and sound - Artificial Intelligence in Hazardous Applications*. AAAI Press - The MIT Press, 2000.

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

[9] S. Daly and S. Orme, “The reliability of the Sizewell ‘B’ reactor protection system,” *Electrical and Control Aspects of the Sizewell B PWR, 1992., International Conference on*, pp. 208–214, 1992.

[10] F. Py and F. Ingrand, “Dependable execution control for autonomous robots,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),Sendai, Japan*, 2004.

[11] D. Berthelier, C. Chicher, M. Narmada, D. Dalemagne, O. Boudillet, C. Veltz, R. Chemel, M. Yu, and G. De Rivals Mazeres, “Automated Transfer Vehicle (ATV) critical software overview,” in *53rd International Astronautical Congress of the International Astronautical Federation (IAF), Houston, TX; USA*, 2002.

[12] D. Essame, J. Arlat, and D. Powell, “Tolérance aux fautes dans les systèmes critiques,” LAAS-CNRS, Tech. Rep. 00151, 2000.

[13] M. Kim, I. Lee, U. Sammapun, J. Shin, and O. Sokolsky, “Monitoring, checking, and steering of real-time systems,” in *2nd International Workshop on Run-time Verification*, 2002.

[14] E. Baudin, J.-P. Blanquart, J. Guiochet, and D. Powell, “Independant safety systems for autonomy,” LAAS-CNRS, Toulouse, France, Tech. Rep. 07710, 2007.

[15] F. Maraninchi and Y. Rémond, “Mode-automata: About modes and states for reactive systems,” *Lecture Notes in Computer Science*, vol. 1381, pp. 185–200, 1998.

[16] D. Henrich and S. Kuhn, “Modeling intuitive behavior for safe human/robot coexistence cooperation,” in *Proceedings of the International Conference on Robotics and Automation*, 2006.

[17] M. Perhinschi, M. Napolitano, G. Campa, B. Seanor, J. Burken, and R. Larson, “Design of safety monitor schemes for a fault tolerant flight control system,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 42, no. 2, pp. 562–571, 2006.

[18] J. Guiochet, G. Motet, C. Baron, and G. Boy, “Toward a human-centered UML for risk analysis - application to a medical robot,” in *Proc. of the 18th IFIP World Computer Congress (WCC), Human Error, Safety and Systems Development (HESSD04)*, C. Johnson and P. Palanque, Eds. Kluwer Academic Publisher, 2004, pp. 177–191.