Safety-critical advanced robots: A survey

Jérémie Guiochet, Mathilde Machin, Hélène Waeselynck

Dependable advanced robots: a survey

Jérémie Guiochet[a,b], Mathilde Machin[a,b], Hélène Waeselynck[a]

[a] *Université Toulouse, LAAS-CNRS, France*
[b] *Université Toulouse, UPS, France*

Abstract

Developing advanced robotics applications is now facing the confidence issue for users, which is a main limitation for their deployment in real life. This confidence could be justified by the use of dependability techniques as it is done in other safety critical applications. However, due to specific robotic properties (such as continuous human-robot physical interaction or non deterministic deliberative layer), many techniques need to be adapted or revised. This paper reviews the main issues and research work in the field of dependable robots, making the link between the dependability and robotics concepts. It also presents main challenges for increasing robot dependability.

Keywords: Dependability, safety, collaborative autonomous robot

1. Introduction

Even if fictional fantasies are still far from real robots, technological improvements make them approaching reality. Besides ethical discussions, how to build such systems is a crucial issue. But if we plan that some of these fantasies come to reality in next decades, another issue can be raised, which is *how can we trust them*? It is already a main challenge in critical applications, from transportation to aeronautics, and it will be obviously a core challenge for robots deployment. A main approach for increasing trust is the use of dependability techniques. Nevertheless, if such systems actually belong to more general classes of systems such as embedded or cyber-physical systems, the *collaborative* and *autonomous* properties, for instance, induce important issues in dependability techniques application.

Email address: jeremie.guiochet@laas.fr (Jérémie Guiochet)

The dependability issue, and more specifically the safety issue, became a major challenge in the research. For instance, several recent research European projects consider safety as the main challenge of human-robot cooperation like [1, 2, 3, 4] or as a key objective in [5, 6, 7, 8]. National projects such as [9] in the UK, [10] in Germany, [11] in the US, and dedicated research teams (e.g., [12] in US) or institutes (e.g., [13] in Japan) also focus on robot dependability. Although many robotic functions may impact safety (for instance gripping issues or collision avoidance), we focus in this review on work which have a direct and explicit link with dependability by considering faults avoidance and treatment.

In Section 2, we introduce elements of new robotics, such as autonomy and collaboration, that are fundamental to carry out analyses of hazards and risks. Then, Section 3 deals with European standards for robot safety. We present major work done for dependability in robotics in Section 4. Section 5 concludes with main challenges in the field of dependable robots.

2. From industrial to advanced robots - Hazards and risks

Among the large diversity of robotics applications and their associated ethical issues [14], safety is not a new concept. It has been studied for years by industry, particularly for manufacturing robots. Nevertheless, the development of advanced robots has to lead us to consider new paradigms inducing new hazards as presented in Table 1.

2.1. Autonomy

One major new paradigm is the development of the deliberative software layer, able to plan and to make decision. Many hierarchical architectures for autonomous robots are then split in three layers (see Figure 1):

Decisional layer : It receives objectives from another system, or an operator and generates some plans according to an abstract representation of the system and its environment. Functions for deliberation (e.g., planning, learning or goal reasoning [16]) are usually based on knowledge specific to the application domain (such as heuristics or an environment model) and an inference mechanism used to solve problems by manipulating this knowledge. Execution time is not guaranteed and outputs/results are not deterministic. The use of heuristics is not guaranteed to be optimal or perfect, but sufficient to find solutions.

	Industrial robotics	Advanced robotics	New hazards examples
Motion	No robot motion in human presence	Simultaneous motion (human and robot)	Bad synchronization between human and robot / Non-human-legible movements
Human-robot closeness	Human is far	Human is close / Physical contact	Collisions, contact forces too high
Human-robot interaction	Teach pendant	Advanced interaction (cognitive)	Mode confusion / communication errors
Robot control	Automatic	Autonomous	Hazardous decisions
Mechanical architecture	Heavy / Stiff / Powerful	Light / Compliant / limited power ("intrinsically safe" [15])	Precision hazards / energy storage due to compliance
Task complexity	Mono-function	Multi-functions	Safety rules not adapted (diverse and evolving rules)
Workspace	Structured	Non-structured (uncertainties)	Adverse situations / uncertainties in perception

Table 1: Core properties of industrial and advanced robotics, and examples of induced hazards

Executive layer : It converts plans sent by the decisional layer, into primitive functions for the functional level.

Fonctional level : It is in charge of feedback control loops coupling sensors to actuators, of perception facilities and trajectory computation.

Each level sends to the highest level the results of task execution (including errors that cannot be managed at the lowest level). Hybrid versions with combined layers or direct communication links between functional and decisional layers also exist, but this simple three-layer description is representative of most of the current hierarchical architectures.

2.2. Collaboration

Removing the protective fences around robots, led to the development of human-robot collaboration, where human and robot share task execution and may interact to synchronize their actions. Such collaboration is based on human robot interactions (HRI), which are based on remote devices (teach pendant, buttons, user interface), cognitive signals (voice, posture), or physical contacts (also called pHRI, physical Human Robot Interaction). These

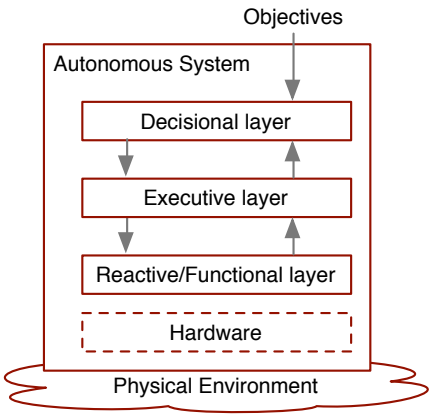

Figure 1: A three layer architecture for autonomy

interactions have an impact on safety, but it is not explicit in this classification. We proposed in the PHRIENDS project [1] to use a more precise classification, called a closeness level (medical robots are not covered by this classification, as they have a similar classification defined as active medical devices in the European Directive [17]):

Far (no pHRI possible): Human and robot are not sharing the same workspace; a direct physical contact is not possible. The interaction with a far robot is usually carried out via remote communication.

Close (accidental pHRI possible): In this case the human and robot are sharing the same workspace. Since the human is within the robots reach there is a risk of unwanted, potentially harmful physical contact.

Touching without simultaneous movement : The robot shares its workspace with the human. Both are simultaneously moving through the workspace, but physical contact with the moving robot is avoided. In this category, pHRI only takes place when the robot stops.

Touching with simultaneous movement : The robot shares its workspace with the human. Both are moving simultaneously and physical interaction is possible and intended. An example of direct pHRI may be a robot which is programmed by being manually guided through the workspace.

Supporting : Here the physical interaction occurs continuously over extended periods of time, usually in the form of exoskeletons which are worn by the user, or when the robot is carrying a human (for example in healthcare applications or rescue operations).

We believe that these six levels should be part of a safety analysis. It is indeed obvious that the consequence of failure of the system will be strongly related to the type of closeness.

2.3. Harms, Risks and Hazards

The first and obvious concern when dealing with robot safety, is the possible harm due to an unwanted collision between a human and a robot. Even if harm is defined in [18] as *a physical injury or damage to the health of people, or damage to property or the environment* (the property includes the robot itself), most work done on harm induced by robots are biomechanical analyses of human robot contact inducing impact, crushing, cutting, etc. and associated control loop or actuators for reducing harm severity (e.g. see [15, 19, 20, 21, 22]). Some results of these researches are part of ISO/TS 15066, which has been analyzed in [23]. Authors note that it is still difficult in this document to validate the forces calculation, as the situations in terms of probability of exposure and complexity of interaction (human moving or not, which direction, etc.) are difficult to describe. Hence, it is nearly impossible to determine an acceptable speed or force, without knowing the robotic application.

Besides these researches specific to robotics, a more generic approach to study safety is based on the concept of *risk* defined as [18] the *combination of the probability of occurrence of harm and the severity of that harm.* In a risk management process, the main activity is actually focusing on identifying the hazards defined as any potential sources of harm.

Main works aiming at identifying hazards present in robotics applications are reviewed hereafter. Even if the study [24] states that crushing and clamping are the major hazards in robot cells, an important challenge is to identify all possible hazards induced by the task and the context. Robots considered as *machines* (according to the European Directive [25]), induce the same hazards as other industrial machines (electric shocks, cut, etc.). However, they also induce more specific hazards that can be found in [26], Annex A.

Autonomous systems face hazards whose sources belong to software, electronic or mechanical parts of the system itself, as well as to human errors, or

to uncertain and stressful environment. Moreover, when autonomy increases, so do the failures of the software part. For instance, [27] presents the implementation of the autonomous museum tour guide RoboX9 and a study of its failures during five months of operation. 96% of failures were caused by the software components (80% due to the non-critical human interaction process, and 16% due to the critical navigation and localization process). Similar conclusions were drawn in [28], which presents a review on faults detected on 17 robots of the Robocup. Failures of the mission goal are considered. Software faults in these systems are more frequent than hardware faults, and belong to operational system, middleware or robot controller (including localization, or planners). In the context of autonomous systems software faults, in [29] we stated that the decisional layer could contain faults both in the inference mechanism and in its knowledge representation

3. Robot safety standards

In Europe, in order to commercialize a machine (including an industrial robot), the only requirement is to get a CE certification following the European Directive on machinery [25]. This is done through a process, from an auto-declaration of the manufacturer to a complete reviewing process by independent regulation bodies. ISO standards (e.g., [30, 31] for machine safety) are highly recommended as they give confidence to the regulatory bodies to deliver certification. The generic standard [32] dedicated to safety-related hardware and software based on the concept of Safety Integrity Level (SIL) might also be applicable. But due to required physical contact between human and mobile parts of the robot, such directives or standards are not entirely applicable.

Recently, standards specific to robotics have been released : ISO 10218:2011 [33, 34] for robots in industrial environment[1] and ISO 13482:2014[26] for personal robots. A dedicated standard for collaborative robots is under development [36]. The standard ISO13482 [26] gives a list of typical safety-related functions: emergency stop, protective stop, limits to workspace, speed control, force control, hazardous collision avoidance. For each function, a Performance Level (PL) is assigned resulting in a set of recommendations listed

[1] In the US, the safety standard [35] is an adaptation of ISO 10218:2011 Parts 1 and 2, providing guidance on the proper use of the safety features embedded into robots, as well as how to safely integrate robots into factories and work areas.

in [30] (for software it is mentioned to refer to Safety Integrity Level, SIL, as defined in IEC61508 [32]). This approach is appropriate when it is possible to clearly identify and separate the safety functions from the main robot controller, and when the safety function can obviously switch the system in a safe state. Nevertheless, if we consider for instance a manipulator with allowed human-robot physical interaction, the safety-related function "hazardous collision avoidance" should be part of the main robot controller. Indeed perception, decision and reaction features are required to make the difference between a required interaction and a collision. Hence, the main robot controller should be assigned to a high integrity level, which is too demanding for manufacturers.

Until now, very few robots have been certified. For instance, the technical documentation of the UR5 from Universal Robots [37] specifies that 15 safety functions have been tested by the TÜV (*Technischer Überwachungs-Verein*) in accordance with the "EN ISO 13849:2008 PL d, and EN ISO 10218-1:2011, Clause 5.4.3". It is important to note that this certificate only validates the presence of a safety function (clause 5.4.3), with PL d (equivalent to the medium level SIL 2 in [32]). This does not guarantee safety in the context of a given task and environment.

Thus, widely accepted methods for certification of robots, and particularly for autonomous robots, lack. In [38, 39, 40], the authors conclude that even if some formal methods can be efficiently applied to autonomous systems, it is not sufficient to build a safety argumentation to obtain certification. Even if important efforts have been done in recent standards towards new robotic systems, certification of collaborative robots with a decisional layer is still an open issue.

4. Dependability means

Dependability is defined in [41] as the "ability to deliver service that can justifiably be trusted". It encompasses many attributes, such as reliability, safety or availability. To avoid service failures that are more frequent and more severe than acceptable, dependability proposes four means:

Fault prevention : to prevent the occurrence or introduction of faults, including techniques coming from system engineering and good practices from system designing (Section 4.1)

Fault removal : to reduce the number and severity of faults mainly using validation and verification techniques (Section 4.2).

Fault forecasting : to estimate the present number, the future incidence, and the likely consequences of faults. It includes risk analysis methods. (Section 4.3)

Fault tolerance : to avoid service failures in the presence of faults using redundancy, error detections, etc. (Section 4.4)

The term "fault" used in the names of those categories actually mean failure, error or fault, which are three different concepts [41]. We will also consider as "fault" any threat to the dependability, like uncertainties in perception, an heuristics limit, or an unexpected adverse situation.

4.1. Fault prevention

In a hierarchical architecture, developers have to deal with heterogeneous models and abstractions. As in other domains, fault prevention in the software of autonomous system is mainly carried out through the modularity of software components and development tools appropriate to heterogeneity. Component-based software and modularity first appear in architectures such as LAAS [42], RAX [43], CLARAty [44] or IDEA [45]. These layered architectures can be supported by middleware like ROS (*Robot Operating System*) [46, 47], OROCOS [48, 49], or Genom [50, 51, 52]. They provide reuse facilities, communication functions, and code generation.

Other environments, providing tools for formal specification and verification, has also been applied in the context of robotics (see ControlShell [53], ORCCAD [54] or SIGNAL [55]), but they were based on specific languages, which are not interfaced with current robotic development tools. Associated to such tools, model-driven engineering can be used to prevent specification or design faults. The Robotic Application development Process (RAP) [56] proposed in the context of the project [6], is motivated by the absence of such methods in autonomous software development.

Besides the previous model-based approach for the software development, other approaches in robotics contribute to the prevention of faults. For instance, reducing robot performances, such as the degree of freedom in the medical field [57], or power in the context of "intrinsically" safe manipulators

[15] [2], also contributes to the prevention of hazardous situation occurrence. An important work is also developed in order to reach softer and more human aware movements of the robots. This is done through the study of compliant actuators [58, 59, 60] or computation of human-legible movements [61].

4.2. Fault removal

Fault removal aims to reveal, diagnose and remove faults in the considered system. Revealing faults requires to verify the system either dynamically (run tests and detect faults through analysis of logs or with a run-time monitor) or statically (static analysis, model checking, theorem proving). As mentioned by [62, 63, 64], the classic issues faced by verification in control systems are exacerbated for autonomous systems, due to an uncertain execution context and system reaction. It is also hard to validate a decisional mechanism, the assessment of which is strongly dependent on the complete architecture.

4.2.1. Dynamic verification

Testing is the most intuitive way to reveal a fault: a test case is provided to system inputs, then its outputs are analyzed to determine whether they are correct, which constitutes the oracle issue. When a complete behavioral model exists, it is used as an oracle: system and oracle outputs are compared. Otherwise, a partial oracle is used, which verifies properties of outputs. Mainly two types of test can be carried out. Conformance testing aims at revealing faults and robustness testing aims at assessing system resistance to stressful environmental conditions. In robotics, it is difficult to completely specify all situations the system is designed for. During navigation testing, defining the boundary between conformity and robustness testing is quite impossible (e.g., when do environment conditions switch from "normal" to "stressful"?).

According to [62], for autonomous systems, "scenario-based testing provides a very limited coverage". Indeed, its role is often limited to debugging rather than thorough validation the system. Especially in the case of research platforms, developers check correct execution of the system for few scenarios. This issue of test coverage for autonomous robots is also discussed as *situation coverage* in [65]. Intensive testing was however carried out on the RAX architecture for the DS1 project [66]: six test benches were implemented and used

[2] See for instance, products such as the LWR LightWeight Robot III commercialized by KUKA, or UR5 from Universal Robots

for 600 tests. The authors underline the relevance of intensive testing, but acknowledge particular difficulties regarding autonomous systems, notably to define suitable test oracles. This oracle issue has been addressed by [67, 68] where a framework has been developed to generate test cases for robustness testing of mobile autonomous systems. It is based on a model of system tasks (represented by UML sequence diagrams) and on an environment model. In [69], an approach based on genetic mutation is proposed to generate cases to test collision avoidance between two drones. Considering that the oracle is based on the estimation of a distance between drones equivalent to collision, the fitness function is easily implemented. [70] also generate test inputs including 2D worlds (map and obstacles), using procedural content generation as it is done in video games.

A major improvement for testing is the development of simulators. Testing in robotics is costly in terms of time, and can be harmful for the system or its environment (when testing safety for instance), and is usually performed with a limited set of environmental conditions. Simulators cope with these issues, by allowing to plug robot controllers into a simulated mechanical and hardware architecture of the robot in a simulated environnement. Currently, few simulators are sufficiently generic to integrate several software controller architecture, able to simulate gravity, frictions, and dynamic environment. We can cite [71, 72] based on the 3D engine Blender [73], or Gazebo [74] (see a comparaison in [75]). Most work using those simulators for robotics aims at testing a function in relatively simple conditions, rather than fault identification or robustness estimation [76]. Nevertheless, we can forecast that such testing campaigns in autonomous and collaborative robots using simulators will increase.

Another research direction in dynamic verification, is the use of runtime verification techniques reviewed in [77, 78]. This technique generates an oracle from properties (mainly temporal properties), which are specified by adding code usually into the controller software. Verification is then performed during operational life of the system. Such an approach, used in cyber-physical systems (e.g., [79], [80]), has been applied by [81] for non regression testing of planning in robotics.

4.2.2. Static verification

Contrary to dynamic verification, static verification guarantees that all executions of a system are correct regarding requirements. Nevertheless they are generally based on a system model, which is an abstraction of the real

system. Static verification encompasses static analysis, theorem proving (see [82] for obstacle avoidance algorithm proving) and model checking, which represent most of the works addressing robotics.

Model checking consists in the verification of properties of execution traces (or a reduced set) of a dynamic model (usually a state machine). Temporal logics, like CTL (Computation Tree Logic), are widely used to define these properties. In computer science, the main drawbacks of these approaches is the error-prone modeling step and the model representativity issue. Tools also suffer from combinatory explosion. Nevertheless, increasing performance of calculators and algorithms should reduce this limitation, and improve model checking applicability in the future.

In robotics, [83] and [84] propose to use model checking with an extension to estimate the probability that the properties are satisfied. To avoid modeling the software, the functional layer is checked directly as Java code in [85]. [86] present an approach to verify the decomposition and synchronization of the controller tasks written in C++, using the model checker NuSMV.

Static verification of the planners is also an important issue in robotics. One way to validate a planning model is to define an oracle as a set of constraints that characterize a correct plan: plans satisfying the constraints are deemed correct. Such a technique was used for thorough testing of the RAX planner during the NASA *Deep Space One* project [66], and is supported by the VAL validation tool [87]. Some works [88, 89] have attempted to validate application-specific models by means of model checking, which usually implies a manual conversion of the model into the syntax accepted by the model checker. This requires an intimate knowledge of the model checker and it is thus usually carried externally by a formal method expert, rather than by the system designer. However, some research has studied how this model transformation can be automated [90]. More generally, [91] show how planning and verification may contribute to each other.

Theoretically linked with model checking, the supervisor synthesis was originally defined by [92] and [93]. Properties to check are combined with a dynamic model of the system in order to synthesize correct-by-design control software while providing formal guarantees of correctness and performance. Such an approach has been used by [94] and [95] in order to guarantee properties like deadlock absence or data freshness. In [96], the synthesis of a robot controller taking into account uncertainties in sensing and actuating is studied (more generally robot controller synthesis is studied in [12]).

4.3. Fault forecasting

Fault forecasting aims at estimating the cause-consequence chain of fault occurrence. It encompasses well-known risk analysis techniques usually classified into two categories :

- *Bottom-up*: a fault effect on the system is estimated in terms of cause-consequence, severity and probability, e.g. FMECA (Failure Modes Effects and Criticality Analysis), HAZOP (Hazard Operability). These methods are based on the use of tables listing deviations (or failure modes), their consequences and possible corrective actions.

- *Top-down*: determination of faults (and their combination) inducing an identified unwanted effect. FTA (Fault Tree Analysis) is used to deduce and represent with a logical tree the combinations of events (like faults) leading to an unwanted top event.

Such methods have been widely used for industrial robots development [97, 98, 99, 100, 101]. However, several challenges appear when applying them to advanced robots:

- Causality analysis is limited due to the complexity and non-determinism of the decisional layer.

- Probabilities of some unwanted events (e.g., software failures, human errors, adverse situations occurrence) are difficult to estimate.

- Hazardous situations may appear in the long term due to a sequence of decisions, instead of a logical combination of events.

- Uncertainty in perception, heuristics and human-robot interactions may induce hazardous behavior, which is difficult to analyze with the current risk analysis techniques usually focusing on fault propagation.

A few studies in robotics consider these issues. In [102], FMEA and FTA are applied to a collaborative robot (not autonomous) focusing on the safety-related functions (emergency stop, etc.) using SIL (Safety Integrity Level) from [32]. The conclusion is that new approaches are needed to analyze human-robot interactions risks. A similar approach is used for medical robots by [103], where risk analysis is slightly adapted without taking account the previous issues. In [104], the system is decomposed into components

and functions, and perform an analysis using HAZOP. In [105], a variant of HAZOP for software, SHARD (Software Hazard Analysis and Resolution in Design) is used, associated with a predefined list of hazardous environmental conditions in the context of mobile robotics. A method called STPA (System Theoretic Process Analysis [106]), which provides guidance to users combining guide words and fault models, is applied to models, based on a process/controller/actuator/sensor representation. It has been used for several safety-critical systems, including robots [107]. Taking into account the importance of the environment, a specific method is developped in [108]. Called ESHA (Environmental Survey Hazard Analysis), it analyses environmental hazardous situations that may occur (due to terrain, obstacles, etc.), without taking account the mission, or the robot tasks. In this paper the authors mentioned that the method HAZOP-UML developed at LAAS [109, 110, 111, 112, 113] is the only safety analysis approach focusing on human-robot interaction. It is based on the hazard identification technique HAZOP, coupled with a system description notation UML (Unified Modeling Language).

Association of several techniques is proposed in [114]. Hazard list templates and ETBA (Energy Trace and Barrier Analysis) are combined. This technique starts from an unwanted release of energy, to infer the causes of this physical event. HAZOP and FFA (Functional Failure Analysis) are used to analyze functions and data flow. Then, a FTA is performed using the results of the previous techniques. Combining all these techniques aims at creating a reasonable approach for autonomous systems analysis, but as mentioned by the authors, further studies are required to improve applicability to autonomous systems. They also suggest in [39] to use the safety case approach and the GSN (Goal Structure Notation) for safety argumentation in autonomous system. This approach has the advantage to integrate in a single argument all evidences in favor of safety, which is particularly interesting when no standards are applicable.

4.4. Fault tolerance

Fault tolerance is rarely explicitly mentioned in literature about autonomous robotic systems, where the concept of *monitoring* is preferred when referring to planning (see [16] for a discussion on the subject). Although some techniques for error detection (such as temporal control by a watchdog, model-based diagnosis monitoring, redundancy and voting) or system recovery (error containment, positioning in a safe state, and hardware and software recon-

figuration) are quite common, we believe that their use is far from systematic, partly because most autonomous systems are still research platforms focusing on autonomous function development rather than dependability. Moreover, fault tolerance increases significantly the cost for the development in terms of physical space or power autonomy, which are all critical for embedded systems, and *a fortiori* for autonomous robots. Several fault tolerance mechanisms are presented in the following sections, according to the layer they are implemented in (see Figure 1).

4.4.1. Functional layer

At the functional level, fault tolerance in robotics has been experimented for actuators, sensors or perception software errors. For instance, [115] propose to develop dedicated monitors for each software component, which is also done in [116]. In these papers, timing or reasonableness checks are performed for hardware and software modules as in embedded systems, but with robotic specific recovery actions impacting the decisional level (for instance, reduce the autonomy level of the robot). Works at this level of architecture may be comparable to the ones in safety-critical embedded systems. Nevertheless, recovery mechanisms at the functional layer and their consequences on the decisional level are a specific issue.

4.4.2. Executive layer

Although [117] do not explicitly mention the three-layer architecture, the faults from environment and sensors are detected and recovered in the layer responsible for action sequencing and execution. In case of error detection, the corresponding function is executed in a fall-back mode. Other functions are chosen to deliver the same task or the level of autonomy is reduced by switching to a tele-operated mode. In this case, the decisional layer is disconnected.

In [118], a layer has been developed (conceptually close to supervisor synthesis) to observe events coming from both decisional layer and functional layer, and to block requests from decisional layer or interrupt execution of functional modules. Inconsistent requests regarding the environment and some errors in functional modules are thus covered. A comparable approach, with completely different technologies is used in [119], where a robot controller is synthesized using the BIP technology (Behavior, Interaction, Priority).

4.4.3. Decisional layer

Detecting plan execution errors is known in robotics as execution monitoring [120, 121, 122]). These works actually do not focus on faults in the planner itself but rather on the planner capacity to cover errors coming from other layers. For instance, in [123], the decision level integrates mechanisms to deal with environment hazards. The planner has a model of reachable states, and it checks if safety properties are respected. It computes a distance between intermediary states and hazardous states. Authors of [124] point out that the decisional layer may also cover faults in the hardware layer. Observations and actuator states are compared to a supposed system state. A belief management system establishes some hypothesis, which are transmitted to the planner.

Very few papers consider faults of the planner itself. In [125], a measure for planner reliability is proposed. Theoretical results are compared to experimental ones, showing a necessary tradeoff between temporal failures (related to tractability of decisional mechanisms) and value failures (related to correctness of decisional mechanisms). Later work [126] addresses this tradeoff through concurrent use of planners with diversified heuristics: a quick but dirty heuristic is used when a slower but more suitable heuristic fails to deliver a plan in time. [127, 128] propose a fault tolerance approach for temporal planners which are a major class of decisional software components. The mechanism covers residual development faults in planning models and heuristics. Recovery from possible errors is achieved using redundant diversified planning models.

4.4.4. Independent safety monitoring layer

A popular form of fault tolerance dedicated to safety is *safety monitoring*, through which the functional system is forced to a safe state (recovery) should some hazardous behavior be detected (error detection) by an external and independent layer. Safety monitors appear in the literature in robotics and decisional systems under many different terms: *safety manager* [129], *autonomous safety system* [130], *checker* [118], *guardian agent* [131], or *emergency layer* [132]. In [133], safety of a museum tour-guide robot is managed through several mechanisms like operating system exception handling, a redundant monitoring software, and a redundant monitoring hardware. In most of these works, the specification of the safety rules is done without any generic method. On the contrary, an approach based on risk analysis is proposed in [105]. In case of uncertainties or when safety rules are not verified,

commands to actuators are filtered, or the robot is stopped. However, the mechanism is not completely independent from the main controller for observation means, and thus its own system state representation can be erroneous due to failures of the main controller.

In [134, 135] a complete framework for the generation of these safety rules taking advantage of the concept of safety margin. It starts from a hazard analysis, and is based on formal verification techniques to automatically synthesize consistent safety rules.

5. Challenges for dependability of autonomous systems

As presented in [136], the roadmap in the USA at 15 years for collaborative robots is to achieve the commercialization of systems that can recognize, work with, and adapt to human or other robot behaviors in an unstructured environment (e.g. construction zones or newly configured manufacturing cells). If we mix this roadmap with the one of autonomous vehicles, we get an objective of a robot that is also capable of moving in any environment in which humans can be. Robots will be able to learn on their own how to move in previously unseen scenarios (e.g., extreme weather, sensor degradation). It is of course implicit that such services should be delivered with a justified level of trust, i.e., with an acceptable level of dependability.

To achieve such objectives, important efforts should be done in several directions. Focusing on dependability, we propose the following ones:

Adaptative safety monitoring Adaptation to extreme conditions, or hazardous situations is of particular interest and an important issue. For instance, while the system accomplishes its missions, safety rules must be checked online, and should also change and be adapted according to the context.

Modeling and simulation for safety analysis This vast field is a key issue in safety analysis in robotics. Model-based safety analysis will allow analysis at the first steps of development and might thus have a great impact on the system design. The development of simulators integrating more accurately physical phenomena, and able to test the robot software, is also important in order to promote and increase testing methods.

Formal methods for verification Verification of robot controllers is a real challenge, as many techniques used in embedded systems are hardly applicable due to the decision layer in autonomous architectures. For instance, verification of planners using formal methods is still an open issue.

Correct-by-construction control and planning Besides verification, we also point out the area of supervisor synthesis, which should lead to more confidence in the software of the controllers. Some works are on progress, but usually focusing on the functional layer, and not on the decisional one.

Perception of hazardous situations Hazardous situation perception can be really complex for autonomous collaborative systems. The integrity of perception mechanisms is still an open issue, for robotics that may evolve indoor to outdoor, or in physical interaction with user.

Compliant mechanisms and actuator Researches on compliant actuators will have an important impact on robot behavior, and particularly on safe reaction strategies in case of failures.

Human-robot interaction models A main issue is the development of usable human-robot interaction models, in order to perform model-based risk analysis.

Certification Due to the fact that such systems behavior and environmental conditions will never be deterministic, applying design standards (and adapting them) will not be sufficient. New tools to build safety argumentations for such systems are needed.

Compared to other safety critical domains, robotics has gigantic open issues for the functions of perception, deliberation and reaction, which draw most of the research work. However, deploying new robotic applications requires the joint development of both robotic functions and adequate dependability techniques, as presented in this paper.

[1] PHRIENDS, Physical Human-Robot Interaction: Dependability and Safety, Project supported by the European Commission under the 6th Framework Programme (STReP IST-045359), `http://www.phriends.eu`, accessed: 2015-04-30 (2006-2009).

[2] SAPHARI, Safe and Autonomous Physical Human-Aware Robot Interaction, Project supported by the European Commission under the 7th Framework Programme, http://www.saphari.eu, accessed 2015-05-17 (2011-2015).

[3] SAFROS, Patient Safety in Robotic Surgery, Project supported by the European Commission under the 7th Framework Programme, http://www.safros.eu/safros/, accessed: 2015-04-30 (2009-2013).

[4] ROBOT-PARTNER, Seamless Human-Robot Cooperation for Intelligent, Flexible and Safe Operations in the Assembly Factories of the Future, Project supported by the European Commission under the 7th Framework Programme, http://www.robo-partner.eu/, accessed: 2015-04-30 (2013-2016).

[5] ROSETTA, RObot control for Skilled ExecuTion of Tasks in natural interaction with humans; based on Autonomy, cumulative knowledge and learning, Project supported by the European Commission under the 7th Framework Programme, http://www.fp7rosetta.org/, accessed: 2015-04-30 (2009-2013).

[6] BRICS, Best of robotics, Project supported by the European Commission under the 7th Framework Programme, http://www.best-of-robotics.org, accessed: 2015-04-30 (2009-2013).

[7] CHRIS, Cooperative Human Robot Interaction Systems, Project supported by the European Commission under the 7th Framework Programme, http://www.chrisfp7.eu/, accessed: 2015-04-30 (2008-2012).

[8] CARLOS, Cooperative mobile robotics, Project supported by the European Commission under the 7th Framework Programme, http://carlosproject.eu/who, accessed: 2015-04-30 (2012-2014).

[9] ROBOSAFE, Trustworthy Robotic Assistants, EPSRC-funded project, UK, http://www.robosafe.org/, accessed: 2015-07-30 (2013).

[10] SIMERO, Safety strategies for human-robot cooperation, Partially funded by the German Research Foundation, http://www.ai3.uni-bayreuth.de/projects/simero/, accessed: 2015-07-30 (2003).

[11] NREC, National Robotic Engineering Center, Carnegie Mellon University, http://www.nrec.ri.cmu.edu/capabilities/safety_ops/, accessed: 2015-07-30 (2015).

[12] Verifiable Robotics Research Group, Sibley School of Mechanical and Aerospace Engineering, Cornell University, http://verifiablerobotics.com/, accessed: 2015-07-30 (2015).

[13] Robot Innovation Research Center, Dependable Systems team , Advanced Institute of Advanced Industrial Science and Technology (AIST), https://unit.aist.go.jp/rirc/en/team/dependable_systems.html, accessed: 2016-01-01 (2016).

[14] L. Royakkers, R. van Est, A literature review on new robotics: Automation from love to war, International Journal of Social Robotics (2015) 1–22.

[15] K. T. Ulrich, T. T. Tuttle, J. P. Donoghue, W. T. Townsend, Intrinsically safer robots, Tech. rep., Barrett Technology Inc. (1995).

[16] F. Ingrand, M. Ghallab, Deliberation for autonomous robots: A survey, Artificial Intelligence.

[17] 93/42/EEC, Council directive of the 14th of june 1993 concerning medical devices, Journal Officiel des Communautés Européennes (JOCE) L169 (1993).

[18] ISO/IEC-Guide51, Safety aspects - Guidelines for their inclusion in standards, International Organization for Standardization (1999).

[19] M. Zinn, O. Khatib, B. Roth, J. Salisbury, Playing it safe [human-friendly robots], Robotics Automation Magazine, IEEE 11 (2) (2004) 12–21.

[20] B. Povse, D. Koritnik, T. Bajd, M. Munih, Correlation between impact-energy density and pain intensity during robot-man collision, in: 2010 3rd IEEE RAS and EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob), 2010, pp. 179–183.

[21] S. Haddadin, Towards Safe Robots, Approaching Asimovs 1st Law, Vol. 90 of Springer Tracts in Advanced Robotics, Springer, 2014.

[22] S. Haddadin, Physical safety in robotics, in: R. Drechsler, U. Kühne (Eds.), Formal Modeling and Verification of Cyber-Physical Systems, Springer Fachmedien Wiesbaden, 2015, pp. 249–271.

[23] HSE, Collision and injury criteria when working with collaborative robots, Tech. rep., Prepared by the Health and Safety Laboratory for the Health and Safety Executive (HSE), UK (2012).

[24] T. Malm, J. Viitaniemi, J. Latokartano, S. Lind, O. Venho-Ahonen, J. Schabel, Safety of interactive robotics : Learning from accidents, International Journal of Social Robotics 2 (3) (2010) 221–227.

[25] 2006/42/EC, Council directive 2006/42/ec on machinery, Official Journal of the European Union (JOCE) L157 (2006).

[26] ISO13482, Robots and robotic devices – safety requirements for personal care robots, International Organization for Standardization (2014).

[27] N. Tomatis, G. Terrien, R. Piguet, D. Burnier, S. Bouabdallah, K. O. Arras, R. Siegwart, Designing a Secure and Robust Mobile Interacting Robot for the Long Term, in: Proceedings of the 2003 IEEE International Conference on Robotics & Automation, Taipei, Taiwan, 2003, pp. 4246–4251.

[28] G. Steinbauer, A survey about faults of robots used in robocup, in: RoboCup 2012: Robot Soccer World Cup XVI, Springer, 2013, pp. 344–355.

[29] B. Lussier, A. Lampe, R. Chatila, J. Guiochet, F. Ingrand, M. O. Killijian, D. Powell, Fault Tolerance in Autonomous Systems: How and How Much?, in: Proceedings of the 4th IARP/IEEE-RAS/EURON Joint Workshop on Technical Challenge for Dependable Robots in Human Environments, Nagoya, Japan, 2005.

[30] ISO13849-1, Safety of machinery – safety-related parts of control systems – part 1: General principles for design, International Organization for Standardization (2006).

[31] ISO12100, Safety of machinery - general principles for design - risk assessment and risk reduction, International Standard Organisation (2010).

[32] IEC61508, Functional safety of electrical/electronic/programmable electronic safety-related systems. dition 2, International Electrotechnical Commission (2010).

[33] ISO10218-1, Robots and robotic devices – safety requirements for industrial robots – part 1: Robots, International Organization for Standardization (2011).

[34] ISO10218-2, Robots and robotic devices – safety requirements for industrial robots – part 2: Robot systems and integration, International Organization for Standardization (2011).

[35] A. R15.06-2012, American national standard for industrial robots and robot systems - safety requirements (revision of ansi/ria r15.06-1999) (2012).

[36] ISOTS15066, Robots and robotic devices – safety requirements for industrial robots – collaborative operation, International Organization for Standardization (2001).

[37] UR5-Robot, Ur5 technical specifications, Tech. rep., Universal robots (2015).

[38] R. Alexander, M. Hallmay, T. Kelly, Certification of Autonomous Systems under UK Military Safety Standards, in: Proceedings of the 25th International System Safety Conference (ISSC'07), Washington DC, USA, 2007.

[39] R. Alexander, N. Herbert, T. Kelly, Structuring safety cases for autonomous systems, in: Proceedings of 3rd IET International System Safety Conference, Birmingham, UK, 2008.

[40] R. Alexander, B. Gorry, T. Kelly, Safety lifecycle activities for autonomous systems development, in: 5th SEAS DTC Technical Conference, 2010.

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

[42] R. Alami, R. Chatila, S. Fleury, M. Ghallab, F. Ingrand, An Architecture for Autonomy, International Journal of Robotics Research 17 (4) (1998) 315–337.

[43] N. Muscettola, P. P. Nayak, B. Pell, B. C. Williams, Remote Agent: To Boldly Go Where No AI System Has Gone Before, Artificial Intelligence 103 (1-2) (1998) 5–47.

[44] R. Volpe, I. Nesnas, T. Estlin, D. Mutz, R. Petras, H. Das, CLARAty: Coupled Layer Architecture for Robotic Autonomy, Tech. Rep. D-19975, NASA - Jet Propulsion Laboratory (2000).

[45] N. Muscettola, G. A. Dorais, C. Fry, R. Levinson, C. Plaunt, IDEA: Planning at the Core of Autonomous Reactive Agents, in: AIPS 2002 Workshop on On-line Planning and Scheduling, Toulouse, France, 2002.

[46] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng, Ros: an open-source robot operating system, International Conference on Robotics and Automation (ICRA), Workshop on open source software 3 (3.2) (2009) 5–11.

[47] ROS, `http://www.ros.org`, accessed July 2015 (2015).

[48] H. Bruyninckx, Open robot control software: the orocos project, in: IEEE International Conference on Robotics and Automation (ICRA), Vol. 3, 2001, pp. 2523–2528 vol.3.

[49] Orocos, `http://www.orocos.org`, accessed July 2015 (2015).

[50] S. Fleury, M. Herrb, R. Chatila, Genom: A tool for the specification and the implementation of operating modules in a distributed robot architecture, in: International Conference on Intelligent Robots and Systems (IROS), Vol. 2, 1997, pp. 842–849.

[51] A. Mallet, C. Pasteur, M. Herrb, S. Lemaignan, F. Ingrand, Genom3: Building middleware-independent robotic components, in: International Conference on Robotics and Automation (ICRA), 2010, pp. 4627–4632.

[52] Genom, https://www.openrobots.org/wiki/genom, accessed July 2015 (2015).

[53] S. A. Schneider, V. W. Chen, G. Pardo-Castellote, H. H. Wang, ControlShell: A Software Architecture for Complex Electromechanical Systems, The International Journal of Robotics Research 17 (4) (1998) 360–380.

[54] J. J. Borrelly, E. Coste-Manire, B. Espiau, K. Kapellos, R. Pissard-Gibollet, D. Simon, N. Turro, The ORCCAD Architecture, The International Journal of Robotics Research 17 (4) (1998) 338–359.

[55] E. Marchand, E. Rutten, H. Marchand, F. Chaumette, Specifying and Verifying Active Vision-Based Robotic Systems with the SIGNAL Environment, The International Journal of Robotics Research 17 (4) (1998) 418–432.

[56] G. Kraetzschmar, A. Shakhimardanov, J. Paulus, N. Hochgeschwender, M. Reckhaus, Specifications of architectures, modules, modularity, and interfaces for the brocre software platform and robot control architecture workbench, Tech. rep., BRICS FP7 project deliverable D-2.2 (2010).

[57] D. Glauser, P. Flury, C. Burckhardt, M. Kassler, Mechanical concept of the neurosurgical robot Minerva, Robotica 11 (6) (1993) 567–575.

[58] R. Filippini, S. Sen, A. Bicchi, Toward soft robots you can depend on, Robotics Automation Magazine, IEEE 15 (3) (2008) 31–41. doi:10.1109/MRA.2008.927696.

[59] A. Albu-Schaffer, O. Eiberger, M. Grebenstein, S. Haddadin, C. Ott, T. Wimbock, S. Wolf, G. Hirzinger, Soft robotics, Robotics Automation Magazine, IEEE 15 (3) (2008) 20–30. doi:10.1109/MRA.2008.927979.

[60] F. Flacco, A. De Luca, I. Sardellitti, N. G. Tsagarakis, On-line estimation of variable stiffness in flexible robot joints, Int. J. Rob. Res. 31 (13) (2012) 1556–1577.

[61] J. Mainprice, E. A. Sisbot, T. Siméon, R. Alami, Planning safe and legible hand-over motions for human-robot interaction, IARP Workshop

on Technical Challenges for Dependable Robots in Human Environments 2 (6) (2010) 7.

[62] C. Pecheur, Verification and validation of autonomy software at nasa, Tech. rep., NASA (2000).

[63] A. Tiwari, P. Sinha, Issues in v&v of autonomous and adaptive systems, in: Canadian Conference on Electrical and Computer Engineering, Vol. 2, 2003, pp. 1339–1342.

[64] T. Menzies, C. Pecheur, Verification and validation and artificial intelligence, Advances in Computers 65 (2005) 153 – 201.

[65] R. Alexander, H. Hawkins, D. Rae, Situation coverage–a coverage criterion for testing autonomous robots, Tech. Rep. YCS-2015-496, University of York, Department of computer science (2015).

[66] D. E. Bernard, E. B. Gamble, N. F. Rouquette, B. Smith, Y. W. Tung, N. Muscettola, G. A. Dorias, B. Kanefsky, J. Kurien, W. Millar, P. Nayal, K. Rajan, W. Taylor, Remote Agent Experiment DS1 Technology Validation Report, Ames Research Center and JPL (2000).

[67] Z. Micskei, Z. Szatmári, J. Oláh, I. Majzik, A concept for testing robustness and safety of the context-aware behaviour of autonomous systems, in: Conference on Agent and Multi-Agent Systems. Technologies and Applications (AMSTA), Springer, 2012, pp. 504–513.

[68] G. Horányi, Z. Micskei, I. Majzik, Scenario-based automated evaluation of test traces of autonomous systems, in: Workshop on Dependable Embedded and Cyber-physical Systems (DECS) in the International Conference on Computer Safety, Reliability and Security (SafeComp), 2013.

[69] X. Zou, R. Alexander, J. McDermid, Safety validation of sense and avoid algorithms using simulation and evolutionary search, in: International Conference on Computer Safety, Reliability, and Security (SafeComp), 2014, pp. 33–48.

[70] J. Arnold, R. Alexander, Testing autonomous robot control software using procedural content generation, in: F. Bitsch, J. the, M. Kaniche (Eds.), Computer Safety, Reliability, and Security, Vol. 8153 of Lecture

Notes in Computer Science, Springer Berlin Heidelberg, 2013, pp. 33–44.

[71] Morse, http://www.openrobots.org/morse, accessed July 2015 (2015).

[72] G. Echeverria, N. Lassabe, A. Degroote, S. Lemaignan, Modular open robots simulation engine: Morse, in: Robotics and Automation (ICRA), 2011 IEEE International Conference on, IEEE, 2011, pp. 46–51.

[73] Blender3D, http://www.blender.org, accessed July 2015 (2015).

[74] Gazebo, http://gazebosim.org/, accessed July 2015 (2015).

[75] D. Cook, A. Vardy, R. Lewis, A survey of auv and robot simulators for multi-vehicle operations, in: IEEE/OES Autonomous Underwater Vehicles (AUV),, 2014, pp. 1–8.

[76] D. Powell, J. Arlat, H. N. Chu, F. Ingrand, M.-O. Killijian, Testing the input timing robustness of real-time control software for autonomous systems, in: European Dependable Computing Conference (EDCC), 2012, pp. 73–83.

[77] M. Leucker, C. Schallhart, A brief account of runtime verification, Journal of Logic and Algebraic Programming 78 (5) (2009) 293–303.

[78] N. Delgado, A. Q. Gates, S. Roach, A taxonomy and catalog of runtime software-fault monitoring tools, Transactions on Software Engineering 30 (12) (2004) 859–872.

[79] A. E. Goodloe, L. Pike, Monitoring distributed real-time systems: A survey and future directions, rapport technique, NASA/CR-2010-216724 (2010).

[80] A. Kane, T. Fuhrman, P. Koopman, Monitor based oracles for cyber-physical system testing: Practical experience report, in: International Conference on Dependable Systems and Networks (DSN), 2014, pp. 148–155.

[81] A. Goldberg, K. Havelund, C. McGann, Runtime verification for autonomous spacecraft software, in: Aerospace Conference, 2005, 2005, pp. 507–516.

[82] H. Täubig, U. Frese, C. Hertzberg, C. Lüth, S. Mohr, E. Vorobev, D. Walter, Guaranteeing functional safety: design for provability and computer-aided verification, Journal Autonomous Robots 32 (3) (2012) 303–331.

[83] S. Pathak, L. Pulina, G. Metta, A. Tacchella, Ensuring safety of policies learned by reinforcement: Reaching objects in the presence of obstacles with the icub, in: International Conference on Intelligent Robots and Systems (IROS), 2013, pp. 170–175.

[84] M. O'Brien, R. C. Arkin, D. Harrington, D. Lyons, S. Jiang, Automatic verification of autonomous robot missions, in: International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR), Springer, 2014, pp. 462–473.

[85] S. Scherer, F. Lerda, E. M. Clarke, Model checking of robotic control systems, in: International Symposium on Artificial Intelligence, Robotics and Automation in Space (SAIRAS), 2005.

[86] R. Simmons, C. Pecheur, G. Srinivasan, Towards automatic verification of autonomous systems, in: International Conference on Intelligent Robots and Systems (IROS), Vol. 2, 2000, pp. 1410–1415.

[87] R. Howey, D. Long, M. Fox, VAL: Automatic Plan Validation, Continuous Effects and Mixed Initiative Planning using PDDL, in: ICTAI, Boca Raton, Florida, 2004.

[88] L. Khatib, N. Muscettola, K. Havelund, Mapping Temporal Planning Constraints into Timed Automata, in: TIME, Cividale del Friuli, Italy, 2001, pp. 21–27.

[89] J. Penix, C. Pecheur, K. Havelund, Using Model Checking to Validate AI Planner Domain Models, in: SEW, Greenbelt, Maryland, 1998.

[90] A. Cesta, A. Finzi, S. Fratini, A. Orlandini, E. Tronci, Validation and verification issues in a timeline-based planning system, The Knowledge Engineering Review 25 (Special Issue 03) (2010) 299–318.

[91] S. Bensalem, K. Havelund, A. Orlandini, Verification and validation meet planning and scheduling, International Journal on Software Tools for Technology Transfer 16 (2014) 1–12.

[92] P. J. Ramadge, W. M. Wonham, Supervisory control of a class of discrete event processes, Society for Industrial and Applied Mathematics (SIAM) journal on control and optimization 25 (1) (1987) 206–230.

[93] W. M. Wonham, Supervisory control of discrete event systems (2005). URL `www.control.utoronto.ca/DES/`

[94] E. Rutten, A framework for using discrete control synthesis in safe robotic programming and teleoperation, in: IEEE International Conference on Robotics and Automation (ICRA2011), Vol. 4, 2001, pp. 4104–4109.

[95] S. Bensalem, L. d. Silva, F. Ingrand, R. Yan, A verifiable and correct-by-construction controller for robot functional levels, Journal of Software Engineering for Robotics 1 (2).

[96] B. Johnson, H. Kress-Gazit, Analyzing and revising synthesized controllers for robots with sensing and actuation errors, I. J. Robotic Res. 34 (2015) 816–832.

[97] B. Dhillon, Robot reliability and safety, Springer-Verlag, 1991.

[98] B. Dhillon, O. Anude, Robot safety and reliability: a review, Microelectronics and Reliability 33 (3) (1993) 413–429.

[99] B. Dhillon, A. Fashandi, Safety and reliability assessment techniques in robotics, Robotica 15 (1997) 701–708.

[100] I. Walker, J. Cavallero, Failure mode analysis for a hazardous waste clean-up manipulator, Reliability Engineering and System Safety 53 (1996) 277–290.

[101] L. Visinsky, J. Cavallero, I. Walker, Robotic fault detection and fault tolerance: A survey, Reliability Engineering and System Safety 46 (1994) 139–158.

[102] L. Suwoong, Y. Yamada, Risk assessment and functional safety analysis to design safety function of a human-cooperative robot, in: M. Inaki (Ed.), Human Machine Interaction - Getting Closer, Intech, 2012.

[103] P. Kazanzides, Safety design for medical robots, in: Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 2009, pp. 7208–7211.

[104] P. Böhm, T. Gruber, A novel HAZOP study approach in the RAMS analysis of a therapeutic robot for disabled children, in: Computer Safety, Reliability, and Security, Springer, 2010, pp. 15–27.

[105] R. Woodman, A. F. Winfield, C. Harper, M. Fraser, Building safer robots: Safety driven control, Internatioanl Journal of Robotics Research 31 (13) (2012) 1603–1626.

[106] N. G. Leveson, Engineering a Safer World, Systems Thinking Applied to Safety, The MIT Press, 2011.

[107] H. Alemzadeh, D. Chen, A. Lewis, Z. Kalbarczyk, R. Iyer, Systems-theoretic safety assessment of robotic telesurgical system, in: 34th International Conference on Computer Safety, Reliability and Security, 2015.

[108] S. Dogramadzi, M. E. Giannaccini, C. Harper, M. Sobhani, R. Woodman, J. Choung, Environmental hazard analysis-a variant of preliminary hazard analysis for autonomous mobile robots, Journal of Intelligent & Robotic Systems 76 (1) (2014) 73–117.

[109] J. Guiochet, D. Martin-Guillerez, D. Powell, Experience with model-based user-centered risk assessment for service robots, in: IEEE International Symposium on High-Assurance Systems Engineering (HASE'2010), IEEE Computer Society, San Jose, CA, USA, 2010, pp. 104–113.

[110] D. Martin-Guillerez, J. Guiochet, D. Powell, C. Zanon, UML-based method for risk analysis of human-robot interaction, in: 2nd International Workshop on Software Engineering for Resilient Systems (SERENE2010), London, UK, ACM, 2010.

[111] Q. A. Do Hoang, J. Guiochet, D. Powell, M. Kaniche, Human-robot interactions: model-based risk analysis and safety case construction, in: Embedded Real Time Software and Systems (ERTS2 2012), Toulouse, France, 2012.

[112] J. Guiochet, Q. A. Do Hoang, M. Kaniche, D. Powell, Model-based safety analysis of human-robot interactions: The MIRAS walking assistance robot, in: Rehabilitation Robotics (ICORR), 2013 IEEE International Conference on, 2013, pp. 1–7.

[113] J. Guiochet, Hazard analysis of human-robot interactions with HAZOP-UML, Safety Science 84 (2016) 225237.

[114] R. Alexander, N. Herbert, T. Kelly, Deriving safety requirements for autonomous systems, in: SEAS DTC Technical Conference, 2009.

[115] D. Crestani, K. Godary-Dejean, L. Lapierre, Enhancing fault tolerance of autonomous mobile robots, in: Journal of Robotics and Autonomous Systems, Elsevier, 2015.

[116] S. Zaman, G. Steinbauer, J. Maurer, P. Lepej, S. Uran, An integrated model-based diagnosis and repair architecture for ROS-based robot systems, in: Robotics and Automation (ICRA), 2013 IEEE International Conference on, 2013, pp. 482–489.

[117] B. Durand, K. Godary-Dejean, L. Lapierre, R. Passama, D. Crestani, Fault tolerance enhancement using autonomy adaptation for autonomous mobile robots, in: International Conference on Control and Fault Tolerant Systems (SysTol), 2010, pp. 24–29.

[118] F. Py, F. Ingrand, Dependable execution control for autonomous robots, in: International Conference on Intelligent Robots and Systems (IROS), 2004, pp. 1136–1141.

[119] S. Bensalem, M. Gallien, F. Ingrand, I. Kahloul, T.-H. Nguyen, Toward a more dependable software architecture for autonomous robots, IEEE Robotics and Automation Magazine 16 (2009) 1–11.

[120] A. Bouguerra, L. Karlsson, A. Saffiotti, Monitoring the execution of robot plans using semantic knowledge, Robotics and Autonomous Systems 56 (11) (2008) 942 – 954.

[121] O. Pettersson, Execution monitoring in robotics: A survey, Robotics and Autonomous Systems 53 (2) (2005) 73 – 88.

[122] J. P. Mendoza, M. Veloso, R. Simmons, Mobile robot fault detection based on redundant information statistics, in: Workshop at IROS'12 on "Safety in human-robot coexistence and interaction: How can standardization and research benefit from each other?", Vilamoura, Portugal, 2012.

[123] P. Ertle, D. Gamrad, H. Voos, D. Soffker, Action planning for autonomous systems with respect to safety aspects, in: IEEE International Conference on Systems Man and Cybernetics (SMC), 2010, pp. 2465–2472.

[124] S. Gspandl, S. Podesser, M. Reip, G. Steinbauer, M. Wolfram, A dependable perception-decision-execution cycle for autonomous robots., in: International Conference on Robotics and Automation (ICRA), 2012, pp. 2992–2998.

[125] I. R. Chen, F. B. Bastani, T. W. Tsao, On the Reliability of AI Planning Software in Real-Time Applications, IEEE Transactions on Knowledge and Data Engineering 7 (1) (1995) 14–25.

[126] I. R. Chen, Effects of Parallel Planning on System Reliability of Real-Time Expert Systems, IEEE Transactions on Reliability 46 (1) (1997) 81–87.

[127] B. Lussier, M. Gallien, J. Guiochet, F. Ingrand, M.-O. Killijian, D. Powell, Planning with diversified models for fault-tolerant robots, in: Proc. of The International Conference on Automated Planning and Scheduling (ICAPS07), Providence, Rhode Island, USA, 2007, pp. 216–223.

[128] B. Lussier, M. Gallien, J. Guiochet, F. Ingrand, M.-O. Killijian, D. Powell, Fault tolerant planning for critical robots, in: 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN07), Edinburgh, UK, 2007.

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

[130] S. Roderick, B. Roberts, E. Atkins, D. Akin, The ranger robotic satellite servicer and its autonomous software-based safety system, IEEE Intelligent Systems 19 (5) (2004) 12–19.

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

[132] S. Haddadin, M. Suppa, S. Fuchs, T. Bodenmller, A. Albu-Schffer, G. Hirzinger, Towards the robotic co-worker, in: C. Pradalier, R. Siegwart, G. Hirzinger (Eds.), The 14th International Symposium on Robotics Research (ISRR2011), Springer Berlin Heidelberg, 2011, pp. 261–282.

[133] N. Tomatis, G. Terrien, R. Piguet, D. Burnier, S. Bouabdallah, K. O. Arras, R. Siegwart, Designing a secure and robust mobile interacting robot for the long term, in: International Conference on Robotics and Automation (ICRA), 2003, pp. 4246–4251.

[134] M. Machin, F. Dufossé, J. Blanquart, J. Guiochet, D. Powell, H. Waeselynck, Specifying safety monitors for autonomous systems using model-checking, in: A. Bondavalli, F. D. Giandomenico (Eds.), The 33rd International Conference on Computer Safety, Reliability and Security (SAFECOMP2014), Springer International Publishing, 2014, pp. 262–277.

[135] M. Machin, F. Dufossé, J. Guiochet, D. Powell, M. Roy, H. Waeselynck, Model-checking and game theory for synthesis of safety rules, in: 16th IEEE International Symposium on High Assurance Systems Engineering, HASE 2015, Daytona Beach, FL, USA, January 8-10, 2015, 2015, pp. 36–43.

[136] Robotics-VO, A roadmap for u.s. robotics 2013 edition, Tech. rep., Robotics Caucus Advisory Committee of the U.S. Congress (2013).