Safety-critical advanced robots: A survey

Article *in* Robotics and Autonomous Systems · April 2017

DOI: 10.1016/j.robot.2017.04.004

3 authors, including:

Jérémie Guiochet
French National Centre for Scientific Research
62 PUBLICATIONS **736** CITATIONS

SEE PROFILE

Safety-critical 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 safety issue for users, the environment, and the robot itself, which is a main limitation for their deployment in real life. This safety 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 decisional layer), many techniques need to be adapted or revised. This paper reviews the main issues, research work and challenges in the field of safety-critical robots, linking up dependability and robotics concepts.

Keywords: Dependability, safety, collaborative autonomous robot

1. Introduction

Even if fictional fantasies are still far from real robots, technological improvements make them approaching reality. Technical development of the functions of 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 *what is our confidence in such systems*? A major contributor for this confidence is the justification of achieved safety. It has already been a main challenge in critical applications, like ground transportation, aeronautics or nuclear applications, the deployment of which has relied on a corpus of dependability means, as defined by [1]. Safety will obviously be a core challenge for robots deployment as well. Nevertheless, even if such systems actually

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

belong to more general classes of systems such as embedded or cyber-physical systems, the *collaborative* and *autonomous* abilities induce important issues in the application of dependability techniques.

Dependability, and more specifically safety, has become a major challenge in robotics research projects. For instance, several recent European projects consider safety as the main challenge of human-robot cooperation like [2, 3, 4, 5] or as a key objective together with maintainability in [6, 7, 8, 9]. National projects such as [10] in the UK, [11] in Germany, [12] in the USA, and dedicated research teams (e.g., [13] in USA) or institutes (e.g., [14] in Japan) also focus on robot reliability and safety. Although many work in the robotics community focus on robot functions linked with safety (e.g. intrinsically safe robot[1] [15], actuators compliance [16, 17, 18], collision avoidance control or human aware motion [19]), we focus in this survey on work which addresses safety of robotic abilities by considering dependability means such as fault avoidance and treatment techniques. By safety, we will not only consider human integrity, but also the environment or the robot itself integrity.

We first introduce in Section 2 the context of our survey, i.e. abilities of considered robotic systems, such as autonomy and interaction, then some examples of induced hazards, and current European robotic safety standards. Then, we present a survey on dependability means for such robots in Section 3. Section 4 provides a selection of main challenges in the field of safe robots and Section 5 concludes this survey.

2. From industrial to advanced robots - New hazards

Among the large diversity of robotics applications and their associated social and ethical issues [20], safety is not a new concept. It has been studied for years in manufacturing applications, particularly for industrial robots. But the emergence of advanced robots with new abilities, such as decisional autonomy and physical interaction with humans, forces consideration of hazards that did not exist in traditional industrial robots. Table 1 presents a comparison between industrial and advanced robotics. The distinguishing characteristics in terms of autonomy and collaboration, as well as the induced new hazards, are detailed in the subsections below. We then discuss the status of safety standards with respect to advanced robots.

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

		Industrial robotics	Advanced robotics	New hazards examples
Autonomy	**Robot control**	Automatic	Decisional autonomy	Hazardous decisions
	Workspace	Structured	Non-structured (uncertainties)	Adverse situations / uncertainties in perception
Collaboration	**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 interaction	Collisions, contact forces too high
	Human-robot communication	Remote device	Advanced interaction (cognitive)	Mode confusion / communication errors
Task	**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)

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

2.1. Autonomy and collaboration

Autonomy is made possible by the introduction of a decisional software layer in the robot architecture. Such a layer exists in service or field applications, and in robotized systems like UAVs (Unmanned Aerial Vehicles), spacecraft, or self-driving cars. These systems are able to act deliberately regarding their mission, in diverse environments (referred as "non structured" workspace in Table 1). It exists a wide range of degrees between what we can call automatic systems (automatic control for industrial robots in Table 1) and fully autonomous systems (see the 11 levels in the European SPARC roadmap [21], 3 in [22], or 5 for vehicles in the USA roadmap [23]). Nevertheless, in this survey, we mainly use a basic automatic/autonomous dichotomy, without intermediate degrees. Cases where the degree of autonomy may impact the applicability of a dependability technique are discussed specifically in Section 3. Technically, these degrees of autonomy may be implemented in a variety of robotic architectures. Figure 1 presents the well-known abstraction into three layers:

Decisional layer : It receives objectives from another system, or an oper-

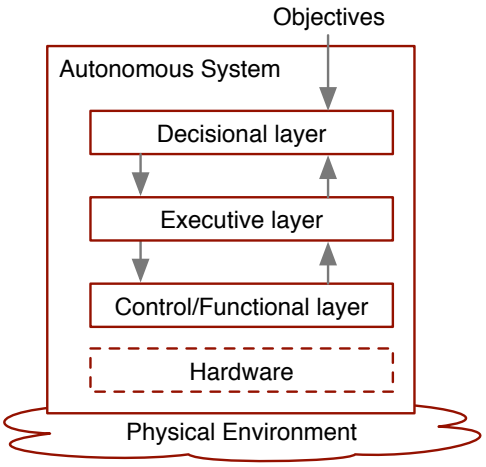

Figure 1: A three layer architecture for decisional autonomy

ator 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 [22]) 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.

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

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

Removing the protective fences around robots, led to the development of human-robot interaction, where human and robot share task execution and may interact to synchronize their actions. As presented in Table 1, such a collaboration is based on human robot closeness (far for industrial robots, and *physical Human Robot Interaction* - pHRI - for advanced robots), on communication means (remote devices or cognitive signals such as voice or posture) and on simultaneous motion of the robot and the human. We proposed in the PHRIENDS project [2] to use an interaction classification [24]

mixing the *closeness* and *motion* properties of Table 1 (for medical robots, defined as active medical devices, such a classification is given in the European Directive 93/42/CEE [25]):

Far : no pHRI possible, human and robot are not sharing the same workspace; a direct physical contact should be not possible.

Close : accidental pHRI possible, 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 : pHRI only takes place when the robot stops, the robot shares its workspace with the human. Both are simultaneously moving through the workspace, but physical contact with the moving robot is avoided.

Touching with simultaneous movement : pHRI possible and intended, the robot shares its workspace with the human. Both are moving simultaneously and physical interaction is possible and intended.

Supporting : continuous pHRI, physical interaction occurs continuously over extended periods of time

It is important to note that this classification adresses human safety, which is the main concern when dealing with collaborative robots. Nevertheless, we also consider in this survey any technique that improves safety for the environment and the robot itself.

Table 2 illustrates a set of possible applications classified according to the autonomy and interaction abilities. Industrial robots, which are classified as "far" from users and automatic, are already well-covered by safety standards. The considered advanced robots in this survey belong to the other classes of this table.

2.2. Examples of new hazards

Even if industrial robots are still a source of accidents (see the fatal accident with an industrial robot on a Volkswagen assembly chain in Germany in July 2015), we focus here on hazards induced by the new properties of advanced robots presented in Table 1, focusing on autonomy and collaboration.

	Interaction levels				
	Far	**Close**	**Touching no move**	**Touching with move**	**Supporting**
Automatic	Industrial robots	Vacuum cleaners	Education robots	Therapeutic aid	Exoskeleton
Decisional autonomy	Exploratory robot	Surveillance robot	Museum guide	Smart co-worker	Self driving vehicles

Table 2: Examples of applications according to the two properties: autonomy and interaction

The first and obvious concern when dealing with robot dependability is about its safety for users due to pHRI. 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, 26, 27, 28, 29]). Some results of these researches are part of ISO/TS 15066, which has been analyzed in [30]. Authors note that it is still difficult 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. Even if the study [31] states that crushing and clamping are the major hazards in robot cells or collision for personal care robot [32], an important challenge is still to identify all possible hazards induced by the task and the context.

A lot of work is still on-going in laboratories to deal with hazards induced by pHRI, but the pragmatic approach for commercialized robotic systems is still today to remove power and stop the robot to avoid any pHRI with simultaneous movement. But this is no longer possible in many new robotics applications (e.g., removing power of a supporting robots may not be safe during a movement).

Regarding the interaction, and the proximity with users, an example of a new hazard is a bad synchronization or communication mishap with robot interface. For instance, the education robot *Little Chubby* injured a person during the China Hi-Tech fair in 2016, due to a human error who activated an unwanted movement. Such an accident becomes possible when the complexity of communication means increases (interpretation of cognitive signals is an important challenge).

Very few studies are focusing on the impact of the presence of a decisional software layer on safety. Up to now, in commercialized robotized applica-

tions, only one fatal accident has been reported due to a bad behavior of the system: the self driving Tesla car accident in May 2016. In that case, it was actually the perception and treatment of an unplanned situation which were responsible of the crash. This accident exemplifies the two major challenges faced by autonomous robots: the adequate perception of the environment in spite of sensing uncertainties, and the adequate reactions to unexpected situations.

Additionally to the intrinsic decisional difficulty, it is also important to mention the residual faults (bugs) in the software as another source of hazard. Indeed, when autonomy increases, so does the software complexity and thus the likelihood that it contains faults.

For instance, [33] 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 [34], 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 operating system, middleware or robot controller (including localization, or planners). It is still difficult to find such studies, while they would be an interesting source of information in order to treat these faults. For instance, [35] argue that the decisional layer could contain faults both in the inference mechanism itself and in its knowledge representation but no study presents types and proportions of such faults.

2.3. Robot safety standards

"Robots have to be safe. But how safe is safe enough?" [36]. In order to solve this issue, most standards rely on a generic approach based on the concept of *risk* defined as the *combination of the probability of occurrence of harm and the severity of that harm* [37]. In a risk management process, a major activity is to identify the *hazards* defined as *any potential sources of harm*, and then to determine the associated risks and their acceptability or not.

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 2006/42/EC [38], which states that a risk management process should have been realized. ISO standards (e.g., ISO13849

[39] and ISO12100 [40] for machine safety) are highly recommended as they give confidence to the regulatory bodies to deliver certification. The generic standard IEC61508 [41] 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. The effort required to attain the highest SILs may also be too important for most current robotic projects.

Nevertheless, the previous integrity approach (functions are ranked according to their failure impact, and development techniques are recommended according to their rank) is widely adopted in many standards (including robotics ones). Recently, robotics standards have been released, e.g., ISO 10218:2011 [42, 43] for robots in industrial environment[2] and ISO 13482:2014[45] for personal robots. Dedicated standards for other domains are under development (see [46] for collaborative robots, or [47] in the agriculture domain). The core idea of these standards is to provide domain-specific safety functions, and associated confidence level that should be guaranteed. For instance, the standard ISO13482 [45] provides 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 in [39] (for software it is mentioned to refer to Safety Integrity Level, SIL, as defined in IEC61508 [41]). 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 might be too demanding for manufacturers.

Hence, until now, very few robots have been ISO certified. For instance, the technical documentation of the UR5 from Universal Robots [48] spec-

[2]In the US, the safety standard ANSI-RIA [44] is an adaptation of ISO 10218:2011 Parts 1 and 2.

ifies 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 IEC61508[41]). This does not guarantee safety in the context of a given task and environment. Moreover, in [49, 50, 51], 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.

A safety culture and widely accepted methods for certification of robots are laking, and it is particularly true for autonomous or collaborative robots in many application domains. For now, many manufacturers of robotic applications only use the European Machine Directive to obtain authorizations for deployment.

3. Dependability means

Dependability is defined in [1] as the "ability to deliver service that can justifiably be trusted". It encompasses many attributes, such as reliability, safety, security, availability or maintainability. Our focus is on safety but it is interesting to refer to the broader conceptual framework established for dependability as a whole. This framework is generic, and not domain-dependent. According to it, the means to avoid service failures that are more frequent and more severe than acceptable can be grouped into four categories:

Fault prevention : to prevent the occurrence or introduction of faults, including techniques coming from system engineering and good practices from system designing (Section 3.1)

Fault removal : to reduce the number and severity of faults mainly using validation and verification techniques (Section 3.2).

Fault forecasting : to estimate the present number, the future incidence, and the likely consequences of faults. It includes risk analysis methods. (Section 3.3)

Fault tolerance : to avoid service failures in the presence of faults using redundancy, error detections, etc. (Section 3.4)

This section provides an overview of the use of these means for robotic systems.

3.1. Fault prevention

Many good practices for software development could be cited in this section. We focus on some major ones, either coming from software engineering in general (component-based frameworks, model-driven engineering) or coming more specifically from the engineering of safety-critical embedded systems (dedicated formal languages, coding rules).

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 appeared in architectures such as LAAS [52], RAX [53], CLARAty [54] or IDEA [55]. These layered architectures can be supported by middleware like ROS (*Robot Operating System*) [56, 57], OROCOS [58, 59], or Genom [60, 61, 62]. 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 [63], ORCCAD [64] or SIGNAL [65]), but they are 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) [66] proposed in the context of the BRICS project [7], is motivated by the absence of such methods in autonomous software development.

For the implementation phase, it is also possible to rely on coding rules used in other safety-critical domains. For instance, the MISRA-C [67], which is a C coding standard originally used in the automotive sector, has been used in [68] for the development of a mobile robot demonstrator. This standard is also used by some major robot manufacturers, but as there is no regulation, each manufacturer may actually define its own coding rules.

3.2. 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 [69, 70, 71], 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 regarding it strong dependence on the complete architecture.

3.2.1. 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. According to [69], 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 [72]. Intensive testing was however carried out on the RAX architecture for the DS1 project [73]: six test benches were implemented and used 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 [74, 75] 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 [76], 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. [77] also generate test inputs including 2D worlds (map and obstacles), using procedural content generation as it is done in video games.

Testing robots in the field 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 (also called software-in-the-loop simulation). Currently, few simulators are sufficiently generic to integrate every software controller architecture, and able to simulate gravity, frictions, and dynamic environment. We can cite MORSE [78, 79] based on the 3D engine Blender [80], or Gazebo [81] (see a comparaison in [82]). Most work using those simulators for robotics aims at testing a function in relatively simple conditions, rather than fault identification or robustness estimation [83]. Nevertheless, we can forecast

that such testing campaigns using simulators will increase as it is done to define world difficulty levels in [84].

Another research direction in dynamic verification is the use of runtime verification techniques reviewed in [85, 86]. 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., [87], [88]), has been applied by [89] for non regression testing of planning in autonomous spacecraft.

3.2.2. 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 and model checking. This latter technique represents most of the work addressing robotics (see nevertheless [68] for obstacle avoidance algorithm proving for an autonomous mobile robot).

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.

In the context of the humanoid robot iCub [90] and a mobile autonomous robot [91], the authors propose to use model checking with an extension to estimate the probability that the properties are satisfied. Also in the context of mobile autonomous robots, [92] present an approach to translate task description of the robot into a formal language, and then 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 [73], and is supported by the VAL validation tool [93]. Some works [94, 95] in artificial intelligence

domain 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 [96]. More generally, [97] show how planning and verification may contribute to each other.

Theoretically linked with model checking, the supervisor synthesis was originally defined in control domain by [98]. 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 applied by [99] for programming a mobile excavation robot, and by [100] for generic mobile robots, in order to guarantee properties like deadlock absence or data freshness. In [101], the synthesis of a robot controller taking into account uncertainties in sensing and actuating has been studied. Another approach, also for autonomous mobile robots, with completely different technologies is used in [102], where a robot controller is synthesized using the BIP technology (Behavior, Interaction, Priority). This framework is composed of a language and a tool set, to support a rigorous design, including verification and code generation.

3.3. 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 a given 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 [103, 104, 105, 106, 107]. 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 [108], 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 IEC61508 [41]. The conclusion is that new approaches are needed to analyze human-robot interactions risks. A similar approach is used for medical robots by [109], where risk analysis is slightly adapted without taking account the previous issues. In [110], the system is decomposed into components and functions, and analyzed using HAZOP for a therapeutic robot. In [111], a variant of HAZOP for software, SHARD (Software Hazard Analysis and Resolution in Design) is used for a mobile robotic platform , associated with a predefined list of hazardous environmental conditions in the context of mobile robotics. A method called STPA (System Theoretic Process Analysis [112]), 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 for a telesurgical application[113]. Taking into account the importance of the environment in mobile robot applications, a specific method is developed in [114]. This method, called ESHA (Environmental Survey Hazard Analysis), analyses the 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 [115] 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).

Even if all these previous approaches are real improvements, they are still non sufficient to address the issues of causality, probability, long term deviation and uncertainties analysis presented in this subsection. Association of several techniques is then proposed in [116] for analyzing autonomous robotic systems, where 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 [50] 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.

3.4. 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 [22] 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 reconfiguration) are quite common, we believe that their use is far from being systematic in robotics. We argue that it is partly because work in robotics is still focusing on robot 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 robots.

In the SPARC roadmoap [21], the proposed "dependability levels" actually do not address dependability as a whole, but defines levels of autonomy of the robot regarding fault tolerance (e.g. how the system is able to autonomously manage, even predict, and recover faults). Even if it is an important challenge, already addressed in other domains like management of large networked systems [117], it is still not much addressed in robotics.

Hence, we choose to present several fault tolerance mechanisms in the following sections, according to the layer they are implemented in (see Figure 1).

3.4.1. Functional/Control layer

At the functional level, fault tolerance in robotics has been experimented for actuators, sensors or perception software errors. For instance, [118] propose to develop dedicated monitors for each software component for mobile robots, which is also done in [119]. 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). In [120], data fusion is used to tolerate perception faults of an autonomous vehicle. This issue is an important challenge in current applications, and will certainly increase while mobile robots mix indoor and outdoor tasks. However, most of the work on perception or localization errors, is not focusing on faults but rather on managing uncertainties, usually using data fusion. As part of fault tolerance, error detection has been studied by [121]. They used neural networks to synthesize error detection components after several nominal runs (no faults) and failed runs (with injected faults). This work however does not explore how the system can recover from the detected errors. 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 an open issue.

3.4.2. Executive layer

Although [122] for autonomous mobile robots 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 [123], 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.

3.4.3. Decisional layer

Detecting plan execution errors is known in robotics as execution monitoring [124, 125, 126]). 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 [127], 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 [128] 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 [129], 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 [130] 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. [131, 132] propose a fault tolerance approach for temporal planners which are a major class of decisional software components. This mechanism covers residual development faults in planning models and heuristics used for a mobile autonomous robot. Recovery from possible errors is achieved using redundant diversified planning models.

3.4.4. 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* [133], *autonomous safety system* [134], *checker* [123], *guardian agent* [135], *emergency layer* [136] or safety monitor [137, 138]. In [139], 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 [111]. 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 [140] 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. However, considering multi-task robots, safety rules might change according to the task and also due to long duration of missions (which might include having to deal with unplanned errors). Hence, a lot of work should be done in the direction of more flexible and adaptive safety rules.

4. Challenges for safety-critical advanced robots

As presented in the 2016 roadmaps from USA [36] or Europe [21], a close objective 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). The objectives in these roadmaps for autonomous vehicles or mobile robots is to develop systems capable of moving in any environment in which humans can be and to 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 confidence, i.e., with an acceptable level of dependability including safety.

To achieve such an objective, important efforts should be done in several directions. Focusing on safety, we extract from the previous sections the following ones:

Modeling and simulation for safety analysis This vast field is a key issue in safety assessment in robotics. For instance, 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.

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

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

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.

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.

5. Conclusion

The tremendous expectations for robotic functionalities should not conceal one major non functional requirement: safety. Even if important work is ongoing in this field, and new standards on robot safety are published,

many techniques coming from dependability research and practice should be transferred and adapted to the robotics community.

As a contribution to develop such approches, this survey has presented a broad overview of work in robot dependability aiming at increasing safety. For this, we kept the classical classification of dependability means into fault prevention, forecasting, removal and tolerance. For each class, we presented the main objectives and recent work in robotics applications. All along this survey we identified open issues, from which we extracted a selection of challenges. We expect these challenges to attract increasing interest in the coming years, in order to allow the safe deployment of robots with advanced autonomy and interaction abilities. Progress in addressing these challenges will also be beneficial to other safety-critical fields, such as aeronautics or transportation, in which the introduction of new autonomy capabilities is also considered.

This survey is of course not exhaustive, and as robots are now parts of fleets, connected and integrated in smart workshops, factories or cities, all issues for safety presented in this survey will just get more complex. It is also expected that other dependability attributes, such as security, come to the foreground.

References

[1] 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.

[2] PHRIENDS, Physical Human-Robot Interaction: Dependability and Safety, Project supported by the European Commission under the 6th Framework Programme (STReP IST-045359), 2006-2009.

[3] SAPHARI, Safe and Autonomous Physical Human-Aware Robot Interaction, Project supported by the European Commission under the 7th Framework Programme, 2011-2015, `http://www.saphari.eu` (Accessed 2017-02-26).

[4] SAFROS, Patient Safety in Robotic Surgery, Project supported by the European Commission under the 7th Framework Programme, 2009-2013, `http://www.safros.eu/safros/` (Accessed: 2017-02-26).

[5] 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, 2013-2016, http://www.robo-partner.eu/ (Accessed: 2017-02-26).

[6] 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, 2009-2013, http://www.fp7rosetta.org/ (Accessed: 2017-02-26).

[7] BRICS, Best of robotics, Project supported by the European Commission under the 7th Framework Programme, 2009-2013, http://www.best-of-robotics.org (Accessed: 2017-02-26).

[8] CHRIS, Cooperative Human Robot Interaction Systems, Project supported by the European Commission under the 7th Framework Programme, 2008-2012, http://www.chrisfp7.eu/ (Accessed: 2017-02-26).

[9] CARLOS, Cooperative mobile robotics, Project supported by the European Commission under the 7th Framework Programme, 2012-2014, http://carlosproject.eu/who (Accessed: 2017-02-26).

[10] ROBOSAFE, Trustworthy Robotic Assistants, EPSRC-funded project, 2013, UK, http://www.robosafe.org/ (Accessed: 2017-02-23).

[11] SIMERO, Safety strategies for human-robot cooperation, Project partially funded by the German Research Foundation, http://www.ai3.uni-bayreuth.de/projects/simero/ (Accessed: 2017-02-26).

[12] NREC, National Robotic Engineering Center, Carnegie Mellon University, http://www.nrec.ri.cmu.edu/capabilities/safety_ops/ (Accessed: 2017-02-26).

[13] Verifiable Robotics Research Group, Sibley School of Mechanical and Aerospace Engineering, Cornell University, http://verifiablerobotics.com/ (Accessed: 2017-02-23).

[14] 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: 2017-02-23).

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

[16] R. Filippini, S. Sen, A. Bicchi, Toward soft robots you can depend on, Robotics Automation Magazine, IEEE 15 (3) (2008) 31–41.

[17] 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.

[18] 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.

[19] J. Mainprice, E. A. Sisbot, T. Siméon, R. Alami, Planning safe and legible hand-over motions for human-robot interaction, in: Seventh IARP Workshop on Technical Challenges for Dependable Robots in Human Environments (DRHE), Toulouse, France, 2010.

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

[21] SPARC, Robotics 2020 multi-annual roadmap for robotics in europe, Horizon 2020 Call ICT-2017 (ICT-25, ICT-27 & ICT-28), Release B 02/12/2016 (2016).

[22] F. Ingrand, M. Ghallab, Deliberation for autonomous robots: A survey, Artificial Intelligence (2014) In press.

[23] SAE, Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles, SAE International, Standard: J3016_201609, Revised: 2016-09-30 (2016).

[24] T. Guhl, R. Bischoff, J. Guiochet, D. Powell, Deliverable D1.1 Report on the analysis of safety functional requirements for safe physical robot human interaction, Tech. rep., PHRIENDS FP6-045359 (Oct 2007).

[25] 93/42/EEC, Council directive of the 14th of june 1993 concerning medical devices, Journal Officiel des Communautés Européennes (JOCE) L169 (1993).

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

[27] 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.

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

[29] 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.

[30] 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).

[31] 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.

[32] T. S. Tadele, T. de Vries, S. Stramigioli, The safety of domestic robotics: A survey of various safety-related publications, IEEE Robotics Automation Magazine 21 (3) (2014) 134–142.

[33] 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: IEEE International Conference on Robotics & Automation, Taipei, Taiwan, 2003, pp. 4246–4251.

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

[35] 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: 4th IARP/IEEE-RAS/EURON Joint Workshop on Technical Challenge for Dependable Robots in Human Environments, Nagoya, Japan, 2005.

[36] Robotics-VO, A roadmap for U.S. robotics from internet to robotics, Tech. rep., Robotics Virtual Organization (2016).

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

[38] 2006/42/EC, Council directive 2006/42/ec on machinery, Official Journal of the European Union (JOCE) L157 (2006).

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

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

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

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

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

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

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

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

[47] ISODIS18497.2, Agricultural machinery and tractors – safety of highly automated agricultural machines – complementary element, International Organization for Standardization (2016).

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

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

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

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

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

[53] 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.

[54] 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).

[55] 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.

[56] 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.

[57] ROS, `http://www.ros.org` (Accessed: 2017/02/26).

[58] 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.

[59] Orocos, `http://www.orocos.org` (Accessed: 2017/02/26).

[60] 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.

[61] 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.

[62] Genom, `https://www.openrobots.org/wiki/genom` (Accessed: 2017/02/26).

[63] 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.

[64] 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.

[65] 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.

[66] 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).

[67] MISRA-C, Guidelines for the use of the c language in critical systems, The Motor Industry Software Reliability Association, UK (2012).

[68] 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.

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

[70] 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.

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

[72] 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).

[73] 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).

[74] 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.

[75] 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.

[76] 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.

[77] 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.

[78] Morse, `http://www.openrobots.org/morse` (Accessed: 2017/02/26).

[79] 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.

[80] Blender3D, `http://www.blender.org` (Accessed: 2017/02/26).

[81] Gazebo, `http://gazebosim.org/` (Accessed: 2017/02/26).

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

[83] 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.

[84] T. Sotiropoulos, J. Guiochet, F. Ingrand, H. Waeselynck, Virtual Worlds for Testing Robot Navigation: a Study on the Difficulty Level, in: 12th European Dependable Computing Conference (EDCC 2016), Göteborg, Sweden, 2016.

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

[86] 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.

[87] A. Goodloe, L. Pike, Monitoring distributed real-time systems: A survey and future directions, Tech. Rep. NASA/CR-2010-216724, NASA Langley Research Center (July 2010).

[88] 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.

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

[90] 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.

[91] 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.

[92] 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.

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

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

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

[96] 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.

[97] 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.

[98] 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.

[99] 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.

[100] 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) (2011) 281–281.

[101] 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.

[102] 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.

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

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

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

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

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

[108] 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.

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

[110] 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.

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

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

[113] 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.

[114] 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.

[115] J. Guiochet, Hazard analysis of human-robot interactions with HAZOP-UML, Safety Science 84 (2016) 225–237.

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

[117] D. Ghosh, R. Sharman, H. R. Rao, S. Upadhyaya, Self-healing systems survey and synthesis, Decision Support Systems 42 (4) (2007) 2164 – 2185, decision Support Systems in Emerging Economies.

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

[119] 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.

[120] K. Bader, B. Lussier, W. Schön, A fault tolerant architecture for data fusion targeting hardware and software faults, in: The 20th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC 2014), Singapore, 2014, p. 10.

[121] A. L. Christensen, R. O'Grady, M. Birattari, M. Dorigo, Fault detection in autonomous robots based on fault injection and learning, Autonomous Robots 24 (1) (2008) 49–67.

[122] 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.

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

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

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

[126] 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.

[127] 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.

[128] 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.

[129] 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.

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

[131] 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.

[132] 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.

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

[134] 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.

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

[136] 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.

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

[138] 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, 2015, pp. 36–43.

[139] 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.

[140] M. Machin, J. Guiochet, H. Waeselynck, J. P. Blanquart, M. Roy, L. Masson, SMOF: A safety monitoring framework for autonomous systems, IEEE Transactions on Systems, Man, and Cybernetics: Systems PP (99) (2016) 1–14.