The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

Global methodology in control architecture to improve mobile robot reliability

B. Durand, K. Godary-Dejean, L. Lapierre and D. Crestani
Laboratoire Informatique Robotique Microélectronique de Montpellier - Université Montpellier 2 - CNRS
{durand*,godary,lapierre,crestani}@lirmm.fr

Abstract— **This paper presents a global methodology developed to increase the reliability of mobile robots. An initial analysis of robot functions and their corresponding significant failures enabled us to introduce dedicated observation modules in the embedded architecture to monitor fault events. Once detected, the functioning mode of the robot can be adapted according to the failure severity level to ensure the success of its mission. The methodology was applied in a case study and the experimental implementation and results are detailed.**

I. INTRODUCTION

Mobile robotic missions are becoming increasingly complex, leading to increased robot complexity. Robots are fitted with numerous powerful sensors which provide accurate information about the robot state and its surrounding environment. They also have various locomotion and interaction capacities thanks to efficient and adapted actuators. The control architecture is the central and critical part of the robot which manages increasingly complex robot activities. In a perfect world, a robot would succeed in completing its allocated missions whatever the situation.

Unfortunately, robots are hampered by numerous types of fault. The interesting study of Carlson *et al.* [1] concerning unmanned ground vehicle operating in real environments demonstrated that robots are often unable to achieve their mission. This study covers a large variety of robots and analyses their missions concerning Urban Search and Rescue and Military Operations. The authors concluded that reliability, which is the capacity to ensure the "continuity of correct service" [2], is low due to a huge variety of failures of many origins. Hence, in the real world robots do not always succeed in dealing with some adverse situations.

To improve reliability, it is essential to design robust (capacity to deliver a suitable service in adverse situations due to uncertain system environments) and fault-tolerant (capacity to deliver a suitable service despite faults affecting system resources) robots [2]. Using these dependable principles, a robot should be able to complete its mission, even when there are faults, or to detect that the mission is no longer possible with its operational capacities and must be abandoned. For simplicity, we will not make a distinction between fault, error and failure in this paper they all refer to a malfunctioning.

Robustness and fault tolerance are based on three main principles [2]: fault or adverse situation detection, diagnosis, and recovery. The next section summarizes a few different studies related to these concepts in mobile robotics.

A. State of the art

The first paragraph in this section is related to fault identification and localization with fault detection and diagnosis mechanisms. The second concerns fault recovery, and finally fault tolerance and robustness in control architectures is tackled.

1) Fault detection and diagnosis methods: Several techniques can be used for fault detection [2]: timing checks (watchdogs), reasonableness checks (valid interval value verification), safety-bag checks (command verification), and model-based monitoring and diagnosis (detection of any inconsistency between measured system data and corresponding model values) [3]. Model-based fault detection is widely used to highlight hardware faults ,[4] where the authors used multiple model based methods (bank of Kalman filters) to detect sensor and mechanical faults. In [5], the authors use a particle filter based state identification method to detect hardware faults such as battery voltage drops or motor encoder decoupling. Other approaches exist for example, in [6], a model based diagnosis approach is described using a probabilistic hybrid automaton to model the considered failure modes and the nominal mode, or in [7], who propose a "generate and test approach" to check all possible sensing failure origins of current symptoms. However, multiple model oriented approaches may be hampered by state space handling problems when the number of treated faults increases, especially in an embedded and real-time context. However in robotics, faults are not always hardware related, they can also be associated with the software. Few studies have assessed the detection of complex robot control software faults during runtime. In [8], the author develops a model based approach using Petri nets to monitor component-based systems and detect erroneous components. But there is no evidence that this approach can handle real-time constraints. Weber and Wotawa [9] describe a model-based diagnosis paradigm to detect and localize runtime control software faults. After a fault detection, the failed component is identified using a failure dependency graph. To the best of our knowledge, this proposition has not been tested in the field.

2) Fault recovery: In robotics, recovery solutions after fault detection have been proposed in some previous studies. For example, concerning hardware faults in [6], the authors determined whether a robot should reconfigure, use a de-

978-1-4244-6676-4/10/$25.00 ©2010 IEEE

graded mode or stop on the basis of qualitative constraints on robot components and diagnostic results. In [7], they use exception handling to recover from a detected failure.

Concerning software faults, the usual approach is to stop and restart either the robot, either malfunctioning components and its dependent services [9].

3) Fault tolerant control architectures: This section presents mechanisms currently used in control architectures to prevent, diagnose and/or recover from faults. Timing checks, reasonableness and safety-bag checks are usually implemented in robotic control architectures. They are usually spread over the architecture and directly embedded in the different control algorithms. In their survey [10], Duan *et al.* present different fault tolerant control architectures. The authors principally focused on hardware fault detection and mainly proposed software redundancy to tolerate faults. In the LAAS architecture , the R2C component [11] is in charge of propriety and assertion tests using safety bag checks. Both LAAS and CIRCA [12] architectures detect adverse situations using execution control and propose high level replanning to tolerate faults. In [13], in the IFREMER control architecture, Nana proposed to use an Intelligent Diagnosis System with a dedicated decisional module to detect and react to faults.

B. Proposal

This short analysis of research concerning fault tolerance and robustness in mobile robotics highlights some limitations. Concerning fault detection and diagnosis, the methods mainly deal with hardware faults. Furthermore, these works generally only focus on a few very specific faults which are not always relevant in real operating situations. They do not use methods to identify relevant faults which are essential to detect. Concerning fault recovery, most fault recovery solutions proposed in the literature are generally very basic (often rebooting or stopping). Moreover, recovery mechanisms often only concern a limited number of faults.

Finally, there is clearly no link between the identification of pertinent faults, fault detection methods and fault recovery works. There is also a lack of global structured approaches to efficiently integrate dependable concepts into the design of the robot control architecture.

In this context, we will propose such an approach to improve the control architecture reliability. Firstly, we use a methodology to identify the most significant faults to detect. Then we describe a structured approach to integrate the most relevant fault detection algorithms in the control architecture. Finally, the decisional capacities of the architecture allow relevant recovery reactions to fault events.

This paper focuses on the fault identification and detection aspects of this approach, without detailing the recovery mechanisms of the architecture. The next section presents the different steps of our methodology. Section III describes the experimental context, and section IV describes the application of our methodology to a case study. Before concluding, section V reports the experiment results.

II. THE PROPOSED METHODOLOGY

A. Fault identification

The FMECA (Failure Mode, Effects and Critical Analysis) approach [14] is used to study potential failures, and to delineate the most critical ones. It begins with a functional decomposition of the system. Then the different failure modes of the functions and their corresponding severities are identified is made. In order to identify faults that induce a service failure, we propose to use a functional decomposition. This step of identification sheds light on the different failures to monitor and also defines their severity according to different robot tasks.

B. Fault detection

For detection of each identified fault, we propose to integrate a dedicated monitoring module named Observer in the control architecture. In an Observer, we use the most adapted existing fault detection or diagnosis method to detect a fault event.

The use of specific and independent modules for fault detection first allows inheritance of the modularity, reusability and upgradeability characteristics of the control architecture. Secondly, a modular approach allows flexible management of these Observers so that the fault detection capacity will be adapted as a function of the robot mission, its environment or its available resources. The Observers monitor the performance of the executive level of the architecture and convey information on failures to the decisional level.

C. Reaction to detected faults

Severity identification with the FMECA method allows the user to determine if a detected fault is critical for the current robot tasks. The reaction to a fault thus takes into account this information, as well as knowledge on the still operational robot functionalities, in order to come up with a solution to manage the encountered problem and pursue the mission with the current robot capacities. The control architecture enables a broad range of reactions. For example, if we have a set of robotic algorithms, they can be tuned to consume fewer resources or switched to other control algorithms if they are out of order. Otherwise, the robot autonomy can be tailored to include the operator's capacities. This can be done requesting information from the human operator or switching to a teleoperated functioning mode

III. EXPERIMENTAL CONTEXT

The experiments were carried out with a Pioneer-3DX from MobileRobots with two reversible DC motors. The robot is equipped with two sonar arrays, bumpers and a camera to perceive the environment. It is controlled by a control architecture: COTAMA, hosted on a laptop under a Linux RTAI real-time operating system. This onboard laptop communicates by a WiFi network with a remote supervisor PC which manages the overall mission of the robot and human-robot interactions. The teleoperation mode uses a UDP protocol.

A. COTAMA control architecture

COTAMA (COntextual TAsk MAnagement) [15] is a modular component oriented architecture. It is split into two main parts, i.e. executive and decisional levels. The executive level involves robotic control. The decisional level adapts the robotic control according to the robot mission progress and its environment.

1) Decisional level: It is divided into two sublevels, i.e. global and local supervisors. The global supervisor is in charge of mission execution. Depending on the mission, the environment and robot state, it defines the objectives that have to be carried out by the local supervisor. The local supervisor manages a given objective. It splits the objective into sub-objectives that are implemented by the scheduler. A sub-objective corresponds to a set of modules that have to be executed to achieve this task.

2) Executive level: It is composed of a scheduler and different dependent modules. Each module implements a robotic algorithm. Modules communicate data according to the consumer/producer paradigm. The scheduler manages the module execution of sub-objective with respect to real-time constraints on module and sub-objective execution.

B. The robot mission

The proposed robot mission is to deliver objects in the laboratory upon the users' request. This mission is carried out in a known environment, and the laboratory map is available. However, the environment remains dynamic since, for example, some humans can interact in the vicinity of the robot. A supervisor PC receives requests and schedules the different delivery tasks. It then interacts with the robot to issue requests and receive reports. For this mission, we define three autonomy level functions of the human-robot interaction class, i.e. autonomous, teleprogrammed and teleoperated. For each autonomy level, several functioning modes can be considered depending on the current operational modules. Typically, we can define less efficient degraded functioning modes to deal with some failures. The delivery mission of the robot is divided into three objectives: driving into the laboratory, and receiving or delivering objects (interactive tasks with users). This paper deals with the first part of the mission, i.e. the driving objective, in order to demonstrate the benefits of the proposed approach.

TABLE I

LIST OF IMPLEMENTED ROBOTIC ALGORITHMS

Robotic tasks	Algorithms
Localization	Monte Carlo Localization Robot's odometry
Path Planning	Lazy PRM
Obstacle avoidance	Deformable Virtual Zone - DVZ Safe Manoeuvering Zone - SMZ
Path following	Path following with actuators velocity saturation

Table I presents a list of robotic control algorithms integrated into the architecture. They define an autonomous level where the robot achieves its driving objective in two main sub-objectives: first, it defines its path with the Lazy PRM path planner [16]; second, it executes a path following sub-objective. After the reception of sensors data, in a non-degraded autonomous mode, a simple Monte Carlo localization method localizes the robot in the map. Then the path following algorithm proposed in [17] guides the robot to follow the path. Two obstacle avoidance algorithms with different approaches are available. The DVZ [16] modifies the control command sent to the robot in an obstacle-free direction. The SMZ [18] algorithm punctually defines another obstacle-free path according to sonar sensors and gives it to the path following algorithm. The definitions of degraded functioning modes derived from this autonomous mode are made switching or removing algorithms. For example, without the Monte-Carlo localization, the robot can only use its odometric sensors which are very noisy. Functioning modes at the teleoperated autonomy level use very few algorithms. The operator has to localize and control the robot according to the camera information.

IV. APPLICATION OF OUR METHODOLOGY IN THE CASE STUDY

In this section, we fully describe the application of our methodology within this experimental context.

A. Fault identification

1) Functional decomposition: The decomposition of our robot mission leads to four main functions: to communicate, drive, take and deliver the object. In autonomous mode, the drive function is divided into six sub-functions: the robot has to create its path, perceive the environment from sensors, localize itself on the given map, follow the path while avoiding obstacles and, of course, to actuate the wheels to move. This functional decomposition guides the preceding mission decomposition into objectives and sub-objectives.

2) Identification of failure severity: The failure severity depends on its frequency but also on the context. Indeed, a fault event could not have the same incidence functions of the available resources at a given instant, the functioning mode and the current robot task. We can distinguish transient failures, intermittent failures and permanent failures. We can then qualify the failure severity on a four-level severity scale:

- **Weak:** the current functioning mode remains operational but can be less effective;
- **Medium:** the current functioning mode is highly perturbed. To keep going on the mission, the robot can remain at the same autonomy level but must skip into a degraded functioning mode.
- **Hard:** it is impossible to pursue the mission in the current functioning mode.
- **Fatal:** it is impossible to pursue the mission at all.

3) Ishikawa, cause-and-effect diagrams: After identification of the failure severity of the different functioning modes, we find the origin of these failures. We adapt Ishikawa's classic diagram [19] using failure cause domains based on the fault taxonomy presented in [1]. We retain the following fault domains: Effectors, Sensors, Environment, Human Design, Power and Control Architecture. Fig. 1 presents the Ishikawa diagram of the Drive function.

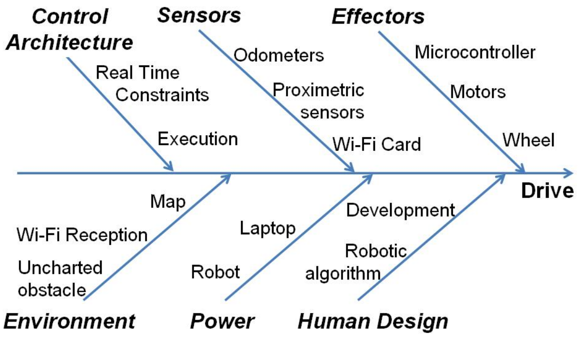


Fig. 1. Ishikawa diagram of the Drive function

B. Fault detection

Fig. 2 shows (continuous line) the controlling loop of the robot in the path following sub-objective in autonomous mode. The lower *Pioneer Communication* module communicates with the robot, forwarding information (sensor data and command values) between the robot microcontroller and the control architecture. Integration of fault detection algorithms implies that the corresponding observation modules have been inserted into the control architecture to monitor the robotic functions of the current functioning mode. Some of the implemented Observers are presented in the figure with dotted lines.

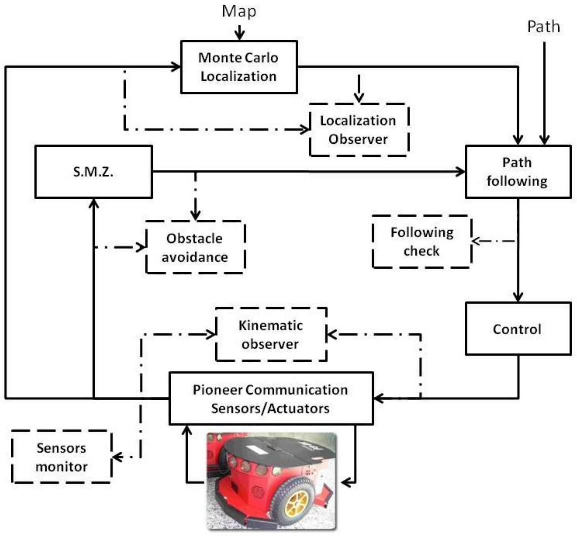


Fig. 2. Path following sub-ojective in autonomous mode

Reasonableness check methods are used in the *Sensor Observer* module, in the *Path following Observer* module and in the *Kinematic Observer* module. The *Localization Observer* checks the consistency between the particle filter results and the sensed odometric data. Finally, the *Obstacle avoidance Observer* checks that the robot avoids collision and does not enter a dead-end loop. A model-based approach (multiple model Kalman filters) is used in the *Kinematic Observer* to detect actuator failures. Fig. 2 only represents the robotic control loop of the path following sub-objective. Other observers are also implemented in the control architecture:

- *Real-time Observer:* monitors the architecture scheduler and verifies the real-time constraints.
- *WiFi communication Observer:* monitors the WiFi connection (quality, bandwith, etc.).
- *Power Observer:* monitors the embedded laptop and the robot power supplies.

C. Reaction to detected faults

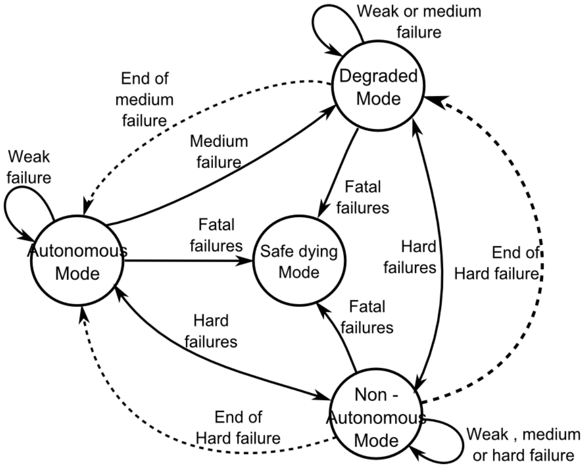


Fig. 3. Failure detection recovery for autonomous mode

All of the observer results are aggregated in a dedicated module named Global Recovery Module (GRM). The GRM module decides the most appropriate reaction according to the severity of the failure, to the current functioning mode, to the available robotic functions and their corresponding modules. The following principles are used, illustrated Fig. 3 for the autonomous mode:

- If the fault severity is weak, a simple reconfiguration of some robotic functions can solve the problem. The current functioning mode remains equally efficient.
- If the severity level is medium, the current functioning level can be maintained but with some degraded robotic functions.
- If the severity level is high, the current functioning mode can no longer be used. A more adapted functioning mode is needed to pursue the mission with the currently available robotic modules, and/or with human-robot interactions.
- If the severity level is fatal, the robot stops its mission properly.

The GRM reacts to fault detection, or to the disappearance of the fault. To manage the functioning mode adaptation, the GRM module generates events to supervisors. These then modify the functioning mode and its corresponding executive modules and/or the autonomy level.

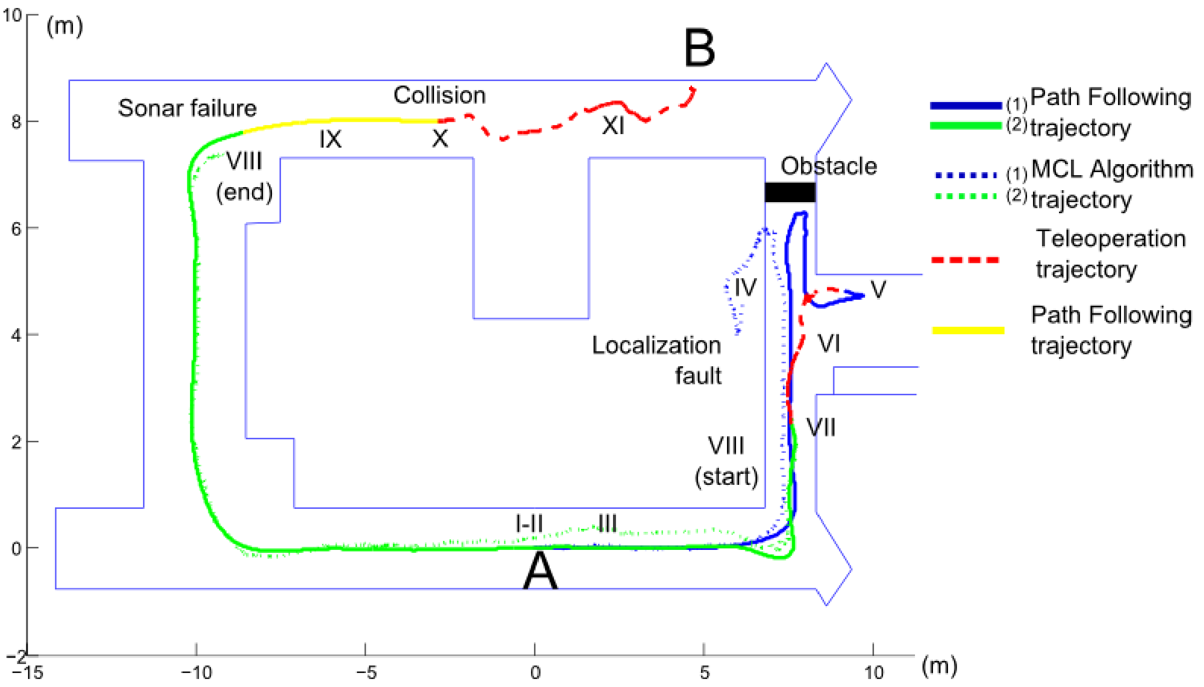


Fig. 4. Experimental mission

V. EXPERIMENTS

In this section, we describe an experimental object delivery mission in the laboratory. It illustrates our methodology in action and highlights the fault detections and the involved reactions. In our setting, some of the observed faults were deliberately created to test the detection of unusual faults (like sonar failure). The considered mission is to deliver object from office A to office B. The object exchange objectives are not described in order to focus principally on analysis of the drive objective from A to B. Fig. 4 presents the recorded experimental robot trajectory and lists the different map points where relevant events were observed.

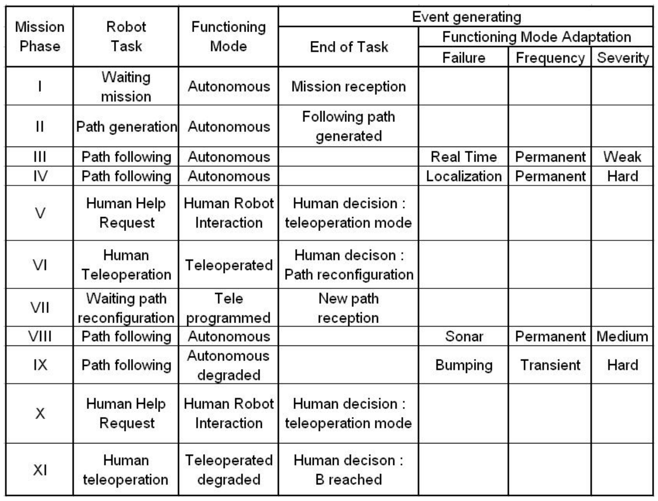
The robot achieved this drive objective in about 2 minutes. The total distance covered is about 50 meters. When moving, the robot speed is 0.3 m/s. The periodic execution of modules involved in sub-objectives is set to 0.1 s. Table II summarizes the different mission phases and details the source event which leads to switching or adapting the robot functioning mode.

- Phases I - II: The mission objective is received and the corresponding path is generated.
- Phase III: A repetitious real-time fault is rapidly observed at the beginning of the path following task. This failure is localized on the particle filter and denotes that the particle quantity is too high. This fault is considered to be weak because it can be eliminated by decreasing the particle number of the algorithm via reconfiguration of the particle filter module.
- Phase IV: During this phase, the path following task works well until a localization fault occurs. Indeed, the robot has encountered an unexpected obstacle. The robot continue avoiding the obstacle (continued blue line) whereas the particle filter (dotted blue line) looses the real localization. This event is considered to be hard in autonomous mode because it reveals that the robot gets lost without an identified reason. The robot thus requests human help and solution.
- Phases V to VII: The human operator decides to observe the robot environment with the on-board camera in teleoperated mode (dashed red line). He detects that an unforeseen obstacle is present and decides to change the following path and to teleprogram it on the robot (teleprogrammed mode).
- Phase VIII: The robot starts to autonomously follow the new path (continued green line). This works until a permanent fault is observed on sonar. This fault is considered to be medium since, to pursue the mission, a degraded autonomous mode can be used without requiring sonar information to localize the robot.
- Phase IX: This degraded mode can potentially be dangerous because obstacle avoidance is unusable. We considered that the mission could be pursued anyway by decreasing the robot speed. The localization is based only on the odometric information (continued yellow line). The limited localization reliability leads the robot to get lost and hit an obstacle. This new failure is hard because there is no longer any possibility of autonomous robot behavior. The robot warns the human operator.
- Phases X - XI: Finally, the human operator decides to end the drive objective using degraded teleoperation (without obstacle avoidance).

This experimental mission shows that the robot was able to detect the different faults that occurred, and to react depending on these faults and actual available resources.

TABLE II

MISSION DESCRIPTION



Mission Phase	Robot Task	Functioning Mode	Event generating			
			End of Task	Functioning Mode Adaptation		
				Failure	Frequency	Severity
I	Waiting mission	Autonomous	Mission reception			
II	Path generation	Autonomous	Following path generated			
III	Path following	Autonomous		Real Time	Permanent	Weak
IV	Path following	Autonomous		Localization	Permanent	Hard
V	Human Help Request	Human Robot Interaction	Human decision : teleoperation mode			
VI	Human Teleoperation	Teleoperated	Human decison : Path reconfiguration			
VII	Waiting path reconfiguration	Tele programmed	New path reception			
VIII	Path following	Autonomous		Sonar	Permanent	Medium
IX	Path following	Autonomous degraded		Bumping	Transient	Hard
X	Human Help Request	Human Robot Interaction	Human decision : teleoperation mode			
XI	Human teleoperation	Teleoperated degraded	Human decison : B reached			

Sometimes the robot opted to ask the human operator for help, and sometimes it solved the problem itself. Finally, the mission was achieved despite fault events.

VI. CONCLUSION

New strategies are now required to make more reliable mobile robots. This paper presents a global methodology to improve robot robustness and fault tolerance. It tries to answer the following questions: Which faults should be focused on? How can they be detected? How should the robot react to these faults in order to pursue the mission?

The proposed methodology merges existing identification and detection techniques with efficient control mechanisms and suitable reactions to construct adaptive control architectures. The FMCEA approach permits identification of the most significant failure sources. They are detected using suitable detection methods integrated into specific observation modules. These modules are included at the executive control architecture level to monitor fault events. Upon fault detection, a dedicated decisional module chooses, considering the current functioning mode, the available robotic functionalities, and the failure severity level, the most suitable functioning mode to pursue the mission. A scripted experiment was presented to illustrate this methodology. The experiment shows relevant fault detection and dedicated reaction. However this experiment has to be repeated numerous times in real environment to analyse success rate of the mission, and relevance of human operator intervention.

Different aspects of the current study could be improved. Firstly, the current decisional mechanism is deterministic. Fuzzy decision making integrating improved use of the robot performance history could enhance the decision making quality. Fuzzy decisional level integrating improved use of the robot performance history could enhance the decisions quality. Secondly, it would be of interest to more thoroughly investigate the problem of human-robot interactions from the human standpoint. Here we just consider that the human operator can solve some robot problems but the robot could be more efficient than the human operator to achieve a given task in some particular contexts. Moreover, it would be interesting to transpose the proposed approach to a robot team that could achieve a given overall mission in spite of some local robot failures.

REFERENCES

[1] J. Carlson and R. Murphy, "How UGVs physically fail in the field ?" *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 423–437, 2005.

[2] B. Lussier, A. Lampe, R. Chatila, J. Guiochet, F. Ingrand, M.-O. Killijian, and D. Powell, "Fault tolerance in autonomous systems: How and how much?" in *proc. of the 4th IARP EURON Joint Workshop on Technical Challenges for Dependable Robots in Human Environments*. Japan: IEEE/RAS, 2005.

[3] M. Hofbaur, J. Köb, G. Steinbauer, and F. Wotawa, "Improving robustness of mobile robots using model-based reasoning," *Journal of Intelligent and Robotic Systems*, vol. 48, no. 1, pp. 37–54, 2007.

[4] P. Goel, G. Dedeoglu, S. I. Roumeliotis, and G. S. Sukhatme, "Fault detection and identifcation in a mobile robot using multiple model estimation and neural network," in *proc. of the Int. Conf. on Robotics and Automation*, vol. 3, 2000, pp. 2302–2309.

[5] C. Valdiviezo and A. Capriano, "Fault detection and isolation system design for omnidirectional soccer-playing robots," in *proc. of the Conf. on Computer Aided Control Systems Design*, 2006, pp. 2641–2646.

[6] M. Brandstötter, M. W. Hofbaur, G. Steinbauer, and F. Wotawa, "Model-based fault diagnosis and reconfiguration of robot drives," in *proc. of the Int. Conf. on Intelligent Robots and Systems*, 2007, pp. 1203–1209.

[7] R. R. Murphy and D. Hershberger, "Handling sensing failures in autonomous mobile robots," *The International Journal of Robotics Research*, vol. 18, no. 4, pp. 382–400, April 1999.

[8] I. Grosclaude, "Model-based monitoring of component-based software systems," in *proc. of the* 15^{th} *Int. Workshop on Principles of Diagnosis*, 2004, pp. 155–160.

[9] J. Weber and F. Wotawa, "Diagnosis and repair of dependent failures in the control system of a mobile autonomous robot," *Applied Intelligence*, vol. 29, pp. 1–18, 2008.

[10] Z. Duan, Z. Cai, and J. Yu, "Fault diagnosis and fault tolerant control for wheeled mobile robots under unknown environments: A survey," in *proc. of the Int. Conf. on Robotics and Automation*, 2005, pp. 3439–3444.

[11] F. Py and F. Ingrand, "Real-time execution control for autonomous systems." in *proc. of the 2nd European Congress ERTS, Embedded Real Time Software*, Toulouse, France, 2004.

[12] R. P. Goldman, D. J. Musliner, and M. J. Pelican, "Using model-checking to plan hard real-time controllers," in *AIPS Workshop on Model-Theoretic Approaches to Planning*, april 2000.

[13] L. T. Nana, "Investigating software dependability mechanisms for robotics applications," *The IPSI BgD Transactions on Internet Research*, vol. 3, no. 1, pp. 50–55, 2007.

[14] *Guide to failure modes, effects and criticality analysis (FMEA and FMECA)*, British Standard Std. 5760-5, 1991.

[15] A. ElJalaoui, D. Andreu, and B. Jouvencel, "Contextual management of tasks and instrumentation within an auv control software architecture," in *proc. of the Int. Conf. on Intelligent Robots and Systems*, 2006.

[16] A. Sánchez, R. Cuautle, R. Zapata, and M. Osorio, *Advances in Artificial Intelligence*. Springer Berlin, October 2006, vol. 4140/2006, ch. A Reactive Lazy PRM Approach for Nonholonomic Motion Planning, pp. 542–551.

[17] L. Lapierre and G. Indiveri, "Non-Singular Path-Following, Control of Wheeled Robots with velocity actuator saturations," in *proc. of the 6th IFAC Symposium on Intelligent Autonomous Vehicles*, Toulouse, France, september 2007.

[18] L. Lapierre, R. Zapata, and M. Bibuli, "Guidance of a flotilla of wheeled robots: a practical solution," in *accepted at the 7th IFAC Symposium on Intelligent Autonomous Vehicles conference*, Lecce Italy, 6-8 September 2010.

[19] K. Ishikawa, *What is Total Quality Control? The japanese Way*. Prentice-Hall, 1985.