Building Safer Robots: Safety Driven Control

Roger Woodman Alan F.T. Winfield Chris Harper
Mike Fraser
Bristol Robotics Laboratory,
University of the West of England,
Bristol, England
http://www.brl.ac.uk

September 26, 2012

Abstract

In recent years there has been a concerted effort to address many of the safety issues associated with physical human-robot interaction (pHRI). However, a number of challenges remain. For personal robots, and those intended to operate in unstructured environments, the problem of safety is compounded. In this paper we argue that traditional system design techniques fail to capture the complexities associated with dynamic environments. We present an overview of our safety-driven control system and its implementation methodology. The methodology builds on traditional functional hazard analysis, with the addition of processes aimed to improve the safety of autonomous personal robots. This will be achieved with the use of a safety system developed during the hazard analysis stage. This safety system, called the Safety Protection System, will initially be used to verify that safety constraints, identified during hazard analysis, have been implemented appropriately. Subsequently it will serve as a high-level safety enforcer, by governing the actions of the robot and preventing the control layer from performing unsafe operations. To demonstrate the effectiveness of the design, a series of experiments have been conducted using a MobileRobots PeopleBot. Finally results are presented demonstrating how faults injected into a controller can be consistently identified and handled by the Safety Protection System.

Keywords: robot safety, hazard analysis, safety protection system, safety-driven control

1 Introduction

Personal robots have long been a desire for those who believe they would make daily tasks easier. Despite the fact that many people would like a robot, compelling need is greatest for those with disabilities or who would otherwise not be able to complete tasks without assistance, and there is increasing evidence that in an ageing population, personal care robots may become essential [1]. Examples of the types of personal robots being considered in this paper are shown in Figure 1.

There are significant barriers preventing robots from being used in people's homes. Arguably the most difficult of these is demonstrating that a robot is acceptably safe for its intended use. Safety implications have always been a concern for robot designers and traditionally the solution has been to prevent the user coming into contact with the robot, by means of physical barriers. For personal robots to become a reality these barriers will need to be removed and more dynamic and flexible safety methods introduced.

Industries which require safety-critical systems have strict processes and standards which must be followed before the system can be put into service. However, as stated by Desantis

Figure 1: Personal robots come in many different forms depending on their use. An interactive mobile robot (left: MobileRobots PeopleBot) may be used as a tour guide, whereas a robot fitted with precision manipulators (centre: Elumotion RT-1) can be used for more complex tasks. Some of the most useful robots are able to explore and manipulate their environment (right: Care-O-bot 3).

et al. [2] and Kulić and Croft [3], there are still no safety standards for complex robots which use Human-Robot Interaction (HRI) to complete tasks. Designers of industrial and commercial robotic systems must consider a wide range of safety risks for their users, the environment and the robot itself. Robotic systems therefore, as with all safety critical systems, require rigorous analysis at all parts of the design to ensure the system is safe. As the first large scale user of robotic systems, the manufacturing industry has developed many of the robotic design methods that are used today. These methods were adapted from design principles and practices from other industrial sectors [4]. Incorporated into the design process were proven techniques such as hazard analysis, failure analysis, rigorous design and extensive inspection and testing. In addition to these, a number of safety standards for industrial robotics have been developed; most notably ISO 10218-1 [5] and ANSI/RIA R15.06 [6]. As discussed by Nokata et al. [7] and Desantis et al. [2], the methods currently employed by robotic designers are not appropriate for designing safe robots operating in unstructured environments. This is due to the high complexity associated with a system that must adapt to changes in its environment and perform actions which cannot always be anticipated during development.

In this paper we present a novel approach for designing robotic systems. Our methodology sets out a procedure for developing a safety system during the hazard analysis stage. This safety system will serve a dual purpose. Firstly, it will be used to verify that safety constraints realised during hazard analysis have been implemented appropriately and that no conflicts are present. Secondly, it will serve as a high-level safety enforcer, by governing the actions of the robot and preventing the control layer from performing unsafe operations. To demonstrate the key safety features of our safety-driven control system a series of experiments have been devised. These experiments have been conducted using a simulated MobileRobots PeopleBot and involve moving items around an environment with and without HRI. Results are presented showing the effectiveness of the system in performing tasks and its ability to handle faults injected into both sensors and controllers.

2 Related Work

Until recently safety analysis and safe control of personal robots was a relatively understudied area. Therefore, little practical data is available in the literature. Some of the best work in the

field has come from the PHRIENDS project (Physical Human-Robot Interaction: Dependability and Safety) [8], which undertook an in-depth study of pHRI. Other notable work has come out of Laboratoire d'Analyse et d'Architecture des Systmes (LAAS), where they have been developing an architecture for autonomous systems [9].

In this section we discuss the main issues associated with designing and developing HRI robots and highlight areas where improvements in safety could have a real impact on getting robots into peoples homes.

2.1 Hazard Analysis

Hazard analysis involves identifying and evaluating potential hazards in a system, which may cause or contribute to an undesirable or harmful event. When a hazard is identified it is analysed to determine what the consequences of the hazard occurring might be. The risk that these consequences pose to both the system and the environment is then established. A safety requirement is subsequently devised with the aim of reducing the risk to an acceptably low level. For safety-critical systems the term often used to describe this is 'as low as reasonably practicable' (ALARP). The safety requirements can specify many types of solution for reducing the risk of the hazard, for example adding extra sensors, adding impact padding, changing how the system behaves and, usually as a last resort, avoiding the hazard by removing the system function altogether.

According to Ericson [10], the major causal factors that can lead to a hazardous event, can be categorised as: (1) hardware, (2) software, (3) humans, (4) interfaces, (5) functions, and (6) the environment. As computers have been used for many decades, hazard analysis methods for hardware, software, interfaces, and functions are well established. Conversely, in robotics identifying hazards associated with the environment and humans which may be present has received little attention. The issues associated with the operating environment of a robot are discussed later in this section.

Functional hazards are directly related to how a system carries out an operation, and requires analysis of the system functions. The system functions define the operations that a system can perform in order to accomplish its objectives [10]. These objectives satisfy the functional requirements of the system as identified from the customer requirements. Conversely, we use the term 'non-functional hazards' to describe everything else, including hazards external to the system such as the users or the environment. A number of hazard analysis techniques exist, many of which evaluate a system using a methodology appropriate for a particular industry. These techniques are generally considered as specialisations of one of the following [11]:

- Failure Modes and Effects Analysis (FMEA)
- HAZard and OPerability studies (HAZOP)
- Event Tree Analysis (ETA)
- Fault Tree Analysis (FTA)

All of these methods use a systematic approach for analysing hazards. The first step in any hazard analysis technique, according to Bahr [12], is to 'understand the physical and functional characteristics of the system under study'. This involves not only looking at the way the system functions, but also the interrelationship of all subsystems and how they may impact the system as a whole. This, as Bahr states, is often a problem area for engineers, who feel they understand how a system works. This can result in an underestimation of how operating conditions and

environment can affect the system. This problem is likely to be much more pronounced for autonomous robots.

Autonomous robots which work cooperatively in close contact with humans have many safety risks not generally associated with industrial robots. Humans can work in ways very similar to each other or can appear to behave unpredictably; in fact it has been observed that people generally operate in patterns similar to other groups of people and rarely operate randomly [13]. It has been shown that humans have a tendency to neglect safety procedures in repetitive tasks [14]. This problem is exacerbated when users don't clearly understand the hazards, or users choose to risk safety for more immediate advantages, such as time saving. Another important issue unique to humans, is how they are able to change the environment in significant ways either intentionally or unintentionally. This may be adjusting lighting, closing doors, adding something to the environment e.g. a toolbox, or interfering with the robot.

A variety of analysis methods have been used for autonomous robots with varying levels of success. ETA and FTA are often used for industrial robots [15, 16] where the environment is structured and HRI is minimal. For HRI situations, Failure Modes, Effects and Criticality Analysis (FMECA) has been shown to capture many of the interactions between the system and the user [17]. Research by Martin-Guillerez et al. [18] applied a modified version of the HAZOP technique to a personal robot. Their findings showed that this technique was more effective than ETA and FTA for identifying hazards associated with users. However, as Böhm and Gruber identify [19] HAZOP and FMECA usually start from a block diagram of the system, which is effective in identifying hazards due to failing components, but lacks the coverage required for identifying hazards associated with completing tasks. Their method was to divide hazard analysis into two parts, 'components view' and 'operations view'. As the names suggest, the components view is concerned with the robot hardware and the operations view with the actions associated with task scenarios. Although their approach identified many of the hazards related to the actual interaction between the human and robot, environmental factors, such as how the robot should interact with other entities that may be present, were not taken into consideration.

In this paper we will be considering autonomous robots that operate in unstructured environments. The next section describes these types of robots and discusses the main safety approaches currently under development.

2.2 Safety of Autonomous Robotic Systems

Autonomous robots are a class of robot system which may have one or more of the following properties: adaptation to changes in the environment; planning for future events; learning new tasks; and making informed decisions without human intervention. Although commercially available autonomous robots are still few, Goodrich and Schultz [20] report that there is increasing demand for both personal robots for the home and service robots for industry.

At present, much of the research into robotic safety is looking at improving safety of specific interactive situations, in particular collision avoidance or failure prevention. Collision avoidance techniques, as the name suggests, aim to prevent robots from coming into contact with surrounding objects. It has been demonstrated that contact avoidance with humans, especially in cooperative situations, requires a higher level of perception compared to other static or dynamic entities [3, 21]. This has led researchers to suggest that safety of human-robot interaction requires both high-precision sensory information and fast reaction times, in order to work with and around humans [22, 23]. Work by Alexander et al. [24] suggests that for autonomous systems to support humans as peers, while maintaining safety, robot actions may need to be restricted, preventing optimum flexibility and performance. In addition to collision avoidance, strategies have been developed to integrate post-contact mitigation into the avoidance scheme.

Work by Ikuta et al. [25] has shown that in robot development, while designing the control systems, it is important to consider safety implications involved with moving external parts of the robot.

Among the requirements of autonomous robots, such as those being discussed in this research, is a certain degree of robustness. This means being able to handle errors and to continue operation during abnormal conditions. To achieve this it is important that the system should be able to support changes to its task specification [26]. These changes are necessary as, in a dynamic environment, the robot will frequently find itself in a wide range of previously unseen situations. To date, the majority of research in this area has addressed this issue by using learning algorithms, often implemented as artificial neural networks (ANNs) [27, 28]. However, as Nehmzow et al. [29] identify, these implementations, although seemingly effective, are difficult to analyse due to the inherent opacity of connection based algorithms. This means that it is difficult to produce an intelligible model of the system structure that could be used in safety analysis. Work by Kurd et al. [30] seeks to address this issue using a hybrid ANN, which is designed to represent knowledge in an interpretable and understandable form.

Research by Bensalem et al. [9] and Lussier et al. [31], has shown that a hierarchical approach to system safety, with different control layers providing planning, task execution and safety supervision, can improve the dependability and reliability of an autonomous robot system. This research is broadly based on the behaviour-based techniques developed by Rodney Brooks. In Brooks' work he introduced a subsumption architecture [32], which demonstrates how different simple behaviours can be combined to produce new complex behaviours. Furthermore, the abstraction of the functional elements of the robot control software allows for modules to be added, removed and amended while retaining control functionality of the robot.

A notable architecture using the three layer approach is the LAAS architecture [9]. This architecture divides the software controlling the robot into three levels: Decisional, Execution and Functional. The distinct feature of the LAAS architecture is the functional level, which encapsulates groups of sensors into modules which can communicate with other modules via a service link. One of the key safety features of this architecture is the execution control level [33]. This level is executed synchronously with the decisional level and is responsible for detecting faults that may occur as a result of a decision being executed. If a potential fault is detected then this decision is prevented from being passed to the functional level.

A report by Alami et al. [8], identifies a number of European robotic manufactures that have recently included software modules to monitor, through external sensing, the space around the robot for any potential dangers. This type of additional monitoring system is known as a 'safety protection system' [34]. A practical example of this, for managing a high-powered laser, has been implemented by Wozniak et al. [35]. Their research found that an architecture which separated safety from control allowed them to more easily configure and extend the safety parameters to meet the requirements of future changes.

3 System Development Methodology

As discussed previously, hazard analysis involves assessing the system requirements, with the aim of identifying potential hazards associated with system operation. Before hazard analysis can take place, the system specification must be produced. This involves first outlining the customer requirements, which can then be used to perform task analysis. These complementary processes result in a document specifying exactly what the system should do and how it will do it. From this document the functional requirements are identified. These requirements relate to the way in which tasks will be performed by the system, and the transformation from system input to system output. Once a requirement specification is available hazard analysis can take

Frequency	Consequence			
	Catastrophic	Critical	Marginal	Negligible
Frequent	I	I	I	II
Probable	I	I	II	III
Occasional	I	II	III	III
Remote	II	III	III	IV
Improbable	III	III	IV	IV
Incredible	IV	IV	IV	IV

Table 1: Risk classification table (based on table B1 of IEC 61508 [37])

place, although as with many development methodologies requirements may be revised at any time.

Our research seeks to bring the development of a Safety Protection System into the hazard analysis process. This, we argue, will allow verification that the safety schemes identified during hazard analysis have been implemented appropriately. This sentiment is supported by the work of Swarup and Ramaiah [36], who state that the most effective way to ensure a system will operate safely, is to build safety in from the start. An overview of our development methodology is shown in Figure 2. As can be seen, the methodology starts by analysing the customer requirements in order to determine the individual tasks that the robot is required to perform. These are subsequently decomposed to identify all subtasks. The method used for this process is hierarchical task analysis, which is discussed in detail in Section 6.1. After this process, as with traditional hazard analysis, the next stage is to identify any potential hazards in the system functionality which could lead to an unsafe event. Each hazard is assessed and the potential consequences of the hazard occurring are identified. Based on the severity of these consequences and the probability of the hazard occurring, a risk value is produced. Finally, a safety requirement is defined, which specifies how the hazard can be either avoided or the risk reduced to an acceptable level. If this is not possible then it may be necessary to re-design part of the system in order to avoid the hazard entirely.

To assess the risk of each hazard consequence identified during hazard analysis, we use a qualitative risk classification matrix based on the example presented in the safety standard IEC 61508 'Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems' [37]. This classification matrix is presented in Table 1: the matrix is organised into a series of columns, which define the hazard consequence severity, and rows denoting the frequency that the hazard could occur. Where a row and column intersect determines the risk class. Four risk classes are possible, from the most severe (Class I) to the least (Class IV). These classes and a description of each are shown in Table 2.

The process of assigning risks classes involves examining each hazard identified during hazard analysis and determining both the frequency with which the hazard is likely to occur and the severity of the consequences associated with the hazard. If the risk class is found to be either Class I (intolerable) or Class II (undesirable), then a safety requirement must be produced, which details how the risk will be reduced to Class IV (negligible) or Class III (tolerable). In Section 3.3 we discuss how these qualitative findings can be interpreted into quantitative values for use in safety functions.

Presently the risk classes are being identified using judgement and experience. This part of the process requires further research, as the difficulty in identifying the probability that a

Class I	Intolerable risk
Class II	Undesirable risk, and tolerable only if risk reduction is impracticable or if the costs are grossly disproportionate to the improvement gained
Class III	Tolerable risk if the cost of risk reduction would exceed the improvement gained
Class IV	Negligible risk

Table 2: Risk classes (see IEC 61508 [37])

specific hazard will occur rapidly increases with the complexity of the robot's environment.

Creating safety requirements involves designers utilising their own experience, adapting solutions from previous projects and following safety standards. To make specifying safety requirements less complicated and easier to verify, we have come up with a standard way of describing them (see Section 3.2). This we believe is a necessary first step for managing the large number of hazards associated with autonomous robots.

The novelty in our approach is how requirements are written, how they are verified and finally how they are used as the backbone of the final system. The remainder of this paper details the design decisions involved in the development of the Safety Protection System and the strategy used to integrate it into the hazard analysis process.

3.1 Identifying Safety Requirements

To reduce the complexity of system interaction with the environment, designers often opt to alter the environment over making changes to the system. This strategy reduces the diversity of potential interactions between a system and its environment by restricting the number and density of features in the environment with which the system can interact. However, as personal robots will be operating in unstructured domains, it will be neither practicable nor desirable to alter them in order to simplify the functionality of the robot, and hence robots will be required to cope with the full complexity of dynamic interactions in the performance of its tasks. The impact of this is that designers will need new methods of identifying all the hazards associated with the robot system and its environment. Here we propose a simple method to aid in the identification of safety requirements for personal robots.

To perform hazard analysis we chose to use elements of a variant of the HAZOP method called SHARD (Software Hazard Analysis and Resolution in Design) [38]; along with a hazard check list that we have devised (Extension 1). The SHARD process, as with HAZOP, involves systematically analysing each system function using a set of key words to guide the process. SHARD is a variant of HAZOP intended for use with software-based systems, and employs keywords that are better suited to identification of software failure behaviour [39]. An example of a hazard analysis document that we produced during this research is given in Extension 2. This document examines the hazards associated with an autonomous robot that is tasked with picking up an object from one location and depositing it in another location. For each hazard consequence that is deemed a risk, one or more safety requirements are proposed. In this document the hazards that are identified as a result of using the hazard check list are highlighted. This is discussed in more detail in Section 6.

The hazard check list is composed of a number of points that robot designers should consider when making design decisions. In particular, this involves consideration of non-mission tasks, which can be thought of as any environmental interaction outside the intended operation of the robot. However, many of the points are applicable to the whole robot life-cycle. Using these two

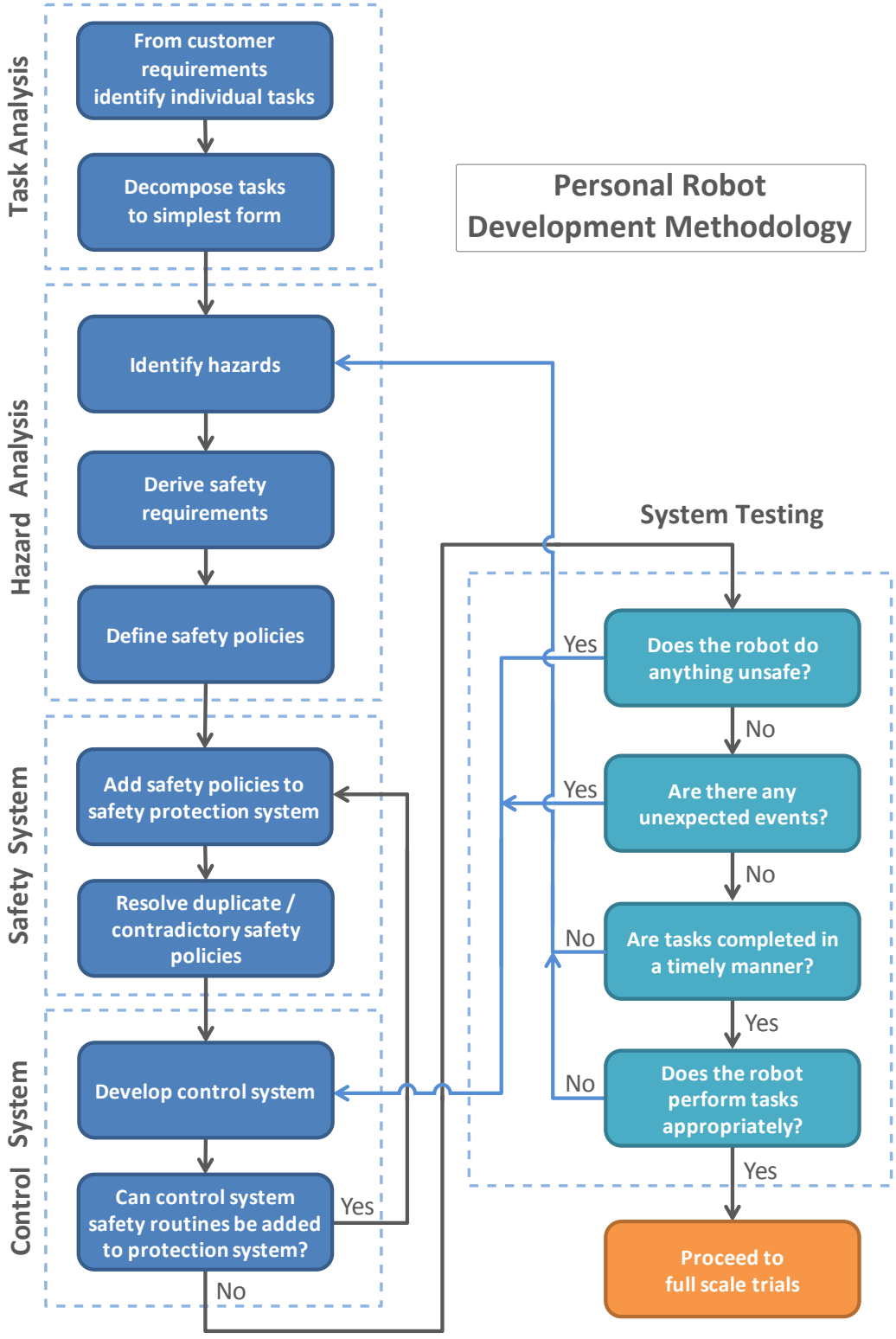

Figure 2: Development methodology for implementing a safety-driven control system.

techniques together, we believe we are able to overcome the problems discussed in Section 2.1.

An extract of the hazard check list is shown in Table 3. It was conceived after careful examination of a number of typical environments that a personal robot could be required to operate in. A combination of experience and brainstorming was used to analyse how a robot may carry out tasks and what hazards it may encounter. The environments considered were: a manufacturing shop floor, a restaurant and a home. These environments are in no way exhaustive and it is likely the check list will benefit from additional examples added over time. Although a robot was not used to perform tasks in these environments, we did consider the dimensions and capabilities of the Care-O-bot 3 mobile robot [40]. This robot was chosen over others such as the WillowGarage PR2 [41], as it is narrower and its manipulator can be stowed away, allowing it to navigate more cluttered environments with less structuring required. An example robot was beneficial, as it helped us understand how a robot would operate within space limitations and in the presence of typical hazards.

Element	**Remark**
Environmental Boundaries: *Floor*	
Movable (grass, soil, gravel, rug)	Wheel slippage
Transparent / translucent (glass walk ways)	May cause issues with identification
Varying heights (major, minor)	May make robot unstable
Environmental Boundaries: *Working Area*	
Doorways (width, steps, change floor surface)	Identify, traverse, drive over
Hot equipment (open fires, boilers)	Recognition, avoidance if necessary
Obstacles [flexible] (curtains, table cloth)	Identify, traverse, drive through
Users: *General*	
Vandalism	Break sensors / safety devices
Children (grabbing, climbing)	May not understand safety instructions
Deaf (fully, partially)	May not hear voice instructions

Table 3: Extract of the hazard check list (see also Extension 1).

As discussed we believe there are currently no hazard analysis methods available that guarantee the identification of all hazards associated with a personal robot. Although the approach we have outlined does not completely solve this problem, it can be shown to capture more potential safety risks than using traditional methods alone. For example none of the systematic approaches we have investigated would identify hazards associated with a human user having wet clothes due to rain. Research by Martin-Guillerez et al. [18] has sought to address this and other issues by using modelling techniques such as UML 'use case' diagrams to visualise the interactions that can occur between the robot and humans. However, it is unclear what benefits these diagrams have on identifying hazards associated with the robot, humans and the environment.

The next section explores our method of encoding safety requirements and details the advantages of the approach. Actual results of safety requirements and safety policies identified for a real robot task are discussed in Section 6.2.

3.2 Implementing Safety Requirements: Safety Policies

In the context of this research paper, a safety policy can be thought of as an interlock implemented in software. These software interlocks, or safety policies, aim to prevent the robot from generating unsafe actions, by means of intervention between the control layer and the actuators.

The initial purpose of developing the safety policies was to create a standard approach for describing and implementing different types of safety requirements. The only restriction being that the safety requirement must be implementable in software. As this required a flexible implementation method, it transpired that not only safety requirements, but a wide range of safety processes could be defined using the same method. The remainder of this section gives details of the implementation method and examines a number of examples.

We have chosen to present safety policies as independent rules, which use facts derived from perception data to impose restrictions on a set of actuators. This idea is based on principles taken from knowledge-based system design [42]. The benefits of this type of design are the inherent parallelism, which treats all rules in the system as separate tasks, all of which are processed simultaneously.

The structured English statement which follows reveals the generic structure for our safety policies. The safety policy object (SP) contains a number of variables, which are compared against to determine whether the associated actuators should be restricted or allowed to operate normally. Sensor functions (SF) provide high-level information about sensor readings, as well as a confidence level that quantifies how confident the sensor function is that its output value is correct. For example, the object distance sensor function could output a value of 300 mm with a confidence level of 0.9, meaning that it is 90% confident that the nearest object is 300 mm away. The method used to calculate the required confidence level for each safety policy is described in detail in the following section.

```
IF    robot_state = SP.required_robot_state
AND safety_state = SP.required_safety_state
AND  comparison of SF.value is inside SP.acceptable_bounds
AND  SF.confidence_level >= SP.required_confidence_level
THEN allow actuators
ELSE restrict actuators
FINALLY return safety_rating based on SF.confidence_level
```

It has been argued that context awareness in a safety-critical system is a crucial way of maintaining both the availability of the system to complete tasks and its safety [23, 43]. This has been implemented in our rule based approach with the use of a 'robot state' object that holds all state information about the robot, which may be of use when making safety decisions. A similar state object called the 'safety state' is used exclusively by the safety policies. This state is employed as a form of memory within the collection of safety policies. An example use is for one policy to set a safety state when a human has been detected. This can then be used in other safety policies to identify a human is in the robot's area, even if it cannot be detected all the time. A timer can be used to reset the safety state if a human is not detected within a certain time frame.

An example safety policy is shown below. This policy is based on a requirement from the robot standards guide ISO 10218-1 [5]. The requirement states that while operating at reduced speed, the speed of the robot is limited to 250 mm/s.

```
IF    robot_state.speed_mode = reduced_speed_control
AND   end_effector_speed <= 250 mm/s
AND   confidence_level >= 0.6
THEN  allow actuators
ELSE  restrict actuators
FINALLY return safety_rating based on confidence_level
```

The following safety policy example, based on an ISO 13855 requirement [44], is used to maintain a separation distance between the robot and a human user. This example and the one that preceded, show that it is possible to explicitly represent requirements taken directly from safety standards. This opens up the possibility that safety standards could not only be used as requirement guidelines, but also as specifications for actual safety constraint implementation. However, it must be noted that many robotic safety standards rarely specify requirements in terms of quantitative values, and instead give general guidelines on the qualities that the final system must possess. In fact the robot safety standard ANSI/RIA R15.06 [6] has the same 'operating at reduced speed' requirement described previously but no others that could be implemented as a safety policy. With the growing interest in personal robotics it may be that in the future we see a shift in robotic safety standards, from the current safety guidelines, to more prescriptive and quantitative requirements. However, as these robots will be required to operate in increasingly dynamic and hard to predict environments, it may make defining quantitative requirements much more difficult.

```
IF    robot_state.operation_mode = collaborative_operation_mode
AND   human_distance >= 100 mm
AND   robot_speed x robot_stopping_time +
      protective_device_minimum_distance < human_distance
AND   confidence_level >= 0.75
THEN  allow actuators
ELSE  restrict actuators
FINALLY return safety_rating based on confidence_level
```

When a safety policy is executed, it either allows its set of actuators to operate normally or imposes restrictions. These restrictions can be in the form of limitations on potential output or as full suppression, preventing the actuators from operating. In both cases a safety rating is produced, which can be used by the control layer to understand the nature of the restrictions, or if none have been imposed, this value is based on the substitution of the sensor function output and the corresponding safety policy comparison values.

The logic used in the safety policies allows only 'and' conditions and not 'or' conditions, this was done so that each rule term in the policy can be evaluated until one returns false. If an 'or' condition is required then a second policy should be used which contains the required conditions. In this context a 'rule term' is a statement which compares two values. If more that one rule term is required then an 'and' is used to separate the terms. Additionally, to aid in the understanding of the policies for the reader, state information should come first, as these dictate the context the policy will be activated in.

As has been discussed, safety policies can be used to implement a number of types of safety process. Presently the main purpose of safety policies is to encode safety requirements identified during hazard analysis. As safety requirements can vary in complexity, one or more policies may be needed. It is important to note that safety requirements that specify changes in the robot construction or to the way in which tasks are completed, cannot be made into a safety policy. Safety policies can only be constructed from requirements that involve the robot software. The

process of composing a safety policy involves analysing a safety requirement to identify what sensor and state information is required and which actuators should be restricted. Some safety polices may only depend on state information; this can be either safety states, as set by other safety policies or robot states, set by the robot control layer. For those policies that use sensor information, the final step is to calculate the confidence level that is required of the sensor functions before the safety policy is activated. This calculation and an example safety policy is given in the section that follows.

In this section the use of safety policies as a method for encoding safety requirements has been introduced. It should be made clear that this part of the proposed methodology (see Figure 2) is still under development. Although, as has been shown, it is quite possible to encode clearly defined safety requirements, generally safety requirements are not well defined and would require them to be further broken down into smaller well defined processes.

3.3 Safety Policy Required Confidence Level Selection

In this section we are only considering safety policies which have a confidence level component. At present those that do not use a confidence level depend only on state information. As previously stated, confidence levels are a value of belief that what is being asserted is true. The values can range from 0.0, which means there is 0% confidence in the assertion, and 1.0 which means that the value is 100% correct. However, it must be remembered that for robots operating in the real world, confidence can rarely be expressed in absolutes and instead depend heavily on the state of the environment.

As safety policies can be used to either restrict actuators or allow them to operate, confidence levels can be used in one of two ways. If the risk associated with a safety policy is high then the designer can choose to use a low confidence level to restrict the actuator or a high confidence level to allow the actuator; this is reversed if the risk is low. In this context, the risk associated with a safety policy is the risk caused by a potential hazard or hazards that are avoided by the actions (restrictions or state changes) taken by the safety policy. The decision to use a 'restricting' or 'allowing' safety policy, depends primarily on other safety policies which are also associated with the same set of actuators. In the cases where either type of safety policy can be used, for example when the safety policy uses a safety state that is altered by another safety policy, then the choice should be made based on which approach is more logical. This subject is discussed further in Section 5.1.

To select appropriate confidence levels for each safety policy, we propose a principled method using risk ratings identified during hazard analysis. These risk ratings and the method we have developed to calculate confidence levels is based on the risk classification matrix (see Table 1). During this process we are essentially interpreting the qualitative results of the hazard analysis as quantitative values required by the safety policies. In addition to this calculation it was decided that the minimum confidence level for allowing an actuator should be 0.5. This was due to the conclusion that if the the confidence was less than 0.5, then their would be a greater probability of the result being incorrect than correct.

The confidence level calculation for safety policies that allow actuators to move is shown in Equation 1, and the calculation used for restricting safety policies is shown in Equation 2.

$$c_a = s * \max(r_h, r_m, r_f) \tag{1}$$

$$c_r = 1 - (s * (\max(r_h, r_m, r_f)) \tag{2}$$

Where c_a is the confidence level for allowing actuators, c_r is the confidence level for restricting actuators, r_h is the risk to humans associated with the safety policy, r_m is the risk to movable

objects, r_f is the risk to fixed objects and s is the potential severity caused as a consequence of the hazard. The severity and hazard consequence values are detailed in a series of tables in Extension 3.

The role of the confidence level calculation is to prevent safety policies being used that are highly hazardous, while making sure that those that are less risky are able to operate freely. To this end the severity values 'None', 'Low', 'Medium', and 'High' were chosen to describe the level of risk. The terms 'direct' and 'indirect' are used to describe hazards that affect humans and objects, either directly or as an indirect result of an initiating event. For example, if the consequence of a hazard is a collision with a human, then this would be considered a direct consequence. Whereas, if a hazard resulted in the robot knocking over an object, which subsequently collided with a human, the human collision would be considered an indirect consequence of the initial event. This is further refined using the terms 'one' and 'many' to describe multiple or single humans or objects affected.

To calculate the risk values used in Equations 1 & 2, a weight value is assigned to each severity value and hazard consequence. These weightings can be seen in the table of Extension 3. There is a monotonic relationship between the weighting values, with larger values being assigned to situations which are deemed more hazardous. For example, if the consequence of a hazard affects both humans and other objects, then only the risk to humans is taken into account, as this represents the highest risk. High risk hazards which require a confidence level of 1.0 (100%) are not allowed on the system and must be redesigned in accordance with IEC 61508 [37]. The weighting values were chosen by examining the types of hazard identified in both the hazard check list (Extension 1) and the hazard analysis document (Extension 2). As this analysis only considered a small subset of all the safety risks associated with a personal robot, it is likely that these risk values will be improved with additional examples added over time.

Given the following safety requirement, *'Robot must move at reduced speed when carrying an object and a human is present.'*, it is possible to construct a single safety policy (see below). This requirement addresses a potential direct risk of harm to one human caused by the robot moving too quickly while carrying an object. It does not pose a risk to more than one human (during a single event) as any collision would activate other safety systems that would immediately stop the robot, thus avoiding harm befalling multiple humans. Similarly there are other safety systems which would stop the robot if an object is sensed in close proximity. Therefore, this safety requirement, although important for the smooth operation of the robot, poses a relatively reduced risk to the human. Using the severity and hazard consequence values presented in Extension 3, it is possible to calculate the confidence level for restricting the actuator as 0.25 and allowing the actuator as 0.75. This is based on multiplying the weighting of a direct risk to one human (0.5) by a medium severity (1.5).

```
IF     robot_state.carrying_object = true
AND    human_detected = true
AND    confidence_level >= 0.25
THEN   actuators.speed_mode = slow
FINALLY return safety_rating based on confidence_level
```

In the following section we discuss how safety policies are implemented in our robot system and how they are used to drive decisions in the control layer.

4 Safety-Driven Control System

Until this point we have focused on issues associated with identifying hazards that may affect personal robots. We now discuss how hazard analyses and the safety requirements they generate can be used to drive decisions within the control layer. This section describes the design of our safety-driven control system architecture and discusses its implementation.

4.1 System Architecture

All computer controlled machines depend on a number of safety functions of one form or another. The idea behind our control system architecture is that these safety functions can be utilised to form a knowledge-base from which active decisions or emergent behaviours can be generated. This resource, as well as being used passively to evaluate possible control actions, can record the state of the robot's environment in terms of safety. In capturing the state of the environment in this way, a model of the robot's working area can be formed that shows where potential areas of risk are located. This model is formed from a combination of safety policies and the sensory data they evaluate.

As Figure 3 shows, the control system has been separated into three distinct layers. A control layer, which is solely responsible for engaging actuators; a safety layer, which has priority over restricting actuator movement; and a sensor layer which provides real-time data to the other two layers. This current design is a modification of an early design by Woodman et al. [45]. As can be seen, the control system is divided up into a series of interconnected modules. The direction of the connecting lines represent the direction data flows and shows the modules which can call functions of other modules. One of the key data flows is between the safety policies and the control mode selection modules. This link allows the control mode selection module to constantly look at the safety restrictions that are being applied at any given time. The diagram in Figure 4 shows a high-level overview of the safety-driven control system architecture. The purpose of this diagram is to illustrate how the main components of the system interact.

The control layer is made up of a number of controllers which provide certain functions, or modes of activity. An example function could be to move the robot to a specific location. This function would carry out the actions until it either completed successfully or a restriction was applied by the safety layer, potentially halting it mid-way. The restriction type can be either a reduction in the actuator output or a complete suppression. In the latter situation the controller would cease all activity, whereupon the control mode selection module would choose another controller based on the rules associated with the current state of the safety layer i.e. which safety policies are being applied. At present only one controller can be selected at any given time. This restriction has been imposed, as it reduces any unpredictable behaviour that may occur as a result of multiple control actions operating concurrently. The main point to understand is that the controllers cannot perform an action which violates one or more restrictions applied by the safety system.

4.2 System Implementation

In this section we discuss the fundamentals of the safety-driven control system implementation. More specifically we show how the control layer makes decisions based on restrictions applied by the safety layer. The design of the safety layer and its implementation decisions are examined in detail in the section that follows.

As has already been established, the control layer is made up of a number of controllers which carry out certain sub-tasks of the robot's main goal. The decision on which of these controllers should be active at any one time is made by the control mode selection module.

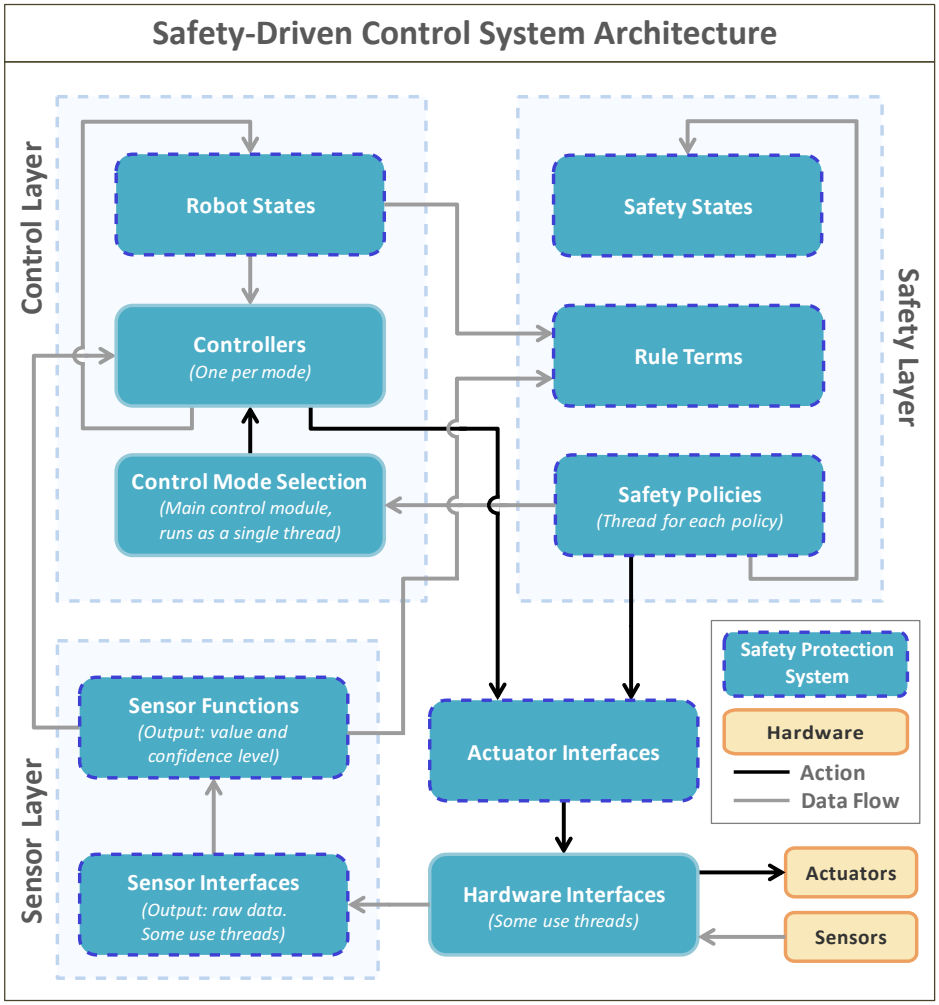

Figure 3: Control system architecture. This illustrates the relationship between safety and control layers. Modules which run concurrently (using execution threads) are marked as these are the best candidate for distributed processing.

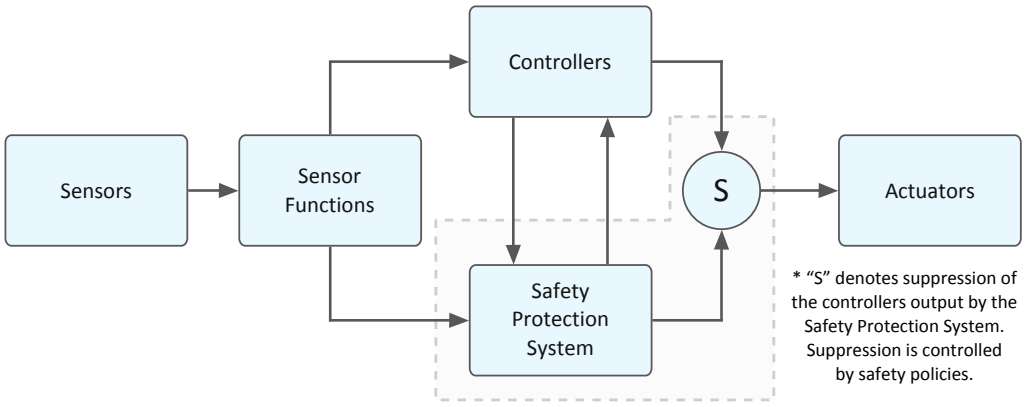

Figure 4: Safety-driven control system high-level diagram.

The implementation of the control mode selection module must be in precise adherence with the safety layer. This means that no element of the control layer should try to violate any restrictions imposed by the safety layer. This should be done, not as it poses a risk to the safe operation of the system, but that not adhering to the safety layer would cause the system to become unresponsive when the safety layer imposes a restriction that the control layer does not know how to handle.

To demonstrate how the different parts of the control layer operate we consider two simplified algorithm extracts. The first extract, which shows part of the control mode selection module, is presented in Algorithm 1. The second extract, presented in Algorithm 2, shows a controller which is called by the control mode selection algorithm.

In Algorithm 1, first the 'MoveToObject' controller (see Algorithm 2) is called, this then performs a number of functions, which first identifies an object and then moves the robot towards it. This process can either be completed successfully, or if a safety policy imposes a restriction, then the controller will return control back to the control mode selection module. At this point if the controller completed its task successfully, the next controller would be selected, for example the controller responsible for picking up objects. Although this would depend on the robot's overall mission goal. This, and all other selections, would follow the same principle as the one presented, and therefore have been omitted.

Algorithm 1 : ControlModeSelection

```
taskCompleted ← false
while taskCompleted = false do
  returnValue ← MoveToObject()
  restrictions ← GetSafetyPolicyRestrictions()
  if returnValue = completed then
    taskCompleted ← true
  else if restrictions contains obstructionDetected then
    NavigateAroundObstruction()
  else if restrictions contains humanInVicinity then
    NegotiateWithHuman()
  end if
end while
```

If the controller fails to complete its process successfully, a list of the current safety policies imposing restrictions is examined. Based on these restrictions a new controller is selected, which is responsible for avoiding the safety hazard that the safety policy addresses. In order to identify each safety policy a unique id is assigned. In Algorithm 1 the safety policies are 'obstructionDetected' and 'humanInVicinity', and the controllers are 'MoveToObject', 'NavigateAroundObstruction' and 'NegotiateWithHuman'. It is important to remember that these two algorithms represent only a small extract of the overall system.

To process both the safety layer and the control layer in parallel, multiple threads of execution are required to run simultaneously. This results in a potential issue where two or more processes could be requesting data from the same resource (usually a sensor function). To overcome this issue we have implemented a 'lock' mechanism. This works as follows: When a new request is made, for example to a sensor function that detects obstacles, a lock is placed on this resource, which prevents any further requests and notifies the requester that a lock is currently in place. When the lock is released all sources that made the request can query the new result simultaneously. This avoids any conflicts between competing threads and reduces the overall processing effort.

Algorithm 2 : MoveToObject

```
if IdentifyObjectLocation() = true then
  if OrientateToFaceObject() = true then
    if MoveTowardObject() = true then
      return completed
    end if
  end if
end if
return failed {The controller could not complete. ControlModeSelection will subsequently
check if this is due to a safety policy being activated}
```

The effect of removing the majority of safety functions from the control layer has greatly reduced the complexity of control routines. However, as there are no real world personal robots of the type discussed in this paper, we are unable to make any quantitative comparisons of processing speed or computational effort. It could be argued that with layered architectures of the type we have presented it is possible to demonstrate the control system in two modes, one with the safety layer operating and one with it not. It would then be possible to make some comparisons of the effectiveness of the design. However, due to the design of our architecture, the safety policies and the restrictions they impose are integral to the control system. Therefore, if the safety layer were to be turned off the control system would simply not function as there would be nothing for the control layer to query against in order to decide what actions it should take.

A potential issue of having a separate safety layer is that a delay is introduced between the time it takes for the safety policy to identify an unsafe event and a restriction being applied. However, this delay is under 0.5 seconds using a 2.8 GHz quad core processor, which did not pose an issue for the types of tasks presented in this paper. Nevertheless, for others implementing this type of architecture it must be given careful consideration. The issue of timing is discussed further in Section 6.5.

At this stage of development the safety policies and the mechanisms which manage them have been designed to enforce safety over availability. It follows that the safest result of any error, failure or unanticipated event is to stop the robot until the issue can be resolved. This can result in the system mission being impeded. It is important to note that this issue is not as a result of having many safety processes (in the form of safety policies), but as a result of the level of caution imposed by each individual safety policy.

The following section explores the design decisions made while developing the safety layer. The designation 'Safety Protection System' is used to define all the processes that are shared between the safety layer, the sensor layer and the actuator interfaces.

5 Safety Protection System

As has already been discussed, the Safety Protection System we are developing is a high-level real-time safety manager, which can intervene to restrict the control layer from activating an actuator in such a way that it could lead to an unsafe event. The design of the Safety Protection System is built on the notion of safety policies (see Section 3.2), acting as rules in a type of knowledge-based system.

This section looks at the main design decisions of the Safety Protection System.

5.1 Design Principles

As the design of the Safety Protection System uses a hierarchical structure, with sensor inputs at the top and actuator outputs at the bottom, a directed acyclic graph (DAG) was chosen as the best method for representing the system. This type of graph is topologically ordered and has no cycles, meaning there is a flow of information from input nodes to output nodes passing through a series of intermediary nodes which alter the data. This method allowed us to create diagrams of the system structure, which made it possible to perform a visual analysis to help identify any potential weaknesses.

The implementation of the graph, or network in its populated form, borrows concepts from both ANNs and Bayesian networks. Bayesian networks are often used for complex high-dimensional problem domains, such as diagnosing medical ailments [46]. Their computational efficiency and inherently visual structure make Bayesian networks ideal for working with complex problems that can be described in terms of causal relationships. A causal graph structure encodes information about the relationships between the nodes of the graph. This means we are able to describe the relationships between sensors and actuators in terms of their connectivity within a networked structure. As with our system, Bayesian networks are generally constructed with knowledge from domain experts with the intention of building a knowledge-base that can be understood by anyone. The knowledge captured is both qualitative, described by the structure of the network, and quantitative, as values stored in the individual nodes. This allows knowledge of different types to be combined in a single representation. The implication of this is that our Safety Protection System, which as previously stated is made up of all the safety functions needed to keep the robot safe, can be fully described by a single network. The comparison with ANNs is more subtle, in fact it is more appropriate to think of our network as a type of ANN with complex nodes. The safety policy nodes use a type of weighting (see Section 5.4), applied to a number of internal rules. This is used to generate an output for one or more actuators.

All rule terms in a safety policy must be evaluated as true in order to be processed fully. Therefore, as was described in Section 3.2, the conditions which can be evaluated quickly, for example state information must be positioned at the beginning of the policy. Those conditions which take more time, usually the sensor function values, must appear last. Each safety policy is processed independently using a separate thread. This allows all policies to be processed in parallel. By using state conditions it is possible to minimise the number of policies which are processed fully at any one time. Presently we are choosing these states manually by carefully examining the relationships between each policy. However, it is likely that this process could be automated in the future.

An example Safety Protection System graph for a mobile robot is shown in Figure 5 (for details of the system modules that populate the Safety Protection System graph see Figure 3). The graph is organised as a series of layers, each layer is composed of nodes of a single type. The top layer represents the robot sensors (S), each of which can be connected to one or more sensor functions (SF). These sensor functions interpret the sensor data and output higher-level information, for example the position of an object. The output of the sensor functions is given as a value (V) and a confidence level (CL). As discussed in Section 3.2, the confidence level is a belief value based on how confident the sensor function is that its output value is correct.

The sensor function information is passed to one or more nodes in the safety policy (SP) layer. These safety policies are implemented using the safety policy rule format described in Section 3.2. Finally, the safety policies are connected to one or more actuator (A) nodes, each of which represents a single motion that an actuator is capable of. For example, a drive motor, which is able to operate in forward and reverse, would be modelled as two actuator nodes. This has been done to increase the availability of the system, so safety policies only intervene on

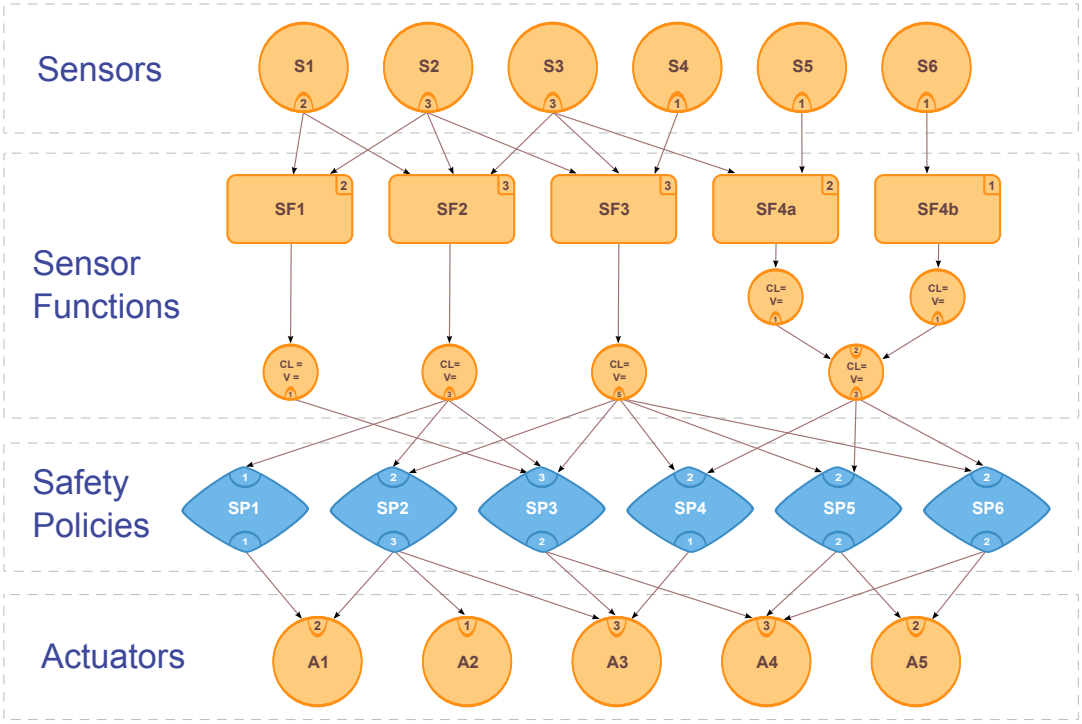

Figure 5: Example Safety Protection System graph. Note: CL = confidence level, V = value.

actions that would put the robot into an unsafe state. An example of this is to allow the robot to move away from danger, but prevent it from moving towards it.

As with any complex system it is important to make certain that the interactions between the various sub-systems do not conflict or compete to create unpredictable behaviour. Due to the design of our system this issue is particularly focused on the Safety Protection System. To avoid any safety policies which compete or conflict, in the first instance is is important that the designers consider carefully the relationship between every safety policy. To aid in this effort, we are working on a tool which can automatically compare every safety policy with each other. The current version of this tool can only identify relationships between two safety policies. For example if two independent safety policies are activated when a human is detected, but one restricts the wheels and the other allows them to move. The aim is to improve this tool to identify the dependencies across multiple safety policies. We are also able to identify conflicts in the restrictions that may occur while the system is operating. This is achieved with a simple mechanism which looks at each restriction being applied. If a new restriction is contrary to a current restriction then the robot will stop moving. At present this is used as another level of safety and ideally it should not occur during normal operation. It is important to note that it would be valid for two safety policies to act on a single actuator at the same time, if this was not a conflicting action. For example one restriction tells the actuator to 'slow' and the other 'stop'. In these types of situation to ensure safety the most severe restriction is always chosen.

To establish the contribution to control theory that our safety-driven control system presents, it is important to consider the interactions between the safety layer and the control layer. As has been shown, the objective of the Safety Protection System is to provide a reference of safety that the control mode selection can use to obtain the desired safe control of the robot. A closed-loop is formed between the control layer and the safety policies, which monitors the sensors and state of the robot and provides feedback in the form of restrictions. This is in addition to the

control layer having direct access to sensor information.

From its inception, we made certain to design the Safety Protection System with all the fundamental requirements that our research had shown a safety system would need. The following sections describe the core design principles that we followed in developing the Safety Protection System.

5.2 Aid the Safety Engineering Process

The Safety Protection System aids the safety engineering process in two main areas. Firstly by having a collection of safety policies all of the same generic type, it is straightforward to iterate through them and verify that they all exist as the specification dictated. The second area, and one which partly justifies the creation of a separate safety system, is that metrics can be taken and used in a quantitative assessment to identify weaknesses in the system. In the example graph of Figure 5, it can be seen that sensor 4 has one connection to sensor function 3, which in turn connects to five of the six safety policies, which finally connect to all the actuator nodes. This means that if sensor 4 failed, the dependent policies could not be evaluated and the system would halt. This shows how areas of the system which may require added redundancy can be identified. This concept will be explored further in the next section.

The ability to analyse the interdependences within a system is an interesting area of study, not just for aiding the hazard analysis process, but also for identifying areas of high and low activity within the system. This kind of information could be utilised to direct improvement efforts into those parts of the system which are most used, and potentially identify those parts which are not required. This bears some similarity to the work of Nehmzow et al. [29], who demonstrated that it was possible to take an existing robot system, and create a model of that system by learning how it functioned. With this model they were able to show how some of the least important functions could be removed, with little to no effect on the system operation.

5.3 Redundancy and Diversity

A major concern for robot system designers are sensor malfunctions and failures [36]. This is a particular concern for those safety critical parts of the system which depend on precise sensor data. Identifying which functions of the system are critical is usually done alongside the hazard analysis process. Assigning a risk level to system functions is one method for declaring the criticality. This method is used in the safety standard IEC 61508 [37] as well as other functional safety standards. The process assigns a level of risk to each function, which must be reduced to an acceptably low level before the system can go into service.

An important method of reducing the risk of critical system functions, is to add redundant sensors. Therefore we wanted to address this issue as seamlessly as possible in our safety architecture. As the example in Figure 5 shows, we have addressed this problem with the use of multiple sensor functions of the same type, SF4a and SF4b. An example of their use could be for monitoring for the presence of a human. These sensor functions could be either identical, using the same sensors and code, or diverse, utilising different sensors and processing the data in a different way. Examining this example further, consider the safety policies SP4, SP5, and SP6, as is shown in Figure 5 these all depend on the sensor function SF4. If these policies were identified as high risk, then a high risk level could be assigned to SF4. To reduce the risk level and the associated risk to the related safety policies, redundancy could be added to SF4 (as is shown in Figure 5). This would mean that SF4 would calculate its value twice using two independent sets of sensors.

We are currently investigating how best to use the output from multiple sensor functions of the same type. The two main approaches we have identified, involve either using the value

from the sensor function with the highest confidence level, or combining the output in some way. Initial trials using a sensor fusion technique described by Schörgendorfer and Elmenreich [47] has shown that a weighted average of multiple sensor function values, can both reduce error while preserving the data diversity obtained from multiple sources. The confidence level output from each sensor function was used as a weighting to give more significance to sensor function values which had a higher confidence level.

A potential problem of using the same sensor and state information in both the safety layer and control layer is the risk of common cause failures [10]. These types of failure can occur when all parts of a system designed to prevent a failure share a common point of reference. For example a sensor which regulates the robot's speed in the control layer may also be used in the safety layer to prevent speeds which are too high. If this sensor gave an erroneous value that was too low, it may cause an acceleration of the robot which is not detected by the safety layer. Therefore it is crucial during the hazard analysis process that the designer identifies all potential common cause failures and adds diversity and/or redundancy as appropriate.

5.4 Confidence Reasoning

Robot sensors and actuators are inherently prone to error. This means that any reasoning about their use, either as data received or actions taken, must incorporate a degree of uncertainty. One of the recent trends in robotics research for handling uncertain information about the robot's environment, is to assign a 'danger index' to objects that the robot perceives could cause a hazard [7, 25, 22]. These indices are continuously updated with the latest sensor readings, with the aim of increasing the robot's confidence in its own understanding. As the robot's confidence increases, so does the reliance on that knowledge, allowing the robot to continue to operate in the presence of potential hazards.

The amount of processing required to produce an accurate view of the world, necessary for navigating safely, is potentially very large. Some have suggested using probabilistic graphical models in the form of Bayesian networks for handling this complex data set [48, 43]. This involves using Bayes rule to combine all the robot data to produce a reasoned output.

For the Safety Protection System we have chosen to use confidence levels (sometimes referred to as confidence factors), which is a method often used in expert systems for dealing with uncertainty [49, 42]. This differs from Bayesian networks as it does not require a priori probability to be assigned to each decisional part of the network. Instead it allows us to assign a value of belief to sensor readings, which can be combined to give an overall confidence level for use in the safety policies. This value can then be compared with the confidence level required by the associated safety policies (see Section 3.3). It is important to note that it is the sensor function output values we are assigning a confidence level to and not the sensor values.

Prior to developing our own confidence equations, we considered a number of other techniques for calculating sensor confidence levels. These can be broadly categorised as those which use only static information about the sensor tolerances and error potential, and those that utilise previous sensor data and readings from multiple sensors. An example which uses a combination of static information and previous sensor data is described by Goebel and Agogino [50]. Their method uses fuzzy logic to combine multiple sensor readings in order to identify uncertainty caused by sensor noise. Another technique, which belongs to the latter category, is presented by Hossain et al. [51]. Their novel approach uses multiple sensors which detect the same type of data. Learning algorithms are used to compare the data over time to identify any differences. A confidence value can then be produced in real-time by comparing the latest sensor values with an expected value.

The main issues with the confidence level techniques that we surveyed is that they do not

take into account the structure of our sensors and sensor function configuration. Therefore, it was decided that at this stage of development it would be more beneficial to use a simpler calculation that is easier to understand and to diagnose any issues that arise. However, it is likely that as development continues a more comprehensive confidence calculation will be required.

To calculate the confidence level of the output from a sensor function, we compare a number of samples in order to identify any errors. The number of samples taken is important, as too few can result in errors not being detected and too many means that the first samples are older and less relevant to the later ones. Therefore, the strategy we used was to base the number of samples taken, on the confidence level required by the safety policy that requests the sensor function value. The proposed calculation is given in Equation 3.

$$n = 2(r/q) \tag{3}$$

The number of sample values is expressed as n. The required confidence level, as dictated by the safety policy, is expressed as r and q is the confidence per query assigned to the sensor function. Sensor functions which produce reliable results, for example bump detection, would have a high confidence level per query, whereas a sensor function with low reliability, such as face detection, would have a low confidence per query. At present the confidence per query for each sensor function is set and fixed each time the robot is started. Choosing the value involves a test campaign specifically designed for each sensor function. For example the face detection sensor function requires a number of detection samples taken from various angles. We are currently working on using information from sensors to adjust the confidence per query dynamically. For example, initial experiments show that using information from light and distance sensors can improve the reliability of the face detection sensor function.

In addition to the number of samples taken, there is also a time restriction within which the sensor function values must be obtained. If the time it takes to retrieve all the required sensor function values is greater than the typical time it takes to retrieve all values then the confidence level -998.0 is returned to identify that a timing problem has occurred. The required processing time is determined by the system designer and is attributed against each sensor function. Additionally, if the value of the sensor function is equal to the sensor functions known failure value, i.e. an error has occurred while calculating the sensor function value due to one or more sensor failures, then the confidence level -999.0 is returned to identify this failure.

The confidence level calculation is shown in Equation 4. The first step of the calculation is to multiply the required confidence level by two to give the maximum possible confidence level. This is then subtracted from the sample standard deviation of all the samples and divided by the number of samples. The sensor function confidence level is represented as c_s and σ is the standard deviation of the samples, calculated using Equation 5. Where n is the number of sensor function samples, x_i represents the sensor function values and $\overline{x}$ is the mean of the sample values.

Standard deviation is used to measure the dispersion between a set of values; the greater the dispersion the larger the standard deviation. This is used in the confidence level calculation to measure the error in the sample values. Another technique which we considered was the variance of the sample, which is the sum of the difference between a set of values. This gave similar results to standard deviation when the samples values had low dispersion. However, it was much more sensitive to large error values.

$$c_s = 2r - (\sigma/n) \tag{4}$$

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \overline{x})^2}{n-1}} \tag{5}$$

To calculate the actual value output of the sensor function, to which the confidence level is assigned, we used the median of the sample values. The mean and mode averages were also considered for choosing the output value, however, we found that both were prone to give extreme values if the sample contained one or more error values.

6 Experiments and Results

To evaluate the proposed safety methodology and system design presented in this paper, we have conducted a number of experiments using a simulated MobileRobots PeopleBot (see Figure 6). The task involves moving items around an environment with and without HRI. Results are presented showing the effectiveness of the system in performing tasks and its ability to handle faults injected into the controllers.

A simulated robot was used instead of a physical robot, as this allowed us to explore the effect of errors in sensors and actuator movement on the safety of the robot. By setting a known sensor error, we were able to adapt the confidence levels of safety policies appropriately. For a real physical robot, sensor errors will need to be established.

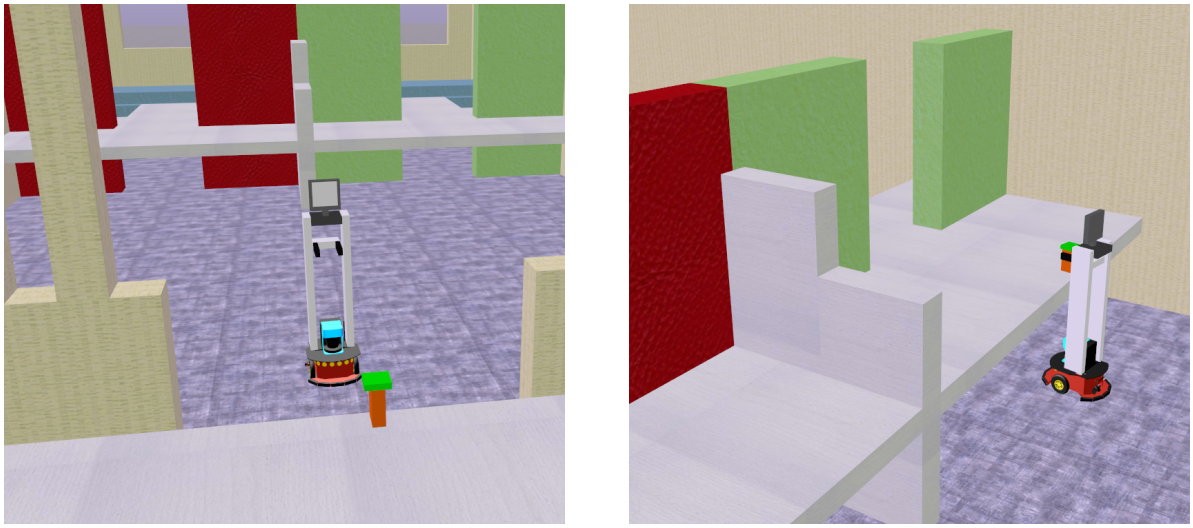

Figure 6: Simulated mobile robot sorting parts (see Extension 4 for full video). The simulation environment was made using Microsoft Robotics Developer Studio.

6.1 Robot Task Example: Part Sorting

To test the viability of the proposed personal robot development techniques, we have devised a robot task, which can be executed with or without human interaction. The idea behind this is that experiments can be performed in an industrial type setting, which almost always requires complete isolation from humans. A human element can then be introduced to the experiment, without any major changes to the setup, allowing for better comparison of results.

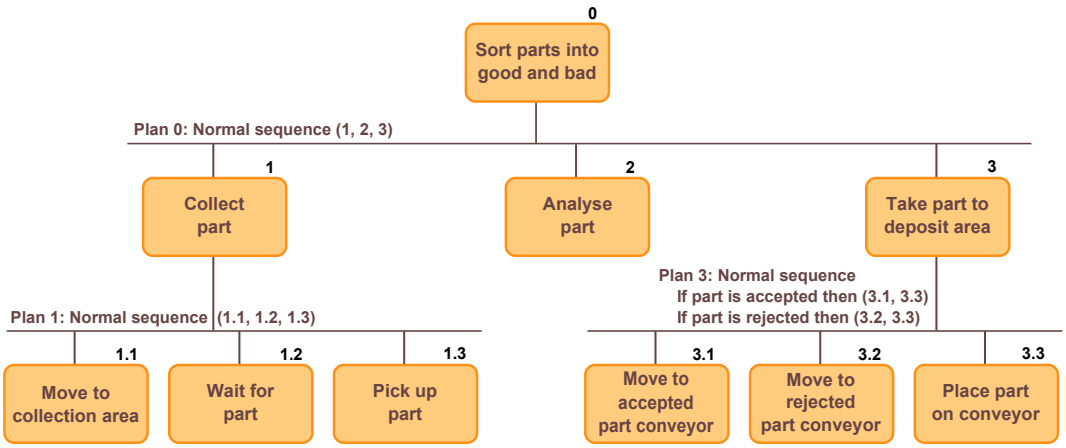

Figure 7: Hierarchical task analysis diagram of a part sorting robot.

6.1.1 Robot Sorting Task – Part 1: Isolation

The robot sorting task involves retrieving parts from a collection area and sorting them into good or bad based on quality. Part 1 of the robot sorting task has the following requirements. The robot must select a part from the part dispenser and place it on either a 'good part' or 'bad part' conveyor. The robot should perform an on-board analysis of the part to determine its quality. The distance from the part dispenser to the conveyors is 3 meters. The robot must be completely autonomous and not tethered or fixed in anyway.

To model the task requirements we used a scenario based technique called 'hierarchical task analysis', currently under evaluation at the Bristol Robotics Laboratory (BRL) [52]. Hierarchical task analysis, developed by Duncan and Annett in the late 1960s [53], is a method of task analysis traditionally used in human factors work for time-motion studies. This involves analysing the way in which humans complete tasks, with the aim of improving efficiency. With this approach there is particular emphasis on task decomposition, which means taking each element of a task and breaking it down to its simplest form [53]. The approach of identifying task-based elements of a process, suggests that hierarchical task analysis may be a promising technique for designing complex robot tasks which need to be performed more dynamically when compared to traditional manufacturing robots. Although we believe this technique has not been used before for describing personal robot tasks, we have found it a useful method for task decomposition.

The diagram of Figure 7 shows it is possible, in principle, to use this technique for modelling customer requirements in a clear and concise manner. In this diagram the robot's tasks are decomposed hierarchically into sub-tasks; the sequencing of sub-tasks at each level is specified as a plan. The sequence of nodes for each plan dictate the logical steps by which each task should be performed.

Hierarchical task analysis would be performed in the early stages of a project in order to record how each task should be completed. The diagrams produced would then be used in the hazard analysis process to help safety engineers both understand the task process and identify any hazardous situations that may arise as a result of the way in which a task is structured.

6.1.2 Robot Sorting Task – Part 2: Human-Robot Interaction

Building on the requirements set-out in part 1, the task example part 2 requires the robot to operate cooperatively with a human user. The additional requirement is as follows. While the

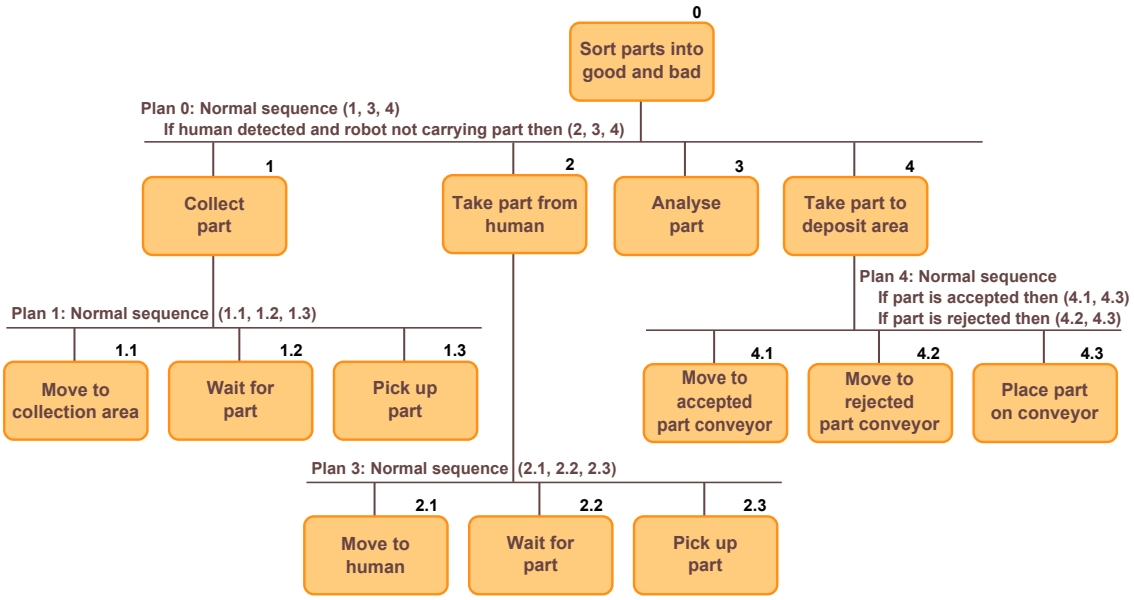

Figure 8: Hierarchical task analysis diagram of a part sorting robot with HRI.

robot is approaching the part collection area, it must be able to identify the location of the human user and stop at a safe distance facing them. The user is then allowed to hold a part for inspection in front of the robot. The robot must negotiate with the human and fully take hold of the part. Finally it should examine the part and take it to the correct good/bad conveyor, all the time maintaining a safe operating speed and distance from the human. This process is presented in Figure 8. Comparing this diagram with Figure 7, it can be seen that a high-level branch, with the starting node 'take part from human', and a new decision condition 'If human detected and robot not carrying part' has been added. These two elements fulfil the new human interactive requirements of the task.

The robot base shall remain stationary when in close proximity to a human. The safe separation distance should be in accordance with the machinery safety standard ISO 13855 [44]. When in close proximity to the user, all actuators, which are not prevented from moving, should operate at a reduced speed mode, in accordance with the robot safety standard ISO 10218-1 [5].

6.2 Robot Sorting Task Safety Policies

Hazard analysis has been performed for the robot sorting task (part 1 and 2) using the method introduced in Section 3. It was clear from this analysis that the hazard check list was an important tool for identifying a number of potential safety risks that were not identified using the traditional hazard analysis approach. However, although the use of the hazard check list has been justified, it must be remembered that at this stage it cannot cover every possible hazard.

The results of hazard analysis for the robot sorting task is presented in Extension 2. This analysis identified a number of safety requirements which were then used to generate 22 safety policies. These safety policies are presented in Table 4. Those safety policies that are based on safety requirements identified using the hazard check list are marked with a '*'. Two examples of these are the 'Unscheduled weight change' and 'Unintended movement' safety policies. These both monitor changes to the robot which should not occur as part of the task, and may be activated as a result of a human either pushing the robot or adding weight unexpectedly.

For each safety requirement that involves the control system, it was found that we could

Safety Policy	Restriction Type	No. Actuators
* Fire detection	Abort	*All*
Crash detection	Abort	*All*
* Maintenance mode activated	Abort	*All*
* Sudden drop detected	Stop	*All*
Human voice command 'stop'	Stop	*All*
Collision with object	Stop	*All*
* Unscheduled weight change	Stop	*All*
* Unintended movement	Stop	*All*
Unable to reach object	Stop	5
Obstruction imminent	Stop	5
At table	Stop	4
Taking object from human	Stop	4
Picking Up object	Stop	4
Depositing object	Stop	4
* Axes orientation	Stop	3
* Floor moisture high	Stop	3
Gripper pressure exceeded	Stop	1
Approaching table	Slow	3
Human detected	Slow	3
Obstruction near	Slow	3
Moving with object	Slow	3
Approaching object	Slow	3

Table 4: Safety policies used for part sorting task (ranked by restriction severity)

satisfy the requirement by creating one or more safety policies, written in the standard format discussed in Section 3.2. However, this is not advisable in some situations when real-time operation is critical for safety. For example a watchdog timer that monitors the responsiveness of the system to prevent processes from hanging. This is due to the large amount of parallel processing which occurs as a result of having a separate process or thread for every safety policy. However, in general this does not cause a safety risk as each safety policy must define a confidence level, which has a time component so confidence in the state of the world decreases rapidly if new samples are not taken. Furthermore, by incorporating confidence levels into the safety policy design, we are able to address many of the safety requirements that are concerned with environmental conditions affecting sensor accuracy.

During hazard analysis we discovered that in order to complete the robot sorting task at no point would two actuators need to be activated simultaneously. Although we did not apply a restriction to prevent this, it is note worthy that a complex robot task like the one discussed

can be controlled in a sequential manner. This could potentially reduce the overall processing effort required to monitor the robot's safety while in operation. This is discussed further in Section 6.5.

6.3 Handling Controller Faults

To demonstrate the effectiveness of the Safety Protection System in preventing the controllers from violating one or more safety policies and potentially causing an unsafe event, we have conducted experiments using fault injection. In this context a fault is a request to activate an actuator which is currently being restricted by one or more safety policies. Ideally controllers should not produce these types of activations, as they should be designed to first check the state of the actuator for any applied safety policies before making action requests.

Fault injection, also known as fault insertion testing, is a widely used experimental validation technique for demonstrating the fault tolerance of a variety of systems [54]. Work by Lussier et al. [55], has shown the effectiveness of using fault-injection for validating an implementation of the LAAS architecture. In their work they examined the impact of adding different levels of redundancy to the planner element of the system by first injecting faults and then analysing how the system behaved in the presence of these faults.

An experiment was conducted using a simulated MobileRobots PeopleBot developed with Microsoft Robotics Developer Studio. The robot sorting task was used for the experiment, which involved picking up an item from one location and depositing it in another location. While the task was being carried out, a fault was injected into a controller which requested the activation of the actuator that moves the robot forward. Although all actuators are treated equally and therefore any could have been used in the experiment, we chose the forward actuator as this type of movement has the most hazards associated with it. The results from the experiment are presented in Figure 9.

As the results show, 72 seconds into the task a forward actuator request is denied. This request was injected into the controller while the safety policy that stops the robot running into the table (where the object to pick up is located) was active. This safety policy's role is to disable the forward actuator if a table is detected in front of the robot. If this request was not suppressed, then the robot would have collided with the table.

The graphs presented in Figure 9, show only a selection of the data produced while the system is operating. From these results it is possible to visualise how the activation of safety policies cause the associated actuators to become inactive. With this data it is possible to identify areas of inefficiency within the controllers. For example it can be seen that the actuators that turn the robot are being constantly activated. We subsequently found that this is due to the threshold that is set for aligning the robot with its target location. To reduce this constant adjusting of angle, this threshold was simply increased.

As has already been alluded to our system is both behavioural, in terms of how the system can react to safety restrictions, and uses planning to determine how tasks should be completed. To validate systems that have reactive elements, traditional methods tend to analyse the systems' mechanisms and not the behaviour. In this context, the mechanisms can be thought of as the system functions, whereas the behaviour is the ways in which the system functions interact during operation. A study by Swarup and Ramaiah [36], observed that analysis of system behaviour in order to identify violations in safety constraints will become an increasingly important aspect of safety-critical system verification, as the complexity of systems continue to increase. By removing safety functions from the control layer we have reduced its complexity and therefore the verification effort required. These processes still need to be verified within the Safety Protection System. However, as discussed in Section 5, this can be tested in isolation

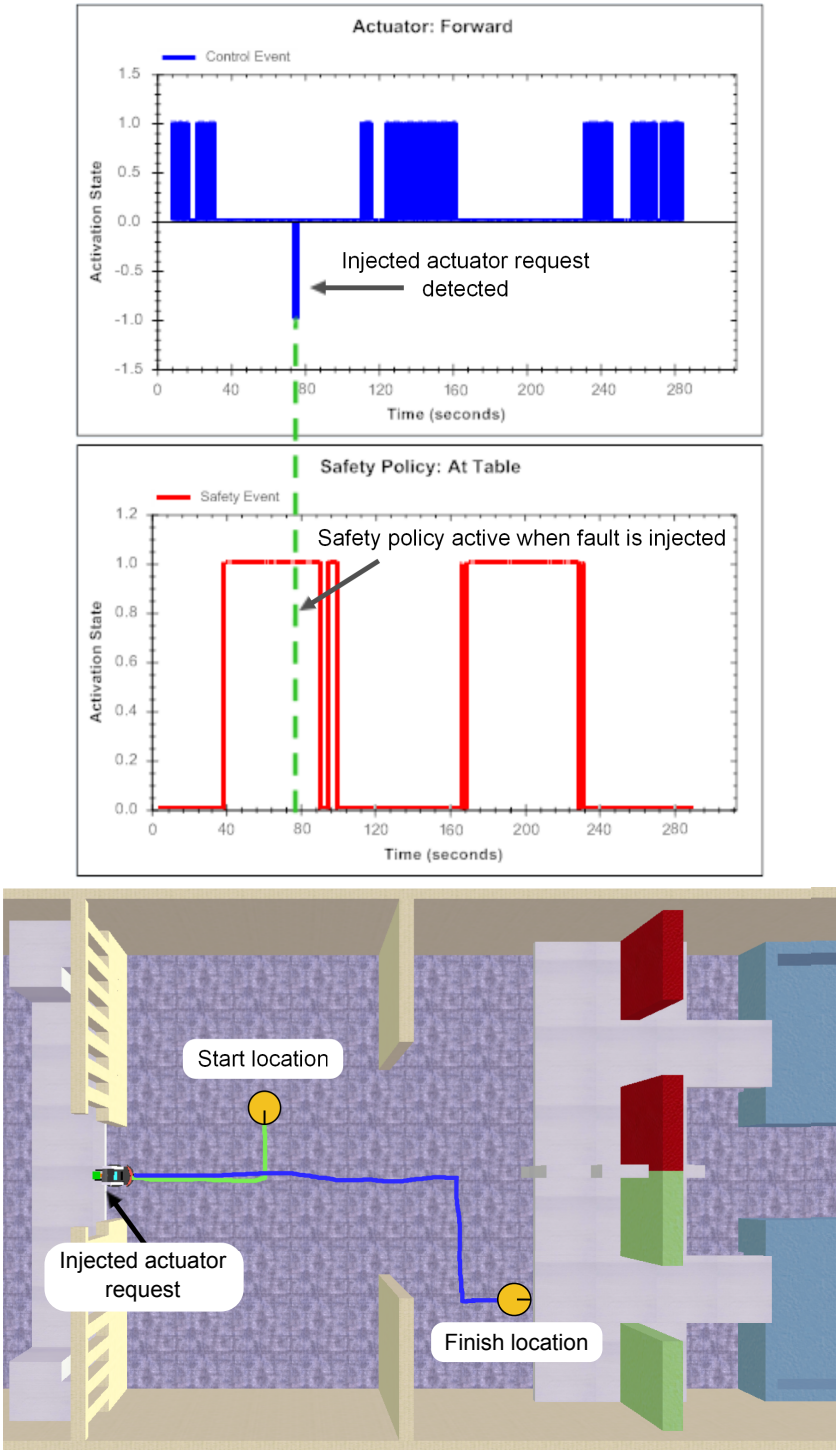

Figure 9: Robot sorting task results: Demonstrating that a forward request injected into a controller, while the Safety Protection System is restricting the actuator, can be detected. Only the safety policy 'At Table' is shown as it is responsible for restricting the forward actuator at the time the request is made.

and potentially be exported for use on other hardware without requiring any changes.

The results from our fault-injection experiment shows that the Safety Protection System is capable of suppressing actuator requests that may invalidate a safety policy and cause an unsafe event. Although these results are to be expected as the safety policy will always activate when a table is detected preventing the robot moving forward. The purpose of this experiment is to demonstrate the importance of the Safety Protection System and how the controllers do not necessarily need to be fault free. What the results clearly show is that it is possible to detect when a controller attempts to invalidate a safety policy. This opens up the possibility that controllers could be made more adaptable by, for instance, using this event information as reinforcement to change how they behave in certain conditions. For example, if a safety policy is activated unexpectedly when a controller is avoiding an obstacle, it could use this information to adjust its control routine. Over time this could allow the robot to navigate around its environment more efficiently. This type of learning could be applied to any type of task, for example it could equally be applied to learning which type of verbal communication is most appropriate in a given situation to get the desired response from the user. Finally, as we have demonstrated, the fact that controllers do not need to be fault free to keep the system safe, gives more scope on how changes to controllers and the control mode selection module can be made.

6.4 Detecting Sensor Faults and Failures

Building on the results presented in the previous section here we discuss how sensor faults and failures are handled by the sensor functions. As before, fault injection is used to simulate how the control system behaves in the presence of a fault.

The experiments used a single ultrasonic sensor, which can detect the distance of the nearest object it is pointed at. A sensor function was then created which sampled the sensor value and outputted it as its own value. In many cases sensor functions are more complicated than this, manipulating data taken from multiple sources. However, to make the experiment clearer a simple sensor function was used.

To demonstrate how the sensor function affected the control system, a safety policy was devised. This safety policy (shown below) compares the sensor function value, named 'obstacle_distance', with the minimum acceptable distance, in this case 200 mm. A confidence level of 0.7 has been given. If either the obstacle distance value falls below 200, or the confidence level falls below 0.7 then the forward actuator will be stopped.

```
IF     obstacle_distance < 200
AND    confidence_level >= 0.7
THEN forward_actuator.speed_mode = stop
FINALLY return safety_rating based on confidence_level
```

The first experiment injected two faults into the sensor output in quick succession. The result of this is shown in the graph of Figure 10. These faults, one at 7300 ms and another at 7350 ms, were produced by means of a software function initiated by clicking the computer mouse. The dashed line at the confidence level 0.7 identifies the point that the safety policy is triggered. As can be seen on the graph the confidence level drops significantly when the fault occurs, which in turn triggers the safety policy.

For the second experiment we considered a total sensor failure. This meant that the sensor produced no output that could be read by the system. The result is shown in the graph of Figure 11. The failure occurred at 6550 ms and again was triggered manually by means of a software function. As a result of the sensor not producing any real output, the sensor interface

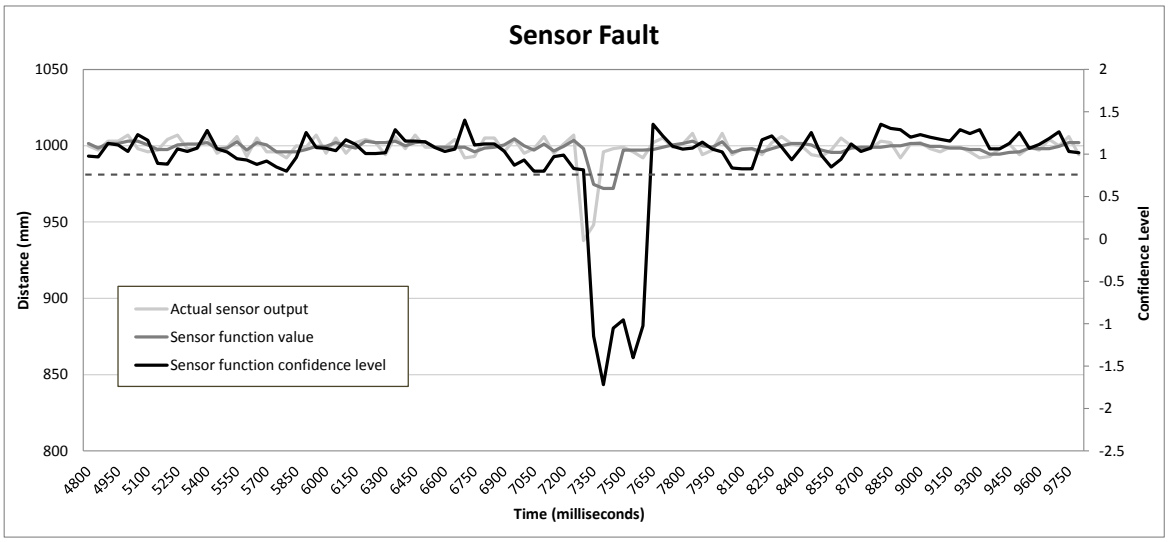

Figure 10: Obstacle distance output – Faults injected at 7300 ms and 7350 ms.

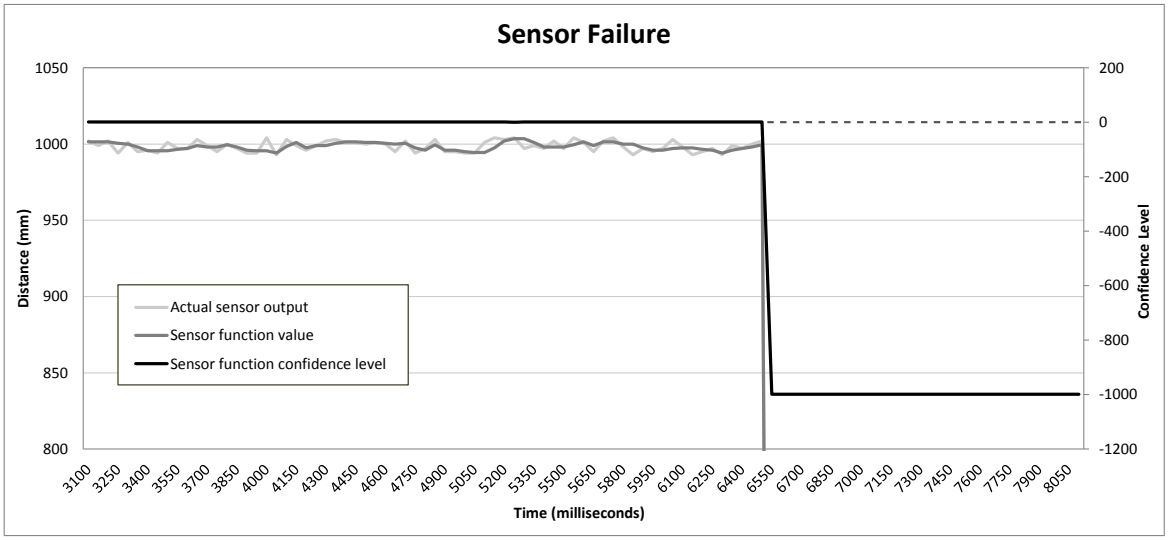

Figure 11: Obstacle distance output – Failure at 6550 ms.

outputs an error value of -1.0. As part of the sensor function process, all sensor values are checked for errors, therefore when this is detected it triggers the sensor function to output its error value, which in this case is also -1.0. As was discussed in Section 5.4, when a sensor function identified an error the confidence level is not calculated and instead the confidence level -999.0 is returned to identify a failure.

Both sets of results demonstrate how the confidence level is an integral part of a safety policy. By specifying a level under which actuators are restricted the Safety Protection System is able to manage sensor faults and failures which may lead to an unsafe event. The following section considers timing issues that arise from using safety policies and confidence levels.

6.5 Timing Issues

The design of the experiments discussed in this paper do not require the robot to operate quickly, nor do they require actions of the robot to be performed in parallel. This was not a

conscious decision and may have come about from using the hierarchical task analysis model described in Section 6.1. As each action of the robot (moving forward, closing gripper, etc.) was performed sequentially, we found that delays caused by safety policies needing to take multiple sensor samples to build confidence did not unduly impact the robot's task mission.

To demonstrate the effect of processing safety policies we have examined the human detection safety policy presented in Section 3.3. This safety policy was chosen as it takes the longest time to process. This policy is responsible for detecting the presence of any humans that may be standing in front of the robot. If a human is detected then the actuators for moving forward are restricted. The time taken to identify a human standing in front of the robot is ≈44 ms, and the number of samples required to confirm the presence of the human is 4. In addition to the processing time of the safety policy, is the overhead of the other polices and control functions executing concurrently. This equates to an average time (based on 20 trials) of 218ms, from the human standing in front of the robot, to the safety policy restricting the actuators. The wheel speed of the robot is restricted to 0.15 m/s, this means that potentially the robot can travel 0.0327 metres (32.7 mm) in the time it takes to restrict the actuators. For the type of mobile robot we are using, this delay between a hazard entering the vicinity of the robot and the restriction being applied presents only a low risk. However, this could present a potential issue for robots which require faster movement. For example a humanoid robot with high-functioning arms.

These timing issues could be alleviated by distributing the processing across multiple computer systems, using more accurate sensors and faster processing algorithms for computationally intensive problems such as human detection.

7 Conclusions

The development costs for safety-critical systems, such as those used in the rail, nuclear and aviation industries, can be significant. For personal robots there is no evidence that suggest that development of safety systems will be any less significant. Therefore it is clear that reducing the effort required to identify and implement safety systems will in turn reduce the costs. The methodology we have developed allows the designer to systemically identify hazards and implement those safety requirement solutions that are associated with the control system. The result of this process is that the Safety Protection System, which in addition to being part of the working system, can also be used to analyse the generated safety policies and identify any inconsistencies.

In this paper we have presented a number of development techniques, which aim to improve the safety of personal robots. The approach we have taken has been based on a study of existing techniques, which found that those currently used in robotic development, have significant limitations for designing safe personal robots. This leads us to propose a process where a high-level safety system is developed during the hazard analysis stage. As described, this Safety Protection System is used to both verify the implementation of the safety requirements and to act as a real-time safety manager, preventing the controllers from performing unsafe actions.

A hazard check list has been developed that aids robot designers in identifying hazards that may be associated with non-mission tasks. This check list, which forms part of our proposed methodology, was used to analyse a number of robotic tasks and has been found to identify hazards that may have not been identified using traditional methods. This check list was developed by examining a number of real-world robot tasks and will likely benefit from the findings of additional tasks added over time.

To test the viability of the proposed methodology, we have outlined a robotic task which allows for experiments to be conducted in two parts. The first part involves a complex robot

task, with the robot working in isolation. The second part introduces a human element and removes restrictions preventing the robot operating in the presence of a human. Using our hazard analysis methodology we analysed the robotic task and identified a number of safety requirements. From these requirements safety policies were devised and implemented to form the Safety Protection System.

To demonstrate the safety-driven control system in operation we have presented a series of results showing how safety and control events are generated over time. From these results it is possible to visualise how the activation of safety policies cause the associated actuators to become inactive.

To establish the degree of fault tolerance achieved by our control system, we performed a number of experiments using fault injection. The first experiment considered how erroneous control actions are handled in the control layer. A fault was injected into one of the controllers in the form of a request, which activated the forward drive actuator. This involved running the system normally and at a time when a safety policy was active, sending a command to the actuator interface from the control layer to switch the forward actuator on. As we have discussed, we are able to both prevent hazardous actuator requests and identify the time and frequency that they occur. A second experiment looked at how sensor faults and failures are managed in the control system by means of confidence levels. This experiment used the same method of fault injection, simulating both an intermittent fault and a complete sensor failure.

We have argued that robots intended for applications such as domestic or personal care will be required to perform a wide variety of complex tasks involving sophisticated sensory processing and action generation. Therefore, the number and diversity of possible interactions with the environment could be much higher than the number required of many contemporary applications. It is foreseeable that in the future, robots will be required to operate in close contact with humans in ever changing environments. These anticipated robotic applications will require the use of adaptive learning or evolutionary computational processes, which change over time in order to continue providing their functions throughout the operational life of the robot. The implication of adaptation features of robot operation is that the manual performance of safety analyses at design time is likely to become infeasible. Since the number of interactions between the robot and its environment is much greater, the effort required for the analysis will be increased. Therefore, it is likely that analysis processes will become impractical without automation, which at the present time does not exist.

We believe that the research presented here provides a logical process for developing and analysing the safety of a personal robot, which could form the basis of a safety proof. In addition, by using safety policies a designer has the flexibility to add or adapt safety systems in order to change how the robot reacts in any given situation. Furthermore, the design of safety policies are such that they are not tied to any specific hardware. This opens up the possibility that a set of safety policies could be created for one or more sensors or actuators and combined to a form hardware/software package that could be fitted to many different types of robot.

8 Future Work

Future work will take the concept of automation further and investigate how adaptive learning could be used to modify safety policies, while maintaining safety, to allow the robot to carry out new tasks. There is a fine line between adaptability and learning. In this paper we have described how the system can react to changes to environmental conditions, both actual and perceived. However, these can be considered as behavioural routines as the system is simply reacting to changes within a pre-defined safe threshold. To move outside that threshold it is key for the system to learn new information.

The main issue we face with developing learning mechanisms is how to make internal changes to the Safety Protection System without causing inadvertent unsafe events. For humans errors and misjudgements are all part of the learning process [13]. However, in the majority of circumstances it will be unacceptable to allow the robot to reach a state where it could cause a hazard.

In developing the simulation for our experiments, we have considered the idea of employing the simulation as an internal real-time task evaluator for a real robot. This would involve the robot perceiving its environment, modelling it in simulation and then trying out various simulated scenarios, evaluating each for possible hazards. The robot could then choose to complete the task in a safe and efficient way. Based on the success of the chosen scenario, reinforcement could be fed back into the simulation in order to improve its results.

Acknowledgements

This research is funded by Great Western Research in partnership with Avian Technologies Ltd.

Appendix A: Index to Multimedia Extensions

The multimedia extensions to this article are at: http://www.ijrr.org.

Extension	Media Type	Description
1	Data	Hazard analysis check list
2	Data	Hazard analysis findings for part sorting robot
3	Data	Confidence level calculations
4	Video	Demonstration of robot performing part sorting task

References

[1] T. Ogure, Y. Nakabo, S. Jeong, and Y. Yamada, "Risk management simulator for low-powered human-collaborative industrial robots," *International Conference on Intelligent Robots (IROS)*, pp. 49–54, 2009.

[2] A. Desantis, B. Siciliano, A. Deluca, and A. Bicchi, "An atlas of physical humanrobot interaction," *Mechanism and Machine Theory*, vol. 43, pp. 253–270, March 2008.

[3] D. Kulić and E. Croft, "Strategies for safety in human robot interaction," *Proceedings of IEEE International Conference on Advanced Robotics*, pp. 644–649, 2003.

[4] M. Hägele, K. Nilsson, and N. J. Pires, *Springer Handbook of Robotics: Industrial Robotics.* Heidelberg, Germany: Springer, 2008.

[5] ISO 10218-1, *ISO 10218-1:2006 : Robots for Industrial Environments – Safety Requirements – Part 1: Robot.* Geneva, Switzerland: International Organization for Standardization, 2006.

[6] ANSI/RIA R15.06, *ANSI/RIA-R15.06-1999(R2009) : Industrial Robots and Robot Systems – Safety Requirements.* New York, USA: American National Standards Institute, 2009.

[7] M. Nokata, K. Ikuta, and H. Ishii, "Safety-optimizing method of human-care robot design and control," *Proceedings of IEEE Robotics and Automation*, vol. 2, pp. 1991–1996, 2002.

[8] R. Alami, A. Albu-Schaeffer, A. Bicchi, R. Bischoff, R. Chatila, A. De Luca, A. De Santis, G. Giralt, J. Guiochet, G. Hirzinger, F. Ingrand, V. Lippiello, R. Mattone, D. Powell, S. Sen, B. Siciliano, G. Tonietti, and L. Villani, "Safe and dependable physical human-robot interaction in anthropic domains: State of the art and challenges," *Proc. IROS'06 Workshop on pHRI - Physical Human-Robot Interaction in Anthropic Domains*, 2006.

[9] S. Bensalem, M. Gallien, F. Ingrand, I. Kahloul, and T.-H. Nguyen, "Toward a more dependable software architecture for autonomous robots," *Special issue on Software Engineering for Robotics of the IEEE Robotics and Automation Magazine*, vol. 16, pp. 67–77, March 2009.

[10] C. Ericson, *Hazard analysis techniques for system safety.* New Jersey, USA: John Wiley & Sons. Inc., 2005.

[11] R. Storey, Neil, *Safety Critical Computer Systems.* Boston, USA: Addison-Wesley Longman Publishing Co., Inc., 1996.

[12] N. J. Bahr, *System Safety Engineering and Risk Assessment: A Practical Approach.* Washington, USA: Taylor & Francis, 1997.

[13] N. G. Leveson, *Safeware – system safety and computers: a guide to preventing accidents and losses caused by technology.* Washington, USA: Addison-Wesley, 1995.

[14] A. Lüdtke and L. Pfeifer, *Human Error Analysis Based on a Semantically Defined Cognitive Pilot Model*, vol. 4680 of *Lecture Notes in Computer Science.* Heidelberg, Germany: Springer, 2007.

[15] Y. Yamada, T. Yamamoto, T. Morizono, and Y. Umetani, "Fta-based issues on securing human safety in a human/robot coexistance system," *IEEE Systems, Man and Cybernetics*, vol. 2, pp. 1058–1063, 1999.

[16] K. Khodabandehloo, "Analyses of robot systems using fault and event trees: case studies," *Reliability Engineering and System Safety*, vol. 53, no. 3, pp. 247–264, 1996.

[17] J. Guiochet and C. Baron, "Uml based fmeca in risk analysis," *European Simulation and Modelling Conference ESMc2003*, pp. 1–8, October 2003.

[18] D. Martin-Guillerez, J. Guiochet, D. Powell, and C. Zanon, "A uml-based method for risk analysis of human-robot interactions," *2nd International Workshop on Software Engineering for Resilient Systems*, pp. 32–41, April 2010.

[19] P. Böhm and T. Gruber, "A novel hazop study approach in the rams analysis of a therapeutic robot for disabled children," *Proceedings of the 29th international conference on Computer safety, reliability, and security*, vol. 6351, pp. 15–27, 2010.

[20] M. Goodrich and A. Schultz, "Human-robot interaction: A survey," *Foundations and Trends in Human-Computer Interaction*, vol. 1, pp. 203–275, January 2007.

[21] J. Heinzmann and A. Zelinsky, "Quantitative safety guarantees for physical human-robot interaction," *International Journal of Robotic Research*, vol. 22, no. 7, pp. 479–504, 2003.

[22] D. Kulić and E. Croft, "Pre-collision safety strategies for human-robot interaction," *Autonomous Robots*, vol. 22, no. 2, pp. 149–164, 2007.

[23] M. Giuliani, C. Lenz, T. Mller, M. Rickert, and A. Knoll, "Design principles for safety in human-robot interaction," *International Journal of Social Robotics*, vol. 2, no. 3, pp. 253–274, 2010.

[24] R. Alexander, N. Herbert, and T. Kelly, "The role of the human in an autonomous system," *Proceedings of the 4th IET System Safety Conference*, 2009.

[25] K. Ikuta, H. Ishii, and N. Makoto, "Safety evaluation method of design and control for human-care robots," *International Journal of Robotic Research*, vol. 22, no. 5, pp. 281–298, 2003.

[26] P. Bonasso and D. Kortenkamp, "Using a layered control architecture to alleviate planning with incomplete information," *Proceedings of the AAA Spring Symposium on Planning with Incomplete Information for Robot Problems*, pp. 1–4, March 1996.

[27] U. Nehmzow, "Flexible control of mobile robots through autonomous competence acquisition," *Measurement and Control*, vol. 28, pp. 48–54, 1995.

[28] T. Larsen and S. Hansen, "Evolving composite robot behaviour – a modular architecture," *Proceedings of RoMoCo'05*, pp. 271–276, 2005.

[29] U. Nehmzow, T. Kyriacou, R. Iglesias, and S. Billings, "Robotmodic: Modelling, identification and characterisation of mobile robots," *Proceedings of TAROS*, 2004.

[30] Z. Kurd, T. Kelly, and J. Austin, "Developing artificial neural networks for safety critical systems," *Neural Computing and Applications*, vol. 16, pp. 11–19, October 2006.

[31] B. Lussier, M. Gallien, J. Guiochet, F. Ingr, M. olivier Killijian, and D. Powell, "Experiments with diversified models for fault-tolerant planning," *IARP'07, Roma, Italy*, 2007.

[32] R. Brooks, *Cambrian intelligence: the early history of the new AI*. Cambridge, MA, USA: MIT Press, 1999.

[33] F. Ingrand and F. Py., "Online execution control checking for autonomous systems," *In Proceedings of the 7th International Conference on Intelligent Autonomous Systems*, pp. 273–280, 2002.

[34] C. Fuller and L. Vassie, *Health and safety management: principles and best practice*. Essex, UK: Pearson Education, 2004.

[35] J. Wozniak, V. Baggiolini, D. Q. Garcia, and J. Wenninger, "Software interlocks system," *Proceedings of ICALEPCS07*, pp. 403–405, 2007.

[36] M. B. Swarup and P. S. Ramaiah, "An approach to modeling software safety in safety-critical systems," *Journal of Computer Science*, vol. 5, pp. 311–322, April 2009.

[37] IEC 61508, *IEC 61508 : Functional safety of electrical/electronic/ programmable electronic safety-related systems*. Geneva, Switzerland: International Electrotechnical Commission, 2000.

[38] P. Fenelon, J. Mcdermid, M. Nicholson, and D. Pumfrey, "Towards integrated safety analysis and design," *Applied Computing Review*, vol. 2, no. 1, pp. 21–32, 1994.

[39] J. A. Mcdermid, M. Nicholson, and D. J. Pumfrey, "Experience with the application of hazop to computer-based systems," in *Compass '95: 10th Annual Conference on Computer Assurance*, (Gaithersburg, Maryland), pp. 37–48, National Institute of Standards and Technology, 1995.

[40] B. Graf, C. Parlitz, and M. Hägele, "Robotic home assistant care-o-bot 3 product vision and innovation platform," in *Proceedings of the 13th International Conference on Human-Computer Interaction. Part II: Novel Interaction Methods and Techniques*, (Heidelberg, Germany), pp. 312–320, Springer-Verlag, 2009.

[41] B. Rooks, "The harmonious robot," *Industrial Robot: An International Journal*, vol. 33, no. 2, pp. 125–130, 2006.

[42] S. Kendal and M. Creen, *An Introduction to Knowledge Engineering.* London, UK: Springer, 2007.

[43] A. G. Pipe, R. Vaidyanathan, C. Melhuish, P. Bremner, P. Robinson, R. A. J. Clark, A. Lenz, K. Eder, N. Hawes, Z. Ghahramani, M. Fraser, M. Mermehdi, P. Healey, and S. Skachek, "Affective robotics: Human motion and behavioural inspiration for cooperation between humans and assistive robots," in *Biomimetics: Nature-Based Innovation*, ch. 15, Taylor and Francis, 2011.

[44] ISO 13855, *ISO 13855:2010 : Safety of machinery – Positioning of safeguards with respect to the approach speeds of parts of the human body.* Geneva, Switzerland: International Organization for Standardization, 2010.

[45] R. Woodman, A. Winfield, C. Harper, and M. Fraser, "Safety control architecture for personal robots: Behavioural suppression with deliberative control," *The Seventh IARP Workshop on Technical Challenges for Dependable Robots in Human Environments*, 2010.

[46] O. Pourret, P. Naim, and B. Marcot, *Bayesian Networks: A Practical Guide to Applications (Statistics in Practice).* Chichester, UK: Wiley-Blackwell, 2008.

[47] A. Schörgendorfer and W. Elmenreich, "Extended confidence-weighted averaging in sensor fusion," *Proceedings of the Junior Scientist Conference*, pp. 67–68, 2006.

[48] N. Marzwell, S. Tso, K, and M. Hecht, "An integrated fault tolerant robotic control system for high reliability and safety," *Proceedings of Technology 2004*, November 1994.

[49] A. A. Hopgood, *Intelligent systems for engineers and scientists.* Florida, USA: CRC Press, 2nd ed., 2001.

[50] K. Goebel and A. Agogino, "An architecture for fuzzy sensor validation and fusion for vehicle following in automated highways," *29th International Symposium on Automotive Technology and Automation*, pp. 203–209, June 1996.

[51] M. Hossain, P. Atrey, and A. El Saddik, "Learning multi-sensor confidence using difference of opinions," *IEEE Instrumentation and Measurement Technology Conference Proceedings*, pp. 809–813, may 2008.

[52] C. Harper, M. Giannaccini, R. Woodman, S. Dogramadzi, T. Pipe, and A. Winfield, "Challenges for the hazard identification process of autonomous mobile robots," *4th Workshop on Human-Friendly Robotics (HFR 2011)*, 2011.

[53] A. Shepherd, *Hierarchical Task Analysis*. London, UK: Taylor & Francis, 2001.

[54] R. Svenningsson, J. Vinter, H. Eriksson, and M. Törngren, "Modifi: A model-implemented fault injection tool," *SAFECOMP 2010*, pp. 210–222, 2010.

[55] B. Lussier, M. Gallien, J. Guiochet, F. Ingrand, M.-O. Killijian, and D. Powell, "Planning with diversified models for fault-tolerant robots," *ICAPS'07*, pp. 216–223, 2007.