Preprints of the 5th IFAC Symposium on Mechatronic Systems
Marriott Boston Cambridge
Cambridge, MA, USA, September 13-15, 2010

MECHATRONICS2010-113

Safe-Guarded Agent Design Pattern for Mechatronic Systems

Dao Ba Phong, Theo J.A. de Vries and Job van Amerongen

Control Engineering, Faculty EEMCS, University of Twente,
P.O. Box 217, 7500 AE Enschede, The Netherlands. Tel: +31 53 4892606
{P.B.Dao , T.J.A.deVries , J.vanAmerongen} @utwente.nl

Abstract: To support the application of real-time Multi-Agent Control Systems (MACS) for mechatronic systems, a combination between the MACS design approach and OROCOS framework has been implemented: the OROCOS-based Implementation Framework for MACS (OROMACS). This paper presents our research results to make the OROMACS framework be easily applicable to develop real-time safe-guarded controller-agents and to maximize the reusability of safe-guarded MACS designs for various types of mechatronic systems. The approach that we advocate is a combination between OROMACS framework and pattern-based design method. Eleven control system design patterns are formed in which the Safe-Guarded Agent, one of the core design patterns, aims at providing a generic and flexible safe-guarded control solution for mechatronic systems. The design patterns are well organized into two reusable generalized safe-guarded control solutions, one for simple mechatronic systems and one for complex mechatronic systems.

Keywords: controller-agent, multi-agent control system, safe-guarded control, design pattern, intelligent control, polymorphic specifications, mechatronics.

1. INTRODUCTION

While designing a safe-guarded control system for a mechatronic system, the designer has to consider the following particular requirements (or specifications):

1. The control system should be designed to meet the functional requirements of a mechatronic system such as: multi-operation modes like start-up, homing, normal operation and shut-down; multiple functionality like detect targets, move to targets, follow a target and concurrently avoid obstacles on the way.
2. The control system not only should have a good control algorithm to meet the requirement of control performances (speed of response, bandwidth, stability, overshoot, sensitivity for disturbances and parameter variations), but also guarantees the safe-guarded control issue for both operator and machine.
3. The safe-guarded control issue has to cover and deal with a variety of fault sources with different criticality levels. It should also be functioned with the functions such as error detection, error handling, graceful degradation, and error recovery along with different degrees of fault tolerance.

Because of the specific design objectives and constraints, these requirements cannot be solved easily and satisfactorily by a single (traditional) control algorithm, i.e. a control system with only one controller such as a PID controller. Several practical solutions have been proposed such that complex requirements can be solved by using multiple models of computation, heterogeneous design techniques, and an integrated approach of multi-disciplinary, while taking into account multiple control objectives (van Breemen, 2001; Fregene et al., 2001; Masina et al., 2004). Therefore, realistic control requirements of mechatronic systems generally results in a multi-controller system that consists of a set of subcontrollers that are combined into an overall controller, such that when the multi-controller system is executed the overall performance specification is met. For each subcontroller, several conventional controller design techniques can be appropriately applied. This strategy of solving a complex control problem by decomposing it into partial control problems is called the divide-and-conquer approach (Johansen and Murray-Smith, 1997). In this area, the research of van Breemen (2001) contributed with a controller design framework for complex control problems named Multi-Agent Controller Implementation Framework (MACIF). Although van Breemen's design framework results in a clear structural overview of the overall control problem, and in a structured way to design and implement the multi-agent control system, it still gives room for improvement: *the design process has to be largely repeated whenever the designer moves to new applications or other mechatronic systems, even when these control problems resemble each other. The main reason is the lack of support for reusability of the design results of multi-agent control systems from previous projects to new projects*. In order to solve this problem, two research questions have been formulated along with strict requirements of safe-guarded control issues taken into account:

1. **How to make the multi-agent control system design approach be easily applicable to design safe-guarded controllers for various types of mechatronic systems?**
2. **How to maximize the reusability of safe-guarded MACS designs for mechatronic systems?**

Copyright © 2010 IFAC

While answering these questions, eleven control system design patterns are formed in which the Safe-Guarded Agent design pattern is presented here. Section 2 contains background information about Agents and Multi-Agent Systems, Multi-Agent Control Systems, OROMACS framework, and Design Patterns in Control Engineering. Section 3 describes the safety issue in mechatronic systems. Section 4 deals with DemoLin setup, a simple mechatronic system, with the approach starting with the design of a safe-guarded MACS for DemoLin and then formulating a reusable generalized solution for other simple mechatronic systems in terms of control system design patterns. Section 5 deals with TriPod setup, a complex mechatronic system, with the approach that supports the design of a safe-guarded MACS for TriPod by reusing the whole complete safe-guarded MACS design for DemoLin; and then a reusable generalized solution for other complex mechatronic systems in term of control system design patterns is formulated. Finally, some conclusions are given in section 6.

2. BACKGROUND

2.1 Agents and Multi-Agent Systems (MAS)

In Information Technology, an "agent" is regarded as an autonomous entity responsible for performing a certain task in coordination with its community. Because the term "agent" is used so frequently in various domains with different purposes, a unanimous precise and technical definition of the term "agent" cannot be formulated easily (Wooldridge, 1999). In spite of the lack of a technical definition, researchers try to come up with a notion of what an agent is, and is not, so as to be able to discuss their work with others based on the notion. A definition that captures the essential aspects of being an agent, what is approved by most researchers, is given in (Franklin & Graesser, 1996): "An autonomous agent is a system situated within and part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future."

Although an agent is presented as an entity that solves problems in order to achieve its own goal, many problems are far too complex to be handled by an individual agent. Only a "society of agents" is capable of solving such problems. In a more technical meaning, a society of agents is called a multi-agent system (MAS). A commonly approved definition of a MAS is given by Wooldridge (2002): "A Multi-Agent System consists of a number of autonomous, intelligent agents, which can interact with one another in order to pursue their own goals or cooperatively solve common problems".

At present, the research community of agent technology has pursued the objective to bring results of academic researches into practical applications. Pechoucek et al. (2006) pointed out that the available agent techniques have performed well in five types of application groups. And recently, a valuable review of industrial deployment of multi-agent technologies has been carried out by Pechoucek and Marik (2008) that gives an overview of the current applications and evolving trends of MAS in the industrial domain. As many numerous control applications based on agent technology have been successfully developed and implemented, a survey of multi-agent systems in control engineering is done by Daneshfar and Bevrani (2009). In the review, they present design methodologies, standards, tools, and supporting technologies to provide an effective MAS-based control design.

2.2 Multi-Agent Control Systems (MACS)

Because of the advantages of agents and MAS in solving complex problems, *agent* and *controller* have been combined to form a new concept named *controller-agent*. It brings the best of both fields together in the form of an *agent-based multi-controller system*. Pursuing this idea, van Breemen (2001) proposes a controller design framework for complex control problems named Multi-Agent Controller Implementation Framework (MACIF). This framework formulates a new design approach that is called Multi-Agent Control Systems (MACS). The MACS design approach results in control systems that have an open character, such that parts (or controller-agents) can be added, modified or removed without re-programming the operation of the remaining parts of the control system. In a MACS, a complex control problem is solved by a pool of controller-agents, in which each controller-agent is responsible for solving a part of the whole problem, thus providing a well-structured problem solving approach. As multiple controller-agents are acting on their own particular problems to solve the overall problem, conflicts between individual agents may arise, as these partial problems are interdependent. These conflicts are resolved by ***coordination mechanisms*** between controller-agents (CA); these mechanisms determine when and how activities of controller agents (i.e., calculation of control signals) are applied to the plant. There are *five coordinator types* mainly used in MACS which are:

1. ***Master-Slave*** (MS) is a subordination dependency in which the Slave-CA depends on the Master-CA, i.e. the Slace can be active only when the Master is active.
2. ***Fixed-Priority*** (FP): each CA is assigned a fixed priority during the operation of the MACS. The operation of CAs in a group is determined relying on their priority levels.
3. ***Parallel*** (P) makes all controller-agents within a group concurrently active.
4. ***Sequential*** (S) makes controller-agents within a group active in succession, for a single round only.
5. ***Cyclic*** (C) makes controller-agents within a group active in succession repeatedly.

2.3 OROMACS Framework

The MACS design approach and the OROCOS framework (Orocos, 2009) complement each other in two main aspects:

- MACS enables to create the hierarchically structured control systems that consist of coordinated elementary and composite controller-agents. In OROCOS, it is not possible to do this because the construction of a composite component is currently not supported.

Copyright © 2010 IFAC

- ❖ OROCOS framework supports a generic hard real-time kernel that can be used to build control applications with reliable determinism real-time behavior. Thus, it is an ideal complement for the MACS design approach.

The reciprocal complement makes a strong motivation for a combination between the MACS design approach and OROCOS framework. As a result, an ***ORO****COS-based Implementation Framework for* ***MACS*** (OROMACS) has been formed to support the development of real-time multi-threaded MACS in all phases from the design and simulation in computer to the realization and application in real model.

2.4 Design Patterns in Control Engineering

The design patterns technology that is widely used nowadays was inspired by the work of Christopher Alexander and colleagues. He is the first person that used what he called "a pattern language" in the architectural work to get such better design solutions. Alexander defines a pattern as "a three-part rule, which expresses a relation between a certain context, a problem, and a solution" (Alexander, 1979, p. 247). Through the literature review on design patterns, we understand that research and application of design patterns technology in control engineering domain is rather rare. It is clearly proved by the little number of relevant publication in journal and conference papers. The main reason is that pattern-based design approach is fairly new in the control theory and control engineering community. Another reason is the insufficient awareness of advantages of using design patterns. Because of that, some experience researchers on pattern-based control engineering point out main benefits to motivate the application of pattern-based design approach in the control engineering field. Sanz and Zalewski (2003) defines the pattern-based approach as "a method of generating solutions based on existing design knowledge". They stress that the pattern-based control engineering is not a control design method in the classic sense but a new way of managing and exploiting existing design knowledge for control systems, leading to better solutions. With the same judgement, Selic (1996) states that design patterns capture proven solutions, which, if applied intelligently, can result in significant benefits in terms of productivity and reliability. Zalewski, Selic and others believe that using this approach leads to control systems that are better designed, i.e. they are more modular, adaptable, understandable, and evolvable.

3. SAFETY ISSUE IN MECHATRONIC SYSTEMS

The safety issues in mechatronic systems (robots and manipulators) are always involved in two aspects: *safety for human or operator* and *safety for machines themselves*.

- The safety issue for human can be guaranteed by placing robots and manipulators in an area where people do not work closely to or directly with; and also by preventing people from entering the machine's working area. However, in some special cases people have to work closely to the machine (e.g. doing experimental research, testing or repairing of the system) and thus the safety issue for operator becomes more complex in these cases.
- Regarding the safety issue for machines themselves, the problem is generally very complicated because many possible sources of faults or errors have to be identified and handled strictly.

While performing a certain task, a robot or manipulator system usually has to cope with variously serious levels of different fault sources that can occur in an unwanted manner. If these faults are not identified correctly and handled strictly, they can bring dangerous situations for both human and machine. Because of that, we identify **18 fault sources** that are common in mechatronic systems. We also classify the fault sources into **three critical levels** those are ***dangerous, serious, and warning*** in which dangerous is the highest hazardous level, the next one is serious, and warning is the lowest hazardous level. *We assess the potential risk level of faults for human and machine to make this categorization*.

- ***8 fault sources at dangerous level***: faults at the highest critical level that can result in hazardous accidents for operator and critical damages for machine; therefore cut-off the power supply as fast as posible is the chosen solution to prevent faults from growing worse.
- ***6 fault sources at serious level***: faults at the critical level that only cause critical damages for machine; therefore, emergency-stop as fast as possible is the desired response to prevent worse damage for the machine.
- ***4 fault sources at warning level***: faults at the lowest critical level that don't cause much dangerous for operator and machine; thus normal stop is the reasonable solution.

Table 1. List of 18 fault sources that are common in mechatronic systems

Faults	Critical level
N1. Control computer get crashed totally	dangerous
N2. Failure of interface cards	dangerous
N3. Interconnecting wiring gets broken	dangerous
N4. Mechanical part is broken	dangerous
N5. Human collision	dangerous
N6. Human's unauthorized access into the working area	dangerous
N7. Failure of the power supply	dangerous
N8. Active emergency-stop	dangerous
N9. Exceeding end-effector working area	serious
N10. Exceeding joint working area	serious
N11. Failure of joint(s) or motor's transmission part(s)	serious
N12. The moving direction of motor(s) is intercepted by obstacle	serious
N13. Obstacle collision	serious
N14. Self-collision	serious
N15. Over-heating of the motor's armature coils	warning
N16. Failure of motor's power amplifier	warning
N17. Large tracking errors	warning
N18. Actuator overload	warning

Note that, we kindly don't want to discuss much about the list of fault sources and the categorization of critical levels

Copyright © 2010 IFAC

because these works are not the main content of the research and they vary according to personal opinion. To detect the fault sources, we propose a list of measures which is the combination of hardware- and software-based solutions. However, these measures are not presented in this paper because of space limitations.

4. DEMOLIN SETUP

4.1 Introduction

DemoLin is a simple mass-spring-mass system which was developed at Imotec B.V. (http://www.imotec.nl/) for the demonstration purpose of controller performances. It has a base plate (motor mass), which is driven by a linear motor. Another mass (end-effector mass) is connected on the top of the base plate with two flexible iron plates. Both the masses are attached to pretension belts and these belts are supported by pulleys mounted on two shafts that drive encoders. Plant model of DemoLin setup is the fourth-order and defined as a Flexible Mechanism of type AR (Coelingh, 2000). DemoLin can be considered as a simple single-axis electro-mechanical motion system; the 20-sim model is given in Fig. 1.

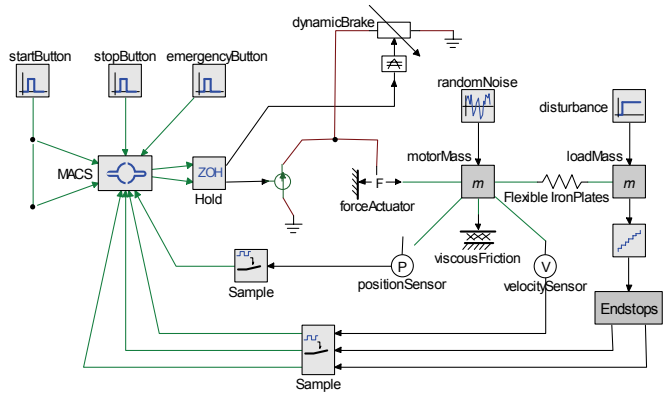

Fig. 1. 20-sim model of the DemoLin setup with the safe-guarded MACS.

4.2 Design the safe-guarded MACS for DemoLin

Before designing a safe-guarded control system for DemoLin setup, *three particular requirements* for the system are given:

1. The control system is needed to have multi-operation modes, which are: Startup (with two sub-operation modes are Initial and Homing), Normal Operation, Shutdown (with three sub-operation modes are Stop, Standby, and PowerOff), and Safe-Guarded in which the Safe-Guarded mode always has the highest operation priority to ensure the safety for the setup and human. Operation modes can have different priority levels (i.e. priority-based operation), control missions (e.g. accurate position control or safe-guarded control), control system configurations (e.g. simple or advanced controller), and motion trajectories (e.g. periodic or non-periodic path).
2. The control system is required to meet the desired control performances (speed of response, bandwidth, stability, overshoot, sensitivity for disturbances and parameter variations); and also to guarantee the safe-guarded control issue for both operator and machine.
3. The controller in each operation mode should perform its mission intelligently and autonomously. In case there is not any fault, the operation modes are normally active in the sequence: Initial, Homing, Normal Operation, Stop, Standby, and PowerOff. However, in case a certain fault occurs, the Safe-Guarded mode is immediately activated to handle the fault. The Safe-Guarded mode should be equipped with capabilities such as error detection, error handling, graceful degradation, and error recovery along with different degrees of fault tolerance. Then, depending on the potential critical level of the present fault, an appropriate safe-guarded activity could be applied.

A safe-guarded MACS is designed for DemoLin that fully meets the above-mentioned requirements (Fig. 2). We apply **a design procedure including four control system design patterns** that is described in a top-down approach hereafter.

Firstly, we use *the SingleApplication-Agent design pattern to initially generate the hierarchically structured safe-guarded MACS*. As a result, we obtain "MACS for DemoLin setup" that consists of a "Local Safe-Guarded Agent" and a "Multi-Operation Mode Agent", coordinated by a Fixed-Priority Coordinator in which the "Local Safe-Guarded Agent" has a higher priority level than the one of the "Multi-Operation Mode Agent".

Secondly, we use *the Local SafeGuarded-Agent design pattern to generate the hierarchical structure for the "Local Safe-Guarded Agent"*. As a result, the "Local Safe-Guarded Agent" consists of a "Local Dangerous Problems Handler", a "Local Serious Problems Handler", and a "Local Warning Problems Handler" that are coordinated by a Fixed-Priority Coordinator in which the "Local Dangerous Problems Handler" has the highest priority level; the "Local Serious Problems Handler" is the next one; and the "Local Warning Problems Handler" has the lowest priority level.

Thirdly, we use *the MultiFunction-Agent design pattern to generate the hierarchical structure for the "Multi-Operation Mode Agent"*. As a result, it consists of a "StartupMode Agent", a "NormalOperationMode Agent", and a "ShutdownMode Agent" that are coordinated by a Sequential Coordinator in which the "StartupMode Agent" is the first one to be active.

Finally, we use *the SingleFunction-Agent design pattern to generate the hierarchical structure for the three operation modes*. As a result, each operation mode consists of a "TrajectoryGenerator-Agent" and a "OperationController-Agent" that are coordinated by a Master-Slave Coordinator in which the "TrajectoryGenerator-Agent" is the Master and the "OperationController-Agent" is the Slave. Note that, in Fig. 2 this structure is not displayed; it is just presented in Fig. 6.

Here, we explain how the Local Safe-Guarded Agent handles a certain fault. According to Fig. 2, the Local Safe-Guarded Agent has the hierarchical structure with three different Local Problems Handlers, coordinated by a Fixed-Priority Coordinator, to deal with three critical levels of faults

Copyright © 2010 IFAC

(dangerous, serious, and warning). Each Local Problems Handler has a hierarchical structure as depicted in Fig. 3, in which ***the Errors Detection-Agent plays the role of the Master-controller-agent with the mission to detect faults and then to classify them into three critical levels; whereas the Single Safe-Guarded Activity-Agent keeps the role of the Slave-controller-agent with the mission to deal with graceful degradation and errors recovery issues***. Hence, it can be seen that whenever a certain fault occurs the Errors Detection-Agent will decide an appropriate Local Problems Handler to handle the fault.

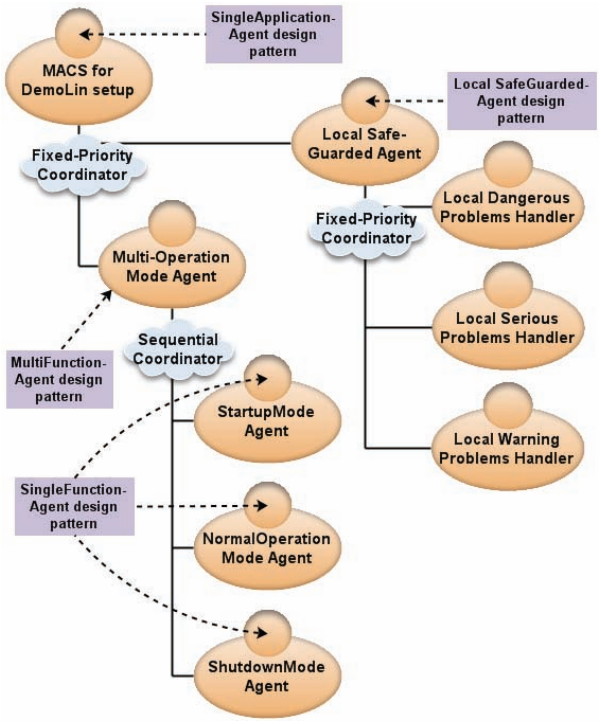

Fig. 2. Safe-guarded MACS for DemoLin.

Note that, only one Errors Detection-Agent element is really existing in all three Local Problems Handlers. That is because ***we reuse the Errors Detection-Agent in all Problems Handlers and use the polymorphic specification approach to make the Errors Detection-Agent structure able to hold multiple functionality.*** In our research, the Errors Detection-Agent is designed to have six polymorphic specifications (see Fig. 3 and Fig. 9). The same approach is applied for the Single Safe-Guarded Activity-Agent. The graceful degradation and errors recovery issues can be done flexibly through a plentiful set of polymorphic specifications that are provided by the Single Safe-Guarded Activity-Agent (Fig. 3). It means that, while designing a safe-guarded MACS, we have to choose a suitable polymorphic specification for each Single Safe-Guarded Activity-Agent of three Local Problems Handlers. For example, in Fig. 3 *the "Brake + PowerOff" specification* is chosen for the Single Safe-Guarded Activity-Agent of the Local Serious Problems Handlers.

To illustrate for the design procedure, *a test case is realized in a scheme that the end-effector hits against an end-limit switch caused by a wrong position reference*. After the Errors Detection-Agent identifies this fault as N10 (exceeding joint working area) and classifies it as a serious level (Table 1), the Local Serious Problems Handler is immediately activated to handle the fault. In this case, a graceful degradation issue is realized by the Single Safe-Guarded Activity-Agent with *the "Brake + PowerOff" specification* (Fig. 3).

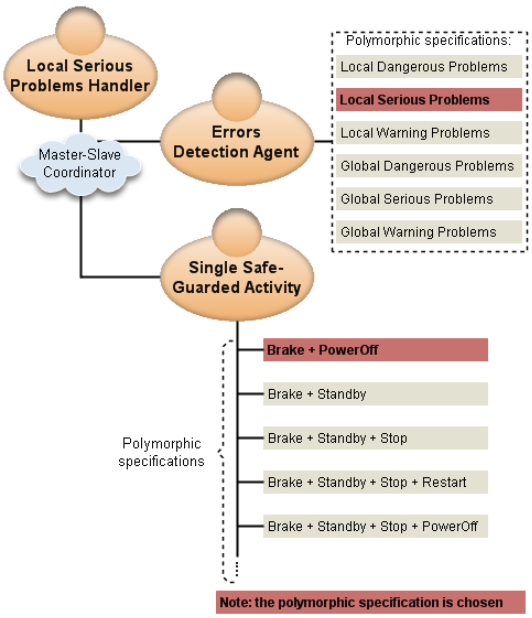

Fig. 3. Hierarchical structure of the Local Problems Handler.

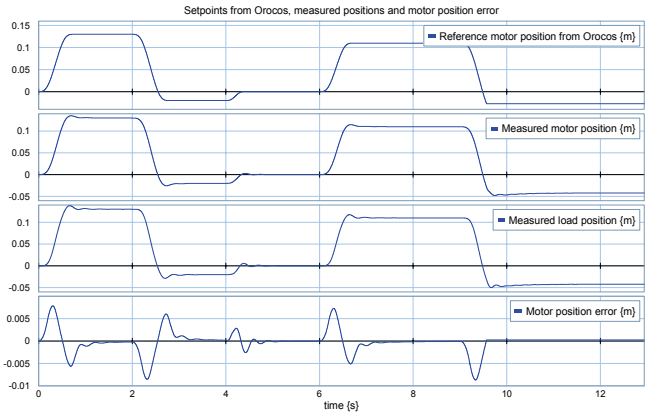

Fig. 4. Simulation result of which the safe-guarded MACS deals with the fault "exceeding joint working area".

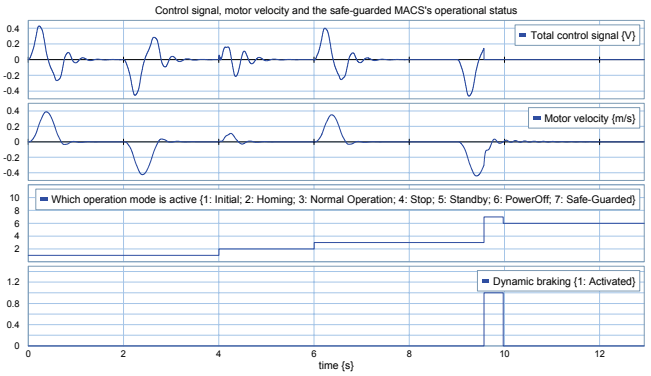

Fig. 5. Status transitions between operation modes of the safe-guarded MACS for DemoLin.

Copyright © 2010 IFAC

As a result, the sequence of operation modes is: from t = 0 [s] to 4 [s] is Initial mode; from t = 4 [s] to 6 [s] is Homing mode; from t = 6 [s] to 9.62 [s] is Operation mode; from t = 9.62 [s] to 9.87 [s] is Safe-guarded control mode with dynamic brake; from t = 9.87 [s] to end is PowerOff mode (Fig. 4 and Fig. 5). The experiment results on the real setup are given in (Phong and de Vries, 2010).

4.3 A generalized solution for simple mechatronic systems

Relying on the safe-guarded MACS design procedure for DemoLin setup, a reusable generalized solution (Fig. 6) for other simple mechatronic systems (single-axis manipulators or single-functionality motion systems, etc.) is formulated by the hierarchically structured application of four control system design patterns as follows:

Design pattern 1: **SingleApplication-Agent design pattern** consists of a Local SafeGuarded-Agent and a MultiFunction-Agent that are coordinated by a Fixed-Priority Coordinator in which the Local SafeGuarded-Agent always has a higher priority level than the one of the MultiFunction-Agent.

Design pattern 2: **Local SafeGuarded-Agent design pattern** consists of a Local Dangerous Problems Handler, a Local Serious Problems Handler, and a Local Warning Problems Handler that are coordinated by a Fixed-Priority Coordinator in which the Local Dangerous Problems Handler has the highest priority level; the Local Serious Problems Handler is the next one; and the Local Warning Problems Handler has the lowest priority level.

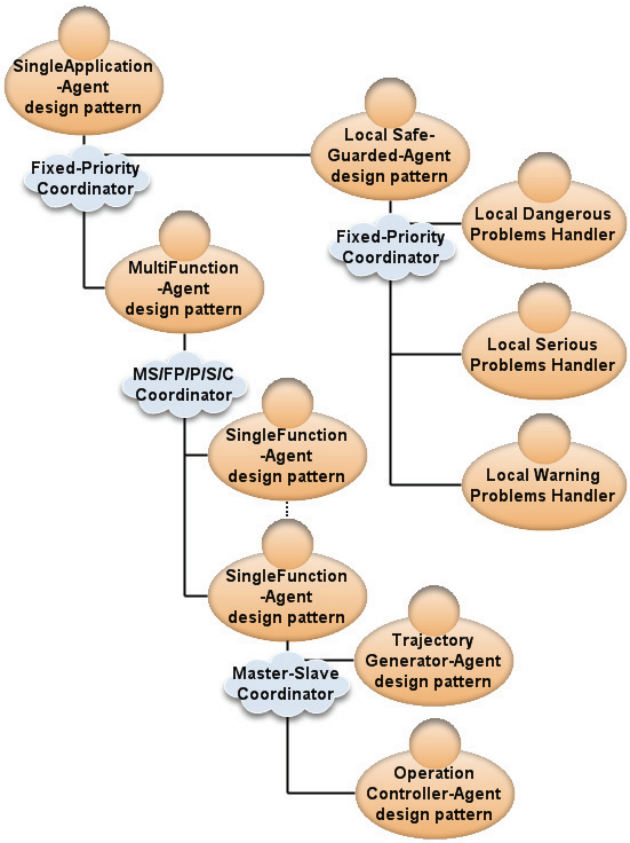

Fig. 6. Hierarchical structure of the reusable generalized solution for simple mechatronic systems.

Design pattern 3: **MultiFunction-Agent design pattern** is the composite controller-agent that consists of several SingleFunction-Agents, coordinated by one of five coordinator types: Master-Slave (MS), Fixed-Priority (FP), Parallel (P), Sequential (S), or Cyclic (C).

Design pattern 4: **SingleFunction-Agent design pattern** consists of a TrajectoryGenerator-Agent and a OperationController-Agent that are coordinated by a Master-Slave Coordinator in which the TrajectoryGenerator-Agent is the Master and the OperationController-Agent is the Slave.

5. TRIPOD SETUP

5.1 Introduction

Tripod is a pick-and-place machine which was also developed at Imotec B.V. (http://www.imotec.nl/) for testing different types of advanced controllers. It consists of three linear motors, which can move up and down within their safe operating regions. A pair of rods is connected to each linear motor, and the other side of these rods is connected to a platform at the top. Due to the constrained movement of the rods, the platform cannot rotate but only translate. The position of the platform is determined by the positions of the three linear motors. So we see that TriPod has three identical parts. Each part consists of one linear motor attached to the platform through a leg, thus forming a fourth order plant model which can be categorized as a Flexible Mechanism of type AR (Coelingh, 2000). As a result, each leg of TriPod has the same plant model as DemoLin. TriPod can be considered as a complex multi-axis electromechanical motion system (three axes) with the 20-sim model is given in Fig. 7. Moreover, because of its specific structure, TriPod setup has some special properties such as: variable load mass and variable springs due to the coupling between three axes that make the load forces variable; the strong coupling between the end-effector space and the joint spaces of three legs.

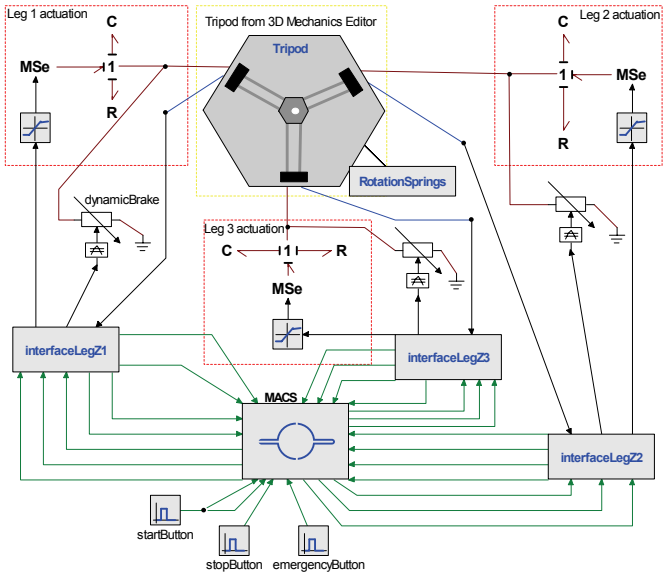

Fig. 7. 20-sim model of the TriPod setup and the safe-guarded MACS.

Copyright © 2010 IFAC

5.2 Design the safe-guarded MACS for TriPod

In general, the requirements of the safe-guarded control system for TriPod setup are almost the same as the ones for DemoLin setup. The reason is that the safe-guarded control problem of each axis of TriPod setup can be considered the same as the one of DemoLin setup. However, a new issue has come up in this case where the safe-guarded control problem for multi-axis operations related to all three axes of the TriPod setup should be taken into account. In order to distinguish this kind of safe-guarded control problem with other safe-guarded control issues for single-axis operations, we categorize the safe-guarded control problem for multi-axis operations in a new context, i.e. a ***Global influence sphere***. Hence, the requirements while designing a safe-guarded control system for TriPod setup consist of both Local and Global safe-guarded control issues. A safe-guarded MACS (Fig. 8), designed for TriPod setup to meet the particular requirements, which is based on the design procedure: **apply three control system design patterns; and then reuse the whole complete safe-guarded MACS design for DemoLin into the design for TriPod setup**. The design procedure is explained according to a top-down approach:

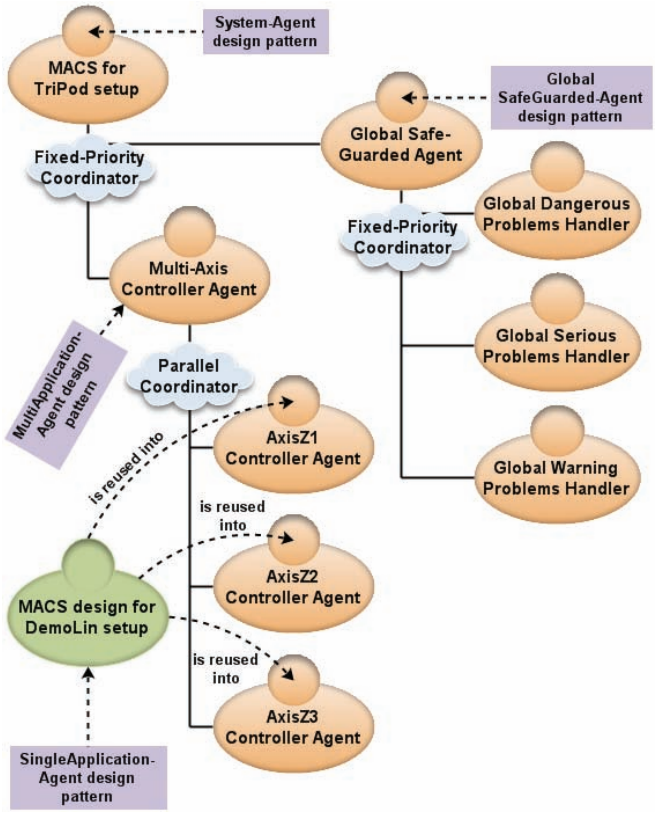

Fig. 8. Safe-guarded MACS for TriPod.

Firstly, we use *the System-Agent design pattern to initially generate the hierarchically structured safe-guarded MACS*. As a result, we obtain "MACS for TriPod setup" that consists of a "Global Safe-Guarded Agent" and a "Multi-Axis Controller Agent", coordinated by a Fixed-Priority Coordinator in which the "Global Safe-Guarded Agent" has a higher priority level than the one of the "Multi-Axis Controller Agent".

Secondly, we use *the Global SafeGuarded-Agent design pattern to generate the hierarchical structure for the "Global Safe-Guarded Agent"*. As a result, the "Global Safe-Guarded Agent" consists of a "Global Dangerous Problems Handler", a "Global Serious Problems Handler", and a "Global Warning Problems Handler" that are coordinated by a Fixed-Priority Coordinator in which the "Global Dangerous Problems Handler" has the highest priority level; the "Global Serious Problems Handler" is the next one; and the "Global Warning Problems Handler" has the lowest priority level.

Thirdly, we use *the MultiApplication-Agent design pattern to generate the hierarchical structure for the "Multi-Axis Controller Agent"*. As a result, the "Multi-Axis Controller Agent" comprises a "AxisZ1 Controller Agent", a "AxisZ2 Controller Agent", and a "AxisZ3 Controller Agent" that are coordinated by a Parallel Coordinator.

Finally, we *reuse the whole designed safe-guarded MACS for DemoLin into each axis of TriPod*. It means: AxisZ1 Controller Agent, AxisZ2 Controller Agent, and AxisZ3 Controller Agent use the safe-guarded MACS with the same hierarchical structure as DemoLin setup. ***The only thing that remains to be done is to modify application-specific settings*** (e.g. error bound, controller parameters, polymorphic specifications, etc.).

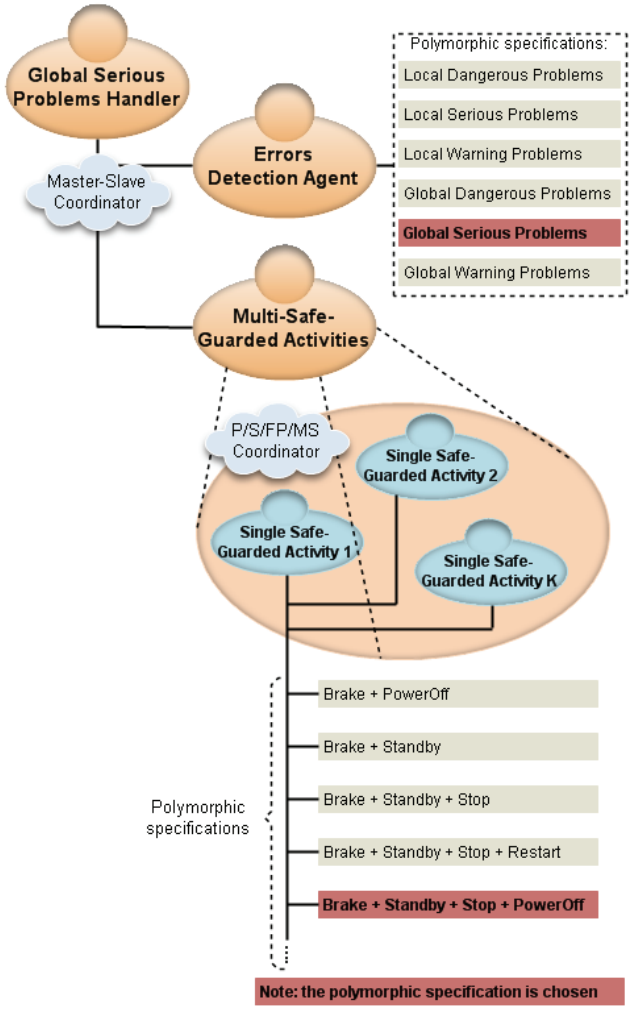

Fig. 9. Hierarchical structure of the Global Problems Handler.

Copyright © 2010 IFAC

Here, we make clear how the Global Safe-Guarded Agent handles a certain fault. By comparison of Fig. 2 and Fig. 8, we see that the Local Safe-Guarded Agent and the Global Safe-Guarded Agent have the same hierarchical structure. *The difference is only between the Local Problems Handlers (Fig. 3) and the Global Problems Handlers (Fig. 9) that is the Global Problems Handlers use the Multi-Safe-Guarded Activities-Agent, instead of the Single Safe-Guarded Activity-Agent.* It can be seen that a Multi-Safe-Guarded Activities-Agent consists of a pool of Single Safe-Guarded Activity-Agents which are coordinated by one of four coordinator types: Parallel (P), Sequential (S), Fixed-Priority (FP), and Master-Slave (MS). The different types of coordinators allows designers to flexibly choose an appropriate safe-guarded control strategy for each mechatronic application. Single Safe-Guarded Activity-Agents of a Multi-Safe-Guarded Activities-Agent can use the same or different polymorphic specification. For example, in Fig. 9 *the "Brake + Standby + Stop + PowerOff" specification* is used for all Single Safe-Guarded Activity-Agents.

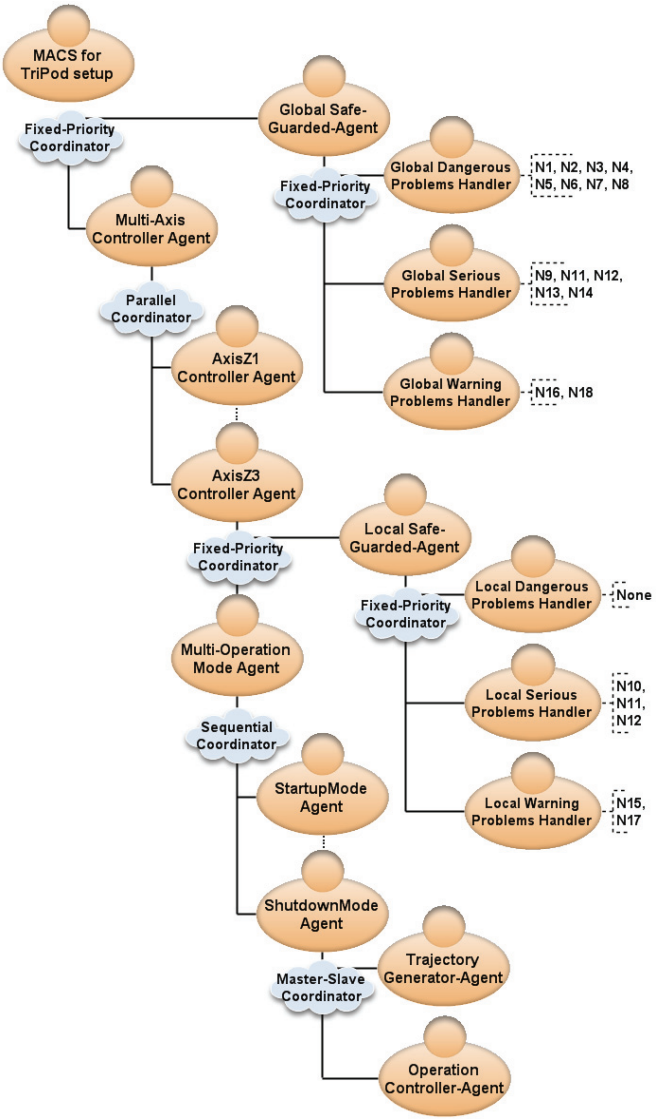

Fig. 10. Overall hierarchical structure of the safe-guarded MACS for TriPod.

It is noticed that the Global Safe-Guarded Agent always has a higher priority level than the Local Safe-Guarded Agent. It means the Global Problems Handlers have right to take over the active authority from the Local Problems Handlers. Hence, the designer must be very careful while designing the Global Problems Handlers because it can cause problem by taking over the active authority from a Local Problems Handler which maybe is handling a critical fault. To avoid this problem, we suggested a temporary solution that is design the Global Safe-Guarded Agent with only the Global Dangerous Problems Handler. This solution can work well because the faults with Global influence sphere occuring in practical applications generally are problems related to the operation of multiple manipulators, robots, or production stations, which are normally considered as dangerous problems. However, this problem has been fully studied to bring out a better solution.

Because both Local Safe-Guarded Agent and Global Safe-Guarded Agent present in the safe-guarded MACS design for TriPod setup, the safe-guarded control missions have to be appropriately assigned to six Problems Handlers. In Fig. 10, we map the list of 18 common fault sources of mechatronic systems (Table 1) into three Local Problems Handlers of the Local Safe-Guarded Agent and three Global Problems Handlers of the Global Safe-Guarded Agent. The map is based on the influence sphere of faults (Local or Global) and critical level of faults (Dangerous, Serious, or Warning). As the Errors Detection-Agent plays the role of the Master-controller-agent with the mission to detect and classify faults, this map is done through programming six polymorphic specifications of the Errors Detection-Agent. *When moving to a new application, the map will be different.*

To illustrate for the design procedure, *a test case is realized in a scheme that* ***the 1st linear motor*** *works with a wrong trajectory (it goes down too much and hits against the lower end-limit switch).* After the Errors Detection-Agent identifies this fault as N10 (exceeding joint working area) occuring on the 1st linear motor and classifies it as a serious level (see Table 1), the Local Serious Problems Handler of the AxisZ1 Controller Agent is immediately activated to handle the fault. In this case, a graceful degradation issue is realized by the Single Safe-Guarded Activity-Agent with *the "Brake + PowerOff" specification* (Fig. 3).

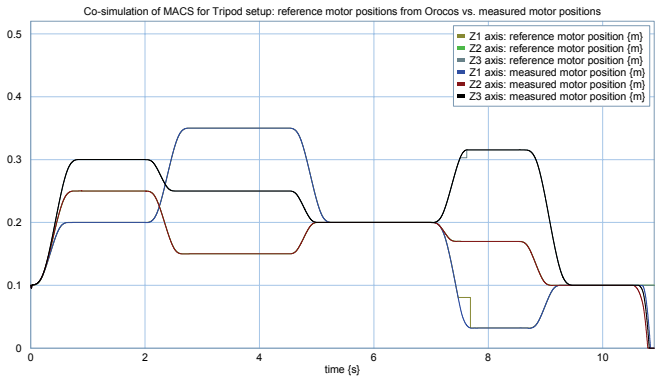

Fig. 11. Simulation result of which the safe-guarded MACS deals with two consecutive faults (N10 and N9).

Copyright © 2010 IFAC

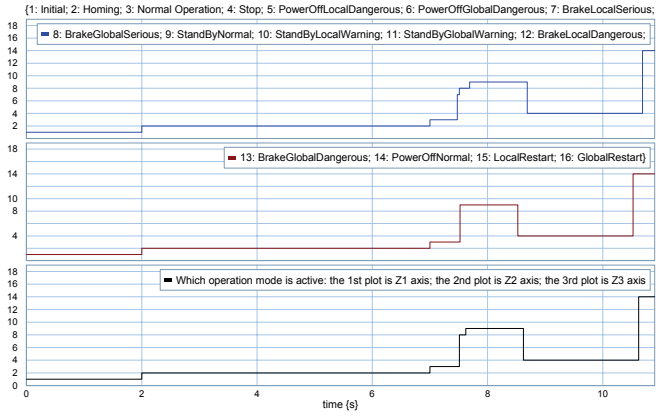

Fig. 12. Status transitions between operation modes of the safe-guarded MACS for TriPod.

However, *the fault N10 of the first axis indirectly causes another fault for TriPod; it makes the end-effector move out of the safe working area*. The Errors Detection-Agent identifies this fault as N9 (exceeding end-effector working area) and classifies it as a serious level (Table 1). Moreover, the Errors Detection-Agent has been already programmed to "understand" this kind of fault has a ***Global influence sphere***. It means that "exceeding end-effector working area" must involve all three axes of TriPod setup. Therefore, the Global Serious Problems Handler is immediately activated to handle the fault. A graceful degradation issue is realized by the Multi-Safe-Guarded Activities-Agent with *the "Brake + Standby + Stop + PowerOff" specification* is chosen for all Single Safe-Guarded Activity-Agents of three axes of TriPod setup. In Fig. 12, we can see the transition from Local Safe-Guarded Agent to Global Safe-Guarded Agent on the AxisZ1 Controller Agent, i.e. from "BrakeLocalSerious" with number 7 to "BrakeGlobalSerious" with number 8.

5.3 A generalized solution for complex mechatronic systems

Relying on the safe-guarded MACS design procedure for TriPod setup, a reusable generalized solution (Fig. 13) for complex mechatronic systems (such as multi-robot or multi-manipulator stations with multiple functionality and/or multi-operation modes, etc.) is formulated based on two things:

1. The hierarchically structured application of three control system design patterns:

Design pattern 1: **System-Agent design pattern** consists of a Global SafeGuarded-Agent and a MultiApplication-Agent that are coordinated by a Fixed-Priority Coordinator in which the Global SafeGuarded-Agent always has a higher priority level than the one of the MultiApplication-Agent.

Design pattern 2: **Global SafeGuarded-Agent design pattern** consists of a Global Dangerous Problems Handler, a Global Serious Problems Handler, and a Global Warning Problems Handler that are coordinated by a Fixed-Priority Coordinator in which the Global Dangerous Problems Handler has the highest priority level; the Global Serious Problems Handler is the next one; and the Global Warning Problems Handler has the lowest priority level.

Design pattern 3: **MultiApplication-Agent design pattern** is the composite controller-agent that consists of several SingleApplication-Agents, coordinated by one of five coordinator types: Master-Slave (MS), Fixed-Priority (FP), Parallel (P), Sequential (S), or Cyclic (C).

2. Reuse the whole complete safe-guarded MACS design for simple mechatronic systems, which are based on *the reusable generalized solution for simple mechatronic systems* (section 4.3), into the current safe-guarded MACS design for a new complex mechatronic system.

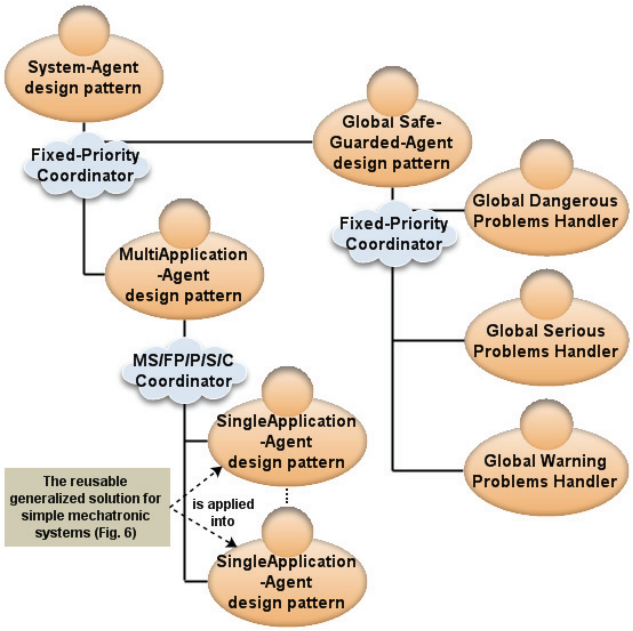

Fig. 13. Hierarchical structure of the reusable generalized solution for complex mechatronic systems.

6. CONCLUSIONS

The control system design patterns based on the polymorphic specifications approach form a highly applicable and reusable **design patterns based safe-guarded MACS design approach** that answers well to the proposed research questions. Moreover, through an analysis and design process of the safe-guarded MACS for DemoLin and TriPod setup, two reusable generalized safe-guarded control solutions, one for simple mechatronic systems and one for complex mechatronic systems, are well defined. With support of this design approach, two advantages can be achieved while designing the safe-guarded MACS for mechatronic systems, thus shorten the control system development time.

1. The approach ***enables to quickly generate the hierarchically structured safe-guarded MACS for mechatronic systems with various complex levels***.
2. The approach ***maximizes the reusability at three levels***:
   - *Reuse coding parts containing control algorithms* of a controller-agent into another controller-agent.
   - *Reuse controller-agents* of a safe-guarded MACS design into another safe-guarded MACS design.

Copyright © 2010 IFAC

- *Reuse the whole complete safe-guarded MACS design* for a simple mechatronic system (e.g. DemoLin) into another safe-guarded MACS design for a complex mechatronic system (e.g. TriPod).

The research makes the MACS design approach be easily applicable in practice and achieves a good reusability that can significantly reduce the control system development time for mechatronic systems. However, we have been faced with a conflicting problem between Global and Local Safe-Guarded Agent. This problem will be solved in our future research.

REFERENCES

Wooldridge, M. (1999). Intelligent agents. *Multiagent Systems,* pp. 27-77. The MIT Press, Cambridge, Massachusetts.

Franklin, S. and Graesser, A. (1996). Is it an agent, or just a program?. *In the 3rd International Workshop on Agent Theories, Architectures and Languages,* pp. 193-206.

Wooldridge, M. (2002). An Introduction to Multiagent Systems. John Wiley & Son Ltd., Chichester, England.

Alexander, C. (1979). The Timeless Way of Building. Oxford Univ. Press, New York.

Sanz, R. and Zalewski, J. (2003). Pattern-Based Control Systems Engineering. *IEEE Control Systems,* vol. 23(3), pp. 43-60.

Selic, B. (1996). An Architectural Pattern for Real-Time Control Software. *In Proc. PLoP'96 3rd Annual, Pattern Languages of Programming Conf.*, Monticello, IL.

Johansen, T.A. and Murray-Smith, R. (1997). The Operating Regime Approach to Nonlinear Modelling and Control. Taylor & Francis.

van Breemen, A.J.N. (2001). An Agent-Based Multi-Controller Systems: A design framework for complex control problems. *PhD thesis*, 238 pages, University of Twente, Enschede, The Netherlands. ISBN 9036515955.

Pechoucek, M., Thompson, S., Baxter, J., Horn, G., Kok, K., Warmer, C., Kamphuis, R., Marík, V., Vrba, P., Hall, K., Maturana, F., Dorer, K., and Calisti, M. (2006). Agents in Industry: The Best from the AAMAS 2005 Industry Track. *IEEE Intelligent Systems,* Vol. 21, No. 2, pp.86-95, ISSN: 1541-1672.

Pechoucek, M. and Marik, V. (2008). Industrial Deployment of Multi-Agent Technologies: Review and Selected Case Studies. *International Journal on Autonomous Agents and Multi-Agent Systems,* ISSN: 1387-2532.

Daneshfar, F. and Bevrani, H. (2009). Multi-Agent Systems in Control Engineering: A Survey. *Journal of Control Science and Engineering*, Volume: 2009. ISSN: 1687-5249. DOI: 10.1155/2009/531080.

de Vries, T.J.A. (1994). Conceptual Design of Controlled Electro-Mechanical Systems: A modeling perspective. *PhD thesis*, 169 pages, University of Twente, Enschede, The Netherlands. ISBN 90-900-6876-7.

Coelingh, H. J. (2000). Design Support for Motion Control Systems. *PhD thesis*, 218 pages, University of Twente, Enschede, The Netherlands. ISBN 90-365-1411-8.

Fregene, K., Kennedy, D.C., and Wang, D.W.L. (2001). HICA: A Framework for Distributed Multiagent Control. *In Proc. IASTED Int. Conf. Intelligent System Control*, pp. 187-192.

Masina, S., Lee, K.Y., and Garduno-Ramirez, R. (2004). An Architecture of Multi Agent System Applied to Fossil Power-Fuel Power Unit. *In Proc. of the IEEE Power Engineering Society General Meeting*, Denver, CO.

Phong, D.B. and de Vries, T.J.A. (2010). MACS for DemoLin setup. *Technical report*, Control Laboratory, University of Twente, Enschede, The Netherlands. To be submitted.

OROCOS (2009). http://www.orocos.org.

Controllab Products, 20-sim (2009). http://www.20sim.com.

Copyright © 2010 IFAC