RTRobMultiAxisControl: A framework for real-time multi-axis and multi-robot control

H. Fischer, Margot Vulliez, P. Laguillaumie, P. Vulliez, J.-P. Gazeau

RTRobMultiAxisControl: A framework for real-time multi-axis and multi-robot control

H. Fischer, M. Vulliez, P. Laguillaumie, P. Vulliez, J.P. Gazeau
Institut Pprime, dept. GMSC
Université de Poitiers
Poitiers-Futuroscope, France

***Abstract*—This paper presents the challenges met in the development of a new framework for multi-axis and multi-robot control. The increasing demand for multi-robot collaboration and robotic assistance, both in industry and service applications, has raised the level of interactions between robots and humans in a shared environment with real-time constraints. Some scientific and technological issues need to be unlocked to ensure safe human-robot interactions: to guarantee the response time during the robot perception-action process, to cope with dynamic interactions with the robot environment, to secure the collaborations between several machines and humans, and to improve the integration of the robots at home and in open zones of production lines. The specifications of a new hardware and software framework are set with respect to these observations. The framework will meet complex research and industrial issues for the future of multi-axis and multi-robot control. Before introducing our approach, a brief survey of existing frameworks and robotic middleware is given.**

The foundations of our approach are based on the following requirements: the framework must be real-time, transferable, maintainable and multi-manufacturer. The framework design has also to guarantee the robustness of the machine interactions in a dynamic and collaborative environment. To evaluate the feasibility of our design strategy and assess its performance, we have developed mechatronic devices with a high level of human-machine collaboration. This paper outlines two robotic applications which require multi-robot real-time synchronization and are based on the proposed framework. A temporal analysis demonstrates its robustness for multi-axis and multi-robot control.

Note to Practitioners: This paper was motivated by the problem of proposing a transferable framework dedicated to real time multi-axis and multi-robot control. Most of existing middleware solutions do not unify accesses to the axis level and to the robot control level with a common programming approach and with real time capability. This is a key problem for achieving multi-robot synchronization and reaction times compatible with dynamic collaboration. Our framework is based on open robotics and object-oriented programming and implements the industrial standards for motion control. Multi-robots control experiments with a high number of synchronized axis and heterogeneous hardware demonstrate the efficiency of the approach.

Keywords — Multi-axis Control, Multi-robot Control, Collaborative Robotics, Industrial Ethernet, Real-time, Open Robotics, Framework, Middleware.

This work has been sponsored by the French government research program "Investissements d'Avenir" through the Robotex Equipment of Excellence (ANR-10-EQPX-44). It is also supported by the Nouvelle Aquitaine Region (program "CPER Numeric"), in partnership with the European Union (FEDER/ERDF, European Regional Development Fund) and French National Research Agency (ANR) through the SEAHAND program (ANR-15-CE10-0004).

I. INTRODUCTION

Manufacturing uses more and more lean and agile solutions to adapt to changes in the marketplace. Innovative technologies must be proposed to meet the demands on production. Traditional automation solutions do not offer such capability in flexibility, and in customizing the products on demand.

During the last four decades, industrial robots have gradually replaced human workers for tedious and routine jobs or in harmful environments which may cause industrial accidents, diseases or MSDs. In this substitution process, although intrinsically flexible, the industrial robot was effectively specialized, working alone in a closed cell, and could only meet one kind of preset task. The robot was tuned for painting, for arc welding, for spot welding, or for grasping one specific product and the end-effector was specially designed with respect to the target task.

Ten years ago, a new type of robot appeared: the collaborative robot (Fig. 1). The collaborative robot aims at assisting the worker. This new interaction feature between human and machine offers many perspectives in working environments and especially in SMEs, as discussed in [1]. It introduces a new way of organizing the production lines by associating, in the same workspace, the strength, precision and robustness of the robot and the decision making, problem solving capabilities, and expertise of the human being.

Iiwa (Kuka)	YuMi (ABB)
Sawyer (Rethink Robotics)	UR3 (Universal Robot)

Fig. 1. Industrial collaborative robot (with courtesy of Kuka, ABB, Rethink Robotics and Universal Robot)

The emergence of collaborative robotics does not mean the end of industrial robotics. Both types of robots will coexist and work together in order to address the issues of flexibility and customization in the production flow.

The need for flexibility in the industry of the future requires new software solutions that will enable the development of robotic applications with real-time multi-robot control in a dynamic environment. As these applications may involve industrial robots, collaborative robots and humans in a shared working zone, it is a major challenge to propose a common framework meeting new industry issues.

Such a challenge can only be addressed if all the mechatronic components in the production process are able to communicate and exchange data with a dynamically consistent real-time scheduling. Real-time communication and operating systems are needed to ensure the determinism of the application and secure the interactions between robots and humans in a shared environment.

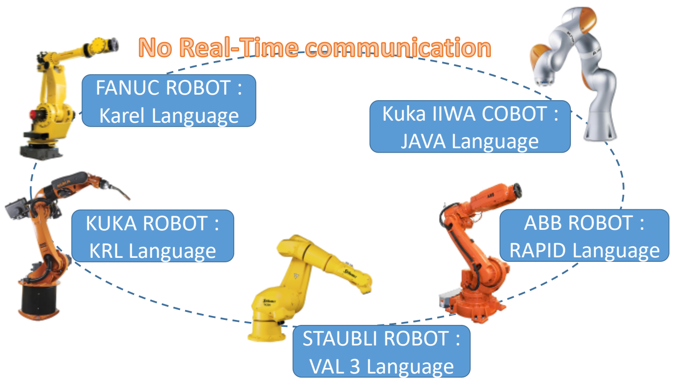

No Real-Time communication
FANUC ROBOT : Karel Language
Kuka IIWA COBOT : JAVA Language
KUKA ROBOT : KRL Language
ABB ROBOT : RAPID Language
STAUBLI ROBOT : VAL 3 Language

Fig. 2. Industrial robots and associated programming languages

Current industrial robots from most manufacturers do not meet these issues; Figure 2 shows that most of them are limited to their own universe; they can only be programmed in the manufacturer language (Kuka uses KRL, Fanuc, KAREL, Stäubli, VAL3 …).

The software components used for programming do not enable reusability and sharing of know-how between robots. It becomes impossible with such an approach to synchronize several robots from different manufacturers in a deterministic scheme in the same robotic cell. Real-time communication and a traversal programming approach are essential to meet flexibility requirements for the development of multi-axis and multi-robot applications.

Hence, this way of working is incompatible with the future of industry. All robots have to communicate and interact with their environment, with other robots and with humans in a dynamic way, under robustness, reactivity and reliability constraints. A common framework is proposed in this paper to meet these requirements and ensure the flexibility of the robotic cell. Our framework is based on open robotics and unifies hardware and software components by using industrial standards and object-oriented programming.

After a brief survey of robotic middleware solutions, we will compare communication performances of ROS and hard real-time Operating System environments in the objective of multi-robot application. From this analysis, we will then draw requirements to design an efficient robotic framework. In the fourth section, the RTRobMultiAxisControl framework is presented. Its different components are described according to open robotic and real-time standards. Sections V and VI will present the multi-robot platforms used to validate the design of the robotic framework and assess the inter-component communication efficiency.

II. Overview of Robotic Middleware Frameworks

A difficulty that rises when designing a framework is to manage the hardware heterogeneity while willing to simplify software design by using common rules in the development of a robotic application. Several robotic middleware solutions have been developed worldwide for that purpose such as OPRos, Orca, MIRO, Player, Marie, Aseba, CLARAty, UPNP, RSCA, MRDS, SmartSoft, Skilligent, Webots, Irobotaware, Pyro, Carmen, and RoboFrame.

As discussed in [2] and [3], a middleware is defined as a layer of software between the operating system and the application level, which provides a common programming abstraction across a distributed system. In [2, 5, 6, 7], surveys of current robotic middleware frameworks are proposed. Attributes, focusing on the architecture, simulation environment, standards and technologies, security for accessing modules, real-time and coordination capabilities, open-source are discussed. Most of the existing robotic middleware are detailed. We chose to present some of them with regards to our multi robot control purpose.

The architecture of the middleware is the organization of the layers, components, libraries, or modules required for the middleware implementation. For example, MIRO middleware for mobile robotics is based on three layers [8]: a device layer that is platform dependent, a service layer with service abstractions for sensors and actuators, a framework layer that contains modules like navigation, self-localization, path planning, and visualization. Player/Stage system [9] is also a middleware for mobile robots. It is composed of two components; the first one, Player, is a device repository for robots, sensors and actuators. The second component, Stage is only for the graphical simulator which models devices. ERSP [10] consists of three layers: a hardware abstraction layer, a behavior execution layer and a task execution layer. Orocos [11] (Open Robot Controls Software) is a Real-Time Toolkit with an infrastructure and functionalities to develop C++ applications. It consists of the three following components: the Orocos Components Library (OCL) that offers some ready-to-use control components, the Orocos Kinematics and Dynamics Library (KDL) that provides real-time calculation of kinematic chains, and the Orocos Bayesian Filtering Library (BFL), with tools such as Kalman Filters and Particle Filters. Another middleware named Orca grew out from Orocos at KTH Stockholm in 2003. Orca developers have replaced CORBA, a communication middleware [12], developed since 1991 and used in Orocos with a modern framework named ICE (Internet Communication Engine). RT-Middleware [13], or Robotics Technology Middleware allows the designers to build quickly and efficiently their own robotic product. Each component is called a RT-Component and they can be easily connected to each other. A state machine rules the state of a component, however RT-middleware does not provide a hard-real-time support. ROS middleware [14] is a "metaoperating system" for developing robotics. ROS is probably the most used robotic middleware and proposes many packages able to integrate the widest library of devices. ROS consists of nodes (software modules) which communicate over topics (passed peer to peer) sending messages. For synchronous communication Services are used. New packages for ROS were recently proposed for control: meta-ROS [15] and ROS Control [16]; but they do not offer hard real time support for multi-axis or multi-robot control. A new version of ROS, named ROS 2 is currently under heavy development [17] for that purpose. This new version is not compatible with the previous version as it was necessary to review the foundations of ROS to allow hard real-time functionalities.

This survey underlines the fact that all middleware solutions relies on distinct architectures. These middleware are mostly based on the three standard communication middleware as shown in table 1: MOM, RPC or CORBA [18]. The components, layers, libraries or modules available are quite different, do not target the same implementations (mobile robotics, artificial intelligence, robot and machine control…), use different programming approaches, and most of them do not offer real-time capabilities or real-time communication. UDP or TCP/IP communication using sockets is often used, inducing temporal indeterminism in the communication flow.

The table 1 highlights different issues of the existing middleware in the field of robotics. These concerns constitute the requirements of an optimal framework. Multi-axis and multi-robot control requires a high level of synchronization and temporal determinism in a dynamic environment. Hence the robotic software framework should provide real-time capability. Besides, the programming approach should be based on industrial standards, which would bring robustness, reusability and support for most of hardware components used in industrial motion control.

Most of these middleware solutions are also far from the controller base level, and do not offer solutions able to unify the access to the axis control. This is a key problem for achieving multi-robot synchronization with reaction times compatible with dynamic collaboration schemes. The next section analyses the temporal behavior of one of the most popular middleware used for the development of robotic applications: ROS.

III. Evaluation of ROS Middleware Communication for Multi-Robot Control

ROS is a widely known robotic middleware which gives access to many robots, sensors and other components from many manufacturers.

ROS is usually implemented on standard PCs with non-Real-Time Operating Systems (RTOS) in many robotic applications. It seemed relevant to us to highlight the limits of this architecture. The tested version is ROS INDIGO; the software is running on the Linux Ubuntu 14.04 operating system on a Intel Quad Core i7 PC (CPU M620 2.7GHz, 4Go RAM).

To evaluate the feasibility of dynamic multi-robot control with ROS we have designed an experimental set-up featuring two modern robots and high-speed Ethernet-based networks.

A high level of coordination and synchronization between several axes or several robots is required to ensure temporal determinism and robustness in a multi-robot cell.

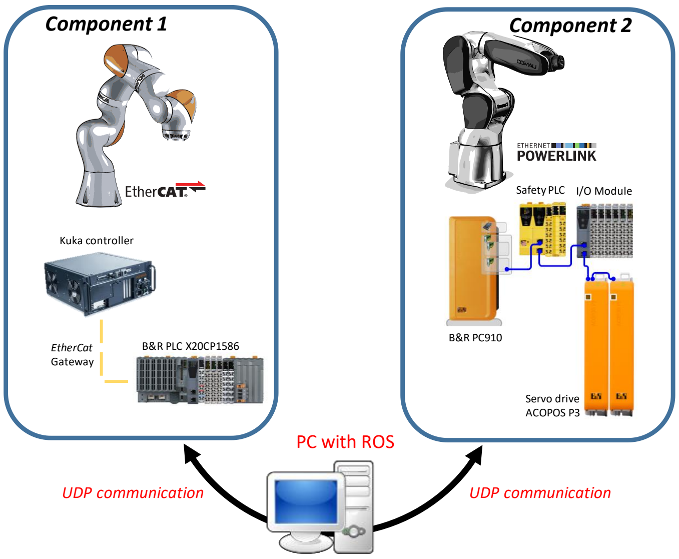


Fig. 3. Architecture used for the evaluation of ROS communication

For our evaluation purpose, we used the following devices (Fig. 3):

- Component 1: an industrial PC (B&R Automation PC910) that controls a Comau Racer 3 robot and its peripherals (gripper and sensors) over POWERLINK fieldbus;
- Component 2: an industrial PLC (B&R Automation X20CP1586) that controls a Kuka iiwa robot equipped with EtherCAT fieldbus.

The objective of the test is to connect these two complex components with the ROS middleware in order for them to share their respective states at a frequency suitable for cooperation. The data exchanged between the two robots are needed for dynamic collaboration purpose.

A. *Evaluation of ROS middleware communication*

ROS manages the cell components with "nodes" and "topics" objects. Nodes are processes that perform computations to achieve tasks (e.g. read sensors, control motors, localize or plan paths), while topics are used to identify the contents of messages. The messages are routed via a publish/subscribe transport system. A node sends out a message by publishing it to a topic, and another node reads it by subscribing to that topic.

As we wish to evaluate the communication between two components, one node is defined per component. All nodes are used for communication, one node attends to data emission (component 1) and the other for data reception (component 2) by using streaming topics.

Table 1: Characteristics of the main robotics middleware

Middleware	Communication Technology: framework used	Hard Real time support	Motion Control Standard	OS	Installation & development	Layers or modules
RT-Middleware	CORBA	No	No	Linux Windows	Needs loader *RTC-Daemon* for RTC components management C++ / Java / Python programming	1)low level layer: motors, sensors 2) middle level layer: vision and image processing 3)high level layer: robot modeling
Orocos	CORBA	Yes, by using Xenomai	No	Linux Windows Mac OS X	Orogen (language for components management) Needs *RTT* (Real Time Toolkit) for components management by using scripts language) Four C++ libraries	1)Components Library 2)Kinematics and Dynamics Library 3)Bayesian Filtering Library
ROS	MOM RPC	No	No	Linux Windows (Cygwin)	Needs loader *ROS-Run* for nodes and topics management C++ / Java / Python programming	12 versions, 7 unmaintained versions, 1600 packages, modules (robots, peripherals, sensors, motion planning, …)
ROS2.0	MOM RPC	Yes, by using Xenomai	No	Linux Windows Mac OS X	Needs loader *ROS-Run* for nodes and topics management C++ / Java / Python programming	Small number of packages

To measure the communication consistency, we chose a basic application exchanging one integer data: component 2 needs to know continuously the state of a sensor associated to component 1. Component 1 sends periodically sensor data (blue dots in Figures 4 and 5) to component 2 through ROS nodes and topics. Component 2 receives the sensor data (red dots in Figures 4 and 5) and the transfer time is evaluated.

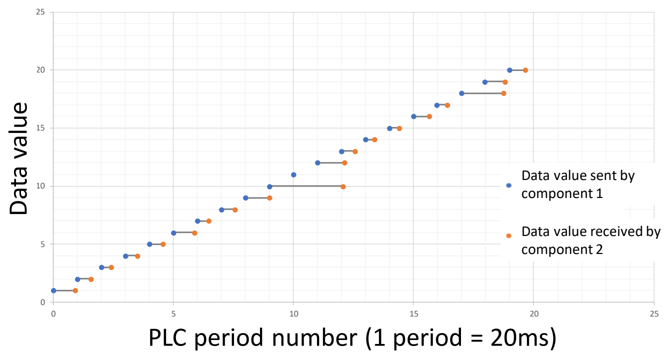


Fig. 4. Data received with a ROS-based architecture with a 20ms sending period

Figures 4 and 5 illustrate the fact that the communication time between the two components is not deterministic whatever the period is. And we also may lose data if the exchange frequency is high. As an example, for a 20ms period data value number 11 was lost (Fig. 4); for a 2ms period, 50% of the sensor values were lost during communication. The loss of data and the lack of determinism may be a cause of failure in a robotic application with response-time constraints.

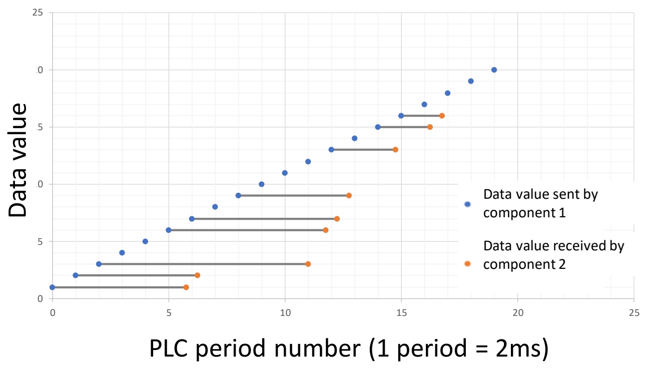


Fig. 5. Data received with a ROS-based architecture with a 2ms sending period

This study concerning ROS middleware evaluation for multi-robot control clearly states the fact that ROS is not made to guarantee temporal determinism in the data flow between robots. However, this condition is crucial to reach a high level of synchronization between axes and robots and to develop reactive control in a dynamic environment.

Therefore, we believe that the first constraint for a unifying framework is to ensure temporal determinism in the perception-action process thanks to hard real-time capability. The definition of a Real-Time System (RTS) can be stated, following [19], as: “*a real-time system is a system that satisfies explicit (bounded) response-time constraints or risk severe consequences, including failure*”. A hard RTS is a system where it is imperative that responses occur within the specified deadlines (for examples, aircraft control or process control applications).

B. *Evaluation of ROS 2 middleware communication*

ROS 2 has an architecture close to the one of ROS. Indeed, it uses Nodes to perform computation and topics are used to share message between several nodes.

To evaluate ROS 2 we have used the same architecture: one node per component. In our case, the first node receives data and publish it on the topic. The second one subscribes to the topic and sends data through the UDP communication. We have modified the protocol for this test, we now send 2000 communication frames from the component, 1 per 2ms. In this case, we highlight the latency in terms of cycle numbers. The first 500 values are shown in Figure 6.

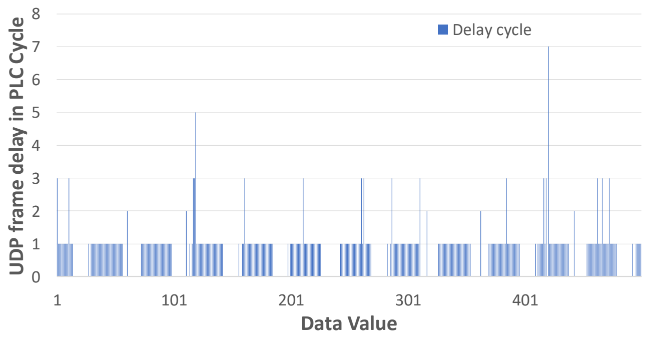


Fig. 6. Delay in cycle between the data sent and the data receive with ROS2 with a 2ms sending period

This test has been performed about twenty times and no data was lost. However, there is a latency in the exchange period, which is not constant. These results illustrate the non-determinism of ROS 2.

C. *Evaluation of RTOS communication*

As a comparison, we performed a new experimentation with the same components, but the ROS PC was replaced with an industrial PC with a real-time operating system (RTOS) (Fig. 7); and the UDP communication was replaced with a hard-real-time Ethernet communication. Among the five major technologies in industrial Ethernet communication (Profinet, POWERLINK, Ethernet /IP, EtherCAT, Sercos III) [20], only three technologies provide hard real time communication: SERCOS, EtherCAT and POWERLINK. We chose to use the POWERLINK communication as it is a BSD-license open-source solution and has be adopted as an IEEE standard (IEEE 61158).

The industrial PLC is built on CPU ATOM 1.6 GHz with 512 MB DDR2 RAM and is running a proprietary runtime based on VxWorks Operating System.

The results with this architecture are illustrated in Fig. 8 with a 2ms sending period. They show no loss of sensor data and the total communication time remains the same whatever the exchange frequency; it means that responses occur within the specified period. This behavior is typical for a hard-real-time system, since it is a requirement for RTOS design specification.

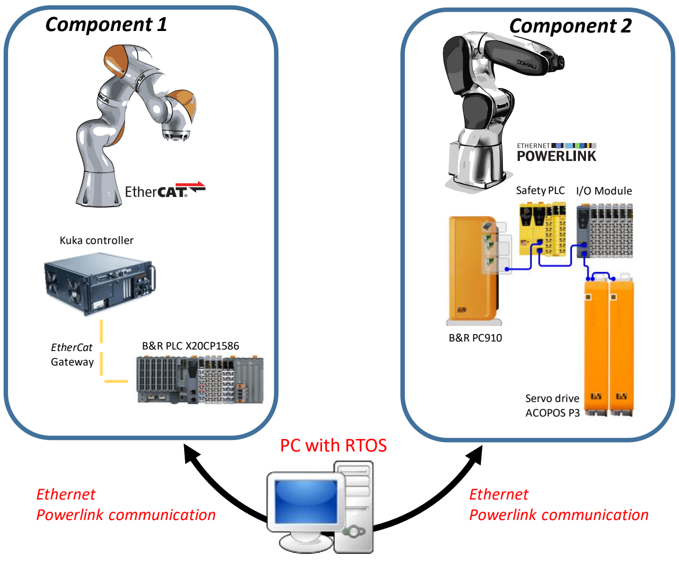


Fig. 7. Architecture used for the evaluation of the RTOS communication

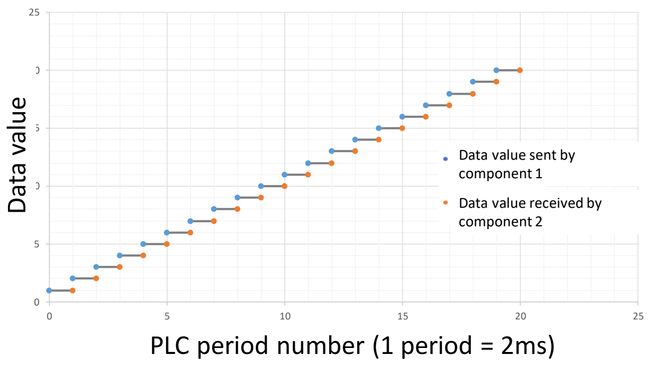


Fig. 8. Data received with RTOS based architecture with a 2ms sending period

This section showed that the ROS temporal behavior will not enable the level of synchronization required for multi-axis control in the field of production machine design. Temporal determinism is a major requirement when designing motion controllers. The next section introduces our new framework and describes its architecture.

IV. RTROBMULTIAXISCONTROL : A REAL-TIME FRAMEWORK FOR OPEN ROBOTICS AND MULTI-AXIS CONTROL

A. *A standard-based strategy*

The success of a given middleware design approach will depend on the ability to transfer the framework outside the lab for multi-axis machine control applications. In [21], the authors develop MURDOCH, a novel method of dynamic task allocation built on publish/subscribe, and they optimize the task completion time. In [22] and [23], the importance of managing time constraints with the use of Contract Net Protocol is emphasized. We also believe that it is necessary to guarantee completion time and robust communication to achieve cooperation between robots operating in dynamic environments.

As an example, a method with dynamic task allocation for groups of robots [24] will not be efficient enough to ensure a high level of synchronization and a deterministic behavior if the robots do not provide real-time data exchange or if their controllers do not give real-time access to low level control loops (position, velocity, or torque control loops). The goal of our framework is to unify the access to robot resources from an exterior RTOS manager.

Currently, the programming of industrial machines relies on the IEC 61131 standard which is a viable alternative to the extensive use of proprietary languages for each robot brand. Efficiency and robustness of machine and process control software are also increased using a real-time communication between motion controllers and robot controllers. Unfortunately hard-real-time Ethernet protocol is not an option available on most of the control cabinets of the leading industrial robot manufacturers.

A wide variety of systems and solutions can be found on the motion control market, but most of them are incompatible with one another. This leads to additional costs, even for the end-users. Without reusable software, developing an application with new hardware begins with a white paper. In the area of PLC programming, the first goal of PLCopen international organization is to provide widely accepted industrial standards such as PLCopen Motion for motion control applications. A recent add-on of PLCopen guidelines concerns the creation of function block libraries [25]. These recommendations aim at shortening the development time and at easing software use and maintenance. The hardware choice becomes less important because the libraries are reusable, the development cost and time, and the support cost decrease.

A compromise must be found between performance, functionality and standardization to have an appropriate solution. There are three kinds of development strategies. The first one consists in developing programs closely coupled to the hardware. Thanks to this, high performance is expected, but the program cannot be reused, so the cost and the time of the next development will not be reduced.

The second one is to develop numerous functionalities, which is helpful but at the expense of performance. The last one focuses on the standardization and the reusability of the project components. For axis control, most of the automation manufacturers integrate PLCopen Motion libraries applied to their hardware solutions.

B. Software architecture

The previous subsection introduced the different design constraints of our new framework. They consist in using:

- IEC 61131: the widely-available and standardized IEC 61131-3 languages or a consistent ANSI C / C++ integration in the IEC world ensure the development of reusable components;
- PLCopen Motion: this provides all useful single-axis functions and makes it possible to use hardware configurations from any manufacturers (Siemens, Beckoff, Schneider, B&R,…);
- Hard Real-Time communication standard [26]: both POWERLINK IEEE 61158 and EtherCAT meet this hard-real-time capability. The use of industrial Ethernet protocols is the keystone of our hard-real-time middleware. As an example, POWERLNK open source standard sets a real-time communication, with adjustable period, between automation components or CPUs and it respects IEEE 802.3 Ethernet standard with no modified Ethernet media access. By using such protocols, the real-time scheduling of the complete robotic cell is guaranteed at every stage of software development.

1) Basic objects for a real-time implementation

Modern software development needs to adopt a structured approach and re-usable codes to deal with the increasing complexity of programs. The 3rd edition of IEC61131-3 includes the aspects of Object-Oriented Programming (OOP) to meet these objectives. OOP compartmentalizes data into objects (data fields) and describes object contents and behaviors through the declaration of classes (methods). Our framework benefits the core concepts of OPP, namely encapsulation, polymorphism and inheritance.

The software architecture of the framework is based on two virtual and abstract objects. The first one, the ‘RbParam’ class, manages all static data, which are the parameters of a robot, a component or a task. The other, ‘RbFbAbstract’, handles the real-time and deterministic control of the robotic cell components. These classes are derived to model all hardware components, from low-level sensors to complex systems, such as a robot or a robotic cell.

The parameters associated with each object are represented by a structure ensuring the compatibility of the robot manager software with other programs written in any IEC 61131-3 language. This structure is the template of the C++ abstract virtual class ‘RbParam<Param_typ>’. It is composed of three virtual public methods: Verify, Read, and Write as shown in Fig. 9.

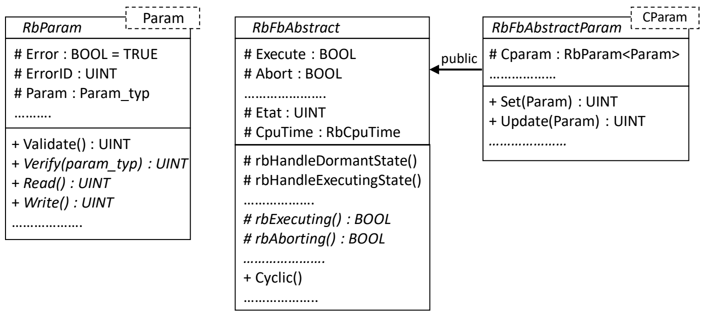


Fig. 9. Partial UML specification of the base classes

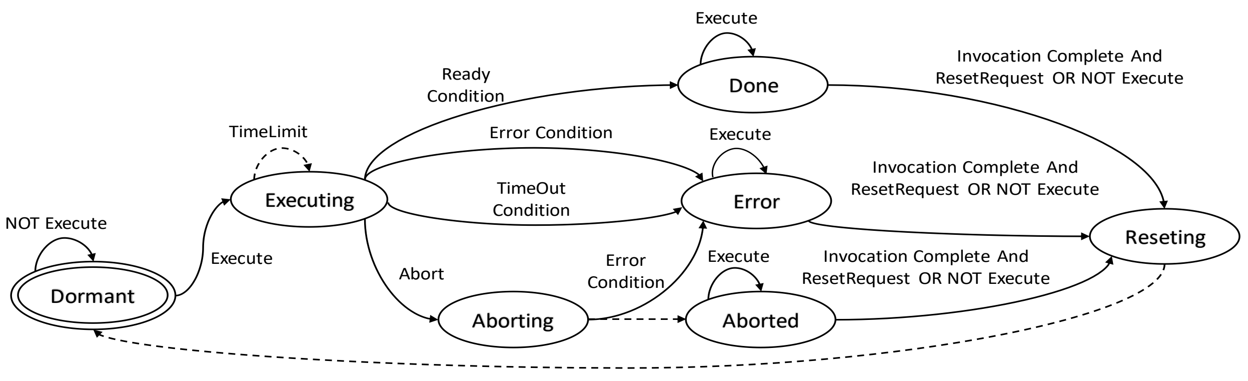


Fig. 10. Real-time function generic block state chart (complying with PLCopen recommandations)

All functions of real-time multi-axis control software are based on the abstract classes "RbFbAbstract" and "RbFbAbstractParam". The "RbFbAbstract" class has a public cyclic method that implements the state chart represented in Fig. 10. This state chart is compatible with PLCopen's compliant library writing recommendations [27]. It includes the two functioning modes 'Edge triggered FBs' and 'Level controlled FBs' described in the standard (the first one is illustrated in Fig. 11). These two basic models of Function Blocks (FBs) with the minimum inputs and outputs have two options for activation:

- "Execute" input triggers the execution of the FB, and "Done" signals that the FB has completed the command;
- "Enable" input is level sensitive, and "Valid" expresses that the other outputs can be safely used.

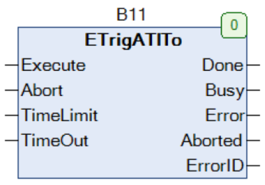


Fig. 11. Edge Triggered functional block

2) A unified and standard axis control

The foundations of the framework offer the programmer nine evolutive packages enabling the management of all multi-axis control applications. The useful FBs for motion control are implemented in the framework by using virtual classes inheriting from 'RbFbAbstractParam' or 'RbFbAbstract'. These virtual classes unify the access to the hierarchically-superior complex objects. With such an approach, inputs and outputs of the PLCopen Motion standard FBs are identical whatever the hardware and topologies. This standard approach provides a unified access to the different components (robots, controllers) in the higher-level.

Motion FBs are included in the 'RbMcPart1' package, while the axis control is ensured by the 'RbRegul' and 'RbMcAxe' packages. These motion control FBs are compliant to PLCopen Motion Parts 1 & 4 [27]. Figure 12 describes the PLCopen Motion functionalities covered by the framework. Four additional FBs have been added to access to the dynamic control of a drive motion, for position, velocity or torque command ('RbMC_MoveCyclicPosition', 'RbMC_MoveCyclicVelocity', 'RbMC_CyclicTorqueControl', and 'RbMC_CyclicLoad').

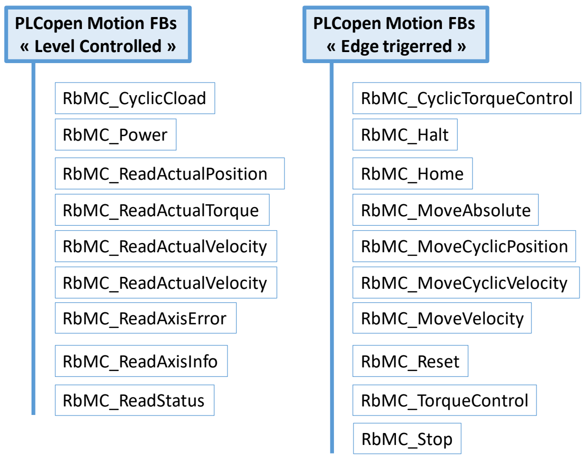
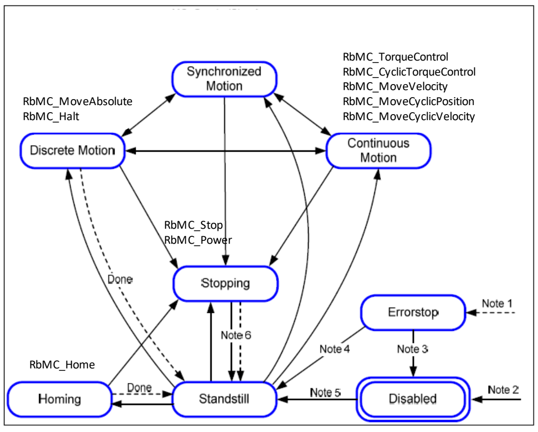


Fig. 12. The PLCopen Motion FBs implemented through virtual classes



Note 1:	From any state. An error in the axis occurred.
Note 2:	From any state. MC_Power.Enable = FALSE and there is no error in the axis.
Note 3:	MC_Reset AND MC_Power.Status = FALSE
Note 4:	MC_Reset AND MC_Power.Status = TRUE AND MC_Power.Enable = TRUE
Note 5:	MC_Power.Enable = TRUE AND MC_Power.Status = TRUE
Note 6:	MC_Stop.Done = TRUE AND MC_Stop.Execute = FALSE

Fig. 13. PLCopen Motion state machine impelemented in RTRobMultiAxisControl motion function blocks

The package “RbRegul” contains utility classes for control such as: trajectory interpolation, prediction, PI regulation, feedforward. The online trajectory planning is based on the interpolation strategy proposed in [28].

The virtual class “RbMcAxe” ensures the axis control and sequentially performs the following tasks:

- Error check;
- Management of the PLCopen Motion generic state machine detailed in Fig. 13;
- Trajectory planning;
- Position, velocity, and torque control.

According to the axis controller characteristics, “RbMcAxe” provides the management and access to various levels of control. Four types of physical topologies can be used, as described in Figure 14. The hardware may be accessed through:

- An H bridge: the position-velocity-torque control loops are then handled by the software;
- Open torque control: position and velocity control loops are handled by the software;
- Open velocity control: position loop is handled by the software;
- Open position control; the framework only takes charge of the generation of controlled positions.

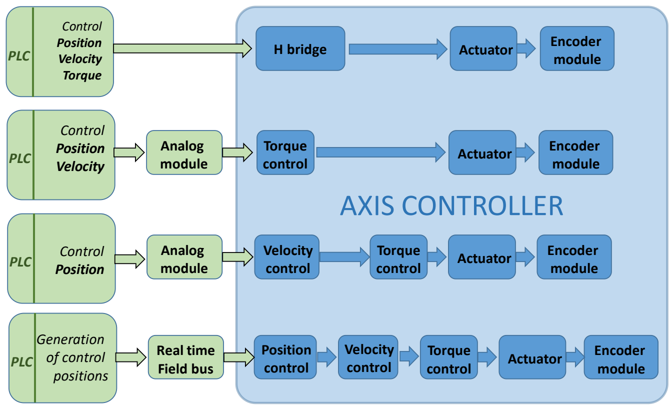


Fig. 14. Four types of topologies for axis control

The framework is also able to handle off the shelf industrial axis controllers; as an example, B&R Automation ACOPOS controllers can be used with the same unified access to the hardware using the PLCopen Motion FBs provided by the manufacturer.

The low-level packages “RbRegul”, “RbMcAxe” and “RbMcPart1” provide a unified and standard interface to control any axis. All higher-level objects can be developed by the user regardless of the type or the manufacturer of the axis.

3) *An open robotic solution*

The virtual class ‘RbMC_AxeRob’is designed to manage the different axes of a whole robotic system. This class enables the use of unified command and information data flow generic structures as described in Fig. 15.

Thanks to the framework standard FBs (Fig. 12), “RbMC_AxeRob” manages the operating modes and ensure the axis safety. All robot axes can be controlled manually or automatically in position, velocity, or torque. The virtual class enables to dynamically switch between these three control strategies, without stopping the axis.

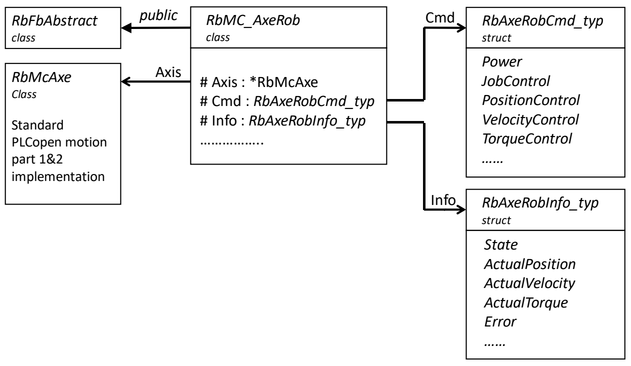
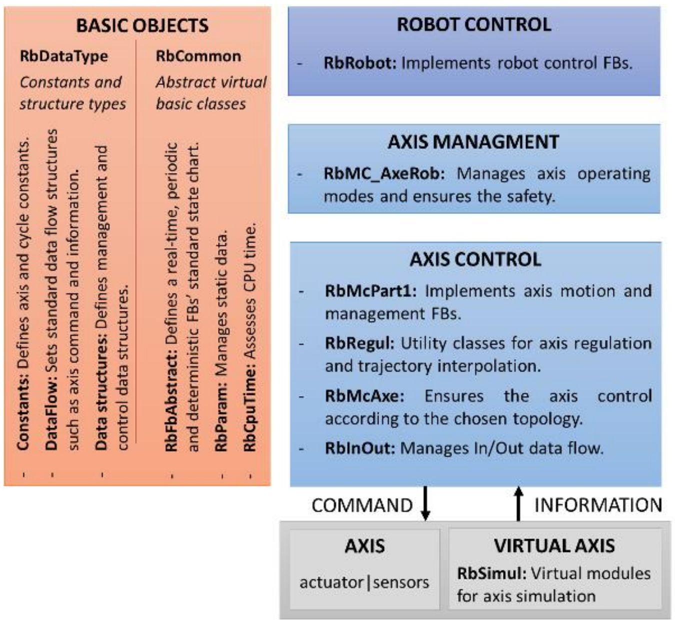


Fig. 15. Virtual class “RbMC_AxeRob”

All control approaches commonly used in robotics can be easily implemented by means of the RTRobMultiAxisControl framework, whose structure is presented in Fig. 16. The framework is a standard and unified foundation for all multi-axis systems and can be used within any hardware architecture. A virtual class “RbRobot” is currently under development to integrate high-level FBs such as axis synchronization, operational space force and position control or on-line trajectory generation.


BASIC OBJECTS
RbDataType
Constants and structure types
RbCommon
Abstract virtual basic classes
- Constants: Defines axis and cycle constants.
- DataFlow: Sets standard data flow structures such as axis command and information.
- Data structures: Defines management and control data structures.
- RbFbAbstract: Defines a real-time, periodic and deterministic FBs' standard state chart.
- RbParam: Manages static data.
- RbCpuTime: Assesses CPU time.
ROBOT CONTROL
- RbRobot: Implements robot control FBs.
AXIS MANAGMENT
- RbMC_AxeRob: Manages axis operating modes and ensures the safety.
AXIS CONTROL
- RbMcPart1: Implements axis motion and management FBs.
- RbRegul: Utility classes for axis regulation and trajectory interpolation.
- RbMcAxe: Ensures the axis control according to the chosen topology.
- RbInOut: Manages In/Out data flow.
COMMAND
INFORMATION
AXIS
actuator|sensors
VIRTUAL AXIS
RbSimul: Virtual modules for axis simulation
Fig. 16. Structure of the RTRobMultiAxisControl framework

In this section, RTRobMultiAxisControl, a novel framework based on a hard-real-time Operating System and industrial standards, has been detailed. PLCopen Motion and IEC 61131 programming are used to seamlessly transfer the framework on different targets for multi-axis control. This standard approach eases the evolutions of the framework components. The framework introduces a new way to unify hardware and software, as will be illustrated with the experiments in the next section.

V. DEXTROBC : A ROBOTIC PLATFORM FOR INDUSTRY 4.0

As a validation test bench, the framework is implemented on the targets of the French national robotic platform "DextRobC" which is dedicated to Industry 4.0. The objective of this technical platform is to demonstrate innovative robotic solutions and designs that may be transferred into the industry of the future. To prove the efficiency of the proposed framework, two complex robotic cells, involving the coordination of several robots, have been used. They are presented in this section.

A. *A dexterous manipulator for flexible and fine tasks*

To increase the flexibility of robots, it becomes necessary to design new grippers able to meet a wide range of tasks. However, reproducing the dexterity of the human hand remains a key challenge. With this objective in mind, the new RoBioSS dexterous hand was developed and patented (CNRS Patent FR 1459956, 10 16, 2014) [29, 30]. The hand has been designed as an end-effector for industrial robots or collaborative robots. The integration of a fully actuated dexterous robotic hand requires the control of a high number of actuators. As each finger behaves like a robot, a high level of synchronization is needed between them to achieve proper inside hand manipulation tasks.

A deterministic and robust communication is a necessary element to fulfill this requirement. The RoBioSS hand with its multi-axis control software and reliable mechanical design encompasses the high capabilities required for fine manipulation tasks.

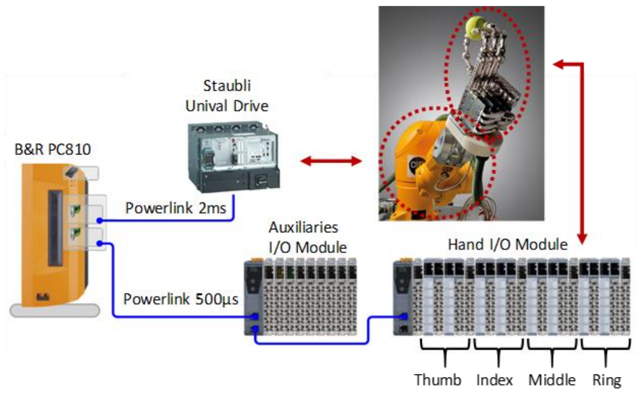


Fig. 17. Dexterous hand-arm system hardware architecture

The hardware architecture of the manipulation cell is detailed in Fig. 17. The hand is mounted on a TX60 6-axis Stäubli serial robot powered by a uniVAL drive industrial robotic controller. This controller offers the innovative feature of a possible integration with an external controller. All 22 axes - 16 axes of the hand and 6 axes of the Stäubli robot - are driven by a B&R industrial PC. Communication between the robot controller, the DC motor power stage and the PC runs through a high-speed POWERLINK fieldbus which guaranties the real-time requirement. Low-level control loops for the hand motors also run on the industrial PC.

As described in Fig. 18, the robot-level software of the manipulation cell is relies on the RTRobMultiAxisControl framework. The main program implements the state machine of the task while the axis manager generates the axis commands for the application. All commands are set thanks to motion and control FBs integrated into the packages of RTRobMultiAxisControl. The framework manages all 22 axes of the manipulation cell. The use of standards during the development of the libraries enabled have a unique control software for this multi-axis robotic cell regardless of various hardware and industrial components used. The finger motors are powered by H-bridges whereas the uniVAL drive cabinet provides the robot joint commands with respect to the controller set values. For the hand control, all axes of the fingers are given 0.4ms periodic position commands while the period for the 6-DOF Stäubli axes is 2ms.

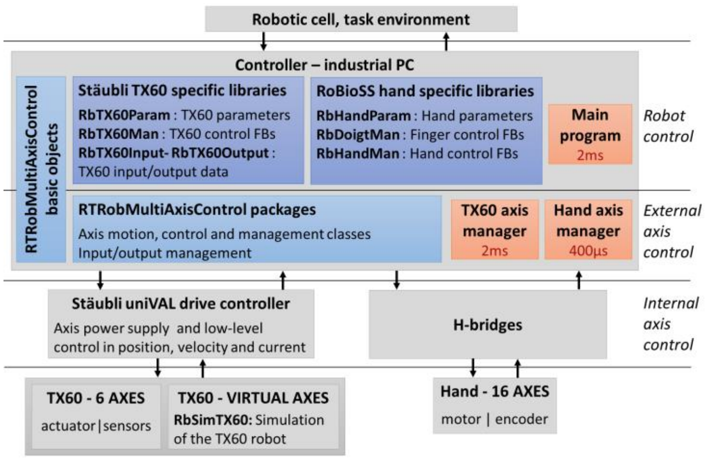

Robotic cell, task environment
Controller – industrial PC
RTRobMultiAxisControl basic objects
Stäubli TX60 specific libraries
RbTX60Param : TX60 parameters
RbTX60Man : TX60 control FBs
RbTX60Input- RbTX60Output :
TX60 input/output data
RoBioSS hand specific libraries
RbHandParam : Hand parameters
RbDoigtMan : Finger control FBs
RbHandMan : Hand control FBs
Main program 2ms
Robot control
RTRobMultiAxisControl packages
Axis motion, control and management classes
Input/output management
TX60 axis manager 2ms
Hand axis manager 400µs
External axis control
Stäubli uniVAL drive controller
Axis power supply and low-level control in position, velocity and current
H-bridges
Internal axis control
TX60 - 6 AXES
actuator|sensors
TX60 - VIRTUAL AXES
RbSimTX60: Simulation of the TX60 robot
Hand - 16 AXES
motor | encoder

Fig. 18. Hardware and software architectures of the multi-robot collaborative cell

B. *A collaborative cell integrating human expertise*

RoBioSS team also explores how humans can interact with machines in the industry context. Different approaches are studied to include the human behavior within a production line. The developed schemes aim at improving the biomechanical comfort and the safety of the human operator. A robotic cell was developed for this purpose; it creates a scene where a high level of collaboration between human and machine can be tested. Two types of human-robot interactions are investigated: the human may be in contact with a cooperative robot acting as an assistant or the human can teleoperate a distant robot through a haptic device. The haptic control system makes it possible to accurately achieve tasks with a proper kinesthetic sense.

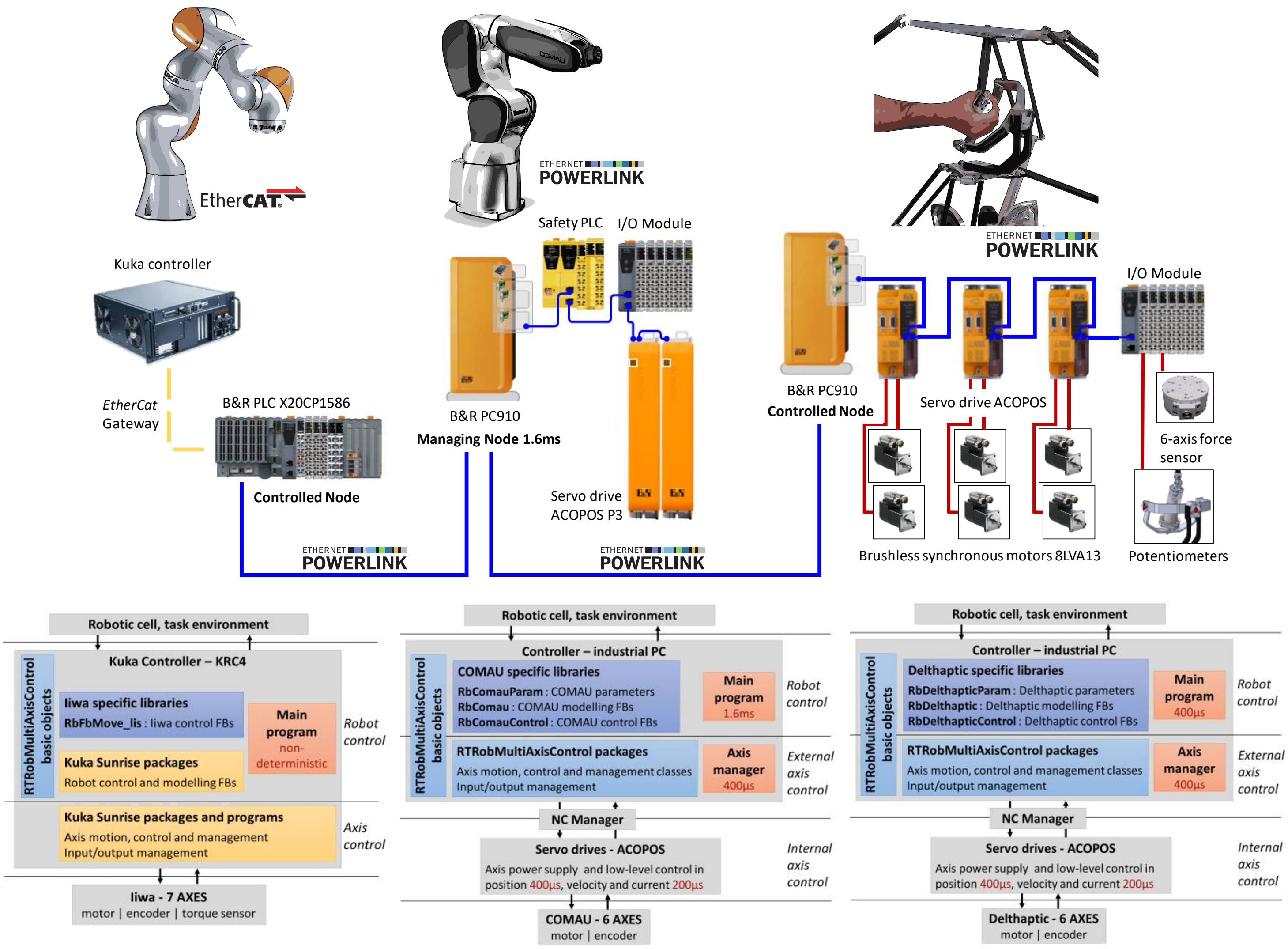

EtherCAT
Kuka controller
EtherCat Gateway
B&R PLC X20CP1586
Controlled Node
ETHERNET POWERLINK
Safety PLC
I/O Module
B&R PC910
Managing Node 1.6ms
Servo drive ACOPOS P3
B&R PC910
Controlled Node
Servo drive ACOPOS
I/O Module
6-axis force sensor
Brushless synchronous motors 8LVA13
Potentiometers
Robotic cell, task environment
Kuka Controller – KRC4
RTRobMultiAxisControl basic objects
iiwa specific libraries
RbFbMove_iis : iiwa control FBs
Kuka Sunrise packages
Robot control and modelling FBs
Main program non-deterministic
Robot control
Kuka Sunrise packages and programs
Axis motion, control and management
Input/output management
Axis control
Iiwa - 7 AXES
motor | encoder | torque sensor
Controller – industrial PC
COMAU specific libraries
RbComauParam : COMAU parameters
RbComau : COMAU modelling FBs
RbComauControl : COMAU control FBs
Main program 1.6ms
RTRobMultiAxisControl packages
Axis motion, control and management classes
Input/output management
Axis manager 400µs
External axis control
NC Manager
Servo drives - ACOPOS
Axis power supply and low-level control in position 400µs, velocity and current 200µs
Internal axis control
COMAU - 6 AXES
motor | encoder
Delthaptic specific libraries
RbDelthapticParam : Delthaptic parameters
RbDelthaptic : Delthaptic modelling FBs
RbDelthapticControl : Delthaptic control FBs
Main program 400µs
Delthaptic - 6 AXES
motor | encoder

Fig. 19. Collaborative multi-robot cell hardware and software set-up

Figure 19 presents the implementations of the two interaction types. An iiwa cobot with its Kuka controller plays the role of an assistant for the operator in a shared environment. A Comau Racer 3 industrial robot is remotely controlled by the operator thanks to the Delthaptic, a novel haptic device designed in the lab [31]. Both the Comau robot and the Delthaptic are driven by B&R industrial PCs and ACOPOS servo drives. The iiwa cobot is driven by the proprietary Kuka Sunrise cabinet.

The Comau robot and the Delthaptic software architectures are similar, as presented in Fig. 19. Both robot controllers implement the RTRobMultiAxisControl framework and use internal control functions of the servo-drives.

The main program plans dynamic trajectories with respect to the task while the axis manager yields the axis commands thanks to motion and control FBs integrated into the framework packages. As the iiwa robot has a closed controller, the proprietary Kuka Sunrise packages must be used to drive the robot and programs are written within the manufacturer’s development suite. A specific library was developed to manage the high-level control of the iiwa and to access to the Sunrise packages; this library is based on the RTRobMultiAxisControl basic objects to homogenize the control of all the components in the collaborative cell.

All the cell components must cooperate in real-time under several control schemes:

- The teleoperating mode between the Comau robot and the Delthaptic;
- The motion control of the Racer when following preset CNC trajectories;
- An exchange task between the Racer and the iiwa;
- A collaborative task between the human and the iiwa.

Each cell component needs to share its state and time-dependent data with others. A robust communication with a wide bandwidth is required to provide these control modes and guarantee the safety of the human within the cell. The choice of hard real-time communication is the only solution able to synchronize our robots. The iiwa robot uses EtherCAT communication standard whereas the Delthaptic and Racer 3 are included into a POWERLINK communication network. A

PLC gateway had to be inserted to exchange data between POWERLINK and EtherCAT networks.

VI. Experimental Results

The RTRobMultiAxisControl framework has been used to control different multi-axis systems in two applications, as presented in the previous section. These very different implementation cases demonstrate the versatility, the robustness, and how the real-time behavior of the base components is kept at the machine level.

A. Synchronization of the fingers for accurate inside hand manipulation

The robustness of the dexterous hand was validated through demonstrations, proposed continuously during three days at SPS IPC Drives 2016 international exhibition in Nuremberg. The video link [32] illustrates the hand efficiency. Different tasks were performed, such as grasping a glass, switching-on a lamp or led ribbons with different kinds of buttons, or screwing a light bulb. A second video illustrates the inside hand manipulation capability of the hand [33]. The gripper must be able to generate fine object motions with respect to its palm. The fingers thus need to collaborate to produce the desired inside-hand motion while ensuring the object stability. The accuracy of the motion has been assessed and the synchronization is clearly visible in the video during the translational and rotational motions of the bottle cap.

The development of such a complex manipulation cell became possible because the mechanical design meets the grasping and manipulation objectives. Fine manipulation tasks can only be achieved thanks to a robust and hard real-time controller. The command motions of the 6-axis industrial robot and the 16 joints of the hand are simultaneously set during the process. The deterministic approach of the framework ensures that the axis commands (position, velocity or torque) are periodically sent, even though the networks run at two different periods (0.4ms and 2ms), without latency. The unified approach allows to seamlessly synchronize axes of different hardware topologies with the same software function blocks.

This manipulation cell, as a whole, shows that the framework can be implemented at different levels of control (Fig. 18). If the robot hardware architecture includes an internal axis controller, such as the uniVAL Drive of the Stäubli robot, the framework only generates trajectories for the task. Otherwise, the framework will also handles all the control loops. This is the case for the hand controller where all the control levels of the RTRobMultiAxisControl are running. The hand control part of the software exhibits a very small computation time of 40µs for the managing all four fully actuated fingers.

B. Real-time cooperation between robots and humans

A typical application of collaborative robotics has been developed to demonstrate the benefits of a collaborative platform for Industry 4.0, with the idea of relocating the human worker at the heart of the production process.

Such a cooperation level between several robots and human operators requires hard real-time communication between all the cell components and real-time Operating Systems. This deterministic behavior of the robotic cell is paramount to provide a proper functioning of the demonstration set-up and to ensure safety of the human operator. For this purpose, POWERLINK and EtherCAT hard real-time communication protocols are used between the Delthaptic, the Comau and the iiwa robots. Four data flow structures where created to exchange significant data for the process (state information, robot dynamic data, sensor data…).

The use of a safety PLC to secure the Comau robot motions sets the communication rate to 1.6ms for all exchanges between the components over POWERLINK networks. The EtherCAT communication rate of the iiwa robot is fixed by Kuka to 4ms. A repetitive delay alternates between 17.6ms and 19.2ms for a poll request cycle from the POWERLINK network (Racer3 and Delthaptic) to the iiwa. This delay originates from the two communication rates, from the application cycle time, and from the POWERLINK – EtherCAT data conversion in the PLC gateway. The variation in the latency is due to non-consistent cycle times of the two networks.

Both Racer 3 and Delthaptic controllers are fully based on the RTRobMultiAxisControl framework (Fig. 19). The programs cyclically compute the robot models and control schemes for the application. All computation times of the main programs and the axis managers are very small and they are compatible with the communication rate of the network. Although the Kuka controller is based on Sunrise, a closed programming environment, the specific motion FBs developed for the iiwa robot rely on the framework basic objects. The use of virtual and abstract objects guarantees that all the function blocks used in the controllers of the cell components share a deterministic behavior. In this way, the real-time feature of the Operating Systems are transferred to all software components.

In the cobotic demonstration scenario, each robot controller switches between the different control modes introduced in the previous section. The application is based on the following cycle: a cylinder part is grasped by Racer 3 remotely piloted by the operator through the haptic device. Then, the Racer transmits the part to the iiwa cobot to be handled and visually inspected by a second operator. The video link [34] shows this cobotic application. The whole cycle is composed of 6 steps:

- The human operator teleoperates the Racer robot through the haptic device to grab the part. A torque control of the Delthaptic interface compensates for its own gravity and provide the haptic feedback in operational space.
- Delthaptic returns to the origin of its workspace under position control. During this step, Racer still tracks the position of the haptic device. The tracking of Delthaptic in real-time is accurate, the tracking error being mostly due to the joint kinematic limits of the Racer.

- The Comau robot executes a CNC trajectory to go to the exchange position.
- Step 4 consists in the exchange of the part between the robots: the iiwa moves to the exchange point, grips the part, and asks Racer to release the part by opening its gripper.
- The Comau robot returns to its home position. Iiwa cooperates with the second operator who visually checks the part. This collaborative mode integrates different cooperation functions. The detection of a contact, measured through a force threshold at the end-effector, stops the robot motion for the safety of the operator. A gravity compensation mode allows the operator to move freely the part hold by the robot without carrying its weight. These cooperation functions ensure the safety of the operator and redefine the robot's role as an assistant for the human. Once the part is controlled by the operator, iiwa autonomously adjusts its end-point to the height of the receptacle, drops it and returns to its home position.
- In the last step, the Racer robot waits for the operator to grab the handle of Delthaptic to begin a new cycle.

The programming of the three robots in the cell is based on the presented framework even though their software and hardware are provided by different manufacturers. The framework unifies the access to the management of these robots with a hard real-time behavior. Thanks to this approach, the temporal determinism in the data flow communication and the real-time determinism in the perception-action process enable the implementation of robust and secured interactions between robots and humans in a shared environment.

VII. Conclusion

Within the context of the DextRobC technological platform, we aim at developing innovative robotic devices and at implementing Industry 4.0 principles. Interoperability is a key challenge in the industry of future. As we wish to secure interaction between humans and machine in an open space, the temporal determinism of software architecture and communication between all the components of an industrial environment is required.

This paper focuses on a software development approach that meets this issue. The foundation of our approach is based on strong requirements: unify the programming of the components, use hard-real-time communication and operating system, use motion control standards.

The framework we developed is fundamentally multi-level, multi-agent and manufacturer-independent. The strategy is based on the integration of modern and robust industrial standards: PLCopen Motion (at the axis and coordinated motion levels) and hard real-time communication (POWERLINK or EtherCAT). These standards guarantee the temporal coordination of all agents in the scene, without dependence on the manufacturers. We also chose to benefit from the possibilities offered by the most recent automation software to use Object-Oriented programming.

The RTRobMultiAxisControl framework has been evaluated in a multi-robot and multi-manufacturer environment: on one hand for the coordination of 16-dof RoBioSS dexterous hand with an industrial Staubli 6-dof robot, and on the second hand for the development of a complete cobotic cell with three different robots and two types of human-robot interactions. The implementation of the framework demonstrates its robustness and performances.

Future developments will consist in adding low-level autonomous reflex behaviors to the robots to be able to adapt to dynamic events occurring in the cell. Such a unified framework used for the coordination of multi-agents systems may support recent coordination strategies such as self-organization or shared-autonomy [35].

References

[1] Erik Nieves, "Overview of Collaborative Robots", RIA, International collaborative robots workshop, October 2016,] Cincinnati, USA.

[2] Ayssam Elkady and Tarek Sobh, "Robotics Middleware: A Comprehensive Literature Survey and Attribute-Based Bibliography" May 2012, Journal of Robotics 2012, pp.1687-9600.

[3] D. Bakken, "Middleware," in Encyclopedia of Distributed Computing, J. Urban and P. Dasgupta, Eds., Kluwer Academic, Dodrecht, The Netherlands, 2001.

[4] H. Utz, S. Sablatnog, S. Enderle, and G. Kraetzschmar, "Miro—middleware for mobile robot applications," IEEE Transactions on Robotics and Automation, vol. 18, no. 4, pp. 493–497, 2002.

[5] J. Kramer and M. Scheutz, "Development environments for autonomous mobile robots: a survey," Autonomous Robots, vol. 22, no. 2, pp. 101–132, 2007.

[6] N. Mohamed, J. Al-Jaroodi, and I. Jawhar, "Middleware for robotics: a survey," in Proceedings of the IEEE International Conference on Robotics, Automation and Mechatronics (RAM '08), pp. 736–742, September 2008.

[7] N. Mohamed, J. Al-Jaroodi, and I. Jawhar, "A review of middleware for networked robots," International Journal of Computer Science and Network Security, vol. 9, no. 5, pp. 139–148, 2009.

[8] B. P. Gerkey, R. T. Vaughan, and A. Howard, "The player/stage project: Tools for multi-robot and distributed sensor systems," in In Proceedings of the 11th International Conference on Advanced Robotics, 2003, pp. 317–323. 1

[9] Kranz, M., Rusu, R. B., Maldonado, A., Beetz, M., & Schmidt, A. (2006). A player/stage system for context-aware intelligent environments. *Proceedings of UbiSys*, *6*(8), 17-21.

[10] Ersp 3.1 software development kit, 2010, http://www.evolution.com/products/ersp/.

[11] Soetens, P., & Bruyninckx, H. (2005, April). Realtime hybrid task-based control for robots and machine tools. In Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on (pp. 259-264).

[12] Object Management Group (1998). The Common Object Request Broker: Architecture and Specification Revision 2.2. 492 Old Connecticut Path, Framingham, MA 01701, USA.

[13] Ando, N., Suehiro, T., Kitagaki, K., Kotoku, T., & Yoon, W. K. (2005, August). RT-middleware: distributed component middleware for RT (robot technology). In Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on (pp. 3933-3938). IEEE.

[14] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: an open-source robot operating system," in Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Robotics, Kobe, Japan, May 2009.

[15] https://github.com/bmwcarit/meta-ros

[16] S. Chitta, E. Marder-Eppstein, W. Meeussen, V. Pradeep, A. Rodríguez Tsouroukdissian, J. Bohren, D. Coleman, B. Magyar, G. Raiola, M. Lüdtke and E. Fernandez Perdomo "ros_control: A generic and simple control framework for ROS", The Journal of Open Source Software, 2017; see: https://github.com/ros-controls/ros_control

[17] KAY JACKIE, RODRIGUEZ Tsouroukdissian https://roscon.ros.org/2015/presentations/RealtimeROS2.pdf

[18] Harrisson Fischer, Philippe Vulliez, Jean-Pierre Gazeau, Saïd Zeghloul, "An industrial standard based control architecture for multi-robot real time coordination", IEEE 14th International Conference on Industrial Informatics (INDIN), 2016.

[19] Krishna Kavi1, Robert Akl1, Ali Hurson, "Real-Time Systems: An Introduction and the State-of-the-Art", DOI: 10.1002/9780470050118.ecse344, Wiley Encyclopedia of Computer Science and Engineering, Mars 2009.

[20] Industrial ethernet system comparison : the 5 major technologies, 3rd Edition of Industrial Ethernet Facts magazine, Ethernet Powerlink Standardization Group (http://www.ethernet-powerlink.org).

[21] Brian P. Gerkey and Maja J. Mataric, "Sold!: Auction Methods for Multirobot Coordination", IEEE Transactions on Robotics And Automation, Vol.18, No 5, October 2002.

[22] T. Sandholm, "An implementation of the contract net protocol based on marginal cost calculations," in Proc. Nat. Conf. Artificial Intelligence (AAAI), Washington, DC, 1993, pp. 256–262.

[23] C. Ramos, "A holonic approach for task scheduling in manufacturing systems," in Proc. IEEE Int. Conf. Robotics and Automation (ICRA), Minneapolis, MN, Apr. 1996, pp. 2511–2516.

[24] L. E. Parker, "ALLIANCE: An architecture for fault-tolerant multirobot cooperation," IEEE Trans. Robot. Automat., vol. 14, pp. 220–240, Apr. 1998.

[25] Creating PLCopen compliant libraries, http://www.plcopen.org/pages/pc2_training/fb_libraries.htm

[26] Industrial ethernet system comparison : the 5 major technologies, 3rd Edition of Industrial Ethernet Facts magazine, Ethernet Powerlink Standardization Group (http://www.ethernet-powerlink.org).

[27] PLCopen Motion: standard for single axis and multi axis control, http://www.plcopen.org/pages/tc2_motion_control/

[28] Biagiotti L. and Zanasi R., "Time-optimal regulation of a chain of integrators with saturated input and internal variables: an application to trajectory planning." 8th IFAC Symposium on Nonlinear Control Systems, 2010, p. 1278-1283

[29] Mnyusiwalla, H., Vulliez, P., Gazeau, J. P., & Zeghloul, S. (2016). A New Dexterous Hand Based on Bio-Inspired Finger Design for Inside-Hand Manipulation. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 46(6), 809-817.

[30] Vulliez, P., Gazeau, J-P., Laguillaumie, P., Mnyusiwalla, H., Seguin P. "Focus on the mechatronics design of a new dexterous robotic hand for inside hand manipulation". *Robotica*, 2018, vol. 36, no 8, p. 1206-1224.

[31] Vulliez, M., Zeghloul, S., et Khatib, O. "Design strategy and issues of the Delthaptic, a new 6-DOF parallel haptic device." *Mechanism and Machine Theory*, 2018, vol. 128, p. 395-411.

[32] The RoBioSS hand in Nuremberg SPS-IPC International Exhibit : https://www.youtube.com/watch?v=X87KKuVESS8

[33] The RoBioSS hand video cited in the newspaper "Le Monde" (Keywords : Le Monde – Gazeau) follow the link : https://www.youtube.com/watch?v=O_P69haNA4A

[34] The RoBioSS multi-robot collaborative cell : https://www.youtube.com/watch?v=IJF9DG1Zlqo

[35] Koorehdavoudi, Hana, and Paul Bogdan. "A statistical physics characterization of the complex systems dynamics: quantifying complexity from spatio-temporal interactions.", Nature, Scientific reports 6 (2016): 27602.

H. Fischer received his Ph.D. degree in Robotics (2018) at Pprime Institute, and his M.S. degree in Computer Sciences, Industrial System Engineering (2014), from the University of Poitiers. His current research focuses on the interaction between human and robot, risk assessment and path planning.

M. Vulliez is a postdoctoral research fellow in the Stanford Robotics Lab at Stanford University. She received her Ph.D. degree in Robotics (2018) from the University of Poitiers, her M.S. degree in Mechanical Engineering, Advanced Systems and Robotics (2015), and her M.S. degree in Faculty Training for Higher Education, Mechanical Engineering (2014), from the Ecole Normale Supérieure de Cachan. Her current research interests include mechatronic design, haptics, and HRI.

P. Laguillaumie is a teacher at the Faculty of Sciences of Poitiers University and works as an application engineer at Pprime Institute. He graduated from the Ecole Normale Supérieure de Cachan and received his M.S in Biomechanics in 2000. His fields of expertise cover mechanical design and industrial automation. He is currently involved in the design of a new robotic assistant for human-machine collaboration in industrial workspace and contributes to the development of a new walking robot platform.

P. Vulliez received his engineer degree in Mechanical Engineering (1981) from Ecole Normale Supérieure de Cachan, and his M.S. degree in Mechanical Engineering (1982), from LMT, Cachan (France). He is presently an assistant professor at the Fundamental and Applied Sciences Faculty of the University of Poitiers, where he teaches mechatronics. He is preparing his Ph.D. in Robotics. His main research is focused on mechatronics and on the design and control of dexterous robot hands.

J.P. Gazeau received his M.S. degree in Mechanical Engineering (1994) and his Ph.D. degree in Mechanics (2000), from the University of Poitiers. He is currently a CNRS research engineer at Pprime Institute UPR 3346. His research interests are: object manipulation with mechanical hands, robot control, and electronic design for embedded systems. He oversees the animation of the multi-scale manipulation workgroup in the French Robotics Research Group, GDR Robotique. He is also the team leader of RoBioSS team at Pprime Institute.