Shop-Floor Architecture for Effective Human-Machine and Inter-Machine Interaction

Bjørn Solvang[1], Gábor Sziebig[2], Péter Korondi[3]

[1,2] Narvik University College, PO Box 385, 8505 Narvik, Norway

[3] Budapest University of Technology and Economics
PO Box 91, H-1521 Budapest, Hungary

[1]bjorn.solvang@hin.no, [2]gabor.sziebig@hin.no, [3]korondi@mogi.bme.hu

Abstract: Manufacturing equipment for small and medium scale production is often arranged in a setup of a Flexible Manufacturing Cell (FMC). Normally, such cell members are from various manufactures and they have all their specific capabilities when it comes to man-machine and inter-machine interaction. Existing man-machine interfacing is quite simple and is typically done via a screen/keyboard interface while inter-machine communication is carried out through designated I/O lines. Although such communication channels provide us a certain possibility to program and control the cell members, it is not enough to fully utilize the potential of the coordinated control of the complete unit. Especially in smaller production facilities, where often the FMC components are a mixture of older (poorly documented) and newer machines, there exists no ready-to-use solution for communication between the members of the cell. This article discusses and presents a new architecture for man-machine and inter-machine communication and control. This new architecture allows for the integration of both new and old equipment and is especially targeted for efficient man-machine interaction.

Keywords: shop- floor architecture; flexible manufacturing cells/systems; human-machine interaction

1 Introduction

The development of novel technologies often stems from experience acquired during the production of a previous series by the same manufacturer. This allows for compatibility with older, in-house, versions of software/hardware solutions. Such a strategy is understandable from a vendor's point of view, but it is not enough from the end-user perspective. The typical end-user has many different machines/software and basically wants everything to be compatible, independent of who produced what.

During the last few decades much research has been dedicated towards general manufacturing equipment such as turning- and milling- machines (NC-machines), industrial robots (IRBs) and automated guided vehicles (AGVs). In the case of NC machines, researchers have focused on establishing a common language that can be shared by all machines [1]. On the other hand, language has not been the main focus among researchers in the field of robotics; their main concern has been to allow users to create robot programs more intuitively and rapidly (e.g. [2, 3]). AGV research has in many senses been related to path-planning and the automatic guidance in a semi-unstructured environment. At the same time we have available new, reasonably-priced ICT technologies; TV/screen solutions, mobile and wireless equipment, 3D technologies, scanners, virtual reality and haptic interfaces. The upcoming challenge is how these classical and new technologies cooperate and be combined in an industrial environment.

For a typical small- and medium-sized enterprise (SME), flexibility is the main concern. SME flexibility means most of all: a rapid setup and initial programming phase for a new production series, programming interoperability between different machines, the tracking of changes at any production step, and the possible incorporation of both older and newer equipment into the manufacturing cell. While there are many promising developments related to various equipments, there are still many challenges to be addressed in the case of SMEs:

a) *Outdated equipment*: the typical SME doesn’t have a new and updated machine park, and their equipment ranges from very old to more up-to-date solutions.
b) *No inter-machine communication*: machines are operated in standalone mode, without any structured communication between them.
c) *Manual programming*: machine codes are mainly produced and tested on the shop-floor, without any feedback to production engineers and other decision makers.
d) *Manual labour*: monotonous and repetitive work-tasks are executed by humans instead of industrial robots, often because of the lack of fast and effective programming methods for smaller production volumes.

Flexibility calls for an open shop-floor architecture which allows for efficient man-machine and inter-machine interaction, and, as pointed out in the introduction, an architecture that allows for the incorporation of both typical manufacturing machines with new 3D/mobile/virtual technologies.

Thus, the key point of this paper is the lay-out of a new shop-floor control architecture which allows for the use of a combination of both older and newer technologies. The following criteria are set against the new architecture:

i) Flexibility through on-line system reconfigurability and the implementation of the plug’n’produce paradigm.
ii) Machine invariance, so that old and new equipment, including a wide variety of different sensors, may be used.

iii) The architecture should enable effective man-machine communication.

iv) The architecture should allow for the use of a heterogeneous programming language.

v) The architecture should be capable of tracking changes during manufacturing processes and should provide feedback to previous stages.

The proposed architecture incorporates the future trends in machine development, especially through the incorporation of the new programming standards and the introduction of a software-based middleware technology for inter-machine communication. The following technologies and techniques serve as the basis in our work:

1) low-cost, accurate and efficient electronics (e.g. microcontrollers) [5]
2) high-speed communication links for inter-machine communication [6]
3) machine programming and development software based on open source solutions (e.g. [7])
4) standards for efficient machine communications and control [8]

The remainder of this article is organized as follows; in Section 2, the incorporated standards are shown along with their main characteristics and benefits. This section also describes the importance of using standards, instead of case-by-case specific solutions as building blocks. In Section 3, the proposed shop-floor architecture is introduced in detail. The architecture has a general layout based on existing standards and can be utilized on a large majority of the key elements in flexible manufacturing systems. The practical implementation of the new architecture is discussed in Section 4. Finally, the article is concluded in Section 5.

2 Related Work

In order to establish a highly flexible shop-floor architecture, we believe that a modularized software, component-based solution gives the best result. In this section the key elements of such a controller solution will be introduced.

The software of the controller should hide the details of hardware layers and provide a platform for standardized communication. Further, it should allow for the reconfiguration of manufacturing cells without hardware changes (within certain limits) and should serve as a platform for human-machine interaction. Every member of the shop-floor will typically have a software-component that will manage its life-cycle and provide services on different levels. The layered structure of such an ideal software component is shown in Fig. 1.

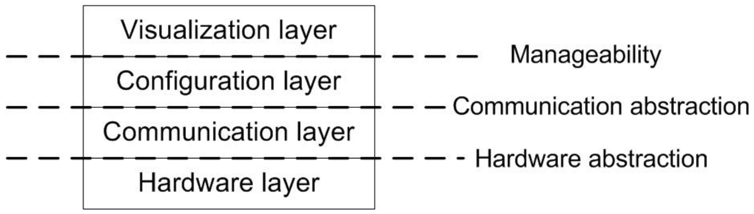

Figure 1
Manufacturing cell member's software layer-structure. Each layer hides the details of its' content. Connection can be established with any of the layers. Command complexity grows from bottom to top.

A member can be constructed from more than one component and, as it is software based, can be assembled on-line for every different task. Note that the definition of tasks within such flexible shop-floor control architectures relies mostly on detailed descriptions of manufacturing processes.

As an example, a 6-axis industrial robot and a 3-axis milling machine can be simultaneously controlled to obtain the functionality of a 9-axis machine. The robot can be used to support and re-orientate the work piece during some machining actions (typically light operations) carried out by the NC-machine.

Until now this would be solved using either I/O signalling and pre-programmed position and orientation information or through manual labour.

In our case, based on the work piece and manufacturing cell information, the shop-floor controller would be set up intuitively and automatically (with or without human-machine interaction) for the given task. To achieve this, two things are needed: 1) a uniform description language for manufacturing processes and 2) a uniform shop-floor/machine/component control language. These two will be introduced in the next subsection.

2.1 STEP-NC Standard

Today, medium and small-scale production is centred around automated and flexible manufacturing cells. These cells usually consist of computer numeric controlled (CNC- or just NC-) machines, industrial robots, material storages, conveyor belts, etc. Each component in the cell has its own control architecture and communication link to other machines or humans. Even in recent NC machines, the communication is limited to Input/Output (I/O) level changes, which are used for signalling from one machine to another or signalling to the shop floor controller. This allows for only severely limited communication and limits flexibility. For example, in many cases the automatic opening and closing of a door clutch still relies on manual interventions to the machine's control system.

As early as in the 1960s, when flexible manufacturing cells appeared, considerable emphasis was laid on their autonomous operation. However, this was only achieved through the careful planning of the product itself, as well as production and machine-specific tasks by a highly qualified human, using the concept of computer integrated manufacturing (CIM) [9].

In the 1970s issues such as sustainability, fast response time to market changes and cost-effectiveness made the utilization of CIM unsustainable [10]. As a result, the concept of CIM reappeared from time to time in new forms, such as holonic manufacturing systems [11], agent-based manufacturing [12], virtual CIM [13], etc. The bottlenecks of most manufacturing cells are the lack of interoperability, program reusability and fast re-configurability. However, the new CIM alternatives focus mainly on macro- and micro-process planning without addressing these important bottlenecks.

The standard for the programming of CNC machines was originally developed in the 1960s (ISO-6983 [14]) and is still used every day with all of its original weaknesses [15]. As pointed out by [16]:

1) It is a low-level language that describes mainly the cutter location of the spindle. Modification of geometrical properties on such a low level is almost impossible. File sizes grow very fast with the level of complexity.
2) It contains several machine-specific language elements (e.g. waiting condition, loops, etc.) which result in a lack of interoperability between machines.
3) The resulting file (to be used uniquely by the machine) is a result of a one-way transformation. It is not possible to feed back any kind of change from the shop-floor to the design stage.

To overcome these problems, new standards were introduced [17] to replace the old one and relieve the user from its constraints. One of the most promising standards is STEP (Standard for the Exchange of Product Model Data - ISO-10303 [18]) and its combination with the STEP-NC - ISO-14649 [8] – (the NC extension of the original STEP).

The benefits of using the STEP-NC standard can be summarized as follows:

a) Shorter time-to-market interval due to self-contained work piece and process planning information in one place
b) Changes can be tracked at any stage even during manufacturing
c) Machines are more efficient and adaptive and hybrid control theories can be applied to manufacturing processes
d) Machine vendor independence

These two standards are the largest and most widespread in the ISO group and are relatively new: STEP was created in 1994 and STEP-NC was created in 2003.

Although these standards have been available to manufacturers for over 8 years, machine manufacturers are seemingly not willing to switch from the older standard. Our guess, as mentioned in the introduction, is that manufactures build on previous versions of their products in order to keep the compatibility. On the other hand, there are several government-funded international projects [19]-[22] that promote standards among large manufacturers (e.g. Boeing, Daimler Chrysler, Volvo). At the same time, many researchers from all over the world [15], [17], [23]-[28] are promoting and developing solutions for STEP and STEP-NC based machining.

It is important to note, however, that dealing with only one aspect of the whole manufacturing cell will not solve every problem. Program portability from one vendor's machine to another's can be solved using newer standards, but the inter-machine communication will still be limited to hard-wired I/O signalling.

2.2 RT-Middleware Framework

Industrial robots have had a slightly different history of development than the typical NC-machine. More specifically, there are two major differences between the two: 1) industrial robots do not share any common language and from the outset, industrial robots were well suited for I/O communication with other machining cell components (this has led to their widespread use as system integrators and shop-floor controllers), and 2) industrial robots use vendor-specific and, in many cases, even model-specific languages.

Typically, NC-machines and industrial robots have proven to be mechanically very long-lasting, while the electrical control systems have become more rapidly outdated. Lately, there has also been a growing quest for advanced robot-sensor integration, at a completely different level than before.

Due to such issues, many researchers have turned to new and open control architectures for robot systems (so-called middleware systems) [7]. The overall goal is to provide a common language and control structure that would allow for user/task-specific sensor integration. The most important middleware solutions can be found in [29]-[40] and a comparison can be found in [41]. [41] also reports that these systems in general provide the user with high-level object-oriented robot programming and control environments. Issues related to specifics, like the number of supported robots, may vary. It is also reported that only one of these middleware technologies - called RT (Robot Technology)-middleware - is under standardization [42].

RT-middleware has proven to be well-adapted to industrial projects and is in use by many different industrial companies (Toshiba, Honda, AIST and also other research institutes) [38].

In 2002, the Japanese Ministry of Economy, Trade and Industry (METI), the Japan Robot Association (JARA) and National Institute of Advanced Industrial Science and Technology (AIST) started a project named "Consolidation of Software Infrastructure for Robot Development". The goal of the project was to implement humanoid robot systems in a modular fashion so that they could meet the diverse needs of users. Further, the project aims to allow system designers or integrators to build versatile robots or systems with relative ease by simply combining selected modular parts [43]. RT-middleware was originally developed for humanoid robots. However, humanoids, industrial robots, and NC-machines are built up from the same building blocks, at a component level (motor/axis control logic form the basis for all of these machines).

The RT-middleware framework combines low-level components (named RTCs, RT-Components) into larger operating units (e.g. like an industrial robot). If the robot needs more sensors or an extra drive axis, software drivers (RTCs) can be written, allowing the sensor/axis to be introduced in the existing robot control architecture. The RT-middleware environment offers various tools for including and even merging together the different RTC units. RTCs are realized as platform independent, CORBA objects. The use of CORBA in shop-floor control architectures has been found to be favourable in other research areas as well [44].

RTCs are the basic, modular units for achieving distributed computing. These elements can be observed as black-boxes. Each black-box has predefined interfaces for communication and the data manipulation and calculation is hidden from the external parties. This results in a highly modular framework, which was already shown in Fig. 1.

More details about the RT-middleware can be found in references [17, 38].

3 Shop-Floor Control Architecture Description

Until now the control of flexible manufacturing systems was limited to I/O level signalling, using an industrial robot or a PLC as an overall controller. The proposed architecture is based on software components (RT-Middleware) and on standards (STEP family). The shop-floor controller is a normal PC connected to the high-speed communication network of the shop-floor. This PC serves as an interface for system assembly and task specifications.

Fig. 2 shows the proposed shop-floor control architecture. At the bottom level (*Cell components*) we have all the cell members represented by NC-machines, industrial robots, cameras, sensors, conveyor belts, feeders, etc.. In general, all cell members that have the ability to communicate are defined as active cell components. Cell members, if possible or needed, are built up from smaller components, forming controllable objects where the communication drivers between the software components are based on the RT-middleware framework. These components represent building blocks for the manufacturing system and are called RT-Components or RTCs. Depending on the specified task, these RTCs can be used as fully standalone cell members or can be assembled in a way that will satisfy the needs posed by a given task. For better understanding, the following example is given (see Figs. 3 and 4): a typical SCARA robot is built up from 4 drive units. 3 drive units are used for positioning and the 4th is used for orientation. The 4 drive units can be represented by 4 standalone RTC components or can be attributed to 1 RTC that has all the necessary functions (kinematics, dynamics, interpolation) included and is capable of the execution of high-level commands (e.g. go to x, y, z position), while the first option uses 4 independent, low-level axis drives that can only understand commands such as turn to the given angle.

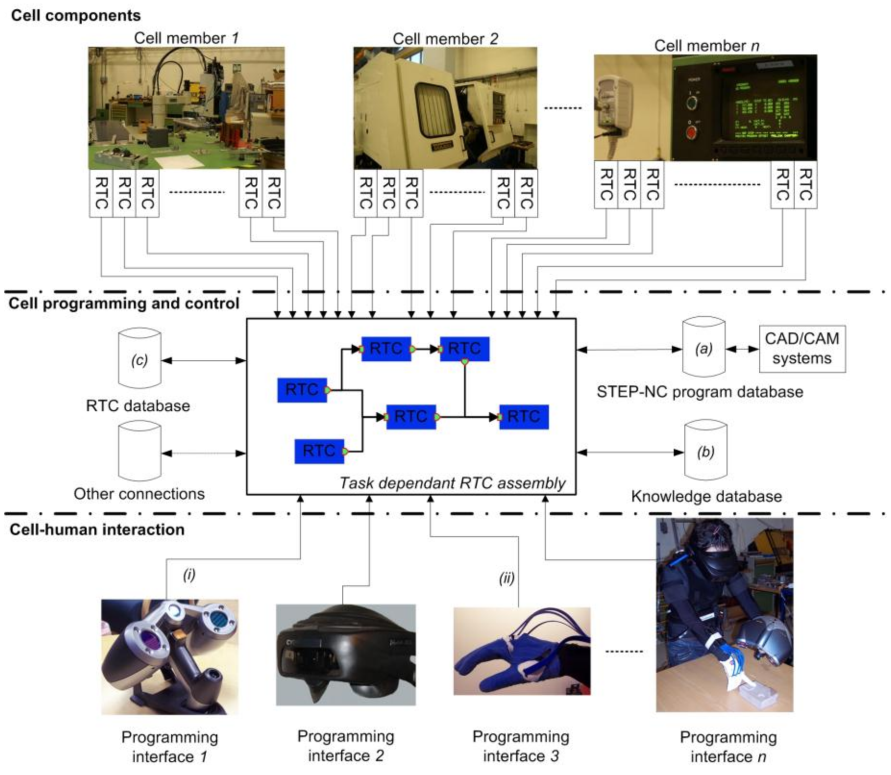

Figure 2

Shop-floor control architecture. The architecture is assembled for every given task, based on the databases contents (STEP-NC work-piece and process description (a), previous knowledge of task executions (b), available cell members and tools (c), etc.) and on the used programming interface (3D scanner (i), motion capture (ii), etc.). New cell members can be created on-the-fly (within certain limits).

If sensors are attached to the SCARA robot's end-effector (e.g. force sensor, which is also an RTC), the measurement values of the sensors can be directly fed to the axis drive RTCs, resulting in a closed-loop controller as low as the drive level for a specific drive or can be fed to the whole SCARA robot, affecting positioning or other parameters in the control.

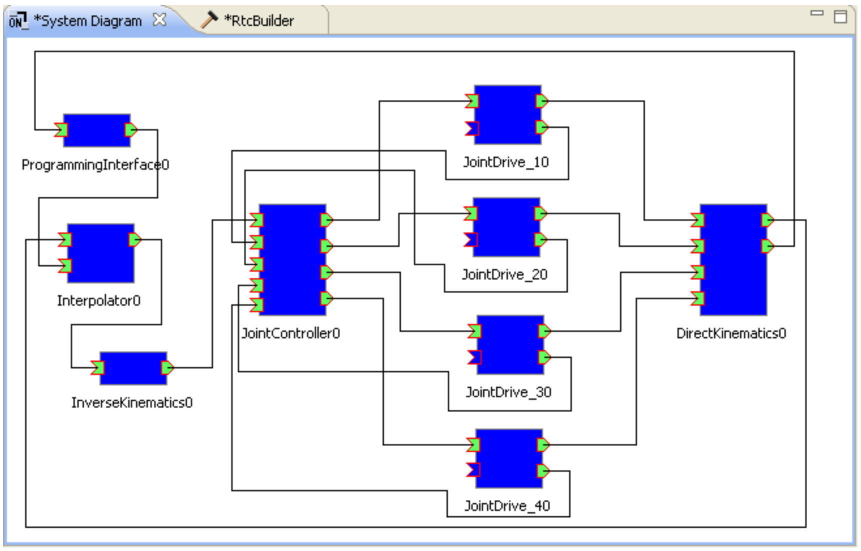

Figure 3

Low level control software component diagram. Every blue box is a standalone RTC. A SCARA robot is built up from 4 independent drive units (JointDrive). Each drive unit gets the reference torque from the JointController, which calculates the desired rotation based on the data received from the InverseKinematics component and the positioning feedback. The movement is split up by the Interpolator, using the feedback data from the DirectKinematics and the instruction from the ProgrammingInterface. The robot can be instructed by the ProgrammingInterface component, which can also represent a joystick or a mouse.

When the necessary RTCs are developed and implemented they are all registered in to the software component database (*RTC database*). From this database, the shop-floor controller can select and combine RTCs according to the specific task-dependent needs. This selection can be automatic or can be based on human-machine interaction using one of the *Programming interfaces*. The decision is also supported from the *Knowledge database*, where the previous task executions are stored with the corresponding RTC assemblies and also from the *STEP-NC program database*, where the actual manufacturing processes and work piece information are defined. A further strength of the software based RTCs is that they can be situated anywhere, as long as they are reachable by the controller (anywhere on the network connected to the controller) and it is possible to receive control messages from other sub-manufacturers (suppliers) or distributors in the supply chain.

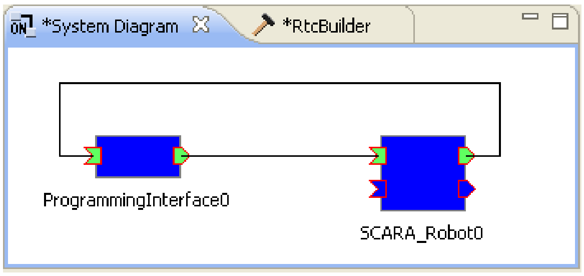

Figure 4
High level control software component diagram. Every blue box is a standalone RTC. With this configuration the SCARA robot is instructed by high level commands (e.g go to given position with linear interpolation).

Combining RTCs into new and higher level RTCs provides us with the opportunity to increase the level and complexity of the commands. This is very important in order to be able to incorporate the STEP-NC standard. The main idea behind the STEP-NC standard is to give the machine the task and let it execute and interpret it independently, based on the local logics and control, which is a much better approach than the previous one, where the predefined movement of each axis was given to the machine. All actions in the manufacturing cell, and all modifications to the geometry of work pieces, are done according to the STEP-NC standard, thus allowing the user to track changes through the manufacturing chain. For these reasons, a STEP-NC interpreter as well as logger RTCs are created, which can be used to convert the high-level commands to joint-level ones and to report changes made to the manufacturing process.

In existing manufacturing equipment, man-machine interaction is often carried out via a screen/keyboard/mouse interface. These communication links will also be important in future manufacturing systems, although more advanced and human friendly interaction systems will most likely be created as well.

The simplest *Programming interface* is the PC's screen, where a special graphical user interface is used to assemble the shop-floor controller (a part of this interface was already shown in Figs. 3 and 4), named RTSystemEditor [38]. More advanced interaction modes can be achieved using a 3D scanner during the inspection procedure or by instructing an industrial robot through human gestures [45]. Distant programming of shop-floor controllers is possible by introducing Virtual Reality techniques [41].

The above described shop-floor control architecture is under implementation at our laboratory. No shop-floors are identical but normally similar components are present (NC machines, industrial robots, conveyor belts, coordinate measurement machines, material feeders, digital cameras, PCs, etc.) and in the next section we would like to share our experiences when it comes to practical implementation towards some typical components.

4 Practical Implementation at a Typical Shop-Floor

In general every member of a flexible manufacturing cell has its own controller (NC machines, industrial robots) or supervisory system (PLC or PC). The proposed shop-floor controller replaces these with software components which can be configured and assembled for every given task. In order to achieve the goals stated in the introduction the cell members can and actually should be constructed from low and high-level software components (e.g. axis drives, sensors). This can be done in two ways: 1) by implementing a wrapper for the given cell member, as one standalone unit (in the beginning this is the most convenient) [4] or 2) by re-engineering the controllers from low-cost highly modularized electrical components [5].

The first solution's biggest drawback is that it is limited to the actual capabilities of the already existing controller. In the simplest case, a PC is connected to the given cell member (e.g. serial link, Ethernet or LPT) and the software component (RTC) is implemented on the PC. The RTC is simply converting the instructions from RT-Middleware to member-specific instructions (e.g. STEP-NC instructions to G code in case of an NC machine [46]).

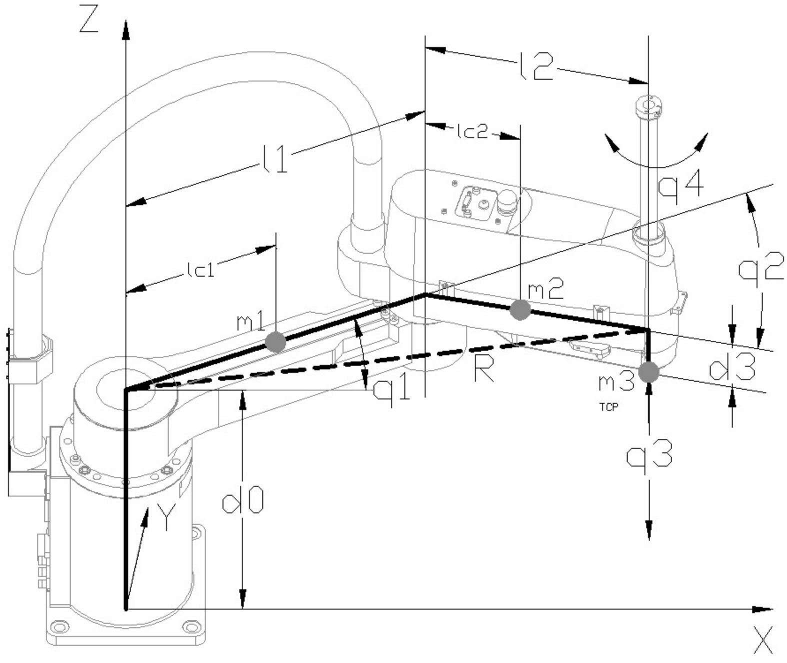

Figure 5

Mechanical drawing of the Adept 604-S SCARA robot. m1, m2, m3 are the masses, l1,l2,d0,d3 are the length, q1,q2,q3,q4 are the angles of the corresponding joints. lc1 and lc2 are the masses position on joint 1 and joint 2, respectively. These data were used in calculations of robot dynamics [47]

In the second case, re-engineering the controller, everything has to be developed and implemented from scratch, but as a result we obtain a high level of flexibility and a truly software-based controller.

In our experimental system, we chose this second option to exemplify the proposed shop-floor controller. The architecture was validated through several experiments.

The experimental shop-floor controller is now limited to one cell member, an Adept 604-S SCARA robot, whose control system is replaced by an RT-Middleware based controller. The mechanical drawing of this SCARA robot is shown in Fig. 5.

4.1 Software and Hardware Layout

To match the mechanical setup of the robot, the following standalone hardware and software components were created (RTCs):

a) *Programming interface*: this is a PC-based graphical user interface, in which the user can enter coordinates for the robot to go to. It also displays the current position of the robot. The input of the component is the actual position of the robot in Cartesian coordinate system. The output is the desired position of the robot.

b) *Interpolator*: based on the actual and desired positions, it plans the trajectory of the robot. The inputs of the component are the desired and the actual position of robot in Cartesian coordinate system. The output is the steps position through the trajectory in Cartesian coordinate system.

c) *Inverse Kinematics*: the component calculates the motor angles for the desired movement based on the actual position. The input of the component is a position on Cartesian coordinate system. The output is the angles of the robot axis for the given position.

d) *Direct Kinematics*: based on the motor angles it calculates the actual position of the robot. The inputs for the component are the angles from the axis drives. The output is the robot’s actual position in Cartesian coordinate system.

e) *Joint Controller*: the component uses the desired motor angles to give torque references for the axis drives. The inputs for the component are the angles for rotation of the motors. The outputs are the torque references for every axis drive.

f) *Axis drive for every joint*: the 4 motor drives are used independently. Each joint has a current control loop inside. The input for the component is the torque reference signal. The output is the angle of the motor, which is calculated from encoder position.

The block diagram based on the above mentioned components is shown in Fig. 6.

This modularized structure of the controller gives the possibility of including sensor data in the control loop on different control levels. The sensor data can be introduced as low as at the *Axis drive* level or at *Joint Controller* level or even higher. The synchronization of the data and the real-time execution of the control loop are guaranteed by the RT-Middleware framework.

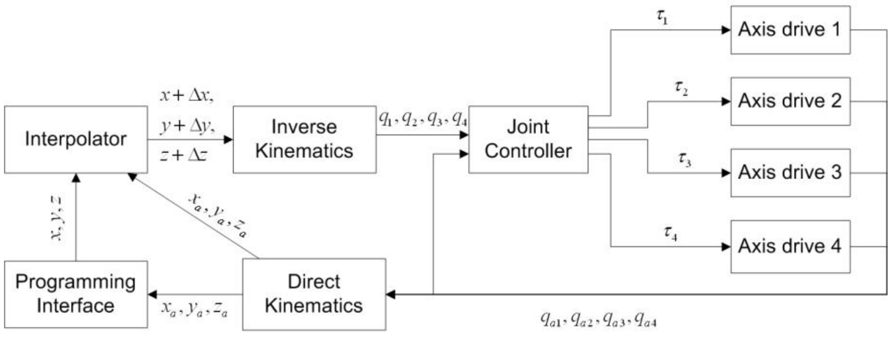

Figure 6
Block diagram of the SCARA robot's controller. x, y, z are the desired point's Cartesian coordinates. x_a, y_a, z_a are the robot actual position's Cartesian coordinates. $x+\Delta x, y+\Delta y, z+\Delta z$ are the robot's next position's Cartesian coordinates. q_1, q_2, q_3, q_4 are rotation angles. $\tau_1, \tau_2, \tau_3, \tau_4$ are torque reference signals. $q_{a1}, q_{a2}, q_{a3}, q_{a4}$ are angles derived from the encoders.

4.2 Experimental Results

In order to validate the control architecture, experiments were carried out on the major joints (Joint 1 and 2) of the SCARA. The software component based controller's main benefit is that each joint is driven independently, but this is also its biggest drawback when designing the controller for the given system.

A robust, decentralized PID control was implemented in the *Joint Controller*. Two different experiments were executed and were repeated two times to ensure the correct measurement. The feedback system's sampling time was 1 *kHz*.

The first experiment tested the step response of the first joint. The results can be seen in Fig. 7. The feedback shows a negligible overshot. Stabilisation time is approximately 200-300 ms, which is to be considered very good for an industrial robot.

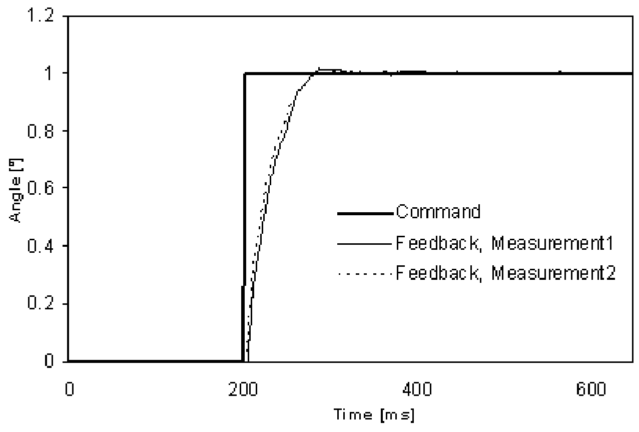

Figure 7

Step response of the first joint. Result of two measurements [48]

The second experiment tested the step response of the second joint in two different arm positions. The first measurement was carried out with straightened arms and the second measurement with the second arm perpendicular to the first. The results can be seen in Fig. 8. The feedback shows a more significant overshoot compared to the first experiment and a longer stabilisation time of 400-500 ms. Such results were expected since this is a SCARA (selective compliance articulated robot arm), especially developed for assembly operations. The compliant structure of joint 2 plays an important role in assembly operations, and the horizontal flexibility of the robot arm is favourable during mounting operations, allowing for initial misalignment between the mating parts.

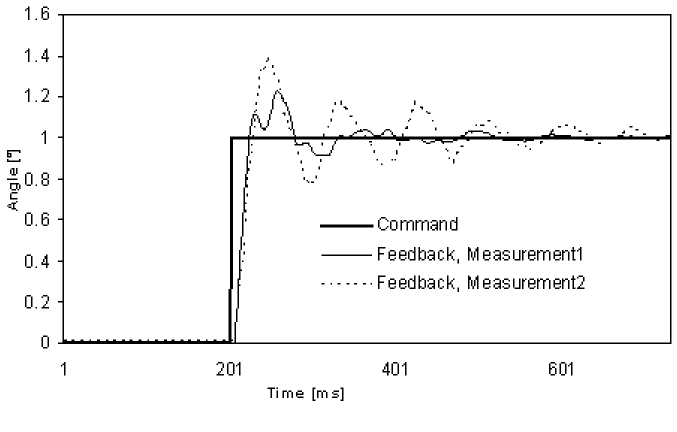

Figure 8

Step response of the second joint, with decentralized PID control. The Measurement 1 is with straightened arms, while the Measurement 2 is with the second joint perpendicular to the first one [48]

Conclusions

This paper presented a shop-floor control architecture for inter-machine communication and control, based on the RT-middleware framework and the STEP-NC standard. The great variety, diversity and complexity of equipment used in manufacturing cells call for an open control architecture capable of integrating all of its members. The proposed methodology is general in its layout and emphasizes openness to the largest extent. The implementation of this new architecture was exampled on a typical shop-floor member (a SCARA robot), which is driven by low and high level commands.

Acknowledgement

This work has been supported by HUNOROB project (HU0045), a grant from Iceland, Liechtenstein and Norway through the EEA Financial Mechanism and the Hungarian Focal Point.

The authors would like to thank Károly Szell, Bence Kovács, Géza Szayer, Balázs Varga, Ferenc Tajti for their contribution to the development process.

References

[1] S. T. Newman, A. Nassehi, X. W. Xu, R. S. U. Rosso Jr., L. Wang, Y. Yusof, L. Ali, R. Liu, L. Y. Zheng, S. Kumar, P. Vichare and V. Dhokia, Strategic Advantages of Interoperability for Global Manufacturing Using CNC Technology, Robotics and Computer-Integrated Manufacturing, Vol. 24, No. 6, pp. 699-708, Dec. 2008

[2] J. N. Pires, G. Veiga and R. Araujo, Programming-by-Demonstration in the Coworker Scenario for SMEs, Industrial Robot - An International Journal, Vol. 36, No. 1, pp. 73-83, 2009

[3] S. Ekvall and D. Kragic, Robot Learning from Demonstration: A Task-level Planning Approach, International Journal of Advanced Robotics, Vol. 5, No. 3, pp. 223-234, Sep. 2008

[5] J. U. Cho, Q. N. Le and J. W. Jeon, An FPGA-based Multiple-Axis Motion Control Chip, IEEE Trans. Ind. Electron., Vol. 56, No. 3, pp. 856-870, Mar. 2009

[6] T. Li and Y. Fujimoto, Control System with High-Speed and Real-Time Communication Links, IEEE Trans. Ind. Electron., Vol. 55, No. 4, pp. 1548-1557, Apr. 2008

[7] J. G. Garcia, J. G. Ortega, A. S. Garcia and S. S. Martinez, Robotic Software Architecture for Multisensor Fusion System, IEEE Trans. Ind. Electron., Vol. 56, No. 3, pp. 766-777, Mar. 2009

[8] International Organization for Standardization (ISO), ISO 14649-1, Industrial automation systems and integration – Physical Device Control – Data Model for Computerized Numerical Controllers Part 1: Overview and fundamental principles, 2003

[9] J. Harrington, Computer Integrated Manufacturing, New York: Industrial Press Inc, 1973

[10] J. Johansen, U. S. Karmarkar, D. Nanda and A. Seidmann, Business Experience with Computer-integrated Manufacturing. A Survey of Current Strategy and Practice, In Proceedings of the Twenty-Eighth Hawaii International Conference on System Sciences, Vol. 4, pp. 970-979, 1995

[11] J. Christensen, Holonic Manufacturing Systems-Initial Architecture and Standards Directions, In Proceedings of the first European conference on holonic manufacturing systems, 1994

[12] D. Ouelhadj, C. Hanachi and B. Bouzouia, Multi-Agent System for Dynamic Scheduling and Control in Manufacturing Cells, In Proc. IEEE International Conference on Robotics and Automation, pp. 2128-2133, 1998

[13] D. Wang, S. V. Nagalingam and G. C. I. Lin, Development of an Agent-based Virtual CIM Architecture for Small to Medium Manufacturers, Robotics and Computer-Integrated Manufacturing, Vol. 23, No. 1, pp. 1-16, 2007

[14] International Organization for Standardization (ISO), ISO 6983-1, Numerical Control of Machine-Program Format and Definition of Address Words Part-1: Data Format for Positioning, Line and Contouring Control Systems, First Edition, 1982

[15] X. W. Xu and Q. He, Striving for a Total Integration of CAD, CAPP, CAM and CNC, Robotics and Computer-Integrated Manufacturing, Vol. 20, No. 2, pp. 101-109, 2004

[16] X. Zhu, Y. Wang, H. Fu, A 3-D Simulation System for Milling Machining Based STEP-NC, in Proceedings of the 2006 IEEE International Conference on Mechatronics and Automation, Dalian, China, 2006

[17] N. Ando, T. Suehiro, K. Kitagaki, T. Kotoku, and W.–K. Yoon. RT-Middleware: Distributed Component Middleware for RT (robot technology), in Proc. Computational Intelligence in Robotics and Automation, CIRA 2005, pp. 457-462, 2005

[18] International Organization for Standardization (ISO), ISO 10303-1, Industrial automation systems and integration - Product data representation and exchange Part 1: Overview and fundamental principles, 1994

[19] ESPRIT Project 8643, Optimised preparation of manufacturing information with multi-level CAM-CNC coupling (OPTIMAL): final report, ESPRIT, 1997

[20] ESPRIT Project 29708, STEP-compliant data interface for numeric controls, ESPRIT, 1999

[21] IMS Project 97006, STEP-compliant data Interface for Numerical Controls, STEP-NC Consortium, 2003

[22] National Institue of Standards and Technology (NIST) 99-01-4035, Model Driven Intelligent Control of Manufacturing, STEP Tools Inc., 1996

[23] S. T. Newman and A. Nassehi, Universal Manufacturing Platform for CNC Machining, Annals of the CIRP, Vol. 56, No. 1, pp. 459-462, 2007

[24] A. Nassehi, S. T. Newman and R. D. Allen, STEP-NC Compliant Process Planning as an Enabler for Adaptive Global Manufacturing, Robotics and Computer-Integrated Manufacturing, Vol. 22, No. 5-6, pp. 456-467, 2006

[25] A Nassehi, S T Newman, X W Xu, R D. Allen and R S U Rosso Jr., Adaptability and Interoperability in CNC Manufacturing, In Proceedings of the 3rd CIRP international conference on Digital Enterprise Technology, pp 1-8, 2006

[26] Xun Xu, STEP into Distributed Manufacturing with STEP-NC, In book Process Planning and Scheduling for Distributed Manufacturing, pp. 393-421, Springer London, 2007

[27] S. H. Suh, B. E. Lee, D. H. Chung and S. U. Cheon, Architecture and Implementation of a Shop-Floor Programming System for STEP-Compliant CNC, Computer-aided Design, Vol. 35, No. 12, pp. 1069-1083, 2003

[28] S. M. Amaitik and S. E. Kilic, An Intelligent Process Planning System for Prismatic Parts Using STEP Features, International Journal of Advanced Manufacturing Technology, Vol. 37, No. 16, pp. 978-993, 2007

[29] S. Magnenat, V. Longchamp, F. Mondada,"ASEBA, an Event-based Middleware for Distributed Robot Control" in Workshops of International Conference on Intelligent Robots and Systems (IROS), 2007

[30] H. D. Nayar, I. A. Nesnas, "Measures and Procedures: Lessons Learned from the CLARAty Development at NASA/JPL," International Conference on Intelligent Robots and Systems (IROS), San Diego, CA, October 2007

[31] J. Jackson, "Microsoft Robotics Studio: A Technical Introduction" in IEEE Robotics & Automation Magazine, Vol. 14, pp. 82-87 ISSN: 1070-9932, Dec. 2007

[32] A. Weitzendfeld, S. Gutierrez-Nolasco and N. Venkatasubramanian, "MIRO: An Embedded Distributed Architecture for Biologically inspired Mobile Robots" in 11th IEEE International Conference on Advanced Robotics (ICAR'03), 2003

[33] Ozaki Fumio, Oaki Junji, Hashimoto Hideaki, Sato Hirokazu, "Open Robot Controller Architecture (ORCA)" in Journal: Nippon Kikai Gakkai Robotikusu, Mekatoronikusu Koenkai Koen Ronbunshu, Vol. 2, 2003

[34] M. Mizukawa, H. Matsuka, T. Koyama, T. Inukai, A. Noda, H. Tezuka, Y. Noguchi, N. Otera, "ORiN: open robot interface for the network - the standard and unified network interface for industrial robot applications" in Proceedings of the 41st SICE Annual Conference (SICE 2002), Tokyo, Japan, Vol. 2, pp. 925- 928, ISBN: 0-7803-7631-55-7, Aug. 2002

[35] Nuno Lopes, Pedro Lima, "OpenSDK - An Open-Source Implementation of OPEN-R", Proc. of 7th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2008), Estoril, Portugal, 2008

[36] H. Bruyninckx, P. Soetens, B. Koninckx, "The Real-Time Motion Control Core of the Orocos Project" in Proceedings of IEEE International Conference on Robotics and Automation (ICRA '03), Heverlee, Belgium, Vol. 2, pp. 2766-2771, ISSN: 1050-4729, ISBN: 0-7803-7736-2, 14-19 Sept. 2003

[37] Matthias Kranz, Radu Bogdan Rusu, Alexis Maldonado, Michael Beetz, and Albrecht Schmidt, "A Player/Stage System for Context-Aware Intelligent Environments" in Proceedings of the System Support for Ubiquitous Computing Workshop (UbiSys 2006), at the 8th Annual Conference on Ubiquitous Computing (Ubicomp 2006), Orange County, California, September 2006

[38] N. Ando, T. Suehiro, K. Kitagaki, T. Kotoku, Yoon Woo-Keun, "RT-Middleware: Distributed Component Middleware for RT (robot technology)" in Proceedings of 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2005), pp. 3933- 3938, ISBN: 0-7803-8912-3, 2-6 Aug. 2005

[39] G. Veiga, J. N. Pires,K. Nilsson "On the Use of SOA Platforms for Industrial Robotic Cells" in Proceedings of Intelligent Manufacturing Systems (IMS2007), Spain, 2007

[40] J.-C. Baillie, "URBI: Towards a Universal Robotic Low-Level Programming Language" in Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2005), pp. 820- 825, ISBN: 0-7803-8912-3, 2-6 Aug. 2005

[41] B. Solvang, G. Sziebig, and P. Korondi, "Multilevel Control of Flexible Manufacturing Systems," in Proc. IEEE Conference on Human System Interactions (HSI'08), pp. 785-790, 2008

[42] Object Management Group (OMG), Robotics DTF, "Robotic Technology Component Specification," 1.0 beta 2, 2007. 08. 18.

[43] G. Sziebig, A. Gaudia, P. Korondi, N. Ando, B. Solvang, "Robot Vision for RT Middleware Framework," in Proc. IEEE Instrumentation and Measurement Technology Conference, pp. 1-6, 2007

[44] J. Shin, S. Park, C. Ju, H. Cho, "CORBA-based Integration Framework for Distributed Shop Floor Control," Computers & Industrial Engineering, Vol. 45, Elsevier, pp. 457-474, 2003

[45] G. Soros, B. Reskó, B. Solvang and P. Baranyi, "A Cognitive Robot Supervision System," In Proceedings of IEEE 7th International Symposium on Applied Machine Intelligence and Informatics, pp. 51-55, Jan. 2009

[46] M. Hardwick and D. Loffredo, Lessons Learned Implementing STEP-NC AP-238, International Journal of Computer Integrated Manufacturing, Vol. 19, No. 6, pp. 523-532, Sep. 2006

[47] B. Kovács, Development of a Universal Robot Controller: Development of FPGA Hardware and Software, Modelling of a Scara robot, B.Sc Thesis, p. 35, Budapest University of Technology and Economics and Narvik University College, 2009

[48] G. Szayer, Development of a Universal Robot Controller: Hardware Development, Designing and Implementing Central Motion Control, B.Sc Thesis, p. 38, Budapest University of Technology and Economics and Narvik University College, 2009