A Real-Time Reconfiguration Infrastructure for Distributed Embedded Control Systems

Alois Zoitl, Wilfried Lepuschitz, Munir Merdan
Automation and Control Institute
Vienna University of Technology
Gusshausstrasse 27-29, 1040 Vienna, Austria
{lepuschitz,zoitl,merdan}@acin.tuwien.ac.at

Mathieu Vallée
Institute of Computer Technology
Vienna University of Technology
Gusshausstrasse 27-29, 1040 Vienna, Austria
vallee@ict.tuwien.ac.at

Abstract

During the last decades production automation research has been focused on improving the flexibility and adaptability in order to cope with the arising challenges of mass customization. Much work was devoted to the higher—planning and scheduling—levels of automation systems. However, the lower level real-time control infrastructure was widely neglected. Therefore, we are currently faced with an adaptive flexible high level control connected to a rigid low level control. This work presents a dynamic reconfiguration architecture for the low level control of industrial automation systems. This architecture allows to reconfigure real-time control applications during full operation of the controlled plant. In an example implementation we prove that this is also possible on small embedded control devices as they are typically used as field devices in industrial automation systems.

1 Introduction

Faced with the mass customization trends, production systems are required not only to ensure a high productivity, but also a rapid response to market changes and customer needs. For addressing these requirements, a reconfigurable manufacturing system is desirable. Such a system should be able to cope with dynamic behaviors, to change quickly and cost-effectively from its current configuration to another configuration without being taken off-line, and to maintain effectiveness when sudden changes in customer demands or unpredictable events such as failures and disruptions occur [9]. It should also integrate heterogenous networked devices, and enables the fast integration of new technology and/or new functions into existing systems as well as the reduction of lead time for launching new systems [13]. A study of USA’s National Research Council identified and highly prioritized reconfigurable manufacturing as one of the six key manufacturing challenges for the year 2020 [15]. Nevertheless, the rigid character and weak adaptation capabilities of the current manufacturing systems, which have centralized and hierarchical control structures, limits their ability to respond efficiently and effectively on dynamic changes.

The multi-agent approach, bringing the advantages of modularity, decentralization, autonomy, scalability, and re-usability, has been widely recognized as an enabling technology for designing and implementing the next-generation of distributed and intelligent production systems [4, 14, 7]. A multi-agent system (MAS) can be defined as a network of autonomous, intelligent entities—agents—where each agent has individual goals and capabilities as well as individual problem-solving behaviors. Following the MAS approach, reconfiguration of a manufacturing system can be supported by a distributed embedded control system, in which control of individual machines is performed in an autonomous way. Research efforts in this direction have especially addressed reconfiguration as modifying planning, scheduling, and resource allocation in a production process [18, 12, 16, 17]. Those works are concerned mainly with system-level reconfiguration, which does not modify the agent’s internal and local control of physical components.

In this paper, we consider the opportunity for self-reconfiguration—internal reconfiguration of the control software within one automation agent—enhancing and extending usual reconfiguration approaches. We present a real-time reconfiguration infrastructure for distributed embedded control software, and we detail its relevant features. The main advantages of our approach are support-
ing reconfiguration in the control software at a fine granularity and satisfying real-time requirements during the reconfiguration process.

This paper is structured as follows. Section 2 introduces substantial motivation examples that deliver requirements which are detailed in Section 3. The reconfiguration infrastructure is presented in Section 4 followed by a description of the actual implementation and its results in Section 5. Finally, section 6 concludes this paper with a summary and an outlook.

2 Reconfiguration Possibilities in a Distributed Control System

We first give some examples of reconfiguration possibilities as a motivation for our proposed reconfigurable control infrastructure. We consider a flexible manufacturing plant as depicted in Figure 1. This example is inspired by the “Testbed for Distributed Holonic Control”, located at the Automation and Control Institute of the Vienna University of Technology. Such a manufacturing plant consists of components such as handling stations for executing actual production tasks like the assembly of parts as well as components for supporting tasks like the transport of pallets carrying parts and finished products.

A distributed intelligent control system is in charge of controlling the operation of this flexible manufacturing plant. In particular, each physical component (e.g., a handling station, conveyor or intersection) is controlled by one automation agent [19]. More precisely, an automation agent is an autonomous part of the system, consisting of the physical component itself, low-level control software (LLC) and high-level control software (HLC). The HLC manages the role of the component in overall production tasks, coordinates with other elements of the manufacturing system and performs diagnosis. The LLC is closely related to the hardware, performs real-time control, and delivers primitive actions used by the HLC functions. The whole manufacturing system thus forms a multi-agent system, whose organization and behavior is adapted depending on the situation and disturbances happening during operation.

Such a control system offers various reconfiguration possibilities. Capacity reconfiguration [17] deals with distributing the work load among available machines. Reconfiguration for error-handling [3] considers alternatives to cope with errors. Redundant paths within the transport system, consisting of conveyors and intersections, ensure fault-tolerance in the case of component failures and provide possibilities for enhanced scheduling algorithms as they allow the prioritization of pallets. A number of works is considering this kind of system-level reconfiguration, strongly related to the overall planning and scheduling of production plans.

Changing the direction of conveyors in the transport system is a specific feature provided by our testbed, which offers rich reconfiguration possibilities. A change of a conveyor’s direction might be necessary to provide the reachability of all possible destinations such as storages or work stations [8]. Specific three-way-switches at the intersections of conveyors are employed that can cope with direction changes. Hence, this type of intersection can act as a diverter or as a junction depending on the system’s configuration. Internal reconfiguration of the intersection’s LLC is desirable to use the appropriate control software depending on the intersection’s role (diverter or junction). Furthermore, reconfiguration is subject to time constraints, as it may only be possible to perform it at certain time points, e.g. for ensuring that no pallet is currently in the intersection.

Besides online reconfiguration necessary to adapt to dynamic changes, we also consider the capability to produce with lot sizes down to 1, leading to a large variety of products. Handling issues like grasping therefore represent difficult challenges as machines are required to grasp these different products in individual ways depending on their individual attributes. Information about the required type of grasping can be stored in the production plan carried by the product itself. As soon as the product reaches the machine, the necessary information is sent to its agent which reconfigures its LLC accordingly to grasp the product. In case the production plan does not include grasping information, the machine agent may have to determine how to grasp the product by using a CAD model or by inspecting the product, which might be especially important regarding disassembly processes. Hence, exact grasping information cannot be anticipated beforehand in such a case. Having a reconfigurable LLC offers the possibility to cope with such a situation as it can be modified according to the product’s attributes.

The assembly or disassembly of heterogeneous products often does not require only different types of grasping but also completely different moving sequences of the handling machines. These can be realized by employing mechanically reconfigurable machines that utilize re-
placeable axes and tools. Hence, mechanically reconfigurable machines offer too many possible configurations for programming rigid control software in advance. Reconfigurable control software approaches are required to cope with the flexibility provided by reconfigurable machines [1].

Especially mechanically reconfigurable machines but also a wide variety of conventional machines rely on different closed-loop control paradigms. While specific ones are used during nominal operation, other ones are applied in the case of a failure. But even during nominal operation several different types of closed-loop control paradigms might be employed as they are designed for different machine states [5], which is especially fundamental in the case of mechanically reconfigurable machines. Closed-loop controllers are most likely to be placed in the LLC as they are real-time critical applications. Hence, a modification of an applied closed-loop controller leads to a reconfiguration of the hosting LLC.

3 Requirements for Reconfiguration Tasks

The described flexible manufacturing plant relies on reconfigurable low level control software to provide functionality both for normal operation as well as measurements in the case of occurred failures. Depending on the actual case, a reconfiguration process can encompass the modification

- of one or several parameters, which is sufficient e.g. for adapting a closed-loop controller without changing its type;
- of an execution sequence, which is needed e.g. for the assembly of heterogeneous products that require different assembly steps at a specific machine;
- of the information flow, which is used for instance to create and modify communication paths when the manufacturing system is extended or reduced by physical components;
- of a physical component’s provided functionality by changing or substituting parts of its control software elements, which is applied for example at intersections that switch from providing diverting functionality to junction functionality or the other way round.

In order to apply reconfiguration processes to control software, specific requirements have to be taken into account. A modular software architecture, like a component-oriented software architecture, is required for dynamic reconfiguration. Such an architecture allows the modification of one component without interfering with the execution of other components. To ensure the structural integrity of the overall system the reconfiguration process of a single component and its impact should have only local effects [20]. As a reconfiguration process is triggered due to a specific event, such as the detection of a component failure at a specific point in time, an execution environment for reconfiguration supporting event-triggered execution is advisable [6]. The LLC has to interact with physical components, real-time constraints have to be taken into account. Therefore, a reconfiguration process has to meet real-time requirements such as timeliness, responsiveness and predictability [20].

4 A new Real-time Reconfigurable Low-Level Control Infrastructure

As pointed out by Brennan et al. [2]:

“Although there has been a considerable amount of work on agent-based approaches to the upper planning and scheduling level of control, very little work has been done on applying these techniques to the lower, real-time control level. The main barriers at the real-time control level result from the difficulty of implementing multiagent systems (MAS) concepts in a stochastic environment where hard real-time constraints must be met to achieve safe system operation.”

Currently no real-time reconfigurable control infrastructure exists that fulfills all the needs and requirements identified for automation agents [20]. In order to overcome these limitations we propose a new LLC infrastructure that on the one hand is in real-time reconfigurable and on the other hand fulfills the requirements of industrial control systems. As IEC 61499 is well suited for supporting reconfiguration we will base our infrastructure on the definitions made in this standard.

A key property of a real-time reconfigurable control infrastructure is that the reconfiguration process does not disturb the controlled process. That means that at no time a wrong stimulus is given to the process and no state change in the process is missed. Kramer and Magee have been one of the first that investigated how a reconfiguration process has to be conducted and what infrastructure is necessary to fulfill this key property [10]. They introduced a so-called configuration manager which is in charge for conducting the reconfiguration process. In order to do this it uses a change specification for transforming the existing system configuration into the desired system configuration. For this the configuration manager needs knowledge of the existing system configuration, the system’s state, and the key properties of the software components involved. Based on this information it determines a sequence for applying the changes and also the prerequisites for each change. As can be imagined this is not an easy task for a typical control application. Their solution utilized large databases containing all required information [11]. Therefore, this approach is not directly applicable to the real-time control layer of industrial automation systems, where control hardware typically is size and performance constrained.
4.1 Programmable Reconfiguration Management

In order to overcome the limitations of the general reconfiguration manager while still having the advantages of this architecture we propose to replace the general reconfiguration manager with a programmable reconfiguration management. Such a programmable reconfiguration management provides the infrastructure for performing reconfiguration to a dedicated application. We call this application reconfiguration application (RCA). An RCA is specifically tailored for each reconfiguration process it has to conduct. During execution the RCA interacts with the application under reconfiguration in order to gather the current application state, change the application, and ensure that all elements of the application are always in a consistent state. The reconfiguration infrastructure provides three dedicated interfaces for RCAs that enable these tasks (see Figure 2).

A typical automation agent’s LLC is a real-time constrained control application meaning this control application has to react within certain time limits to process state changes. For the reconfiguration of such an application this requires from the reconfiguration process to fulfill all timing constraints. Therefore, RCAs are only allowed to perform reconfiguration tasks at specific times during an application’s execution and they need to complete their tasks within a certain time. Hence, also RCAs have to fulfill real-time constraints.

We identified IEC 61499 as base architecture for the LLC of the automation agent. In order to simplify the overall system design it can make sense to utilize IEC 61499 also for RCAs. Apart from the advantage of having only one language for the application specification this has the great advantage that the interaction interface between the RCA and the application under reconfiguration can be greatly simplified. The monitoring of events and data, the setting of certain data values, and the issue of events can now be achieved with event and data connections between these two applications.

4.2 Basic Reconfiguration Services

An important element in our reconfiguration architecture is the interface that allows the RCA to change the application under reconfiguration. This interface on the one hand needs to be powerful enough to allow any reconfiguration process and on the other hand needs to be lightweight enough so that it can be used even on small control devices with limited resources. In [20] we were able to identify a set of so-called basic reconfiguration services that fulfill these properties. These services are the minimal set of reconfiguration services necessary to perform any reconfiguration by executing an according sequence. The basic reconfiguration services can be grouped into the following classes:

- **Structural Services** allow to change the structure of the control application. These services include the creation and deletion of Function Blocks (FBs) as well as connections and the changing of parameters.
- **Library Services** allow to add or remove type definitions (e.g., FBs or data types) in the control device.
- **Execution Control Services** allow to change the execution state of FBs, which is needed to control if FBs shall respond to events or not.
- **State Interaction Services** allow to acquire or change the state of an FB. An FB’s state is represented by its input and output data as well as by its internal data. These services are needed for performing transition management algorithms.
- **Query Services** allow to retrieve the application’s current structure. These services include the retrieval of instantiated FBs, connections as well as their types.

Furthermore, in several cases it makes sense that the reconfiguration entity does not download an actual RCA but directly performs the reconfiguration process. Examples for such cases are very limited devices that have not enough spare capacity for holding an RCA or simple reconfiguration processes that do not need to fulfill real-time constraints. For these cases the infrastructure provides the basic reconfiguration services also as an external interface which can be accessed via communication from external tools.

4.3 Real-time Reconfiguration

The greatest benefit can be achieved from the presented infrastructure when the reconfiguration process can be conducted while the plant is operating and kept operating during the whole reconfiguration process. This avoids costly shutdown and ramp-up times. In order to achieve this a careful planning and implementation of the RCA is...
necessary. The RCA has to ensure that during the reconfiguration process at no time the application is left in an inconsistent state and that no wrong control values are given to the plant. As industrial automation systems are real-time constrained systems a wrongly timed correct control value is as dangerous as a wrong value.

In our analysis of reconfiguration tasks we identified two main elements: tasks that directly change the behavior of the control application and tasks that don’t change it. The second group contains all tasks that perform preparational or clean up work of application elements not directly involved in the control task. This includes the creation of new FBs, interconnecting these, setting their parameters, and the deletion of unused FBs and unused connections. Critical tasks are changing the execution flow, setting parameters of FBs in the execution flow, or transferring the state of an FB to a new one that will replace it. In order to minimize the disturbances resulting from the reconfiguration process, the critical part should be kept as short as possible.

Therefore, we propose to structure an RCA into the following three parts:

1. Setup phase: prepare the control application for reconfiguration
2. Execution phase: switch the execution to new application parts
3. Shut-down phase: clean up the remaining parts of the original control application

This has the advantage that the critical parts are clearly identified in an RCA.

For larger reconfiguration tasks one single RCA may get to large and complicated. Furthermore, the execution phase may get to long to fulfill the non disturbance requirement. For these situations it is better to split the large RCA in a sequence of shorter RCAs. Each of these contains the three phases and leaves the application in a consistent state. This has the further advantage that common reconfiguration tasks (e.g., adapt PID-controller gains) can be implemented once and reused for many reconfiguration processes.

5 Real-time Reconfiguration Example

In order to test our approach we will reconfigure a closed loop control application during full operation. If our approach and assumptions are valid the reconfiguration process must have no effect on the controlled plant.

5.1 Experimental Setup

As target application we have chosen the balancing of an inverted pendulum as shown in Figure 3. The control task is to keep the pendulum erected by moving the carriage to the left or right. The second control task is to move the carriage to a desired position while the pendulum is kept erected. We have chosen this sample application because of two main reasons: firstly, the control application has to produce control values every 5ms which is a typical timing constraint in industrial automation, and secondly, the pendulum is very sensitive to disturbances in the control application. Therefore disturbances introduced by the reconfiguration process should easily be recognizable (e.g., the pendulum falls down).

As control device we use a micro-controller board with the ARM7 micro-controller AT91M55800A from the company Atmel. The microcontroller board is equipped with 1MBytes of Flash and 1MBytes of RAM. The microcontroller is operated at a clock rate of 33MHz and provides several peripheral units like analog to digital conversion, digital to analog conversion, or counter units. This device represents a typical control device used in industrial automation. We have chosen this device to show that the developed approach is suitable for small embedded control devices. The 4DIAC—Framework for Distributed Industrial Automation and Control1—environment provides an IEC 61499 engineering tool and an execution environment capable of executing real-time constrained control applications. We used the 4DIAC environment for both modeling and executing the control application as well as the reconfiguration application. On the control device the 4DIAC execution environment is supported by the real-time operating system eCos2.

5.2 Control Task and Reconfiguration Process

The classical solution for controlling the inverted pendulum is a state space controller. A state space controller uses the system’s current state for calculating the control output. In our case the system’s state consists of the carriage’s position, its velocity, the angle of the pendulum, and its angular velocity. The control output is the desired

---

1 for more information please see http://www.fordiac.org
2 for more information please see http://ecos.sourceforge.org
velocity of the motor driving the carriage. For allowing the operator or a higher control layer to give a desired position of the carriage the control output is combined with a filtered value of the desired position. The upper part of Figure 4 shows this control algorithm implemented in IEC 61499.

For our reconfiguration experiment we will transform this controller to a more sophisticated one. The new control algorithm is again a state space controller, but it has an additional state: the integrated position error (see bottom of Figure 4). This has the advantage that position errors are compensated and it gives a faster movement to new target positions. Such position error compensation is especially helping in the cases of plants with an increased friction, which happens at an increased plant age. Therefore, this reconfiguration process is a typical case for industrial plants.

During the reconfiguration process it is necessary to add two new FBs, replace one FB, create six new connections, rewire five connections, and set six parameters. According to our structure we will split these tasks to the three phases of the RCA as follows:

In the Setup Phase we will create the two new FBs as well as the new state space controller FB. All parameters will be set and all event and data connection between these FBs are created. In order to reduce the effort in the execution phase also the data connections providing the current state and the desired position can be made at this stage as this will not influence the old control application. This works also for the event connection from the new state space controller to the ADD FB.

In the Execution Phase we then just need to rewire the event connection from the old to the new state space controller and rewire the data connection providing the control value, so that the control value from the new controller is used.

In the Shut-down Phase the old controller FB and its remaining event and data connections are deleted.

The well-disposed reader may find a detailed description of the reconfiguration steps in [20].

5.3 Results

This reconfiguration procedure has been applied to the pendulum control several times under two different start conditions: the carriage is holding its current position, and secondly, the carriage is moving to a new position. Figure 5 shows the trace of the position and the angle of one of our reconfiguration experiments. In the shown case the carriage is moving from one position (-8 cm) to a new position (35 cm). At the time instant 2.9 s the reconfiguration process is performed. One can clearly see that the reconfiguration process does not impose any major disturbances. The pendulum movement stays within the range of normal operation. Furthermore, the improved control algorithm provides a faster movement to a new position as can be seen according to the steeper position curve.
For all experiments at no time the balancing of the pendulum could not be achieved, neither any disturbances where recognizable. This shows that even highly unstable control systems can be reconfigured during full system operation. However, the experiments revealed that the planning and execution of the reconfiguration process is elaborate. The control engineer needs to carefully plan the reconfiguration process and program it accordingly. Even for a small reconfiguration process, as the one presented here, many stages are necessary. In order to simplify this task, an agent could take over and perform it autonomously without the need for human interaction. For this purpose means have to be developed that allow an agent to automatically understand the structure of its LLC and the constraints that have to be fulfilled during the reconfiguration process.

6 Conclusion

In this work we showed the needs and requirements for real-time reconfiguration of control applications in the domain of industrial automation. At the HLC side these requirements are well met with software agents. However on the LLC side such functionality is not available. With the real-time reconfiguration infrastructure presented in this work we closed the gap between HLC and LLC. The main element of our architecture is the reconfiguration application (RCA) which is executed considering real-time constraints. During its execution the reconfiguration application modifies the application under reconfiguration by utilizing so called reconfiguration services.

In order to test our concept we applied it to the closed loop control of an inverted pendulum. There the control application was changed during full operation. These tests showed the potential of our approach. However, they also showed that the planning and implementation of reconfiguration applications is rather elaborate.

Our next steps are therefore to connect our LLC with an agent driven HLC and enable the HLC to generate and conduct reconfiguration processes autonomously.

References


