DIPLOMARBEIT

Distributed Control in Process Automation – with the focus on possible applications of IEC 61499

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Diplom-Ingenieurs

unter der Leitung von

Ao. Univ. Prof. Dipl.-Ing. Dr. techn. Markus Vincze
Univ. Prof. Dipl.-Ing. Dr. techn. Bernard Favre-Bulle
Dipl.-Ing. Dr. techn. Alois Zoitl
Dipl.-Ing. Ivanka Terzic

E376
Institut für Automatisierungs- und Regelungstechnik

eingereicht an der Technischen Universität Wien
Fakultät für Elektro- und Informationstechnik

von

Wilfried Lepuschitz
Matr.-Nr.: 9840198
Gersthoferstraße 75–77/17/13
A–1180 Wien

Wien, im Jänner 2008

Abstract

Industrial companies are nowadays confronted with a difficult challenge on the global market. In order to succeed, they have to fabricate high-quality products based on consumer demands with short manufacturing periods at low costs. This challenge affects various domains including process industries. Some fields of process industries such as the petrochemical industries rely on historically grown systems with long life-spans. However, other sectors as for example pharmaceutical and biotechnological industries require flexible production facilities. By applying Batch Processes and modifiable recipes, small amounts of highly differentiated products can already be created.

Even though the applied processes are suitable for a flexible production, the used automation systems are still based on classical approaches that do not support easy modifications concerning their functionality. New control architectures are therefore necessary to further emphasize the gains of flexible production facilities.

Hence, a concept of a new control system for Process Automation has to be developed to meet these demands. To become acquainted with Process Automation, a survey presented in this thesis delivers the requirements and conditions of the target domain. Technical standards as well as existing engineering methods are therefore examined. Moreover state-of-the-art control concepts, that might be used or modified to design the control architecture, are analyzed in regard to the domain of Process Automation.

Based on the survey, this thesis introduces a new Distributed Control Architecture for Process Automation. This architecture is composed of Mechatronic Control Components with software elements based on the technical standard IEC 61499. Furthermore an intuitive engineering method for the development of applications is presented and used to convert a Control Recipe into an IEC 61499 Application.

Finally an actual implementation on a laboratory process plant is performed to evaluate the control architecture as well as the engineering method. This examination reveals certain gains and problems which are discussed to determine any practical impact and possible future research topics.

Kurzfassung

Im Zuge der immer weiter voranschreitenden Globalisierung stellt der heutige Weltmarkt für die Unternehmen vieler Industriezweige eine große Herausforderung dar. Um erfolgreich zu sein, sind diese Unternehmen gezwungen, hoch qualitative auf Kundenwünschen basierende Produkte in möglichst kurzer Zeit und unter möglichst geringem Kostenaufwand zu erzeugen. Auch jene, die auf dem Gebiet der Verfahrenstechnik tätig sind, sehen sich mit dieser Tatsache konfrontiert. Im Speziellen betrifft dies beispielsweise Unternehmen aus den Bereichen Pharmaindustrie oder Biotechnologie. Chargenprozesse und modifizierbare Rezepte werden in diesen Industriezweigen bereits erfolgreich eingesetzt.

Auch wenn diese Chargenprozesse im Vergleich zu kontinuierlichen Prozessen bereits eine flexiblere Produktion ermöglichen, so basieren die eingesetzten Automationssysteme auf klassischen Ansätzen der Steuerungstechnik, welche Modifikationen hinsichtlich ihrer Funktionalität nur bedingt unterstützen. Um die Vorteile der flexiblen Produktion intensiver als bisher nutzen zu können, sind somit neue Steuerungskonzepte notwendig.

Im Zuge dieser Diplomarbeit wird daher ein neuartiges Steuerungskonzept für die Verfahrenstechnik entwickelt. Hierfür ist es zuallererst notwendig, die Rahmenbedingungen der Domäne Verfahrenstechnik zu bestimmen. Technische Standards und existierende Steuerungskonzepte dieser Domäne werden daher untersucht. Weitere Automatisierungskonzepte, die in der Verfahrenstechnik noch nicht eingesetzt wurden, aber möglicherweise für eine Umsetzung in Frage kommen, werden hinsichtlich ihrer Eignung analysiert.

Auf Basis der gewonnenen Ergebnisse der Umfeldstudie wird eine neuartige verteilte Steuerungsarchitektur vorgestellt. Diese setzt sich aus Mechatronischen Steuerungskomponenten zusammen, deren Software auf dem technischen Standard IEC 61499 basiert. Des Weiteren stellt diese Diplomarbeit eine intuitive Designmethode für die Entwicklung von Steuerungsapplikationen vor, die bei der Konvertierung eines Steuerrezepts in eine IEC 61499 Applikation zur Anwendung kommt.

Schlussendlich wird eine Implementierung auf einer verfahrenstechnischen Laboranlage durchgeführt, um die vorgestellte Steuerungsarchitektur, sowie die Designmethode, testen und bewerten zu können. Die hierbei erzielten Resultate werden hinsichtlich gewonnenen Nutzens und etwaiger auftretender Probleme betrachtet und weitere aus dieser Arbeit abgeleitete Forschungsthemen vorgestellt.

Acknowledgement

First of all I wish to thank everyone who encouraged and supported me during my years of studies and throughout the period of composing this thesis – may this have been in either a technical, personal or whatsoever manner.

My sincere thanks are given to Professor Bernard Favre-Bulle for sharing invaluable insights concerning numerous aspects of the academic way of life. It has been a pleasure and honour to work alongside him and I look forward to further opportunities of that kind. I am also very thankful to Professor Markus Vincze for the supervision of this thesis.

I would like to express my gratitude to my tutor Dr. Alois Zoitl for his invaluable help and for providing me with guiding feedback towards the completion of this thesis. Furthermore, I would like to thank my first tutor Ivanka Terzic for her support and encouragement during the early stages of this thesis until joyful private matters demanded her full attention.

Moreover I am very grateful to Christoph Sünder, Michael Rainbauer and Franz Mehofer for their contributions to this thesis as well as Rene Smodic, Ingo Hegny and Oliver Hummer-Koppendorfer for their support in various technical matters.

For numberless hours of learning and companionship throughout these years of studies I wish to thank my former fellow students Herwig Wendt, Jürgen Gottwald and Oliver Hummer-Koppendorfer – all of them already finished their studies some time ago. But finally I will be able to call them colleagues again. "Last but hopefully not least", some may say...

My warmest thanks are reserved for my family for their never-ending support during all my years of education. I am unbelievably grateful to my parents Wilfried and Margit Lepuschitz for making my chosen path of life possible and for being open-minded about any of my decisions.

I would like to conclude my acknowledgement with a personal dedication. I plan to finish my studies on the 18th of January 2008 by passing my final exams. Exactly half a year before that day, on the 18th of July 2007, a dear friend of mine left this world much too early in a tragic accident and it may have been this half of a year he missed to complete his studies as well. I am certain, he would have used his skills devotedly for the collective good, because this concern has always been a driving force in his life. In good remembrance of the past, I dedicate this thesis to Gregor Saje.

Wilfried Lepuschitz

Contents

List of Figures

List of Tables

1 Introduction

Industrial companies are nowadays confronted with a never-ending challenge: the fabrication of high-quality products based on consumer demands with short manufacturing periods at low costs.

Especially the European industry faces strong competitors from the Far East due to the globalization of markets. Mass commodities can be produced cheaper in Asian countries than in Europe, therefore the European industry has to focus on highly innovative products and high quality. Nevertheless, a short time to market is indispensable in regard to decreasing product life cycles and the need to serve the market faster than any competitors. Therefore the industry requires prospective technologies that support companies to cope with these challenges [FB05].

In the domain of process industries, the introduction of Batch Processes already brought certain flexibility at least from a process-oriented kind of view since modifiable recipes provide the basis for a broader range of products. Nevertheless the applied control systems are still based on classical automation concepts such as Analog Closed-loop Controls and Programmable Logic Controls (PLCs) that do not facilitate intuitive engineering methods since their structure does not necessarily bear resemblance to the functional structure of the target system. Moreover the automation systems currently applied in the process industry do not emphasize flexibility and easy modifications concerning their functionality.

Therefore the motivation for this thesis is to develop a flexible architecture that allows modifications in case the system's functionality needs to be changed. Furthermore the architecture shall facilitate intuitive engineering processes to ease any design activities.

1.1 General Objectives

This thesis treats the development and implementation of a Modular Distributed Control Architecture with an application based on the standard IEC 61499 for the domain of Process Automation. The focus is laid on Batch Processes, since they bear resemblance to Discrete Processes in terms of scheduling. This allows the possibility to derive existing concepts concerning Discrete Processes to the field of Batch Processes.

Prior to any activity concerning the actual design of the Modular Distributed Control Architecture, the domain of Process Automation needs to be analyzed in order to determine the target domain's attributes, conditions and requirements. Likewise any other automation system, the aim of the presented architecture focuses on automatically executed production processes. Therefore the field of Process Automation is examined in consideration of the occuring processes. Moreover existing engineering methods and concepts, such as technical standards, deliver possible approaches that can be taken into account.

In regard of the gathered knowledge of the target domain, a new Modular Distributed Control Architecture is to be developed and shall lead to the creation of an IEC 61499 Application. However, not only the resulting application has to be analyzed. Also the design process itself needs to be examined in order to reveal possible gains but also possibly occuring problems.

To demonstrate the feasibility of the developed architecture, the created concept is to be implemented on the laboratory process plant FESTO Compact unit. A validation of the actual implementation shall further unveil any gains and problems of the presented concept.

1.2 Task Definitions

The general objectives of this thesis can be summarized in the following tasks:

- Analysis of the target domain Process Automation;
- Determination of existing engineering methods and concepts;
- Design of a Modular Distributed Control Architecture for Process Automation;
- Development of an application based on IEC 61499;
- Implementation and validation of the concept on the laboratory process plant FESTO Compact unit;
- Discussion of possible gains and occuring problems.

1.3 Structure of the Thesis

Following this introduction, current state-of-the-art concepts and standards related to Process Automation are presented in Chapter 2 to determine the conditions needed for the design of a control system. An examination of the standards IEC 61499 and IEC 61512 investigates their value for the design process.

The presented state-of-the-art concepts act as the basis for the design of a Distributed Modular Control Architecture described in Chapter 3. Not only the development of a final application for the further implementation is described. Also the design process itself is analyzed in detail since it comprises the combination of several existing concepts.

Chapter 4 treats the implementation of the developed application on the laboratory process plant FESTO Compact unit. Moreover this chapter presents the created Component Library as well as used hardware and software.

Chapter 5 presents the outcoming results of this work in regard of possible gains and problems. Finally Chapter 6 concludes the thesis with a summary and an outlook concerning further research possibilities.

2 State of the Art

Automation represents the top-most term that might be applied to describe the domain of this thesis. A brief overview of Automation as a general topic and furthermore a description of distributed systems and Mechatronic Control Components as their elements emphasizes the approaches used as a basis for the developed automation concepts. Furthermore this Chapter presents an introduction into the standard IEC 61499 *Function Blocks* which defines architectural concepts for control systems.

As this thesis treats Automation in the context of Process Engineering, the domain of Process Automation is compared to Production Automation and analyzed in terms of occuring processes and existing concepts. The focus of this work is laid on Batch Processes, hence the standard IEC 61512 *Batch Control* proves to be a fitting source concerning structural concepts in regard to these kind of processes.

2.1 Automation Systems

The word *Automation* descends from the greek word *automatos*, which means "self moving, self thinking". Automation implies the use of artificial instruments for enabling a process to run automatically. Regarding a technical installation it refers to equipping the installation with certain machines and instruments to ensure its automatic job execution [FB04].

Ever since the use of primitive technologies to automate different production processes (e.g. the use of a windmill to prepare wheat for the baking-process), applied technologies were adapted either due to new discoveries or changing technical demands. In case of automation the commonly used architecture some decades ago was based on a central processing unit containing the complete control intelligence – hardware as well as software. On account of changing requirements of the industry (e.g. the request for more flexibility) the decentralized control architecture was developed and nowadays serves as the industrial state-of-the-art architecture widely used in a broad range of production domains.

Nevertheless the demands of the industry are not yet fully met. Customerbased products with small batch sizes (even down to one) require flexible production plants and therefore adaptive control systems leading to the development of Distributed Control Architectures. The following sections introduce the concept of *Modular Distributed Control Architectures* and *Mechatronic Control Components* as their elements. Furthermore a brief overview of the standard IEC 61499 *Function Blocks* shows its applicability for Distributed Control Systems.

2.1.1 Modular Distributed Control Architecture

The concept of the *Modular Distributed Control Architecture* is based on so-called *Multidisciplinary Devices*, that contain all relevant hardware and software components to execute required tasks (e.g. job execution as well as communication with other devices). Evidently this concept follows the paradigm of object-oriented programming: the encapsulation of functionality and information into an entity. Although already widely used in the context of software design, this approach seems yet to be rather uncommon for automation as can be seen in the following (rather dogmatic) statement:

> "Automation programming is based on logical functions represented graphically and executed cyclically, whereas general software programming has developed from instructions following each other into objects which have a set of methods that can be invoked." [SSPK06]

However, Distributed Control Systems consist of autonomous intelligent entities (Multidisciplinary Devices), that coordinate their activities by exchanging information to perform a global task. Each device provides functional tasks according to its physical structure (e.g. *"Heat"* is a possible functional task for a heating unit) including diagnostic processing (for example error detection). The combination with other devices and their allocated functional tasks enables more abstract functional tasks (Figure 2.1). For instance the functional task *"React"* requires the tasks *"Control heat"* and *"Circulate"*. Finally the functionality for a whole process plant (e.g. *"produce PVC"*) can be defined by continuing to combine the devices [FB04, Dut03].

The Modular Distributed Control Architecture features several advantages, as devices can easily be added to or removed from the system. Furthermore this approach facilitates a more intuitive engineering process since all structural attributes of the automation systems are derived from the physical structure of the target system. For further information regarding the advantages of this architecture see Chapter 3 in [Dut03].

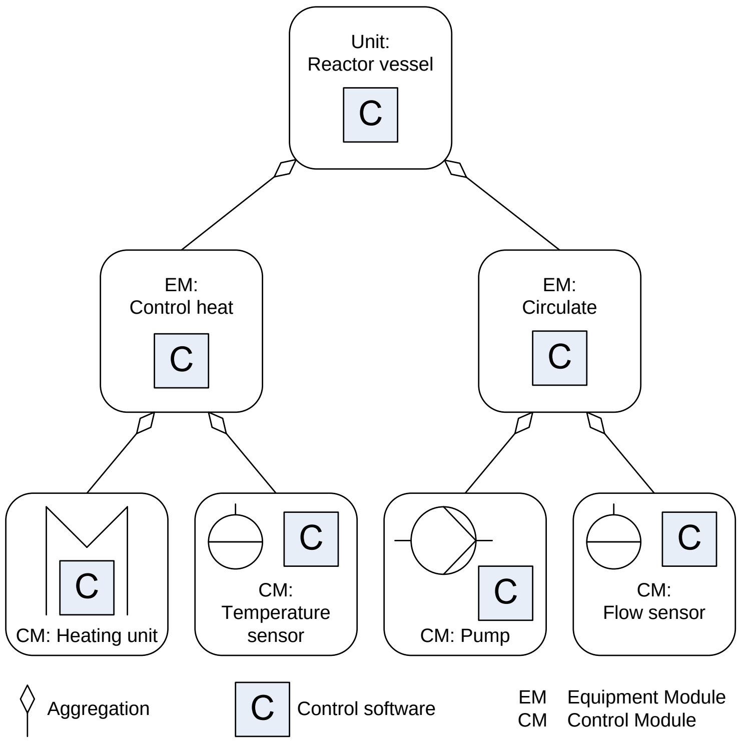

Figure 2.1: Modular Distributed Control Architecture composed of Multidisciplinary Devices

2.1.2 The Concept of Mechatronic Control Components

Mechatronic Control Components incorporate mechanical, electronic as well as software elements and represent one kind of Multidisciplinary Devices that can be used to compose a Distributed Control System (see Section 2.1.1).

Types of Mechatronic Control-Compontents

One type of Mechatronic Control Components interacts directly with the physical processes. These components may generally be referred to as *intelligent sensor/actuator units* (Figure 2.2) and, as indicated by the name, comprise a combined sensor/actuator system [Dut03]. In case of a of a non-intelligent sensor/actuator unit its controller only has to process orders from the superior control and to transfer this information to the information/energy conversation element as well as to prepare sensor data and transmit it to the higher-level control. The controller of an intelligent sensor/actuator unit is able to execute more complex tasks independently of its superior control (e.g. diagnostic functions). For instance a *ball valve* in combination with a controller and two position sensors (to verify if the mechanical elements of the valve are located in a defined end-position) represents an intelligent sensor/actuator unit. The controller not only positions the valve according to the orders sent by the superior control but also checks if the valve reaches the demanded end-position within a certain time.

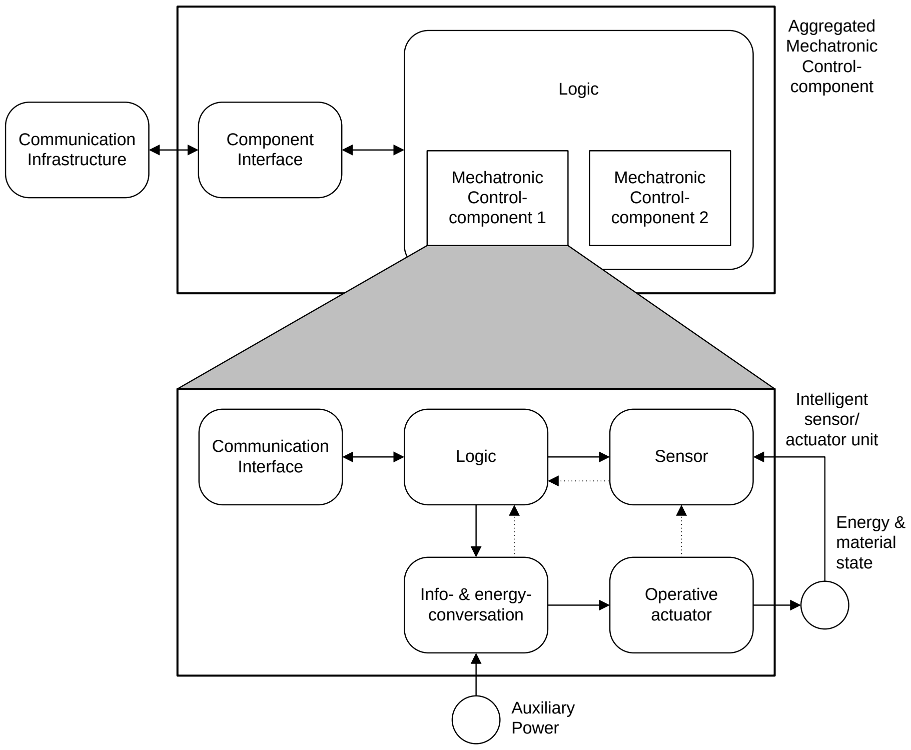
Aggregated Mechatronic Control-component
Logic
Communication Infrastructure
Component Interface
Mechatronic Control-component 1
Mechatronic Control-component 2
Intelligent sensor/ actuator unit
Communication Interface
Logic
Sensor
Energy & material state
Info- & energy-conversation
Operative actuator
Auxiliary Power

Figure 2.2: Mechatronic Control Components based on [ZF00]

Aggregated Mechatronic Control Components provide more abstract functionalities by encapsulating lower-level Mechatronic Control Components (Figure 2.2). For instance a *Process Cell* that contains pumps, various valves and sensors represents an Aggregated Mechatronic Control-Component but this term can even be applied to a whole factory.

2.1.3 IEC 61499 – Function Blocks

As already stated in Section 2.1.1, Distributed Control Systems base on the use of devices with encapsulated functionality. *Function Blocks (FBs)*, which are in fact reusable software components providing a certain functional task, are already widely applied in automation systems. IEC 61131-3 *Programmable Controllers – Part 3: Programming languages* contains the concept of FBs but has to be amended for the use in a Modular Distributed Control Architecture due to a lack of applicability for the distribution. Since control mechanisms in a distributed system are not centralized in one core entity, data will be processed parallel in more than one device. To ensure the correct sequence of activities of a whole system, IEC 61499 introduces an event-driven execution model and therefore dissociates the event flow from the data flow. The aim of IEC 61499 appears to be evident according to the very first sentence of this standard:

> "IEC 61499 defines a generic architecture and presents guidelines for the use of *function blocks* in distributed Industrial-Process Measurement and Control Systems (IPMCSs)." [IEC05]

The IEC 61499 standard is organized hierarchically ranging from the top-most model, the System model, to the FB model as the lowest level.

System model The *System model* forms the highest abstraction layer and describes the relations between *applications* and *devices* (Figure 2.3a). Applications are not restricted to a single device and can be distributed over several devices. Furthermore devices are capable of providing their functionality to more than one application. According to the *Application model* a distributed application is composed of connected function blocks. Since these function blocks may be arranged on different devices, communication services have to ensure the consistency of event- and data-flow between the devices.

Device model A device is composed of one or several *resources* that contain function blocks and enable their independent execution. Each resource may comprise parts of applications (Figure 2.3b). Moreover the device model includes an interface for communication services as well as a process interface to communicate with input or output ports of the physical device.

Resource model A resource contains a function block network and enables its execution by providing a scheduling function to ensure the correct execution

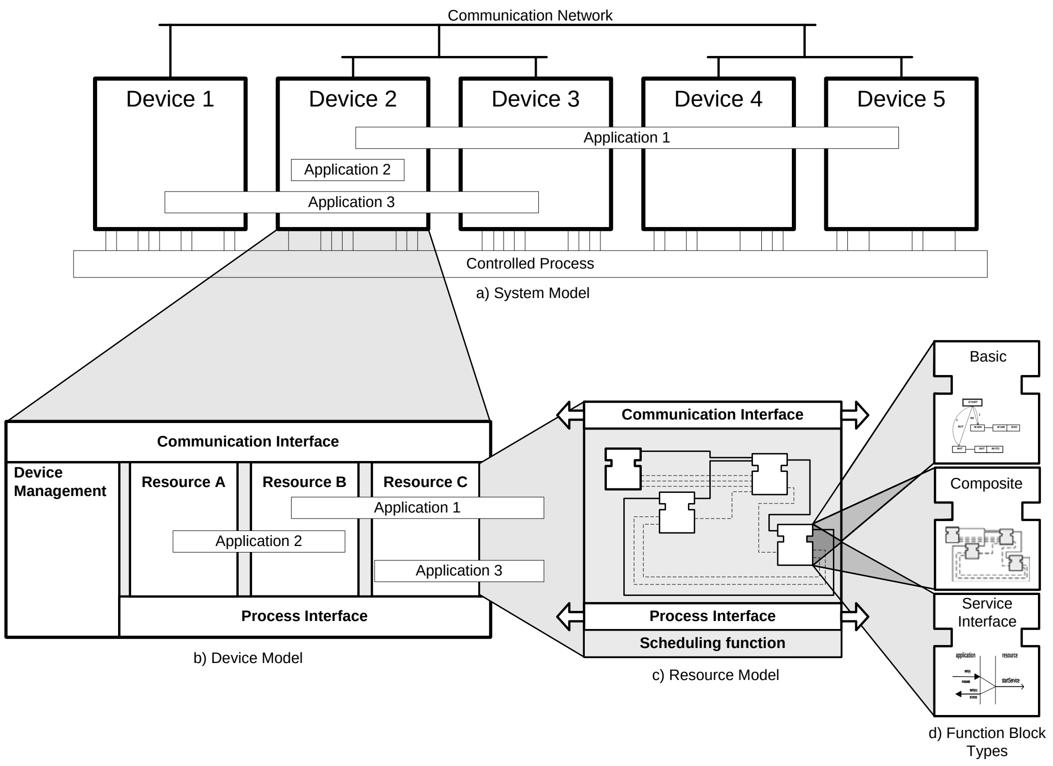

Figure 2.3: Reference models according to IEC 61499 [Zoi07]

sequence of algorithms within the function blocks (Figure 2.3c). Furthermore the resource provides a communication interface as well as a process interface. *Service interface function blocks* can exchange information (events and data) with remote resources or phyiscal I/O ports by accessing these interfaces.

Application model An IEC 61499 *Application* consists of a function block network including event and data connections. These function blocks do not need to be necessarily located within the same device or resource. An application may be distributed over several devices and resources. Function blocks may also be encapsulated within *subapplications* that have the external characteristics of a function block.

> "In practical terms, an application is the entire set of function blocks and interconnections to solve a particular automation control problem. Examples might be: the set of function blocks required to control a production line, a plastics xtruder, or a fermen-

tation vessel." [Lew01]

Function block model According to the IEC 61499 standard, a *Function Block (FB)* is defined as a functional unit of software containing data and algorithms (Figure 2.4). Only knowledge about the interface (event- and data-I/Os) is required to use the encapsulated functionality of a function block. Multiple instances can then be created and executed independently from each other.

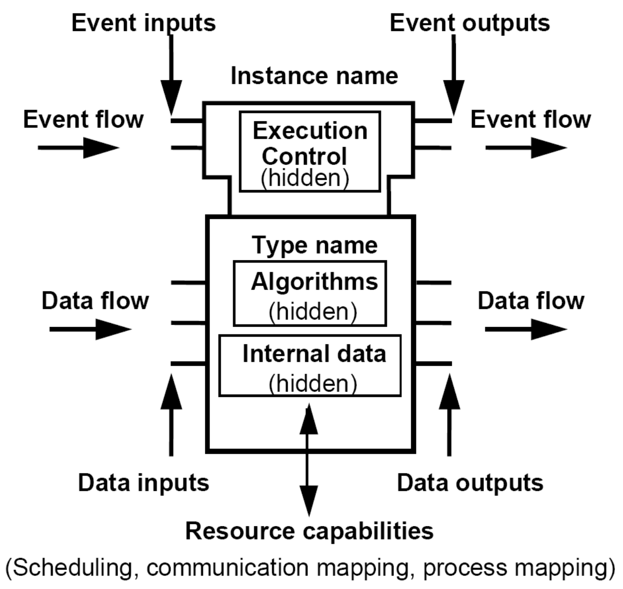

Figure 2.4: Characteristics of a function block [IEC05]

Function block types The different *Function block types* (Figure 2.3d) comprise the behavior and interfaces of instantiated function blocks. This incorporates a type name, input and output events, variable definitions as well as the function block's internal behavior.

Basic function blocks (BFBs) are constituted by their internal algorithms which are executed according to certain input events. An *Execution Control Chart (ECC)* forms a state machine and maps input events on to algorithms. Output events are further created at the occurrence of state changes.

Composite function blocks (CFBs) encapsulate a network of function blocks and their corresponding event and data connections. These function block networks may contain basic as well as composite function blocks and provide the possibility to develop composite function blocks with complex functionalities.

Service interface function blocks (SIFBs) provide the necessary interfaces to enable communication processes between the function block network of a resource and function blocks of a remote resource or to access hardware elements (e.g. physical I/O ports) of a device.

2.2 Process Engineering

Process Engineering generally refers to any creative actions that include the development of processes. In general a process represents the entirety of consecutive interacting activities within a system to transform, store or transport material, energy or information [FB04]. Evidently the term Process Engineering is widely used within technical engineering domains but seems also appropriate for instance within the economical sector. Nevertheless the term Process Engineering is commonly deemed to be closely connected with *Chemical Engineering.* The German word *Verfahrenstechnik*, which represents only one possible translation for Process Engineering, is more specific and indeed only related to technical engineering. This thesis treats Process Engineering within the Chemical Engineering domain which mainly focuses on material transforming processes.

Process Engineering represents a vital part within petrochemical, food, pharmaceutical and biotechnological industries. Though widely used within those different branches, the attitude towards the implementation of new ideas and technologies differs from very conservative within petrochemical industries to open-minded within organic and pulp industries [Ing06]. A look at the largest Austrian oil refinery reveals historically grown systems with life-spans of several decades. PLCs[1] are rarely in use and with only a few exceptions almost all processes are of a continuous kind (see Section 2.2.1 for *Continuous Processes*). Bulk chemical process industries such as the petrochemical sector rely often on a continuous production with large output quantities of undifferentiated products. Flexibility is therefore not deemed to be a cost advantage. On the contrary industries that produce smaller amounts of more differentiated products require flexible multipurpose production facilities to react quickly on changing market demands [Kui99]. Nevertheless resentments towards the implementation of new concepts within all mentioned domains can not be denied as was shown in a summer workshop carried out by a workgroup of the Helsinki University of Technology [SSPK06].

[1]Programmable Logic Control

2.2.1 Process Automation Compared to Production Automation

It is evident that knowledge of the target domain is required to develop any kind of automation system. Various companies and institutes work on the development and improvement of Distributed Control Systems. However, most of these activities are concentrated on applying these systems to the domain of *Production Automation.* In this context *Process Automation* has been treated more like a "red-headed stepchild". Therefore a comparison between appearing processes within Production Automation and Process Automation seems appropriate to draw conclusions for the development of a system concept. Both types of automation are derived from Industrial Engineering but evidently treat different aspects of material handling.

Production Automation

Production Automation aims at the production of individual workpieces with a defined shape and determined measurements. Common activities involve the treatment of surfaces, shaping of workpieces and the assembly of parts to create more complex products. Production Automation incorporates usually *Discrete Processes.*

Discrete Processes contain data of fixed points in time and space to follow determined time-schedules and determined target locations. Discrete Processes can therefore be regarded as non-stationary processes. They appear for instance when using industrial robots for pick-and-place tasks. Though closed-loop controls are inevitable to keep the movement of the robot on track these functions are hidden for the operator.

Process Automation

Process Automation treats the production of shapeless materials which includes typically gas, liquid, paste, powder and granulate. It aims at the alteration of the nature, structure, property or chemical composition of materials. Process Automation incorporates two types of processes: *Continuous Processes* and *Batch Processes.*

Continuous Processes remain in a constant operational state without down times due to a continuous supply of material and energy. Simultaneously these processes produce a continuous output. All characteristic values (pressure, temperature and concentration) along the flow path within the process equipment are locally different but temporally constant. Therefore a Continuous Process runs stationary [UB95].

Batch Processes are neither discrete nor continuous, nevertheless they bear resemblance to both other process types. On the one hand Batch Processes represent a subset-treatment of goods normally treated in Continuous Processes but moreover also follow determined time-schedules likewise to Discrete Processes. Regarding an ideal material mixture all characteristic values within the process equipment are locally equal but change during the dwell period. Hence a Batch Process runs non-stationary [UB95, SV94]. The name is derived from the term *Batch*, which represents the amount of material located in the reactor vessel that is treated as a whole within the process.

2.2.2 Existing Concepts for Process Automation

This section introduces general automation concepts that are commonly used within the sector of Chemical Engineering. Furthermore this section offers a brief view on the standard IEC 61804 *Function blocks (FB) for process control* which applies some aspects of the standard IEC 61499 to Process Automation.

Analog Closed-loop Controls are small closed-loop control units with either a two-position-output, three-position-output or three-position-step-output and are also referred to as *Industry Controls*. Typically they are used to operate heating or cooling units and incorporate only a very limited HMI[2]. Control parameters for certain control paths are sometimes preset already during their production in case of bulk commodity products [Str02].

Digital Compact Closed-loop Controls are different from Analog Closed-loop Controls by providing more functionality and by offering more possibilities concerning the HMI design as well as the integration into higher-level control systems. They are also called *Process Controls*. Digital Compact Closed-loop Controls feature an important advantage over their analog counterparts by achieving their functionality with the use of cost- and space-saving software instead of expensive hardware. Due to the software they are able to execute the tasks of several Analog Closed-loop Controls (for instance in the realization of a cascade control). In most cases Digital Compact Closed-loop Controls are used to operate a PI- or PID-control [Str02].

[2]Human Machine Interface

Programmable Logic Controls are widely in use within the domain of Production Automation to operate Discrete Processes due to their step control ability. Operating a Continuous Process does not require advanced functionality as provided by a PLC therefore these type of controls came into use in Process Automation with the emergence of Batch Processes which require the capabilities to operate time-scheduled (for instance cyclic controls) as well as process-dependent (acting on sensor inputs). In Process Automation PLCs are in use to control for example press or packaging machines as well as climatic systems and show their advantages especially in controlling machines with real-time arbitration. Since PLCs are often operated on the field-level directly next to the controlled equipment, some have to be built rather sturdy to maintain operational status even within difficult environments (for instance PLCs might be exposed to high temperature levels) [Str02].

Distributed Process Control Systems differ from PLCs more gradual than in principle. Modern Process Control Systems (PCS) represent so-called SCADA[3] systems and contain PLCs especially for real-time processes. Therefore they are often also referred to as *Hybrid Control Systems*. While extensive HMIs have not been emphasized for PLCs especially concerning their use outside of Process Automation, the development of visualization components for PCS within their target domain has been deemed important ever since their introduction. Modern PCS are structured according to a decentralized, respectively distributed, architecture and can therefore be adapted to the process structure. This can be considered a plain sailing problem statement for Continuous Processes which require almost only closed-loop controls (Figure 2.5a) whereas for Batch Processes also more complex control tasks have to be taken into account (Figure 2.5b). Hence PLCs are often integrated into the structure of a PCS to meet the demands of complex tasks. More detailed information concerning the PCS and their components can be found in [Str02].

IEC 61804 – Function blocks (FB) for process control

The standard IEC 61804 *Function blocks (FB) for process control* specifies the requirements of Distributed Process Control Systems based on Function Blocks. It is in use in several industries as for example in petrochemical, pharmaceutical, or food and beverage domains. This standard demands the following requirements to be fulfilled by PCS [IEC06a]:

[3]Supervisory Control and Data Acquisition

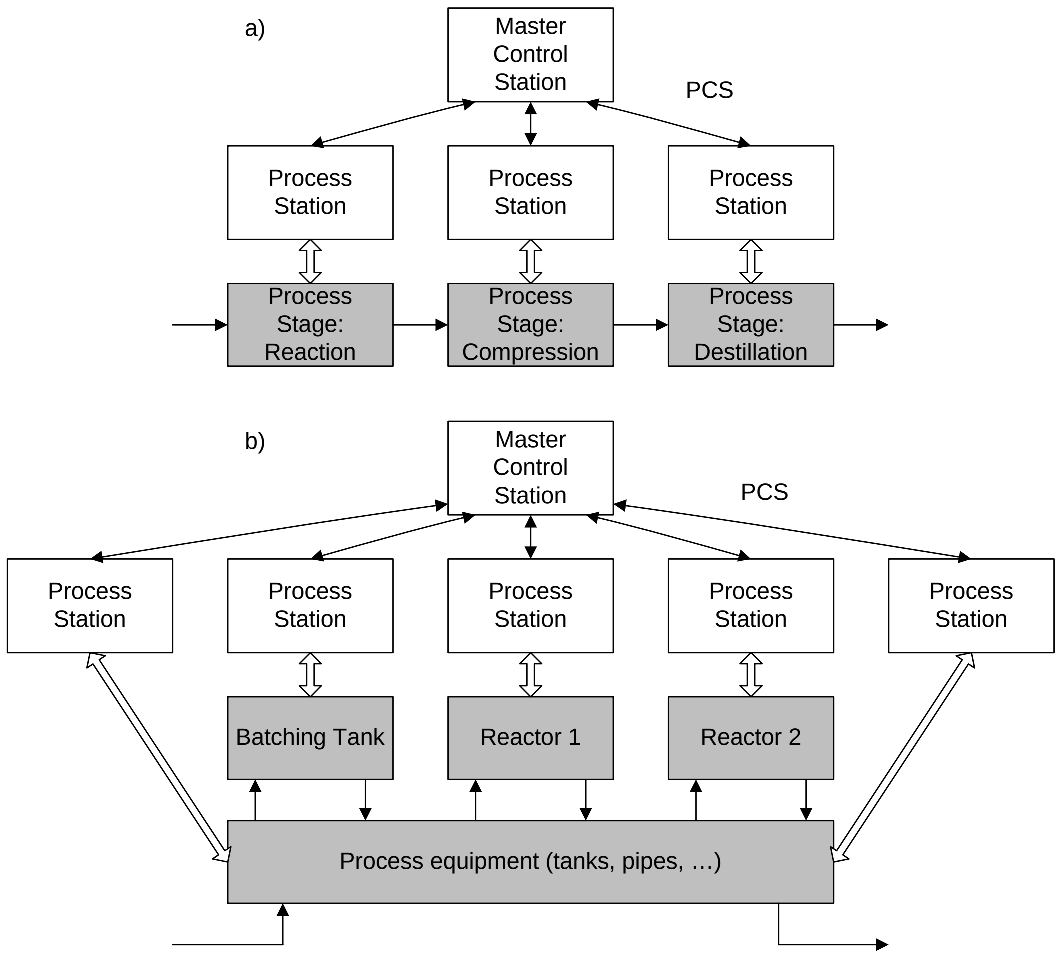

Figure 2.5: PCS for Continuous (a) and Batch Processes (b) – based on [Str02]

- increase security and safety;
- reduce time to market;
- be supportable with available tools;
- reduce costs of development and support;
- minimize training costs;
- support integration of distributed control applications;
- support integrated methodology for implementation;

- have increased maintainability, modifiability, agility, upgradeability, flexibility, ability to validate, accessibility, availability, compatibility of support tools, multi-vendor device/application compatibility, reusability of knowledge and designs, reusability of software components;
- be made up of digital devices that are compatible, interworkable, interconnectable, interoperable and interchangeable with each other.

This standard defines requirements for FBs concerning control, maintenance and technical management as applications that interact with sensors as well as actuators and refers to the generic FB model of IEC 61499 as the basis for developing process control FBs. However, IEC 61804 incorporates also applications designed not with FBs for example such as the HMI or maintenance applications (Figure 2.6).

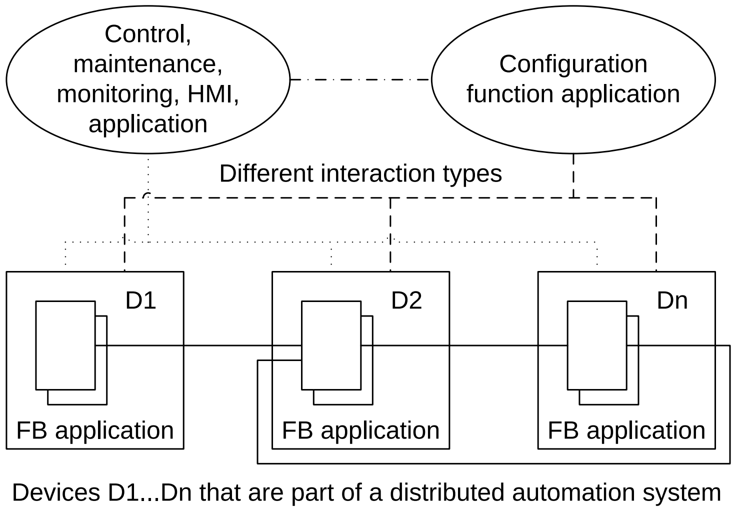
Control, maintenance, monitoring, HMI, application
Configuration function application
Different interaction types
D1
D2
Dn
FB application
FB application
FB application
Devices D1...Dn that are part of a distributed automation system

Figure 2.6: IEC 61804 system overview with FB-based as well as non-FB applications – based on [IEC06a]

IEC 61804 states a small set of basic function block types to be integrated into a device structure for a process industry FB application (Figure 2.7).

The *Resource Block* contains variables that describe the characteristics of the physical sub-component associated with a resource.

General FBs provide device and network independent automation functions for an application. They contain algorithms to process input variables and data received from Technology Blocks to further produce output variables and data which is sent to Technology Blocks.

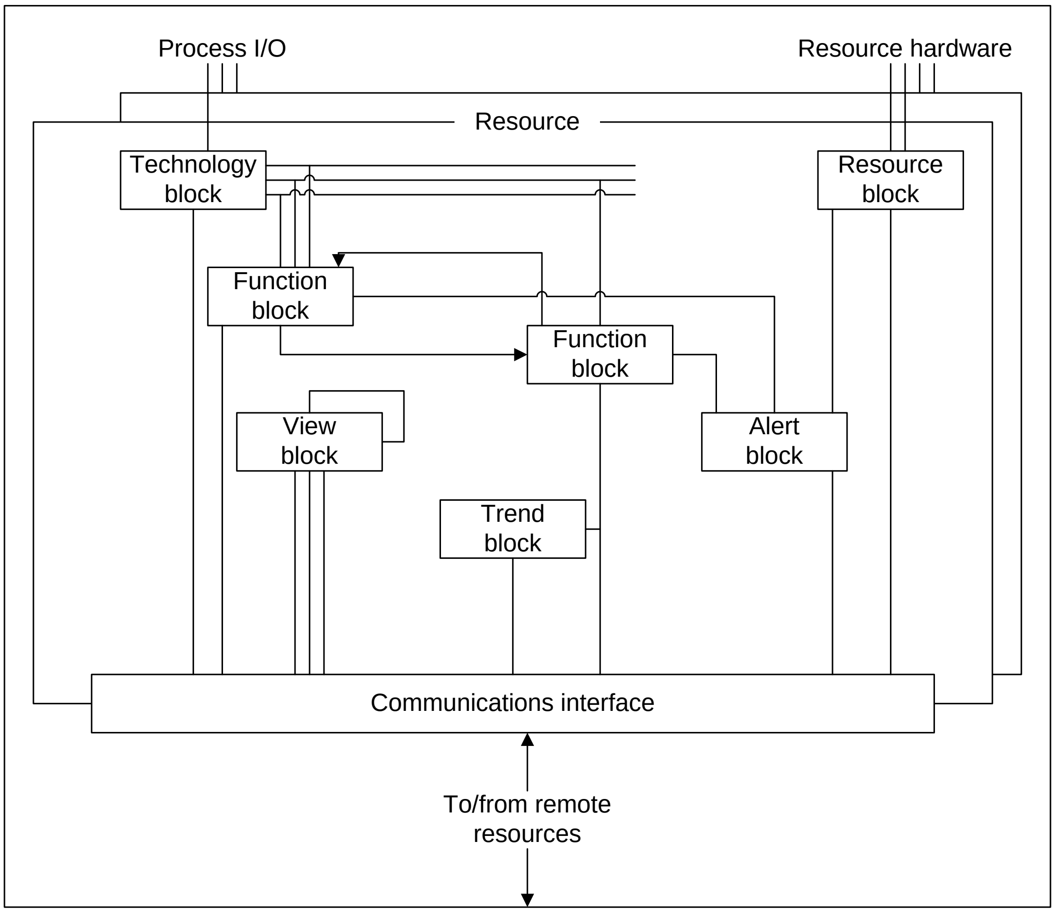

Figure 2.7: Device structure for a FB application for the process industry [IEC06a]

Technology/Transducer Blocks connect General FBs with I/O devices (sensors and actuators) by providing a device independent interface that can be used by FBs. Moreover Technology Blocks provide further funcionality as for example converting I/O data to a device independent representation.

View Blocks allow access to block parameters and act as a gateway for operators to view FB data.

Trend Blocks provide access to data structures that represent the temporal progression of a single block parameter effectively reducing communications as well as system processor overhead that would be needed for scanning parameters at a fast rate for trending.

An alarm is generated if a block leaves a particular state or returns back to that state. *Alert Blocks* are responsible for a controlled transfer of alarms within the system and are usually designed according to standard exchange protocols.

Table 2.1 shows a comparison between components of the standards IEC 61499 and IEC 61804.

IEC 61499	**IEC 61804**
System model	System overview
Application model	FB application
Device	Device
Resource	Resource
FB	General FB
-	Resource Block
Service interface FB to access the hardware	Technology/Transducer Block
Special FB which provides the input data to the communication for the HMI	View Block
Special FB which handles errors and transfers error messages	Alert Block
Special FB which accesses data structures containing the temporal progression of parameters	Trend Block

Table 2.1: Comparison between IEC 61499 and IEC 61804 components [IEC06a]

2.3 IEC 61512 – Batch Control for Process Engineering

In the year of 1992 the NAMUR[4] issued the NAMUR Recommendation NE 33 *Requirements to be Met by Systems for Recipe-Based Operations* describing procedures and system requirements for recipe-based operations related to a general concept for the operation of process plants [NE 03]. Further efforts led to the development of the standard ISA S88.01 *Batch Control, Part 1: Models*

[4] Automation Systems Interest Group of the Process Industry – formerly known as: Normenarbeitsgemeinschaft für Meß- und Regeltechnik in der chemischen Industrie

and Terminology and its counterpart IEC 61512-1 *Batch control – Part 1: Models and terminology.*

Knowledge about the general structure of a batch manufacturing plant as well as its processes is necessary to apply the concept of the *Modular Distributed Control Architecture* (see Section 2.1.1). The standard IEC 61512-1 provides the required information for the design of a Distributed Control System.

> "This part of IEC 61512 on batch control defines reference models for batch control as used in the process industries and terminology that helps explain the relationships between these models and terms." [IEC97]

To ease design, operation and improvement processes of a batch manufacturing plant, this standard offers a consistent terminology and a set of concepts and models for the batch industry to provide a communication basis for all involved work groups (for instance chemists, automation engineers, machine operators, etc.). Therefore the utilization of IEC 61512 shall lead to

- a shortening of the setup time of production facilities to manufacture new products;
- easier development of tools for implementing Batch Control;
- enabling the users to better determine their requirements;
- easier development of recipes without the services of a control systems engineer;
- a cost reduction of automating Batch Processes;
- a decrease of life-cycle engineering efforts.

According to the standard, the described models may be diminished or extended as long as the consistency of each model remains assured.

2.3.1 General Terms and Definitions

This section provides general terms and definitions according to the standard IEC 61512. For terms and definitions related to the structural models and recipe concepts of this standard see Sections 2.3.2 and 2.3.3 respectively.

The term *Batch* implies two meanings. On the one hand one Batch indicates the resulting material that was produced during the execution of a *Batch*

Process. On the other hand a material production process during its execution can be referred to as one Batch.

A *Batch Control* incorporates control activities and functions to process finite amounts of ingredients according to a sequence of process activities by using one or more facilities within a finite time period.

A *Batch Process* refers to a process which leads to the production of a finite amount of material by processing ingredients according to a sequence of process activities by using one or more facilities within a finite time period.

A *Common Resource* is an entity that can serve more than one user (e.g. higher-level resources). While *Exclusive-use Resources* are able to serve only one user at a certain time, *Shared-use Resources* comprise the ability to serve multiple users at any time.

2.3.2 Structural Models and Concepts

IEC 61512 introduces a set of structural models concerning processes, physical equipment and control software. These models are taken as a basis for the structural design of the Modular Distributed Control Architecture.

The Process Model

A Batch Process can be segmented hierarchically according to the *Process Model* of the standard IEC 61512 into *Process*, *Process Stage*, *Process Operation* and *Process Action* (Figure 2.8a).

A *Process* represents the sequence of chemical, physical or biological activities for the transformation, transport or storage of material or energy. IEC 61512 presents the production of polyvinyl chloride (PVC) as a Batch Process example.

A Process consists of one or more *Process Stages* that usually run independent from each other in a serial or parallel arbitration to conduct a scheduled sequence of transformations of the processed material. Regarding the PVC-process, typical Process Stages could be the polymerization of vinyl chloride monomer to polyvinyl chloride or the drying of PVC-powder.

A Process Stage contains a number of *Process Operations*. Each Process Operation refers to a larger process activity which in fact means usually one chemical or physical transformation of the processed material. Typical Process Operations of the PVC-process could be the preparation of the reactor or the reaction itself.

Process Actions represent smaller process activities that are combined to a Process Operation. The Process Operation “Reaction” could consist of the following Process Actions:

1. fill a certain amount of catalyst into the reactor;
2. fill a certain amount of vinyl chloride monomer into the reactor;
3. heat up the reactor to a temperature between 55 and 60 degrees celsius
4. hold the reactor temperature between 55 and 60 degrees celsius until the reactor pressure drops.

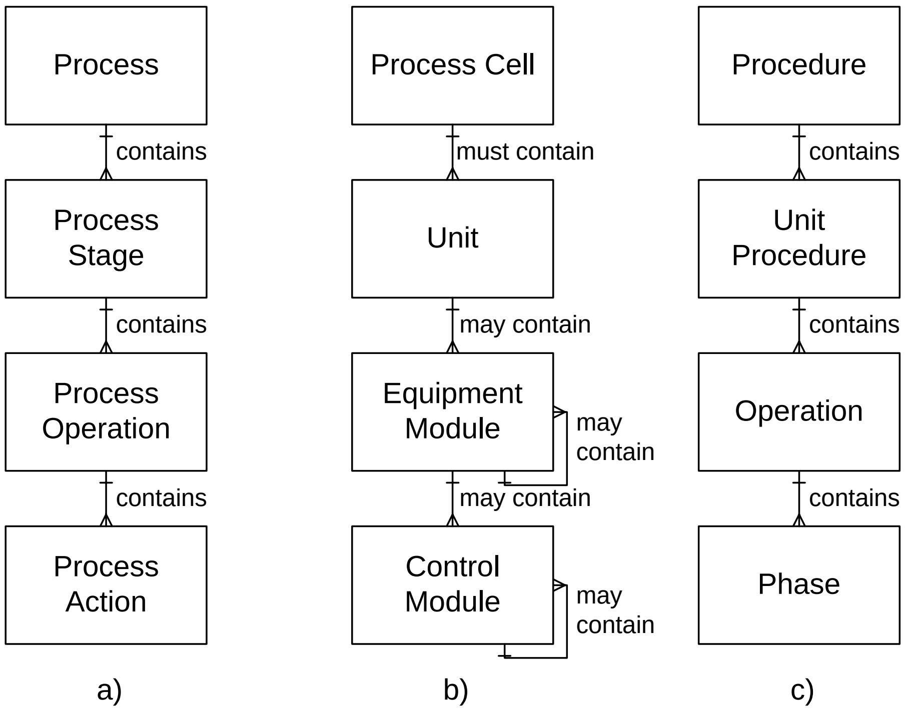

Figure 2.8: Process Model (a), the lower four levels of the Physical Model (b) and Procedural Control Model (c) [IEC97]

The Physical Model

The *Physical Model* abstracts the physical organization of a batch manufacturing company into a hierarchical structure (Figure 2.8b). Usually groupings of lower levels are merged to form a higher level but on some occasions groupings of a level may be merged into other groupings of the same level.

The standard IEC 61512 treats the upper three levels – *Enterprise*, *Site* and *Area* – of the Physical Model only marginally since these levels are connected closer to entrepreneurial than technical regards. The lower four levels – *Process Cell*, *Unit*, *Equipment Module* and *Control Module* – represent delimited groupings of equipment. Combining equipment into functional groupings eases operational tasks since each of these groupings can be treated as a larger single device.

An *Enterprise* is an organization that coordinates the operation of one or more *Sites*. Product decisions are made on this level as well as determinations about the manufacturing location and production methods.

A *Site* represents a constructurally, geographically or logically limited part of a batch producing Enterprise. Generally these limitations are based rather on organizational as well as entrepreneurial than on technical criteria.

An *Area* refers to a constructurally, geographically or logically limited part of a batch producing Site. Likewise regarding Sites these limitations are based on organizational and entrepreneurial criteria.

A *Process Cell* is a logical grouping of equipment to process one or more Batches. Logical control possibilities are defined according to the structure of Process Cells within an Area to develop control strategies (for instance in the case of emergency situations).

A *Unit* contains *Control Modules* and/or *Equipment Modules* to conduct larger process activities of one Batch (e.g. reaction or crystallization of a dilution). A Unit combines all necessary process devices to conduct these activities as an independent equipment grouping and may allocate also Common Resources for its tasks. Generally its elements are arranged around a processing installation (for example a reactor tank). According to IEC 61512 a Unit processes not more than only one single Batch or one part of it at any point in time.

An *Equipment Module* refers to a functional aggregation of devices which is generally arranged around a smaller processing installation (e.g. a filter) to conduct smaller process activities (e.g. dosing of material). It may either be part of a Unit or act as a Common Resource if defined as an independent device.

A *Control Module* represents the lowest level of device aggregation and is capable of performing basic control activities. Typically it contains sensors, actuators and other Control Modules to form a functional unit that can be operated as a single entity (for example a closed-loop control device that consists of a transmitter, a controller and a control valve).

The Equipment Control Concept

The *Equipment Control* enables an entity to be controllable and contains

- the *Basic Control*,
- the *Coordination Control* and
- the *Procedural Control*.

According to the standard IEC 61512 these three types of controls are necessary to enable an entity to provide process functionality.

The *Basic Control* provides the functionality to create or maintain a certain state of the entity or a process. It includes closed-loop controls, locking mechanisms, monitoring, exception handling and repeatable discrete controls. The Basic Control can react to process conditions that influence control outputs or require correction measures. It can be activated, deactivated or modified by either the operator or the other elements of the Equipment Control.

The *Coordination Control* directs, starts and modifies the utilization of the entity as well as the execution of the Procedural Control. For example it may contain algorithms for conflict management regarding the allocation of entities or algorithms to transfer the operational status of an entity.

The *Procedural Control* conducts entity-oriented activities in a scheduled sequence to perform process-oriented tasks. It comprises hierarchically combined procedural elements and enables entities to execute Batch Processes. These procedural elements form the *Procedural Control Model* (Figure 2.8c).

The Procedural Control Model

The Procedural Control Model consists of the four levels *Procedure*, *Unit Procedure*, *Operation* and *Phase*.

A *Procedure* refers to a strategy to conduct a process (e.g. the production of a Batch) and is defined by a sequence of *Unit Procedures*. As an *Equipment Procedure* it is part of the Equipment Control.

A *Unit Procedure* represents a strategy to execute a process within a Unit and contains a sequence of *Operations* as well as start, control and organization algorithms. These Operations cannot be distributed over several Units and only one Operation may be active within its Unit during any point in time. Nevertheless several Unit Procedures of one Procedure can be executed simultaneously if distributed over several Units – one Unit Procedure per Unit. As an *Equipment Unit Procedure* it is part of the Equipment Control.

An *Operation* is an independent process activity and consists of a sequence of *Phases* as well as start, control and organization algorithms. Typically an Operation refers to one chemical or physical transformation. As an *Equipment Operation* it is part of the Equipment Control.

A *Phase* represents the lowest level of procedural elements within the Procedural Control Model. It refers to a small process oriented task and can be segmented into steps and transitions likewise a *Sequential Function Chart (SFC)*. The task of a Phase is the execution of a process oriented action whereas its steps focus on the technical specifications of the used equipment. For example a step could be the activation or deactivation of a closed-loop control. A phase that is part of an Equipment Control is called an *Equipment Phase*.

Equipment Entities

Equipment Entities are composed by combining the lower four levels of the Physical Model with the elements of the Procedural Control Model to provide the necessary functionality to produce a Batch. Based on this context, the terms *Process Cell*, *Unit*, *Equipment Module* and *Control Module* still refer to the physical entities but also include the corresponding Equipment Control.

Figure 2.9 shows the correlations between the Procedural Control Model, the Physical Model and the Process Model. Process functionality can be achieved by mapping the Procedural Control onto the physical equipment. Though all models resemble each other concerning the number of levels (as already mentioned only the lower four levels of the Physical Model are treated in this standard), elements of different levels are combined to provide process functionality of a certain level.

The well-disposed reader may find more detailed information concerning the Equipment Control of the different levels in [IEC97].

2.3.3 Recipes

A *Recipe* contains the minimal necessary information and requirements to produce a certain product. It represents a method to describe a product including its corresponding production procedures and process-related informations.

Recipe Types

A *General Recipe* describes site-independent process requirements. It defines the required raw materials, their relative amounts and the necessary treatment without regarding any technical equipment.

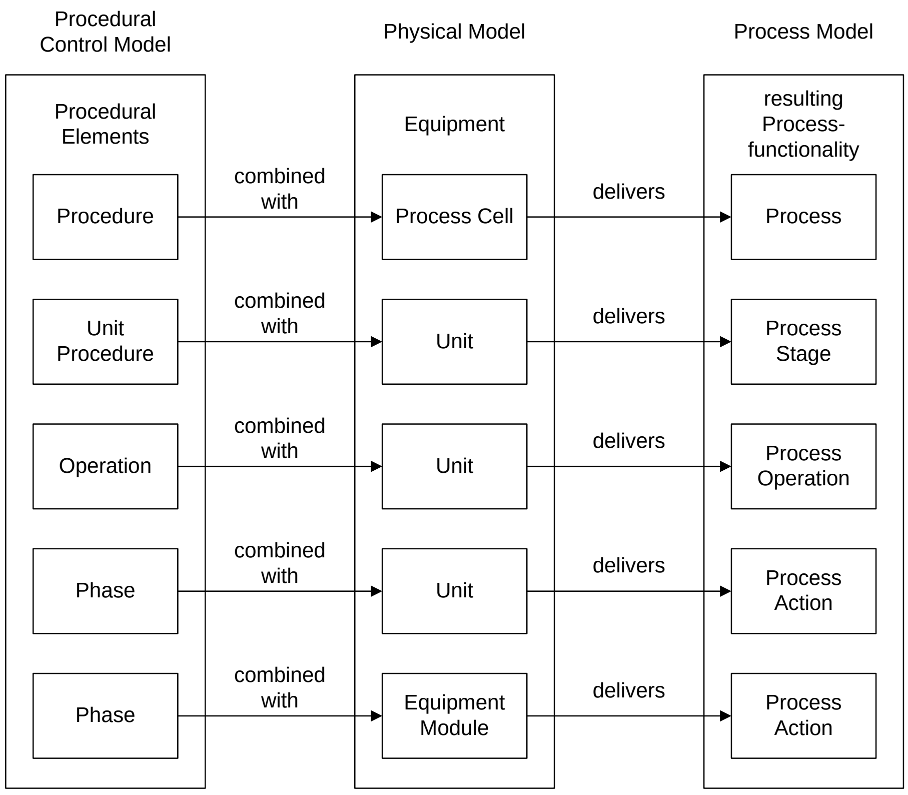

Figure 2.9: Procedural control mapped onto equipment to achieve process functionality [IEC97]

A *Site Recipe* represents a combination of a General Recipe and site-specific information. Usually it is derived from a General Recipe to meet the requirements of a certain production facility and offers details for a long-term production schedule. To manufacture a product in several Sites, more than one Site Recipe can be derived from a General Recipe.

The *Master Recipe* can be derived from a General or a Site Recipe. It is sufficiently adjusted to the attributes of a Process Cell to ensure the correct production of a Batch by combining its information with the provided functionality of the equipment. If various Process Cells execute different parts of the Site Recipe, more than one Master Recipe can be derived from a Site Recipe to meet these requirements.

Created as a copy of a Master Recipe, the *Control Recipe* is modified thereafter according to specific information concerning scheduling and execution to produce a certain Batch. It provides sufficient details to start and monitor the procedural entities of a Process Cell.

Recipe Contents

The *Recipe Head* comprises organizational information including recipe and product identification, version number, recipe draftsman, date of issue, etc. For example a Site Recipe may contain the name and version of the General Recipe it has been derived from.

A recipe further contains *Material and Production Data*:

- *Process Input:* name and amount of ingredients as well as manufacturing resources (e.g. technical equipment and manpower);
- *Process Output:* name and amount of material or energy expected to be the outcome of a recipe's execution;
- *Process Parameters:* detailed information that is not part of the Process Input or Output as for example temperature, pressure and time.

Equipment Requirements define the physical entities that can be used to execute a certain part of a Procedure. While General and Site Recipe specify usable technical equipment only in a general manner (e.g. by naming ingredients and therefore excluding certain technical installations that are not capable of processing these ingredients), the Master Recipe defines necessary requirements more specific to determine the used parts of the available Process Cells. The Control Recipe is used to allocate entities directly by their designation (e.g. tank 101).

The *Recipe Procedure* defines the strategy to produce a Batch. General and Site Recipe Procedures are structured corresponding to the Process Model since these recipe types describe the production process independently from any used technical equipment (Figure 2.10a). By contrast Master and Control Recipe Procedures are described according to the Procedural Control Model since these recipe types are closely related to the used Equipment Entities (Figure 2.10b). Not necessarily a 1:1 correlation has to exist between the process-related elements and their procedural-related counterparts therefore also 1:n or n:1 relations are legitimate. Please see Section 2.3.2 for more details on the Process and Procedural Control Models.

The category *Other Information* covers support information for the Batch Process which is not included in the other parts of the recipe. For instance this could be related to safety or regulatory information (e.g. by the FDA[5]).

[5]Food and Drug Administration

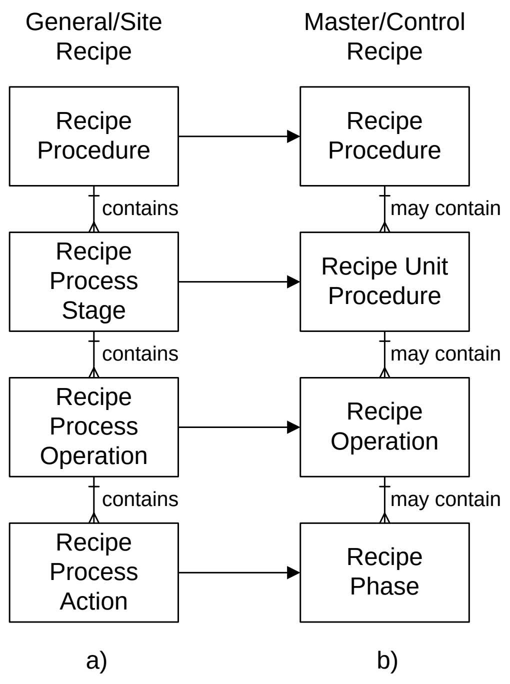

Figure 2.10: Recipe structures based on [IEC97]

Relation between the Recipe Procedure and the Equipment Control

The information of a Control Recipe is not sufficient to operate a Process Cell. Hence it is combined with the Equipment Control to enable the production of a Batch. If procedural elements appear more than once in a recipe every occurrence may be marked explicitly if necessary.

Procedural elements of the Recipe Procedure possess the following data:

- a description of the required functionality;
- information on material and production data as well as further parameters;
- specific requirements concerning the equipment.

By contrast the corresponding elements of the Equipment Procedure comprise the following features:

- an identification to be referred to by the elements of the Recipe Procedure or higher-level elements of the Equipment Procedure;

- a description of the provided functionality;
- variables to store the material and production data as well as further parameters of the recipe;
- execution logic.

2.4 Conclusion

This chapter gives an insight into Automation in general and Process Automation in detail since this thesis is concerned with applying a Modular Distributed Control Architecture to the domain of Process Automation.

Regarding automation concepts, Distributed Control Systems that are based on Mechatronic Control Components feature advantages concerning the modularity of an automation system as it can easily be extended or diminished. They resemble the physical structure of the target system and therefore facilitate an intuitive engineering method. The standard IEC 61499 presents architectural concepts for the implementation of such a control system.

The view on Process Automation delivers knowledge about the domain and its applied processes. As Batch Processes feature attributes of Continuous Processes as well as Discrete Processes, they represent reasonable target processes for the implementation of a MDCA since existing concepts based on Mechatronic Control Components can be derived and adapted to suit the requirements of Process Automation. Moreover Batch Processes prove to be a vital part in several process industries that rely on flexible production systems. The standard IEC 61512 delivers guidelines and structural concepts for the design of a control system for Process Engineering.

The gathered knowledge is taken into account for the development process concerning the design of an IEC 61499 Application to produce a Batch which is described in the following Chapter.

3 Design of a new Distributed Control Architecture for Process Automation

This chapter describes the development of an IEC 61499 Application for Process Automation. Due to the absence of a tangible method for the development process, different approaches are merged to create an intuitive engineering method. The system's structure is based on the structural models shown in Section 2.3.2 and methods to determine this structure as well as the provided functionality are derived from approaches introduced in the standards IEC 61512 and IEC 61804. The components of the system are based on an architecture developed by the ACIN[1] and inspected in reference to the standards IEC 61512 and IEC 61804. Finally a recipe of a simplistic process is created and then converted into an IEC 61499 Application by using the identified structures in conjunction with the Component-based Architecture.

3.1 General Conditions and Objectives

IEC 61512 introduces a general approach for the development of a system that is capable to produce a Batch. This approach is taken into account as the basis for the used engineering method.

The correct execution of a process concerning the production of a Batch requires fully developed equipment structures, process functionality and exception handling for the used equipment. These demands can be met by coordinating activities of the different engineering domains concerning the whole life cycle of a Process Cell.

A dual iterative design process is necessary to determine the suitable equipment objects as well as the fitting procedural elements (Figure 3.1). Considerations of one determination process influence the other one. On the one hand process-oriented considerations regarding the material treatment mark

[1]Automation and Control Institute of the Technical University of Vienna

the main criterion for the determination of the procedural elements that characterize the functionality of the corresponding equipment objects. On the other hand considerations concerning the provided functionality of equipment objects influence the range of possible procedural elements that can be chosen from.

Recipes can be developed by using these procedural elements as well as specific product information. The used equipment objects compose a line but conditions of disposition and allocation have to be taken into account. These activities provide the basic conditions for the production of a Batch.

Basic functions that refer to Equipment Phases (see Procedural Control Model in Section 2.3.2) should be determined in a manner to provide any available functionality of equipment not regarding any used recipes to ensure that new recipes can be developed easily by using existing basic functions. According to IEC 61512 the development and maintenance of these basic functions is a continuous activity to constantly improve the process technology.

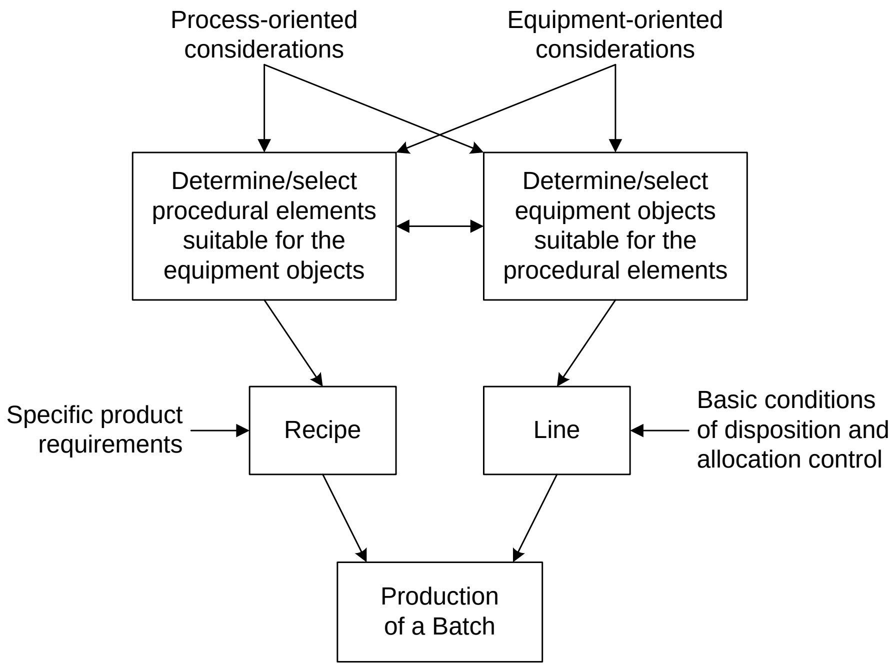
Process-oriented considerations
Equipment-oriented considerations
Determine/select procedural elements suitable for the equipment objects
Determine/select equipment objects suitable for the procedural elements
Specific product requirements
Recipe
Line
Basic conditions of disposition and allocation control
Production of a Batch

Figure 3.1: Parallel determination of procedural elements and equipment objects [IEC97]

3.1.1 The FESTO Compact Unit

The FESTO Compact unit, located in the Odo Struger Laboratory of the ACIN, acts as the target system for the presented control architecture [FES07]. In general this laboratory process plant is used for training and educational tasks to communicate industry-related competences. It is composed of various industrial components which provide training possibilities concerning open-loop as well as closed-loop control technologies [Hel04].

The development process presented in this chapter is described on the basis of the FESTO Compact unit to facilitate a better understanding. Figure 3.2 shows a picture of the laboratory process plant and a derived *Conventional Piping and Instrumentation Diagram (Conventional P&ID)*. Only the elements used later for the actual implementation are taken into account on the shown Conventional P&ID.

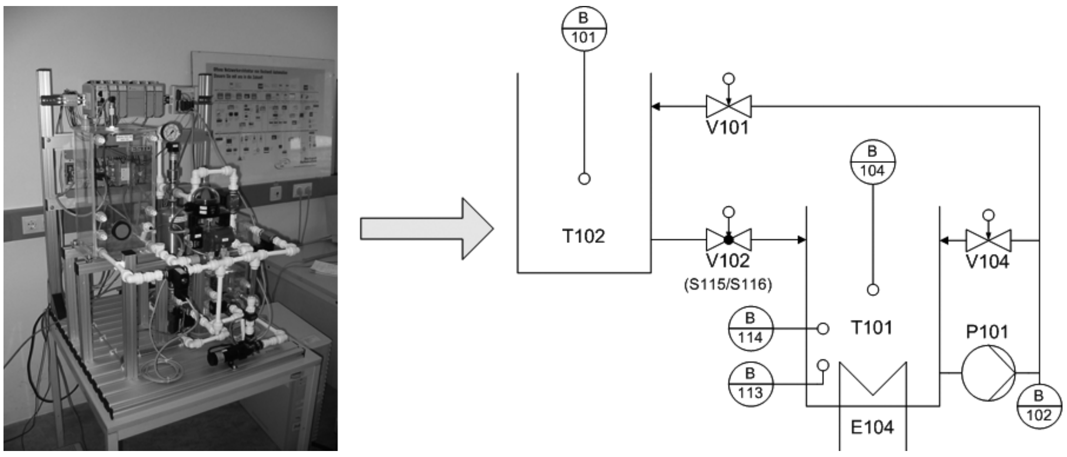

Figure 3.2: The FESTO Compact unit and the corresponding Conventional P&ID

3.2 Structure of the System

It is necessary to identifiy the functionality provided by the physical equipment in order to create procedural elements that can be used to compose recipes. This includes the determination of the system's structure concerning its physical elements. However, some amendments to the structural models introduced in the standard IEC 61512 are made to facilitate a better understanding. The functionality of the physical elements is identified by using an approach based on the standard IEC 61804. Once the functionality of the system is identified on all hierarchical levels, procedural elements can be developed, respectively a Component Libary can be created.

3.2.1 Amendments to the Structural Models and Concepts of IEC 61512

To achieve a consistent design of the Distributed Control Architecture, some amendments have to be made to the models and concepts introduced in the standard IEC 61512.

IEC 61512 defines the correlations between the Procedural Control Model, the Physical Model and the Process Model according to Figure 2.9 in Section 2.3.2. Evidently, elements of different levels of the Procedural Control Model and the Physical Model are combined to form elements of the Process Model. Inconsistencies occur especially regarding the Unit level of the Physical Model as it is combined with three different elements of the Procedural Control Model to form also three different Process Model elements. The approach of this thesis only correlates corresponding elements of the same levels.

As stated already in Section 2.3, the described models treated by IEC 61512 may be diminished or extended as long as the consistency of each model remains assured. Due to the physical limitations of the used equipment, which is presented in Section 4.1.1, it appears to be reasonable to diminish the Procedural Control Model by removing the level Unit Procedure and as a consequence to diminish the Process Model by removing the level Process Stage.

IEC 61512 presents the element Phase as the lowest level of the Procedural Control Model. However, its definition claims that a Phase can further be segmented into steps and transitions. This further aggregation shall be emphasized by introducing the *Step* as the lowest level of the Procedural Control Model. This facilitates the design process regarding the development of sequences by making Steps the elementary components that can be combined to form higher-level elements.

Though Control Modules are introduced as the lowest level of the Physical Model, they are not taken into account concerning the correlations between the structural models. Probably the reason might be their lack of a Procedural Control according to IEC 61512. Nevertheless the design process may be eased by regarding Control Modules within those correlations.

By mapping a Step of the Procedural Control Model onto a Control Module of the Physical Model, functionality on a very low level can be achieved. Therefore it is necessary to extend the Process Model by introducing the *Process*

Step as the lowest element to maintain consistency throughout the structural models of this approach.

Figure 3.3 presents the result of the described amendments. All structural models contain four levels and only elements of the corresponding levels are combined to provide functionality. A consistent structural concept as shown shall facilitate a more intuitive design process of the Distributed Control Architecture.

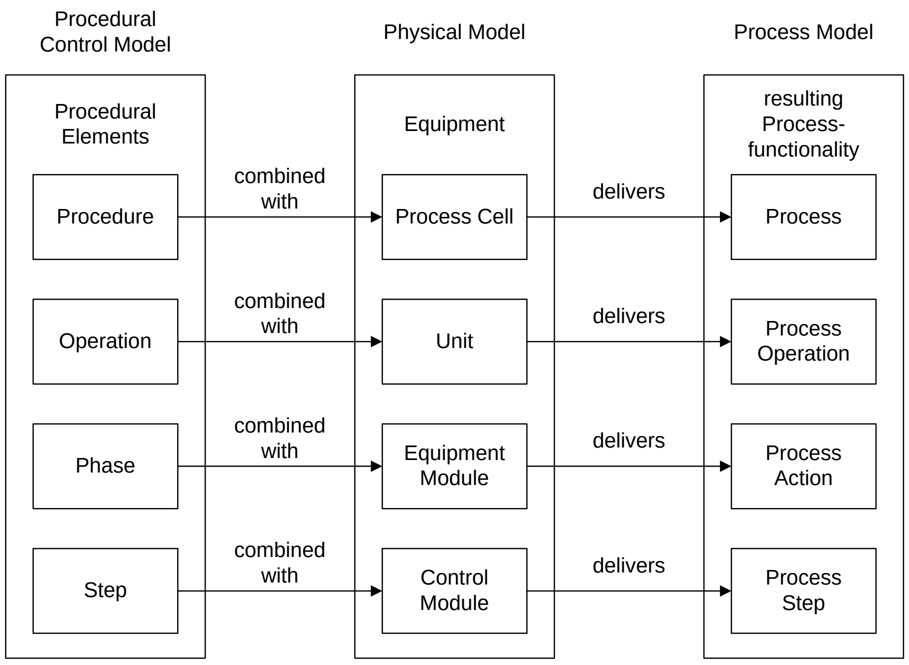

Figure 3.3: Procedural control mapped onto equipment to achieve process functionality

The examples of some structural elements presented in the standard IEC 61512 are not fully delimitable from each other. For instance the standard presents the following examples attached to the definition of a Process Operation:

- Fill: provide destilled water and reagents;
- React: provide vinyl chloride monomer and catalyst, heat up the reactor to a temperature between 55 and 60 degrees celsius and hold the reactor temperature between 55 and 60 degrees celsius until the reactor pressure drops.

As can be seen, material is added to the reactor in both shown Process Operations. This seems logical for the Process Operation *"fill"* as indicated already by its name, whereas *"react"* shows no clear evidence for adding material. More inconsistencies can be found in the presented examples within the standard IEC 61512. The implemented concept of this thesis takes consistency concerning name and structure of the used elements into account to facilitate a more comprehensible system design.

3.2.2 Development of the System Structures

IEC 61804 not only describes FBs for process control and requirements for PCS (see Section 2.2.2) but furthermore a method how to develop a control architecture. This method is used but modified to determine a physical structure of the used equipment based on the Physical Model of IEC 61512. Also the identification of the provided functionality hierarchically based on the Process Model is performed to develop procedural elements based on the Procedural Control Model that can afterwards be used for the creation of recipes.

IEC 61804 states that the requirements for FBs used in the control system are key elements for the design process. These requirements can be determined in the following four steps:

- Process Flow Diagram,
- Extended Piping and Instrumentation Diagram,
- Control Hierarchy Diagram and
- Functional Requirement Diagram.

Process Flow Diagram

The *Process Flow Diagram (PFD)*, which is in fact a Conventional P&ID, represents a graphical schematic view of a part of a process and displays the used equipment therefore referring to the main mechanical units, pipes between these units, remote transmitters and actuators. The information displayed on the PFD shall establish understanding for the process by showing the correlations between the used equipment and the major *Process Elementary Operations*. In general, Process Elementary Operations transform material and energy (for instance to cool water or to make water circulating). They have to be identified in order to create *Control Functions* in the next step. The Process Elementary Operations occuring in the used equipment are:

- fill the reactor with water,
- heat up the water in the reactor,
- circulate the water to ensure a homogeneous heat-up process and
- hand over the warm water.

The Process Cell used for the implementation of this concept consists of only two tanks. Tank T101 acts as the system's reactor vessel while tank T102 represents a storage vessel. In this case T102 stores the input as well as the output material of the reactor vessel which does not resemble a classical process line with separate tanks to store the raw material and the final product. Therefore the storage vessel T102 is virtually duplicated into two separate tanks. Tank T102a is used to provide the raw material which is cold destilled water while T102b receives the warm water from the reactor vessel after the heating process. Hence sensor B101 which is responsible to measure the water level of the storage tank is virtually duplicated and named B101a for measuring the level of T101a as well as B102b for measuring the level of T101b. Separate pipes for transporting the raw material and the final product already exist in the Process Cell, therefore no further additions have to be made.

Extended Piping and Instrumentation Diagram

The *Extended Piping and Instrumentation Diagram (Extended P&ID)* is widely used in chemical and petrochemical industries and represents a PFD marked with allocated Control Functions to display their relations with the process. Four different types of Control Functions are distinguished:

- Measuring Control Functions: to determine the values from one or several sensors;
- Open-loop Control Functions: to control actuators according to one or more sensor values or as a consequence of an order from an operator or a sequence;
- Closed-loop Control Functions: to control actuators according to one or more sensor values and a setpoint;
- Sequence Control Functions: to control actuators and schedule Open- and Closed-loop Control Functions as well as other sequences.

Control Functions are displayed as a graph between symbols referring to inputs, outputs and the control processing which is represented as a separate graphic symbol showing a tag of the corresponding Control Function. Inputs can either be values of sensors or data from other Control Functions. Outputs are either orders to actuators or data for other Control Functions. Therefore the Extended P&ID clearly displays the causal relationships between inputs and outputs of each Control Function.

IEC 61804 only describes an example with identified Control Functions of a single level. Nevertheless the concept treated in this thesis is based on hierarchical structures of IEC 61512 therefore the occuring elements of the Process Model and the Physical Model are now identified according to the four levels displayed in Figure 3.3 using a bottom-up approach.

The lowest level of the Process Model is represented by the newly introduced Process Steps (see Section 3.2.1). Figure 3.4 shows the identified Process Steps and, as can be seen, each Process Step only refers to only one Control Module which means one sensor or actuator.

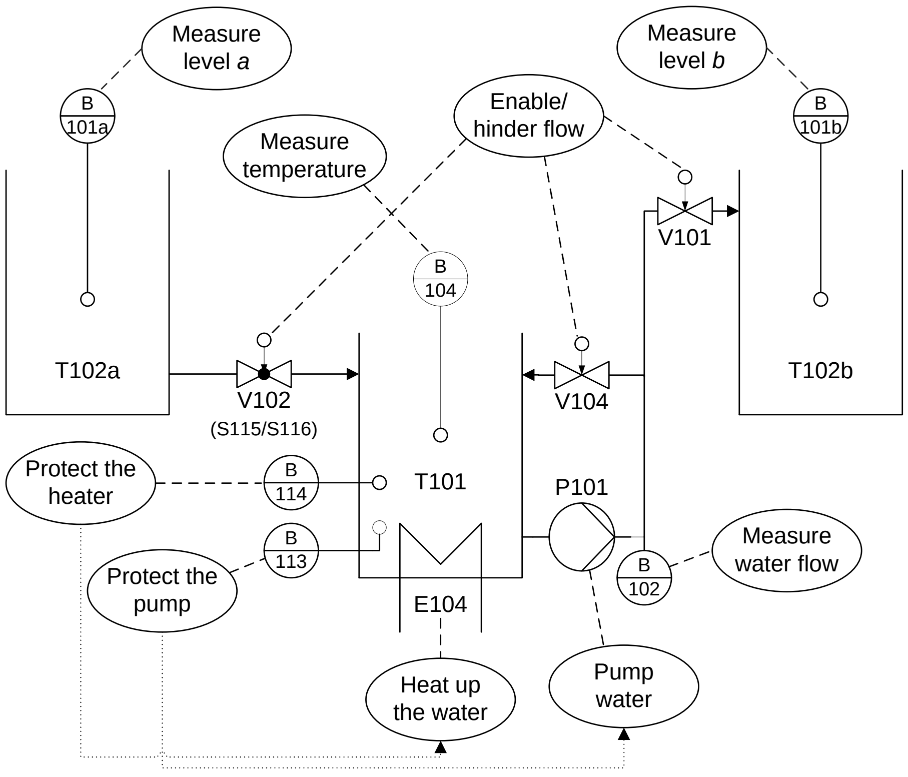

Figure 3.4: Extended P&ID with shown functionality on the Process Step level

Of course the valves V101, V102 and V104 provide their functionality (*“Enable/hinder flow”*) independent from each other. Concerning the development of the procedural elements, the functionality of the sensors B113 and B114 will be integrated into the functionality of the actuators P101 respectively E104 since these protection tasks are of utmost importance for the equipment and shall not be adressed manually by the operator or stated explicitly within a recipe. Evidently two further elements (S115 and S116) exist next to the ball valve V102. These are position sensors that observe the status of the ball valve. Similarly to the already mentioned protection tasks, the functionality of these two sensors is integrated into the functionality of the ball valve V102. Hence, three aggregated Control Modules can be identified:

- CM_P101: pump P101 combined with sensor B113;
- CM_E104: heating unit E104 combined with sensor B114;
- CM_V102: ball valve V102 combined with sensors S115 and S116.

The aggregation of the Process Steps leads to the identification of the Process Actions that rely on Equipment Modules (see Figure 3.5a):

- PA1: pump water with a certain flow rate;
- PA2: heat up water to a certain temperature;
- PA3: hold temperature;
- PA4: measure level difference.

While the sensors B101a and B101b each provide the functionality of the Process Action PA4 solely on their own and can therefore be referred to as Equipment Modules as well as Control Modules, the functionality of the Process Actions PA1, PA2 and PA3 is achieved by combining two Control Modules (one actuator with one sensor). A certain grouping of equipment, in this case the Control Module CM_E104 together with the sensor B104, even provides functionality for two different but related Process Actions. Hence, two aggregated Equipment Modules can be identified:

- EM_P101: CM_P101 combined with B102 delivers PA1;
- EM_E104: CM_E104 combined with B104 delivers PA2 and PA3.

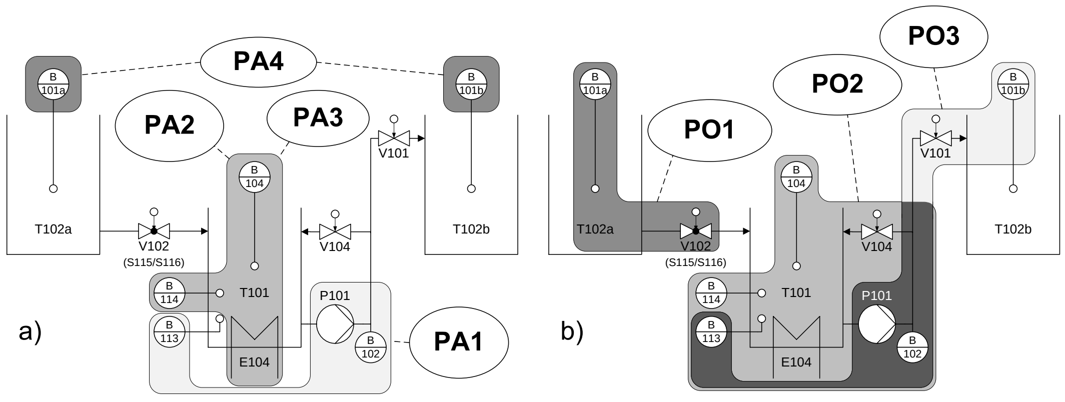

Figure 3.5: Extended P&ID with shown functionality on the Process Action level (a) and on the Process Operation level (b)

Aggregating Process Actions delivers the functionality of Process Operations that rely on Units (see Figure 3.5b). However, according to IEC 61512 it is legitimate to include even lower levels such as Process Steps directly into Process Operations. Three different Process Operations are identified by combining the functionality of Process Actions and Process Steps:

- PO1: fill reactor T101 with water;
- PO2: heat up the water and mix it;
- PO3: hand over warm water.

While PO1 uses equipment exclusively, the Process Operations PO2 and PO3 partially rely on the same actuators and sensors. Hence, the Equipment Module EM_P101 represents a Common Resource. Three different tanks exist therefore three different Units can be identified by using the tags of the tanks for the Units' names:

- U_T102a: consists of the Control Modules B101a and CM_V102 and delivers PO1;
- U_T101: consists of the Equipment Module EM_E104 and uses the Common Resource EM_P101 to deliver PO2;
- U_T102b: consists of the Control Modules B101b and V101 and uses the Common Resource EM_P101 to deliver PO3.

Finally the aggregation of the identified Process Operations delivers the Process itself that uses the complete Process Cell. The Process treated in this thesis can be referred to as “Produce warm water”.

The next step treats the creation of the framework describing the Control Functions after their identification.

Control Hierarchy Diagram

The *Control Hierarchy Diagram (CHD)* represents a graphical schematic and structural view of the Control Functions. The upper layers contain Control Functions to coordinate lower-level Control Functions including those on the lowest level that interact with sensors and actuators. The CHD is achieved by following three steps:

- extraction of the Control Functions from the Extended P&ID,
- definition of the requirements of the Control Functions and
- structuration of the documentation of the Functional Requirement Diagram (FRD).

After the Control Functions have been extracted, the CHD is used to describe their relations with the process interface as well as with sensors and actuators. Figure 3.6 displays the hierarchical structure of the Control Functions according to the Process Model of IEC 61512.

In general the determined requirements for Control Functions and their components such as FBs are constraints as for example accuracy, interface availability, safety or requirements such as the sampling time for control loops. At this stage all requirements are strictly functional without regarding a system of a certain vendor to be implemented. Adequate sampling times for the process elements are stated as follows:

- PA1 needs a control loop to control the flow rate: sampling time 100 ms;
- PA2 requires a cyclical determination of the temperature in the reactor: sampling time 200 ms.
- PA3 needs a control loop to hold the temperature in the reactor: sampling time 200 ms;
- PA4 requires a cyclical determination of the water level within the storage tank: sampling time 500 ms.

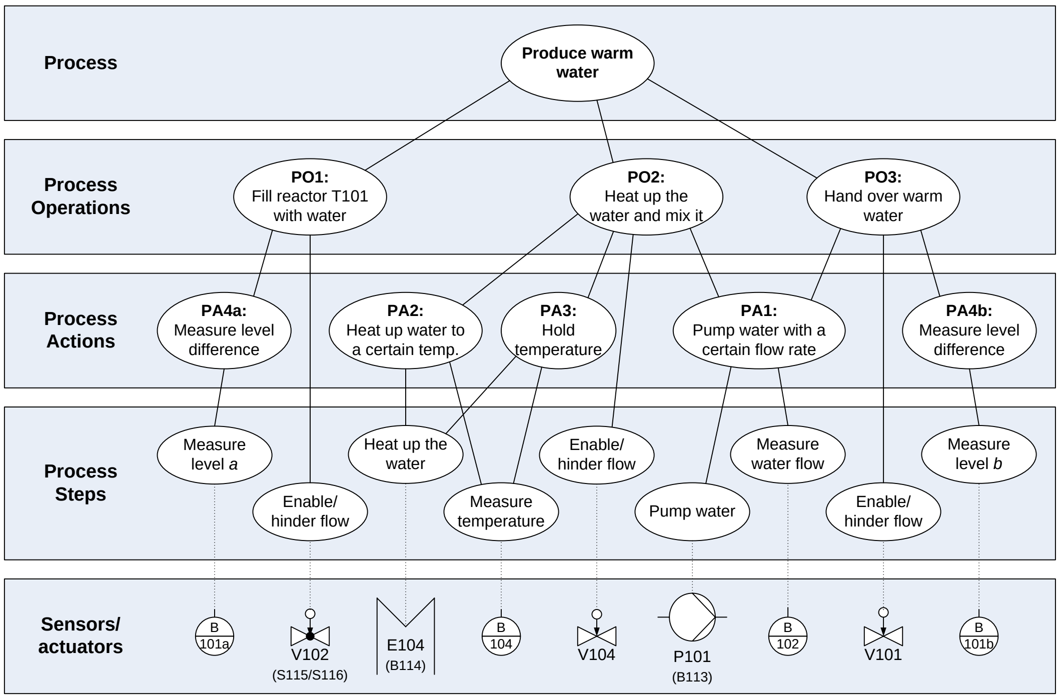

Figure 3.6: Amended Control Hierarchy Diagram with the elements of the Process Model and the controlled sensors and actuators

The Control Functions are then distributed into so-called *Folios* containing one or several Control Functions to create the Functional Requirement Diagram.

Functional Requirement Diagram

The *Functional Requirement Diagram (FRD)* contains a set of Folios that describe the requirements of allocated Control Functions. Each Folio incorporates four sections:

- Title of the Folio;
- Input signals: values received from outputs of other Folios which are used as inputs for some FBs within the Folio;
- Output signals: output values of some FBs within the Folio which are used as inputs for other Folios;

- Network of FBs: represents the behaviour of the Control Function and is responsible for signal processing.

FRDs are not directly drafts for programming but nevertheless specify the Control Functions with their requirements and constraints to control the process. Within the concept of this thesis, each Folio shall contain only one Control Function. The networks of FBs within the Folios will be developed according to the Component-based Architecture described in the following section.

3.3 Component-Based Architecture

A *Component-based Architecture* focuses on the use of Mechatronic Control Components including both intelligent sensor/actuator units as well as Aggregated Mechatronic Control Components (see Section 2.1.2). A Modular Distributed Control System based on the standard IEC 61499 can be developed by applying this architecture. Using a consistent structure for all components regardless of their position within the control system shall facilitate an easier use of this concept for vendors of components respectively automation engineers that develop those components as well as end-users respectively process engineers that focus on the development of recipes.

Using a Component-based Architecture that focuses on a consistently used structure facilitates a range of advantages [SZD06]:

- *Engineering efficiency:* This architecture enables vendors to create components including all necessary disciplines such as hardware control, software control or visualization. A shorter development process of automation systems can then be achieved by using these predefined components. Moreover subsystems can be commissioned separately due to the autonomous character of the components.

- *Flexibility:* Component-based Architectures facilitate fast modifications if requirements concerning the functionality of a plant change. Generally only the coordination logic at higher levels has to be adapted without affecting other levels of the automation system. Furthermore components are easily replacable without the need to modify components of the levels below or above.

- *Exploitation:* The automation system can easily be extended or dimished as components are simply added or removed.

- *Robustness:* Traditional systems with a central device suffer a complete shutdown if a failure occurs within the central device. However a failure of one component in a distributed system entails only local consequences. Moreover if a failure in the network connection occurs, an autonomous component is still able to provide local functionality.
- *Maintainability:* The hierarchical structure of the system facilitates a better understanding of the control program. Exchanging components is supported due to their encapsulation and clearly defined interfaces resulting in only limited change impacts.

3.3.1 Concept of a Component

The structure of the software concept for Mechatronic Control Components used in this thesis was developed by the ACIN and consists of the following elements [SZD06]:

- *Logic:* Is responsible to provide the basic control functionality of the component by combining inputs, outputs and inner device states.
- *Diagnostics:* Is responsible to perform diagnostic functions such as lifetime diagnostic (by counting switching operations of an actuator) or basic error detection (by using plausibility checks of sensor data).
- *Human Machine Interface (HMI):* Is used for interactions between operators and the device and is composed of two parts. One part represents the Mechatronic Control Component within an external visualization device (e.g. a handheld) while the other one is included in the Universal Component Interface and provides a communication interface for the external part.
- *Universal Component Interface (UCI):* The UCI represents an interface for the communication of the component with the external HMI as well as higher-level components. Requests to the component are received by the UCI and sent to the Logic and Diagnostics part. Furthermore the UCI is responsible to forward any responses by these parts to the HMI or higher-level components.

While the Logic and Diagnostics parts of intelligent sensor/actuator units address the physical elements of the used equipment (Figure 3.7a), the corresponding parts of Aggregated Mechatronic Control Components communicate with the UCIs of lower-level components which may themselves be either sensor/actuator units or aggregated components (Figure 3.7b). The automation system is build up hierarchically by intelligent sensor/actuator units on the lowest level and Aggregated Mechatronic Control Components on all other levels with a single component representing the whole facility at the top.

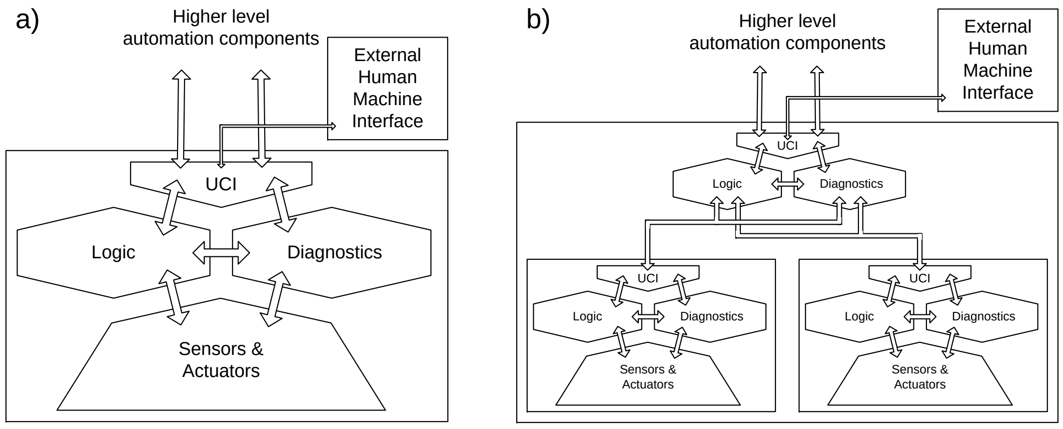

Figure 3.7: Architecture for intelligent sensor/actuator units (a) and Aggregated Mechatronic Control Components (b) [SZD06, Rai06]

In order to ensure the feasibility of the applied approach, the component architecture is inspected in reference to the standards IEC 61512 and IEC 61804.

Regarding the ACIN approach concerning the requirements stated by IEC 61804

The requirements stated by IEC 61804 (see Section 2.2.2) are met by the ACIN approach and the set of basic IEC 61804 Function Blocks, respectively their functionalities, can be implemented as follows:

- *Resource Block:* Variables contained in the Resource Block can be stored in the Logic or Diagnostics part depending on their nature.
- *General FB:* Device and network independent automation functions are carried out within the Logic of a component.
- *Technology/Transducer Block:* The Logic accesses actuators to change the status of the component while the Diagnostics part is responsible to

gather sensor data. Hence the functionality of Technology/Transducer Blocks is distributed over Logic and Diagnostics.

- *View/Trend Block:* The UCI incorporates the functionality of View Blocks since operators can gather data of the component via the internal part of the HMI which is included in the UCI.
- *Trend Block:* The temporal progression of parameters can be stored within the Diagnostics and accessed via the UCI therefore the functionality of Trend Blocks is distributed over these two software parts of the component.
- *Alert Block:* The functionality of Alert Blocks is distributed over the Diagnostics part and the UCI since the former is responsible for error detection while the latter sends these error messages to higher-level components or the external HMI.

Regarding the ACIN approach in compliance with the IEC 61512 Equipment Control Concept

IEC 61512 states that an Equipment Control comprises three types of controls to enable an equipment entity to provide process functionality (see Section 2.3.2). These controls are integrated into the ACIN approach of the software component as follows:

- The Basic Control manages the component's state by affecting actuators. It comprises for example control-loops and reacts on sensor data. Therefore the Basic Control is distributed over the Logic as well as the Diagnostics part of a Mechatronic Control Component.
- The Procedural Control executes procedural elements in a scheduled sequence that can access the Basic Control to change the entity's state. It is integrated in the Logic part of a component.
- The Coordination Control manages the utilization of the entity including the execution of the Procedural Control. It therefore is integrated in the UCI of a component and as such it is able to send orders to the Logic or the Diagnostics part. Furthermore it can transfer the entity's status to higher-level components or to the external HMI.

3.3.2 Realization with IEC 61499

The standard IEC 61499 presents a generic architecture for the use of FBs in Distributed Control Systems. Specific FBs represent the software parts of Mechatronic Control Components. Rainbauer introduces general concepts of FBs for the Logic and Diagnostics parts as well as the UCI (Figure 3.8).

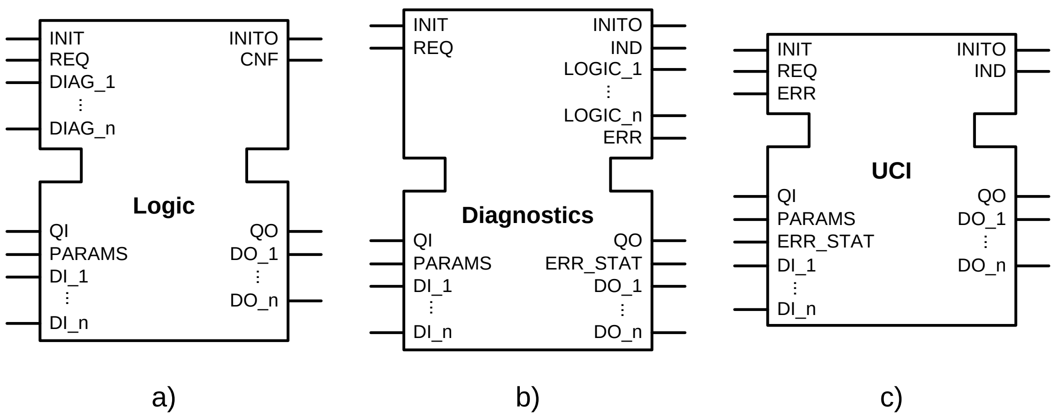

Figure 3.8: Logic (a), Diagnostics (b) and UCI (c) realized with IEC 61499 Function Blocks [Rai06]

The Logic part incorporates the component's basic functionality and contains BFBs and CFBs that comprise the control algorithms as well as SIFBs to access physical I/O ports. The functions of the event I/Os can be described as follows:

- *INIT:* Initializes the FB. Connections to other FBs are established.
- *REQ:* Activates the execution of algorithms in the Logic part. Data from the UCI is processed and physical I/O ports or lower-level components are addressed.
- *DIAG_1...DIAG_n:* Used to receive events triggered by the Diagnostics part.
- *CNF:* Confirms a successful completion of a task to the Diagnostics part or the UCI.

If deemed necessary, the Logic part can be extended with additional event connections for the internal communication of the component. The data I/Os are used for the following tasks:

- *PARAMS:* Are required for the initialization of the FB. They contain either ports and address bits of physical I/O ports in case of intelligent sensor/actuator units or IP addresses of lower-level components in case of Aggregated Mechatronic Control Components. Moreover these parameters can include certain information concerning the execution of algorithms.
- *DI_1...DI_n:* Comprise information sent by the UCI or the Diagnostics part.
- *DO_1...DO_n:* Contain information for the Diagnostics part.

The Diagnostics part contains BFBs and CFBs that comprise certain diagnostic algorithms as well as SIFBs to access physical I/O ports. The functions of the event I/Os can be described as follows:

- *INIT:* Initializes the FB. Connections to other FBs are established.
- *REQ:* Activates the Diagnostics part. Data from the Logic is processed.
- *IND:* Confirms a successful completion of the diagnostic task to the UCI.
- *LOGIC_1...LOGIC_n:* Used to send events to the Logic concerning new available data.
- *ERROR:* Is created in case of an error and forwarded to the UCI.

Likewise the Logic, the Diagnostics part can be extended with additional event connections for the internal communication of the component. The data I/Os are used for the following tasks:

- *PARAMS:* Are required for the initialization of the FB. They contain either ports and address bits of physical I/O ports in case of intelligent sensor/actuator units or IP addresses of lower-level components in case of Aggregated Mechatronic Control Components. Moreover these parameters can include certain information concerning the execution of algorithms.
- *ERR_STAT:* Contains an error message that can be processed in a higher-level component.
- *DI_1...DI_n:* Comprise information sent by the UCI or the Logic.
- *DO_1...DO_n:* Contain sensor data and further information for the Logic or the UCI.

The UCI contains certain SIFBs to coordinate the communication and data transfer between the levels of the system. The functions of the event I/Os can be described as follows:

- *INIT:* Initializes the FB. Connections to other FBs are established.
- *REQ:* Signalizes successful algorithm executions and new data.
- *ERR:* Signalizes the occurrence of an error.
- *IND:* Activates the Logic.

The data I/Os are used for the following tasks:

- *PARAMS:* Are required for the initialization of the FB. They contain the IP addresses of higher-level components and the external HMI as well as the IP address of an error channel solely used for transmitting error data.
- *ERR_STAT:* Contains the error message sent by the Diagnostics part.
- *DI_1...DI_n:* Comprise information sent by the Logic or the Diagnostics part.
- *DO_1...DO_n:* Contain information for the Logic or the Diagnostics part.

Though not stated in the general concept, solely the Logic addresses actuators within the implementation presented in this thesis. Likewise only the Diagnostics part gathers sensor data and forwards it to the Logic if needed. This approach proves to be consistent and works fine for the presented implementation. However, it may cause problems in the case of real-time applications since sensor data has to be sent to the Logic by the Diagnostics part in order to be processed in the Logic's algorithms. Real-time controls may calculate wrong control variables due to the lack of actual sensor information.

3.3.3 Dynamic Index System

Rainbauer implemented the introduced ACIN component architecture on a laboratory clamp sorting installation with Discrete Processes and a statically determined sequence of activities. Moreover each Mechatronic Control Component is addressed only by a single component of a higher level. In order to process events correctly and to achieve a correct sequence of activities, Rainbauer introduced statically determined indexes for each component.

However, a Process Cell shall provide the functionality to serve more than one recipe, therefore statically determined indexes need to be replaced by dynamically determined indexes. This guarantees the correct execution of activities while providing the possibility to modify the application when applying another recipe. Furthermore the functionality of a component at any level may be requested by multiple components of higher levels at different points in time therefore the confirmation event of this certain component must solely be sent to the actual higher-level component that requested the lower-level functionality making it necessary to mark the request distinctly.

Dynamically determined indexes are created during the execution of the application starting with the execution of the first Operation of the recipe. Each request to procedural elements has to be distinctly indexed leading to an easy identification of the momentary state of the activity sequence that can be used as well for locating errors. The Process itself, respectively the Procedure of the Control Recipe, does not need to be indexed.

Using whole numbers may be a rather simplistic approach for indexing the requests of procedural elements, but nevertheless suits the implemented application. The first executed Operation is indexed with the number “1”. Every following Operation receives an index each incremented by one. On request of a lower-level functionality the corresponding component receives the Operation’s index multiplied by ten. The increment by one as well as the multiplication by ten are carried out likewise on lower levels. Therefore the elements below the Procedure are indexed as follows:

- *Operations* are indexed with a single-digit number ranging from 1 to 9.
- *Phases*, respectively *Steps* directly addressed by Operations, are indexed with a double-digit number ranging from 10 to 99.
- *Steps* addressed by Phases are indexed with a triple-digit number ranging from 100 to 999.

Figure 3.9 shows the indexes that are created during the execution of the Procedure. Phase 2 is addressed by two Operations and therefore receives two indexes at different points in time. It further requests the functionality of two Steps which leads to two different indexes for themselves as well. The sequence of activities is well comprehensible due to the index system.

However, to avoid conflicts in the execution sequence, each Operation may address a maximum of ten Phases or Steps. Likewise each Phase may request

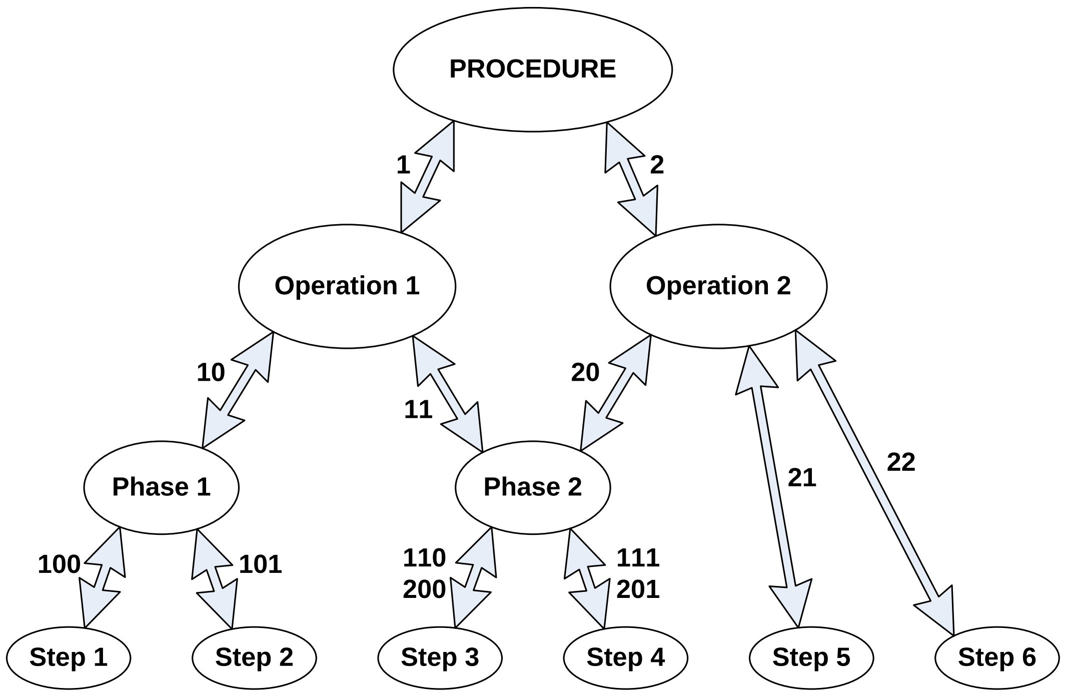

Figure 3.9: Indexes generated at run-time

the functionality of ten Steps at most. These limited numbers of procedural elements may not be sufficient for very complex applications, nevertheless the introduced Dynamic Index System presents a possible solution to ensure a correct sequence of activities.

3.3.4 Utilization of the Components

A *Component Libary* has to be created and is presented in Section 4.2. This library can contain procedural elements for all levels ranging from the top-most element, the Procedure, down to the Steps. The components provided in this library are used to convert a recipe into an application. This conversion can be done on any level of the hierarchical structures.

Performing the conversion on the lowest level requires a Control Recipe which includes every single Step that has to be carried out. An automation engineer only has to provide the components of the lowest level therefore a process engineer would need to determine the exact sequence of Steps to achieve the necessary functionality of the whole Process.

If the conversion would be done on the highest level, a Control Recipe would only contain the request for one specific Procedure. This seems to be an easy task for the process engineer who just has to choose a certain Procedure. However, the automation engineer has to provide the components at all levels in regard of all certain possiblities to create any recipes which is basically impossible.

Therefore a promising compromise proves to be the conversion on the Operation or the Phase level. In case the conversion is performed on the Operation level, the automation engineer has to develop a library that comprises procedural elements ranging from Operations down to Steps. The process engineer has to develop a Control Recipe on the Operation level that can then be transformed into a sequence of Operations to compose a Procedure for the Process Cell. The approach is likewise for the conversion on the Phase level.

An extensive Component Library does not necessarily serve all possible recipes therefore considerations concerning new recipes may require new functionalities not yet integrated in the library. The Component-based Architecture facilitates extensions since new Mechatronic Control Components can easily be added to the library to enlarge the range of provided functionality.

3.4 Development of the Application

While the development of the system structures and the Component Library would be the work of automation engineers, the development of a recipe to create an application is the domain of process engineers. Starting with a general description of a Process, different stages of recipes are composed by combining procedural elements to finally convert the recipe into an IEC 61499 Application that can be implemented on the Process Cell.

3.4.1 Composition of the Recipes

The NAMUR Recommendation NA 46 *Examples of Recipe-Based Operations to NAMUR Recommendation NE 33* describes a Process example starting with the development of a *Source Recipe* [NA 03]. In fact the term Source Recipe used within the NAMUR Recommendations refers to a General Recipe in the nomenclature of IEC 61512. Likewise the NAMUR term Basis Recipe refers to the IEC term Master Recipe. Development steps of both the NAMUR Recommendations as well as IEC 61512 are considered for creating the recipes. However, the nomenclature used for the presented development method is based solely on the standard IEC 61512.

The physical limitations of the used equipment narrow the range of possible

Process applications. Nevertheless even the development of a rather simplistic application indicates the feasibility of the shown design approach. The treated Process heats up distilled water and can be segmented into three stages:

1. Fill a certain amount of distilled water into the reactor;
2. heat up the water to a certain temperature and hold the temperature for a certain time;
3. remove the same amount of now heated water from the reactor.

Development of the General Recipe

The General Recipe is based on specific process-related requirements, product information and material parameters as well as requirements concerning the equipment. The general strategy to achieve the product is broken down into modular procedural elements based on the Process Model (see Section 2.3.2) by describing the required functionality on each level. These modular procedural elements are not bound to a certain recipe and therefore may be usable in several recipes.

The header of the General Recipe contains information concerning the purpose of the recipe and data about the input and output materials (see Table A.1 in Appendix A). A schematic view of the Process (Figure 3.10) displays that distilled water acts as the input material and warm distilled water represents the final product after undergoing the reaction which is in this case simply a heat-up process.

Not only the global values of the recipe but also the used procedural elements have to be described. Table 3.1 shows a brief description of the Process Operation *"Reaction"*.

Process Operation (PO) Description:	
PO-Name	**Operation**
Reaction	Heat up water: 1000 g of water is heated to 40 degrees celsius and shall be held on this temperature for 1 minute.

Table 3.1: Description of the Process Operation of the General Recipe – based on [NE 03]

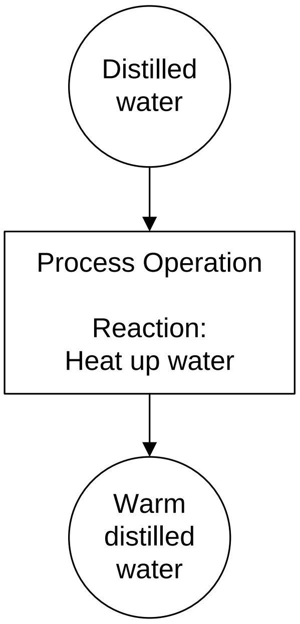

Figure 3.10: Schematic view of the process

The next step would be the development of a Site Recipe by combining the General Recipe with site-specific information such as the used language or certain attributes of the raw material. However, the raw material is just distilled water and the used language is English therefore a more detailed description of this development step is not deemed necessary and skipped.

Development of the Master Recipe

Master Recipes are derived from General or Site Recipes by deciding the method of operation (Continuous or Batch Process) and further by mapping the elements of the Process Model onto the elements of the Procedural Control Model (see Section 2.3.2). Procedural elements of the Master Recipe have to possess the capability to address procedural elements of the Equipment Control once they are used during the execution of the derived Control Recipe. Therefore these procedural elements of the Master Recipe must reflect the provided functionality of the target equipment.

This thesis treats Process Automation with the focus on Batch Processes which therefore represent the used method of operation. The header of the Master Recipe can be derived from the header of the General Recipe but contains additional information such as the run-time or the current status of the recipe (see Table A.2). A *Sequential Function Chart (SFC)* is adequate to describe the structure of the Master Recipe (Figure 3.11a).

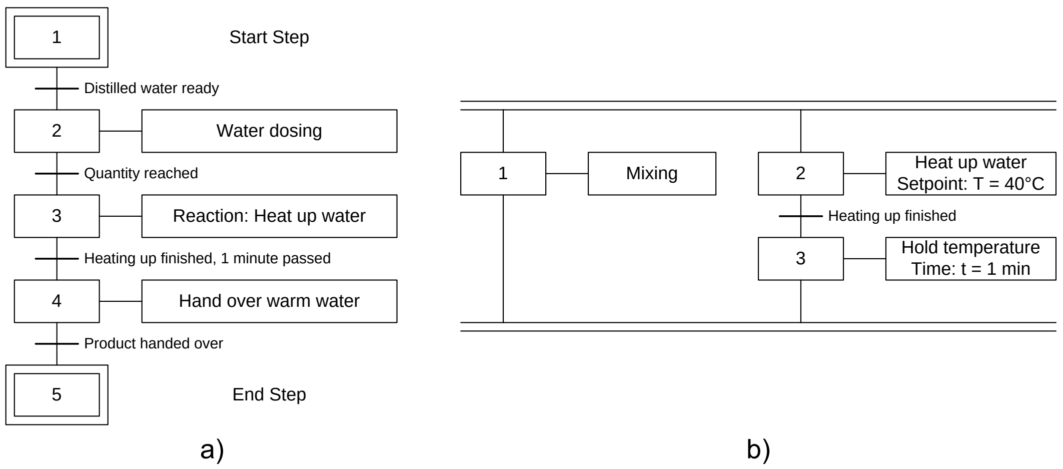

Figure 3.11: Sequential Function Chart of the Master Recipe (a) and of the Operation “Reaction” (b)

It generally begins with a Start Step without any activities. As soon as the conditions of a transition are met, the next step is executed. For instance as soon as the desired quantity of raw material (cold distilled water) is located inside the reactor vessel, the Operation “Reaction” to heat-up the water starts. Likewise Operations themselves are described by using SFCs. For example the Operation “Reaction” contains two parallel sequences of activities (Figure 3.11b). A continuous activity of mixing the water shall ensure a homogeneous distribution of warm water during the heat-up process and the holding time.

Development of the Control Recipe

The Control Recipe can be derived from a Master Recipe by combining it with disposition information and additional input by operators. This includes the verification of the Control Recipe to ensure its execution on the target equipment and the scale of the Control Recipe regarding the demanded batch size according to the scale rules of the Master Recipe and the disposition information. The header of the Control Recipe can be derived from the header of the Master Recipe but contains additional information such as the ordered amount of material or the scheduled start-time (see Table A.3). Likewise the structure of the Control Recipe is derived from the structure of the Master Recipe and described by using a SFC. However, information about the target equipment is added by determining sensors and actuators within the Control Recipe (Figure 3.12).

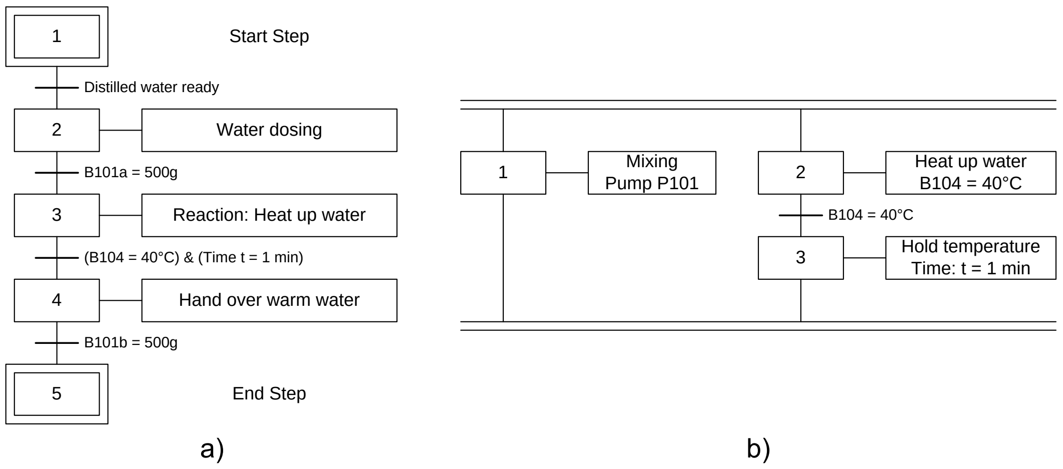

Figure 3.12: Sequential Function Chart of the Control Recipe (a) and of the Operation "Reaction" using reactor tank T101 (b)

3.4.2 Conversion into an IEC 61499 Application

Connecting the recipe with the Equipment Control shall be achieved by determining the procedural elements of the Equipment Control according to the procedural elements of the recipe. The Component Library contains the necessary procedural elements that have to be sequenced to achieve the production of the Batch. The conversion of the Control Recipe delivers the application that controls the Process Cell. Figure 3.13 shows the resulting IEC 61499 Application if the conversion is done on the Operation level.

The IEC 61499 Application resembles the segments of the Process described at the beginning of Section 3.4.1 and consists of three Function Blocks, each of them representing one Operation. If the Control Recipe would have been converted on the Phase level, the resulting application would comprise considerably more FBs, since each of them would only represent one Phase, respectively one Step, since some Steps are addressed directly by the corresponding Operations (Figure 3.6). In the applied recipe, the conversion on the Operation level proves to be a good choice to achieve a clearly arranged IEC 61499 Application.

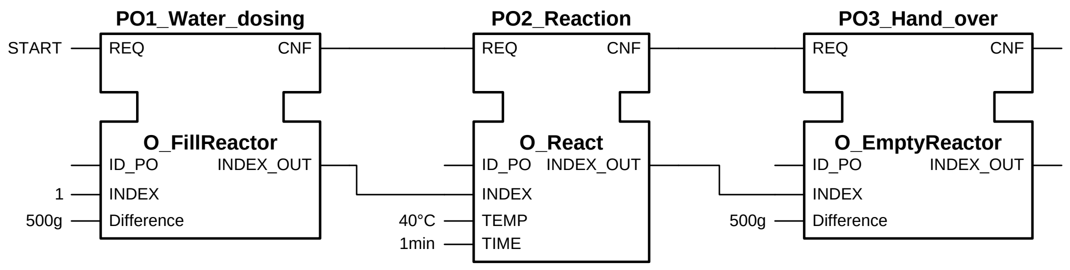

Figure 3.13: The IEC 61499 Application, representing the Procedure of the Process Cell, after converting the Control Recipe on the Operation level

3.5 Conclusion

This chapter explains the development of an IEC 61499 Application for Process Automation to achieve the producion of a Batch. The approaches used as a basis for these development activities are not new but their combination delivers a very intuitive engineering method (Figure 3.14). Two parallel activities can be identified that influence each other.

On the one hand, considerations concerning the used equipment leads to the determination of the system's structures that limit the possible range of applicable recipes. By identifying the functionalities on all levels, a Component Library with procedural elements based on Mechatronic Control Components is created. Furthermore these elements are used to compose an application by converting the elements of the Control Recipe into corresponding IEC 61499 Function Blocks.

On the other hand, considerations concerning the Process lead to the development of a Control Recipe that describes the methods that have to be applied in order to produce the demanded Batch. However, the conversion of the Control Recipe into an application may reveal a lack of functionality. In this case new elements can be added to the Component Library to meet the requirements of the recipe. It may be even necessary to physically extend the Process Cell itself to provide the demanded functionality.

Both activities together lead to the development of an IEC 61499 Application that provides the complete Equipment Control for the Process Cell. To finally produce the required Batch, the application has to be executed on the corresponding Process Cell.

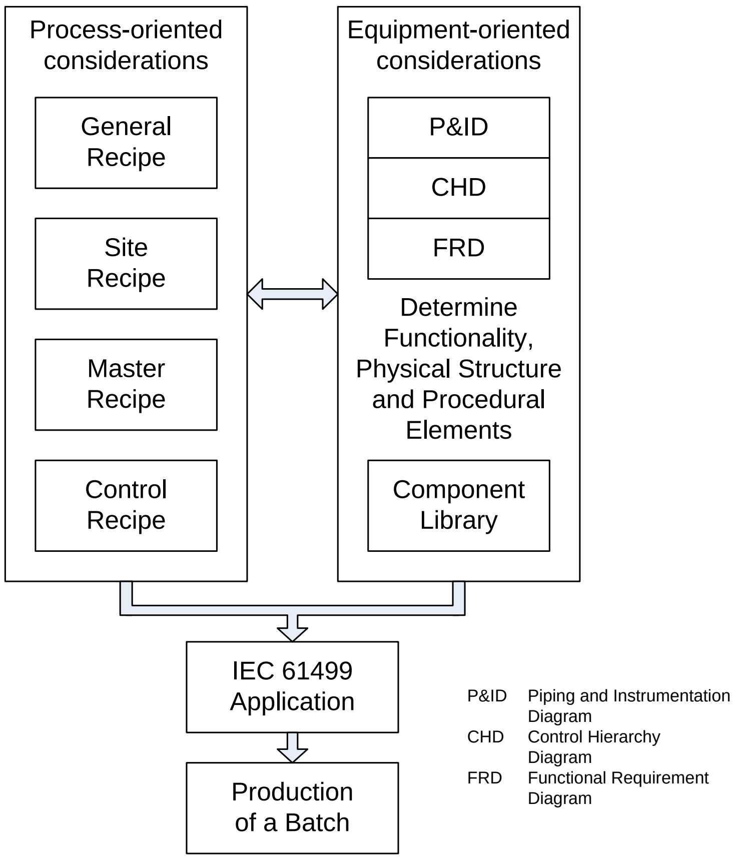
Process-oriented considerations
General Recipe
Site Recipe
Master Recipe
Control Recipe
Equipment-oriented considerations
P&ID
CHD
FRD
Determine Functionality, Physical Structure and Procedural Elements
Component Library
IEC 61499 Application
Production of a Batch
P&ID Piping and Instrumentation Diagram
CHD Control Hierarchy Diagram
FRD Functional Requirement Diagram

Figure 3.14: Structure of the engineering method

4 Implementation of the new Distributed Control Architecture on a Laboratory Process Plant

This chapter describes a first implementation of the designed concept on a laboratory process plant. The plant itself and the used controllers are presented in regard of their attributes as well as their purpose within the system. Accordingly the controllers are allocated to certain equipment groups in order to execute the corresponding software elements.

These software elements are created within the development of the Component Library. The focus lies on developing reusable software components which can be mapped onto more than just one equipment entity. Reusability is reached especially on the lower levels.

After mapping the software elements onto the physical equipment, the functionality for the Process is provided and an IEC 61499 Application is composed to achieve the production of the required Batch.

4.1 Target System and Software Tools

Different physical components as well as software tools are used for the actual implementation of the designed architecture. A laboratory process plant represents the target system for the architecture and a set of controllers provides the necessary execution abilities for the generated software elements. Software tools are used to create and control the application.

4.1.1 Overview of the Laboratory Process Plant

As stated in Section 3.1.1, the FESTO Compact unit acts as the target system for the presented control architecture. However, this implementation does not embrace all sensors and actuators of the Process Cell, therefore only those components actually used are described in this section.

Tanks

The FESTO Compact unit consists of two tanks. The lower tank T101 acts as the Process Cell's reactor vessel while the upper tank T102 represents a storage vessel. As previously stated in Section 3.2.2, tank T102 is virtually duplicated into the tanks T102a which provides the raw material and T102b which receives the final product from the reactor vessel.

Sensors

Various sensors provide the data for diagnostic functions as well as actual values for the closed-loop controls. These sensors deliver digital and analog signals and their tasks are defined as follows:

- Ultrasonic sensor B101, analog (range 0 – 10 V): measures the amount of water stored in tank T102. Likewise tank T102, the sensor is virtually duplicated into the sensors B101a which measures the level of T102a and B101b which measures the level of T102b;
- Flow sensor B102, analog (range 0 – 10 V): measures the water flow which is a result of the pump activity;
- Temperature sensor B104, analog (range 0 – 10 V): measures the temperature of the water located in the reactor vessel T101;
- Capacitive sensor B113, digital (24 V): indicates if the water level of tank T101 is sufficient to serve pump P101;
- Capacitive sensor B114, digital (24 V): indicates if the water level of tank T101 is sufficient to serve heating unit E104;
- Position sensor S115, digital (24 V): indicates if the ball valve V102 is closed;
- Position sensor S116, digital (24 V): indicates if the ball valve V102 is open.

Actors

The Compact unit comprises several valves to provide different flow directions. A water pump is responsible for creating the water flow and a heating unit carries out the actual "reaction" by heating up the water located in the reactor vessel. The tasks can be defined more specific as follows:

- Pump P101, analog (range 0 – 10 V): pumps water either to mix the material located in the reactor vessel or to hand over the final product to the storage tank;
- Heating unit E104, digital (24 V): heats up the water located in the reactor vessel;
- Magnetic valve V101, digital (24 V): opens to enable a water flow for transporting the final product from the reactor vessel to the storage tank;
- Magnetic valve V104, digital (24 V): opens to enable a water flow for mixing the water during the reaction, respectively the heat-up process;
- Ball valve V102, digital (24 V): opens to enable a water flow for transporting the raw material from the storage tank to the reactor vessel.

4.1.2 Controller

For the realization of an intelligent sensor/actuator unit, a controller is needed besides the actual sensor and actuator elements to execute the software components Logic, Diagnostics and UCI. Such an intelligent sensor/actuator unit comprises only few physical elements to control therefore its controller does not require large amount of I/O ports. Furthermore an extremely fast CPU[1] is deemed unnecessary since the implemented application does not include any critical real-time operations. However, analog values have to be gathered from sensors and sent to actuators, therefore the system as a whole requires at least some controllers with analog I/O ports. The system's communication relies on Ethernet therefore all controllers have to possess the capability for such a communication.

IPC@Chip – DK40, FC440 and FC660

The *IPC@Chip* is a controller from the company Beck with an i80186 core and a real-time operating system (RTOS) that supports programs developed

[1]Central Processing Unit

Figure 4.1: The FC660 from FESTO (a) and the DK40 from Beck (b) [FES07, Bec07]

in C and C++ [Bec07]. The IPC@Chip is applied in several control systems by Beck and FESTO such as DK40, FC440 and FC660 which are used for this implementation (Figure 4.1). They offer the following features:

- Ethernet
- RS232
- Memory: 512 kB RAM, 512 kB ROM
- I/O ports:
  - DK40: 8 bi-directional digital I/O ports (24 V)
  - FC440: 16 digital inputs (24 V), 8 digital outputs (24 V)
  - FC660: 32 digital inputs (24 V), 16 digital outputs (24 V), 3 analog inputs (range 0 – 20 mA), 1 analog output (range 0 – 20 mA)

TINI – Netmaster

The TINI[2] represents an embedded Java platform from Dallas Semiconductor and is based on the controller i8051 [Sem07]. It comprises a Java virtual machine and therefore is able to execute Java code.

The TINI is applied in the Netmaster from Elsist which is used for this implementation (Figure 4.2). The Netmaster offers the following features:

- Ethernet
- CAN

[2]Tini INternet Interface

Figure 4.2: The Netmaster controller from Elsist [Els07]

- RS232
- Memory: 512 kB/1 MB RAM, 512 kB ROM
- I/O ports: 12 digital inputs (24 V), 8 digital outputs (24 V), 4 analog inputs (range 0 – 10 V), 2 analog outputs (range 0 – 2.5 V)

Controller Allocation

For a perfect distribution and distinct sensor/actuator units, respectively Control Modules, each controller would control only the physical elements of one sensor/actuator unit (see Section 2.1.2). However, due to limitations concerning the available resources, some controllers are allocated to several sensor/actuator units. Nevertheless the software components of the individual units within such a controller are still separated according to the Modular Distributed Control Architecture. The system comprises

- one DK40,
- one FC440,
- one FC660 and
- two Netmaster controllers (one with 512 kB, the other one with 1 MB RAM).

The DK40 is allocated solely to the Control Module CM_V102. Diagnostic functions that rely on the sensors S115 and S116 are therefore carried out directly within the DK40. The I/O ports are used as follows:

Pin 0 (input)	S115
Pin 1 (input)	S116
Pin 2 (output)	V102

The FC440 is used to handle the Control Modules V101 and V104 which are in fact the magnetic valves. Its I/O ports are used as follows:

A0.0 (output)	V104
A0.1 (output)	V101

The FC660 is allocated to the Control Modules CM_P101 and CM_E104. This includes the diagnostic functions that ensure the safety of the actuators. The I/O ports are used as follows:

E0.2 (input)	B113
E0.3 (input)	B114
A0.0 (output)	P101
A0.2 (output)	E104
analog output	P101

The output port A0.0 has to be set to enable the analog control of the pump. However, the pump requires an analog voltage signal (range 0 – 10 V) but the FC660 is only capable to generate an analog current signal (range 0 – 20 mA) at the analog output port (pin 0). Therefore the current signal of the controller is sent over a 500 Ω resistor in order to deliver a voltage signal to the pump.

One Netmaster (512 kB RAM) gathers the data of the Control Module B101 (ultrasonic sensor). The following input port is used:

analog input E (channel 2)	B101

The other Netmaster (1 MB RAM) gathers the data of the Control Modules B102 (flow sensor) and B104 (temperature sensor). The following input ports are used:

analog input C (channel 0)	B102
analog input F (channel 3)	B104

However, due to the low performance of the Netmaster controllers [Dut03], only the actual SIFBs necessary to gather the sensor data are executed within the controllers, while the corresponding architectural software components Diagnostics and UCI are executed on a PC[3].

[3]Personal Computer

4.1.3 Software

For the development of the Component Library, the conversion of the Control Recipe into an IEC 61499 Application and the actual execution of the final application, certain software tools are required which are described briefly in this section.

Function Block Development Kit

The *Function Block Development Kit (FBDK)* is a freely available tool for creating system architectures based on the standard IEC 61499. It is obtainable on the FBDK website [HOL07].

Device types, resource types and FB types can be designed either with a graphical or textual method. After the development of the system within the FBDK, the FBs are mapped onto the defined devices and resources. To execute the created IEC 61499 Application, the FBDK contains the *Function Block Runtime Environment (FBRT)*. Both, the tool as well as the runtime environment, base on the programming language Java.

However, the parts of the application executed on the IPC@Chip controllers that access the sensors and actuators need a specific runtime desribed in the following subsection.

C++ Runtime

The C++ Runtime, developed for embedded systems at the ACIN, is executed on the IPC@Chip controllers in order to provide the environment for the execution of an IEC 61499 Application [Zoi02]. Though the runtime suits several controllers, it has to be modified according to the actual hardware of the target controller by adding or removing certain SIFBs to correctly access the hardware components.

4.2 The Component Library

The Component Library contains procedural elements, that can be combined to convert a recipe into an application. Even though the conversion might be done on the Operation level, the lower-level elements, Phases and Steps, have to be provided in order to execute the application on the corresponding Process Cell. All FBs have been created with the use of the FBDK.

Apart from a few exceptions on the Step level, all procedural elements comprise a Logic and a Diagnostics part as well as a UCI for the communication (see Section 3.3.1). To emphasize more reusability the procedural elements themselves are created as hardware-independent as possible, nevertheless they need to be mapped onto physical entities in order to provide the required functionality. Hence, for the definition of procedural elements that offer more specific functionalities on a higher level, the actual physical structure of the Process Cell has to be taken into account.

4.2.1 Steps

The Steps represent the lowest procedural elements and are mapped onto Control Modules in order to provide the lowest functionality, the Process Steps. The generated Steps are defined rather general concerning the hardware they can be mapped onto in order to keep them as reusable as possible while the higher-level elements such as Phases and Operations are defined in a more specific manner since they encapsulate more specific functionalities.

Though in general defined with a Diagnostics part that relies on sensors to detect errors, not all Control Modules have to possess sensors that indicate whether an error has occured or not. For instance the magnetic valves V101 and V104 are not equipped with position sensors like the ball valve V102. Nevertheless the Step *Open/close valve* which incorporates a Diagnostics part can be mapped on these Control Modules as well. Though no real sensor data is available, virtual sensors can be simulated by providing adequate data signals for the Diagnostics part to ensure no error message is created.

The general structure of the Steps is exemplified by describing the Step *Open/close valve* and its software structure in detail. The corresponding elements of the other Steps are likewise and therefore explained only briefly.

Open/close valve

The Step *Open/close valve* is used to handle valves. Figure 4.3 shows the software elements that are executed in the allocated controller. However, for a better visbility the INIT connections are not shown. Furthermore the values of the data inputs are not displayed in Figure 4.3 since they are not set until the component is mapped onto the equipment.

The Logic changes the state of the valve according to the orders received by the UCI. The Logic's event and data I/Os can be described as follows:

Event inputs	**Purpose**
INIT	Initializes the FB. Connections to other FBs are established. A flip-flop is set within the Logic to allow activity.
REQ	Changes the state of the valve according to the data input *WORKI* by accessing a physical output port.
DIAG_ERR	Forwarded event by the Diagnostics part in case of an error. The flip-flop is reset and no further valve activity is possible.
HMI_UNL	Forwarded event by the UCI from the operator to unlock and reactivate the valve after an error. The flip-flop is set.
Event outputs	**Purpose**
INITO	Passes the INIT event on to the Diagnostics part.
CNF	Confirms the completion of the required task to the Diagnostics part.
Data inputs	**Purpose**
QI	Needs to be set in order to allow any activity of internal FBs.
WORKI	Contains the required state of the valve: “1” is to open the valve, “0” to close it.
V_AD	Contains the address bit of the physical output port to control the valve.
Data outputs	**Purpose**
QO	Forwarded QI.
WORKO	Forwarded WORKI for the Diagnostics part.

The Diagnostics part checks whether the valve has reached the required end position within a given amount of time by gathering data from the position sensors. If the time is exceeded, an error message is created and sent to the UCI. The Diagnostic’s event and data I/Os can be described as follows:

Event inputs	**Purpose**
INIT	Initializes the FB. Connections to other FBs are established.
REQ	Activates a diagnostic algorithm to check the valve’s state.

Event outputs	**Purpose**
INITO	Passes the INIT event on to the UCI.
IND	Confirms the completion of the diagnostic algorithm in case the valve reached the required end position.
ERR	Confirms the completion of the diagnostic algorithm in case the valve did not reach the required end position.
Data inputs	**Purpose**
QI	Needs to be set in order to allow any activity of internal FBs.
WORKI	Contains the required state of the valve, forwarded from the Logic.
S1_AD	Contains the address bit of the physical input port that accesses the sensor responsible to check the “close” position.
S2_AD	Contains the address bit of the physical input port that accesses the sensor responsible to check the “open” position.
DT	Given time to the valve to change its state until an error event is created.
Data outputs	**Purpose**
QO	Forwarded QI.
S1	Contains the sensor data of the “close” position.
S2	Contains the sensor data of the “open” position.
ERR_NR	Contains the error message in case an error occurred.

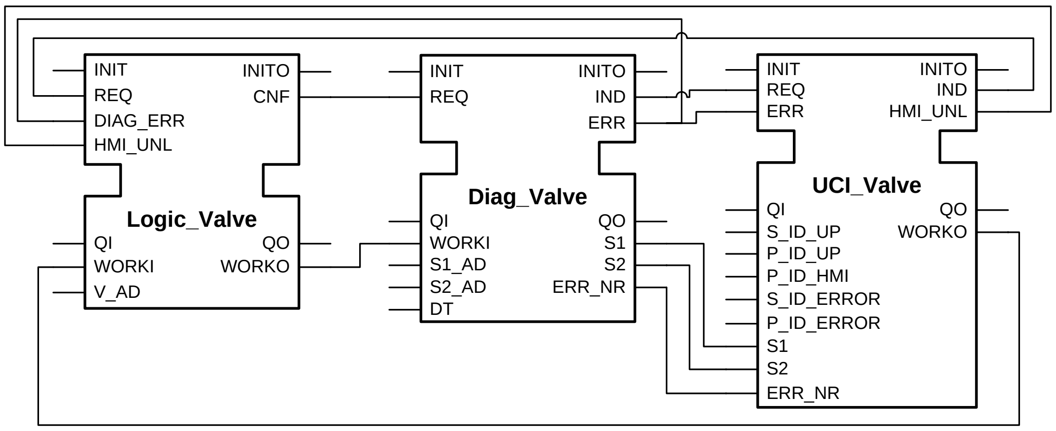

Figure 4.3: Software structure of the Step *Open/close valve*

The UCI receives orders concerning the valve's state from higher-level components and forwards them to the Logic. Furthermore error messages received from the Diagnostics part are sent to the corresponding higher-level components and the HMI. The UCI's event and data I/Os can be described as follows:

Event inputs	**Purpose**
INIT	Initializes the FB. Connections to other FBs are established.
REQ	Activates the UCI to send information to higher-level components and the HMI.
ERR	Activates the UCI to send an error message.
Event outputs	**Purpose**
INITO	Passes the INIT event on.
IND	This event is connected to the Logic to activate it in case new orders are received by the UCI.
HMI_ UNL	This event is forwarded from the HMI in case the operator wants to unlock and reactivate the valve after an error.
Data inputs	**Purpose**
QI	Needs to be set in order to allow any activity of internal FBs.
S_ ID_ UP, P_ ID_ UP	Contain IP addresses and ports for the communication with higher-level components.
P_ ID_ HMI	Contains IP address and port for the communication with the HMI.
S_ ID_ ERROR, P_ ID_ ERROR	Contain IP addresses and ports for the communication on separate error channels.
S1	Contains the sensor data of the "close" position, forwarded from the Diagnostics part.
S2	Contains the sensor data of the "open" position, forwarded from the Diagnostics part.
ERR_ NR	Contains the error message in case an error occurred, forwarded from the Diagnostics part.
Data outputs	**Purpose**
QO	Forwarded QI.
WORKO	Contains the required state of the valve, received from higher-level components.

Turn on/off CM

The Step *Turn on/off CM* is used to handle Control Modules that can be turned on or off (e.g. a heating unit). The Logic controls the state of the unit while the Diagnostics part checks whether it is safe or not to turn on the unit by gathering data from a sensor. An error message is created and sent by the UCI to the corresponding higher-level component in case the safety is not guaranteed if the unit's functionality is required or already in progress.

Activate CM

The Step *Activate CM* is used to handle analog Control Modules (e.g. a water pump). The Logic controls the state of the unit while the Diagnostics part checks if a safe activity of the unit is guaranteed by gathering data from a sensor. An error message is created and sent by the UCI to the corresponding higher-level component in case the safety is not guaranteed if the unit's functionality is required or already in progress.

Read analog sensor value

The Step *Read analog sensor value* is used to gather the value of an analog sensor. The Diagnostics part queries the sensor and forwards the data to the UCI which then transmits this information to the corresponding higher-level component. In the absence of any actuators, no Logic is required for this element.

4.2.2 Phases

The Phases are mapped onto Equipment Modules in order to provide the functionality of Process Actions. Some generated Phases have to be defined more specific than the Steps since they encapsulate more specific functionalities that rely on certain sensors and/or actuators.

The general structure of the Phases is exemplified by describing the Phase *PI-control* and its software structure in detail. The corresponding elements of the other Phases are likewise and therefore explained only briefly.

PI-control

The Phase *PI-control* is used to handle an Equipment Module containing an analog actuator and an analog sensor. Figure 4.4 shows the software elements.

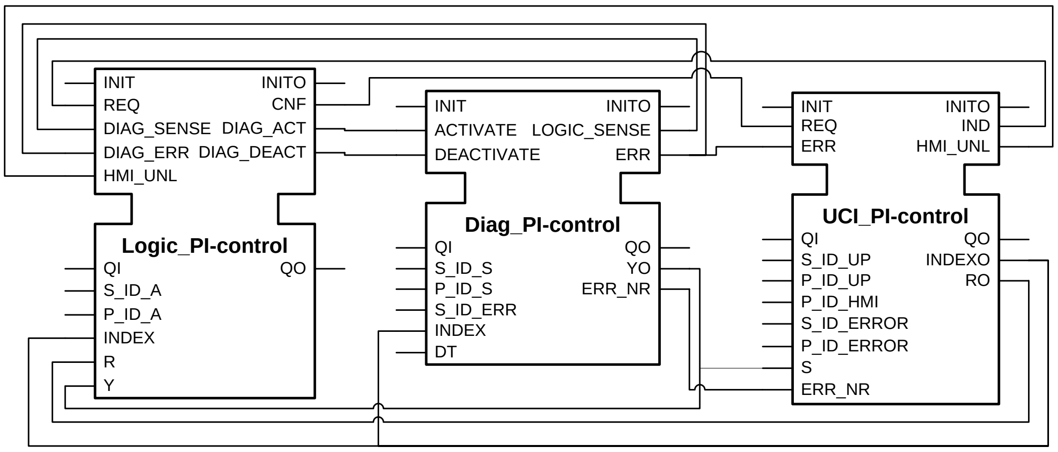

Figure 4.4: Software structure of the Phase *PI-control*

The Logic contains a PI-controller and the Step *Activate CM* to address the actuator. The required reference input is received from the UCI while the measured variable is received from the Diagnostics part. In case, the functionality of the PI-controller is required, the Logic activates a cyclical sensor scan performed by the Diagnostics part with the DIAG_ACT event. If the functionality is not required any longer, the cyclical scan is deactivated with the DIAG_DEACT event. The Logic's event and data I/Os can be described as follows:

Event inputs	**Purpose**
INIT	Initializes the FB. Connections to other FBs are established. A flip-flop is set within the Logic to allow activity.
REQ	Activates the PI-controller in case the data input *R* is not zero.
DIAG_SENSE	Initiates a new calculation of the control input and sets the actuator.
DIAG_ERR	Forwarded event by the Diagnostics part in case of an error. The flip-flop is reset and no further activity is possible.
HMI_ UNL	Forwarded event by the UCI from the operator to reactivate the PI-controller after an error. The flip-flop is set.

Event outputs	**Purpose**
INITO	Passes the INIT event on to the Diagnostics part.
CNF	Confirms the activation of the PI-controller to the UCI.
DIAG_ACT	Activates the Diagnostics part to cyclically gather data from the sensor.
DIAG_DEACT	Deactivates the cyclic activity of the Diagnostics part.
Data inputs	**Purpose**
QI	Needs to be set in order to allow any activity of internal FBs.
S_ID_A, *P_ID_A*	Contain IP addresses and ports for the communication with the Step *Activate CM*.
INDEX	Contains the index received by the UCI from higher-level components to ensure the correct sequence of activities.
R	Contains the reference input, forwarded from the UCI.
Y	Contains the measured variable, forwarded from the Diagnostics part.
Data outputs	**Purpose**
QO	Forwarded QI.

The Diagnostics part contains the Step *Read analog sensor value* to cyclically provide the sensor data to the Logic, respectively its PI-controller, if requested. Furthermore a certain FB is responsible to receive any error messages from lower-level components which are then sent to the UCI. The Diagnostic's event and data I/Os can be described as follows:

Event inputs	**Purpose**
INIT	Initializes the FB. Connections to other FBs are established.
ACTIVATE	Activates the cyclic scan of the sensor.
DEACTIVATE	Deactivates the cyclic scan of the sensor.
Event outputs	**Purpose**
INITO	Passes the INIT event on to the UCI.
LOGIC_SENSE	Confirms a new sensor value to initiate a new calculation of the control input by the Logic.
ERR	This event is created in case an error message is sent by a lower-level component.

Data inputs	**Purpose**
QI	Needs to be set in order to allow any activity of internal FBs.
S_ID_S, P_ID_S	Contain IP addresses and ports for the communication with the Step *Read analog sensor value.*
S_ID_ERROR	Contains IP address and port to receive error messages from lower-level components.
INDEX	Contains the index received by the UCI from higher-level components to ensure the correct sequence of activities.
DT	Sampling time for the cyclic sensor scan.
Data outputs	**Purpose**
QO	Forwarded QI.
YO	Contains the sensor data.
ERR_NR	Contains the error message in case an error occurred.

The UCI receives the reference input from higher-level components and provides it to the Logic in case the functionality of the PI-controller is required. Moreover error messages received from the Diagnostics part are sent to the corresponding higher-level components and the HMI. The UCI's event and data I/Os can be described as follows:

Event inputs	**Purpose**
INIT	Initializes the FB. Connections to other FBs are established.
REQ	Activates the UCI to send information to higher-level components and the HMI.
ERR	Activates the UCI to send an error message.
Event outputs	**Purpose**
INITO	Passes the INIT event on.
IND	This event is connected to the Logic to activate it in case new orders are received by the UCI.
HMI_UNL	This event is forwarded from the HMI in case the operator wants to reactivate the PI-controller after an error.

Data inputs	**Purpose**
QI	Needs to be set in order to allow any activity of internal FBs.
S_ID_UP, P_ID_UP	Contain IP addresses and ports for the communication with higher-level components.
P_ID_HMI	Contains IP address and port for the communication with the HMI.
S_ID_ERROR, P_ID_ERROR	Contain IP addresses and ports for the communication on separate error channels.
S	Contains the sensor data, forwarded from the Diagnostics part.
ERR_NR	Contains the error message in case an error occurred, forwarded from the Diagnostics part.
Data outputs	**Purpose**
QO	Forwarded QI.
INDEXO	Contains the index received from higher-level components to ensure the correct sequence of activities.
RO	Contains the reference input for the PI-controller, received from higher-level components.

Two-state-control

The Phase *Two-state-control* is used to handle an Equipment Module containing an actuator that can be turned on and off as well as an analog sensor. The Logic contains a two-state controller and the Step *Turn on/off CM* while the Diagnostics part contains the Step *Read analog sensor value* to provide the sensor data to the Logic. The two-state controller receives the reference input via the UCI and the sensor data from the Diagnostics. Depending on this sensor data, the two-state controller determines whether the actuator needs to be turned on or off in order to make the controlled system reach the demanded reference value.

Reach temperature

The Phase *Reach temperature* is used to handle an Equipment Module containing a heating unit and a temperature sensor. The Logic contains an algorithm to determine the necessary status of the heating unit and the Step *Turn on/off CM* while the Diagnostics part contains the Step *Read analog sensor value* to provide the sensor data to the Logic. In case the required temperature is reached, the Logic creates a confirmation event to indicate the successful completion of the task.

Level difference

The Phase *Level difference* is used to handle an Equipment Module containing an analog sensor that measures the liquid level of a tank. The Logic contains an algorithm to determine a level difference and creates an event in case the level difference is achieved. The Diagnostics part contains the Step *Read analog sensor value* to provide the sensor data to the Logic.

4.2.3 Operations

The Operations are mapped onto Units in order to provide the functionality of Process Operations. The generated Operations have to be defined according to the used Process Cell since they encapsulate very specific functionalities that rely on the actual physical structure and therefore on certain sensors and actuators of the Process Cell.

The general structure of the Operations is exemplified by describing the Operation *Fill reactor* and its software structure in detail. The corresponding elements of the other Operations are likewise and therefore explained only briefly.

Fill reactor

The Operation *Fill reactor* is used to provide the raw material to the reactor vessel. Figure 4.5 shows the software elements and a detailed view on the Logic. The FB "Status" represents the flip-flop that is set as long as no error occurs. If an error occurs, the flip-flop is reset and the FB "Permit" blocks any incoming REQ-Event. If the flip-flop is set and a REQ-Event is passed on, the Logic opens the valve V102 by addressing its corresponding Step *Open/close valve* with the FB "V102_Open" and therefore enables a water flow from the storage to the reactor tank. Moreover the Phase *Level difference* is addressed by the FB "Level_diff" which generates an event in case the required amount of raw material is provided meaning a certain level difference is achieved. Finally the Step *Open/close valve* is addressed a second time to close V102. The Logic's event and data I/Os can be described as follows:

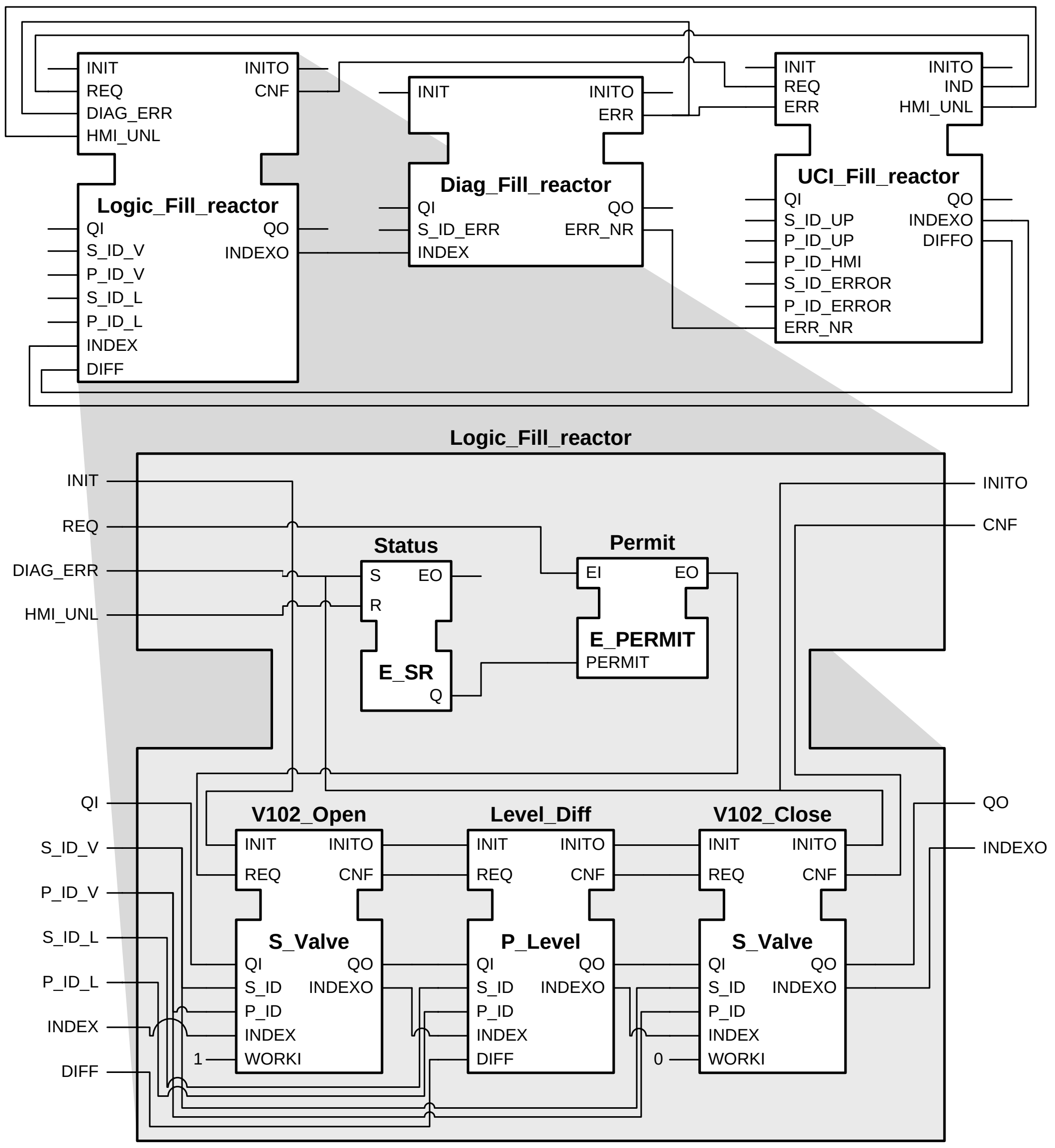

Figure 4.5: Software structure of the Operation *Fill reactor* and detailed view on its Logic

Event inputs	**Purpose**
INIT	Initializes the FB. Connections to other FBs are established. A flip-flop is set within the Logic to allow activity.
REQ	Activates the sequence algorithm which is represented by the internal FBs.
DIAG_ERR	Forwarded event by the Diagnostics part in case of an error. The flip-flop is reset and no further activity is possible.
HMI_UNL	Forwarded event by the UCI from the operator to reactivate this Operation after an error. The flip-flop is set.

Event outputs	**Purpose**
INITO	Passes the INIT event on to the Diagnostics part.
CNF	Confirms the successful completion of the sequence to the UCI.

Data inputs	**Purpose**
QI	Needs to be set in order to allow any activity of internal FBs.
S_ID_V, *P_ID_V*	Contain IP addresses and ports for the communication with the Step *Open/close valve*.
S_ID_L, *P_ID_L*	Contain IP addresses and ports for the communication with the Phase *Level difference*.
INDEX	Contains the index received by the UCI from higher-level components to ensure the correct sequence of activities.
DIFF	Contains the required level difference that is passed on internally to the Phase *Level difference*.

Data outputs	**Purpose**
QO	Forwarded QI.
INDEXO	Contains the index to ensure the correct sequence of activities.

The Diagnostics part solely gathers any occuring error messages from lower-level elements with a certain FB and sends them to the UCI. The Diagnostic's event and data I/Os can be described as follows:

Event inputs	**Purpose**
INIT	Initializes the FB. Connections to other FBs are established.

Event outputs	**Purpose**
INITO	Passes the INIT event on to the UCI.
ERR	This event is created in case an error message is sent by a lower-level component.
Data inputs	**Purpose**
QI	Needs to be set in order to allow any activity of internal FBs.
S_ID_ERROR	Contains IP address and port to receive error messages from lower-level components.
INDEX	Contains the index received by the Logic to ensure the correct sequence of activities.
Data outputs	**Purpose**
QO	Forwarded QI.
ERR_NR	Contains the error message in case an error occurred.

As this Operation is responsible for filling the reactor, the UCI receives the desired level difference from higher-level components and provides it to the Logic. Moreover error messages received from the Diagnostics part are sent to the corresponding higher-level components and the HMI. However, in the 61499 Application actually described in Section 3.4.2 the application itself represents a single Procedure and therefore is the only higher-level element that addresses any Operations. The UCI's event and data I/Os can be described as follows:

Event inputs	**Purpose**
INIT	Initializes the FB. Connections to other FBs are established.
REQ	Activates the UCI to send information to higher-level components and the HMI.
ERR	Activates the UCI to send an error message.
Event outputs	**Purpose**
INITO	Passes the INIT event on.
IND	This event is connected to the Logic to activate it in case new orders are received by the UCI.
HMI_UNL	This event is forwarded from the HMI in case the operator wants to reactivate this Operation after an error.

Data inputs	**Purpose**
QI	Needs to be set in order to allow any activity of internal FBs.
S_ID_UP, P_ID_UP	Contain IP addresses and ports for the communication with higher-level components.
P_ID_HMI	Contains IP address and port for the communication with the HMI.
S_ID_ERROR, P_ID_ERROR	Contain IP addresses and ports for the communication on separate error channels.
ERR_NR	Contains the error message in case an error occurred, forwarded from the Diagnostics part.

Data outputs	**Purpose**
QO	Forwarded QI.
INDEXO	Contains the index received from higher-level components to ensure the correct sequence of activities.
DIFFO	Contains the required level difference, received from higher-level components.

React

The Operation *React* is used for the reaction in the reactor vessel. The Logic opens valve V104 with the Step *Open/close valve* and activates the pump by addressing the Phase *PI-control* which is used for the flow control to ensure the water is evenly mixed. The water is then heated up by the Phase *Reach temperature* and the resulting temperature is held by the Phase *Two-state-control* for a certain amount of time. Afterwards the pump is turned off and V104 is closed. The Diagnostics part solely gathers any occuring error messages from lower-level elements.

Empty reactor

The Operation *Empty reactor* is used to hand over the final product to the storage tank. The Logic opens valve V101 with the Step *Open/close valve* and activates the pump by addressing the Phase *PI-control* which is used for the flow control. Furthermore the Logic contains the Phase *Level difference* which generates an event in case the required amount of the final product is handed over. Afterwards the pump is turned off and V101 is closed. The Diagnostics part solely gathers any occuring error messages from lower-level elements.

4.3 Mapping the Procedural Elements onto the Equipment

In order to achieve the actual Process functionality, the procedural elements have to be mapped onto the physical equipment. Some components are used more than once such as the Step *Open/close valve* which is used for the two magnetic valves and the ball valve. To map the components onto the equipment, they have to be allocated to the controllers. The Steps are executed directly within the available controllers, except those allocated to the Netmaster controllers (see Section 4.1.2). Phases, Operations and the Procedure as a whole are executed on a PC which acts also as the HMI.

To access sensors and actuators, the actual ports and address bits of the controllers have to be provided to the corresponding FBs on the Step level. Furthermore IP addresses and ports have to be assigned on all levels to ensure the communication between the components. For instance the Phase *PI-control* needs to communicate with a Step *Activate CM* which is mapped onto a water pump and a Step *Read analog sensor value* which is mapped onto the corresponding flow sensor in order to act as a flow control.

The following tables describe how the elements of the Component Library are mapped onto the equipment. The Steps are mapped onto the Control Modules to deliver functionality on the Process Step level as follows:

Procedural element: Step	**Control Module**	**Process functionality: Process Step**
Open/close valve	V101	*Enable/hinder flow*
Open/close valve	V102	*Enable/hinder flow*
Open/close valve	V104	*Enable/hinder flow*
Turn on/off CM	CM_E104	*Heat up the water*
Activate CM	CM_P101	*Pump water*
Read analog sensor value	B101	*Measure level a/b*
Read analog sensor value	B102	*Measure water flow*
Read analog sensor value	B104	*Measure temperature*

The Phases are mapped onto the Equipment Modules to deliver functionality on the Process Action level as follows:

Procedural element: Phase	**Equipment Module**	**Process functionality: Process Action**
PI-control	EM_P101	*PA1: Pump water with a certain flow rate*
Reach temperature	EM_E104	*PA2: Heat up water to a certain temperature*
Two-state-control	EM_E104	*PA3: Hold temperature*
Level difference	B101	*PA4: Measure level difference*

The Operations are mapped onto the Units to deliver functionality on the Process Operation level as follows:

Procedural element: Operation	**Unit**	**Process functionality: Process Operation**
Fill reactor	U_T102	*PO1: Fill reactor T101 with water*
React	U_T101 +EM_P101	*PO2: Heat up the water and mix it*
Emtpy reactor	U_T102 +EM_P101	*PO3: Hand over warm water*

In the absence of an available tool to directly convert a Control Recipe into an executable IEC 61499 Application, the FBDK is not only used to create the elements of the Component Library, but also to compose the application itself. Figure 4.6 displays the Procedure of the final IEC 61499 Application as it is shown in the FBDK. The overview on the left shows the devices and their allocated resources. As can be seen, the Control Modules CM_E104 and CM_P101 are allocated to the FC660 controller. This is likewise for the other controllers and their allocated Control Modules. For a better clarity all Phases are executed in one device on the PC. Likewise all Operations are executed within one device on the PC.

The Procedure itself represents one resource within one further device and contains five FBs. The FB "Start" creates the INIT-event to initialize all FBs. The production of the Batch is started, as soon as the FB "Start_Production" receives a signal from the HMI. In this case this FB generates an event which is passed on to the FB "PO1_Water_dosing". When the desired amount of water (500g), is located in the reactor vessel, the FB "PO2_Reaction" is executed. As soon as the required temperature (40°C) is reached and the holding time (1 minute) has passed, the warm water is handed over to the storage tank

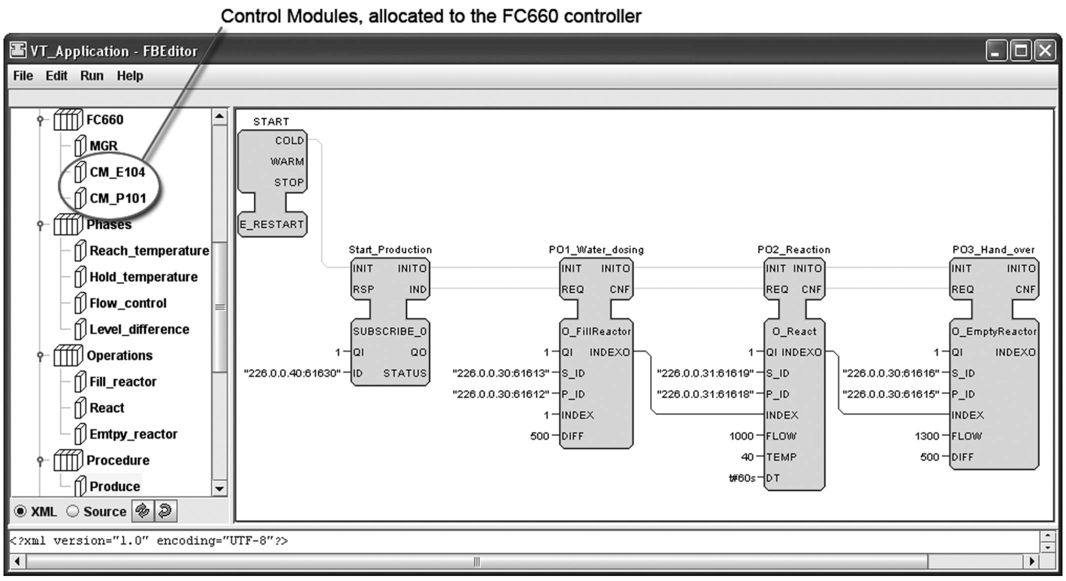

Figure 4.6: The final IEC 61499 Application displayed in the FBDK

during the execution of the FB “PO3_Hand_over”. After the completion of this sequence, further Batches can be produced likewise.

4.4 Conclusion

The FESTO Compact unit provides an adequate range of sensors and actuators for a first implementation of the concept presented in Chapter 3. Using several small controllers that each host the software elements of a few or even just one sensor or actuator emphasizes the distributed design of the architecture.

The FBDK is used to create the elements of the Component Library and to compose the application. In the absence of a tool to directly convert recipes into applications based on the shown approach, both automation engineers and process engineers have to use the same tool to fulfill their tasks.

As the conversion is done on the Operation level, the automation engineer’s work is to create the Component Library and map its elements onto the corresponding equipment entities concerning Steps, Phases and Operations. The process engineer converts the Control Recipe by sequencing the Operations and by adding values to certain data inputs. The final IEC 61499 Application enables the production of the required Batch.

5 Discussion of Results

The Distributed Control Architecture presented in this thesis and its corresponding design method that leads to the development of an IEC 61499 Application offer several gains for the process industry but nevertheless are faced with some problems as well. The following sections discuss the results concerning the design method as well as the actual implementation.

5.1 Gains of the Design Process

The developed design method which leads to the creation of an IEC 61499 Application offers advantages not only in regard to the control system but also concerning technical standards. Though these standards already represent a good basis for any development activities, their modification can lead to further gains.

Amendments to the Structural Models of IEC 61512

The structural models of the standard IEC 61512 provide a solid basis for any structural considerations concerning a control architecture for Process Automation. Nevertheless it is reasonable to modify these structural models to gain more consistency and to better support any design activities. As the elements of the different structural models need to be connected, more consistency can be achieved by correlating solely elements of the same level. Moreover, the introduction of a further level below the existing ones emphasizes the approach of aggregating lower-level components to compose higher-level components. The addition of Steps to the Procedural Control Model and Process Steps to the Process Model leads to the integration of the Control Modules into the general relational structure (as is shown in Figure 3.3). These amended structural models imply more consistency and are suitable for the actual design method shown in this thesis.

An Engineering Method for a Modular Distributed Control Architecture in Process Automation

Section 3.2.2 describes the extraction of the functionality provided by the Process Cell to determine the conditions for a later software component design. The applied method is based on the standard IEC 61804 but modified in order to be compatible to the used structural models. This extraction approach combined with a Component-based Architecture leads to a very intuitive method for the creation of software elements for Mechatronic Control Components.

After the development of a Control Recipe, these Mechatronic Control Components are used for the composition of an executable application based on the standard IEC 61499 (see Section 3.4.2). Due to the Component-based Architecture, this conversion can theoretically be done on any level. However, not all levels prove to be a good choice (see Section 3.3.4). A conversion done on the Step level requires a very detailed Control Recipe and would lead to an application which is neither easy to be understood nor convenient to be modified. The benefits of encapsulated functionalities would simply be lost and the application would bear resemblance to classical PLC programs. However, a conversion done on the Procedure level implicates disadvantages as well since the Component Library would need to cover virtually any possible recipes that could be executed on the Process Cell. The development of such a library is in fact impossible. Therefore it is advisable to convert Control Recipes at either the Phase or the Operation level to optimize the engineering efforts.

In total, the combination of existing approaches leads to a very intuitive engineering method for applying the Distributed Control Architecture to the domain of Process Automation (see Section 3.5).

5.2 Outcome Concerning the Implementation

Though the design method features several advantages, the actual implementation reveals certain problems. The development of the software elements for the Component Library unveils a weakness of the used component concept. Initially the intended aim was to develop software elements that support two types of reusability. However, this aim is reached only partially.

Firstly, created software elements shall be able to handle more than just one type of specific sensor/actuator unit or grouping. While this is partially possible on the lower levels, the software elements for higher-level components have to be designed very specific with regard to the target equipment (see Section 4.2.3). On the contrary the designed software elements on the Step level can be used to handle several different sensor/actuator units (see Section 4.2.1). For instance the Step *Open/close valve* is used for the ball valve as well as the magnetic valves.

Secondly, the software elements shall be independent from any controller type they might be allocated to. The actually created software elements on the Step level rely on internal SIFBs to access the physical I/O ports of the controller. These SIFBs have to be designed specifically for certain controller types, therefore the resulting software elements on the Step level are not independent from the used controller types. However, higher-level components are not bound to specific controller types and can be executed regardless of the controller's hardware configuration.

Nevertheless the instance specific parameterization during the mapping activity proves to be a problem concerning the reusability of software components. However, a certain reusability is evident since some components of the lower levels are suitable for more than just one type of sensor/actuator unit. This offers advantages regarding possible modifications of the physical equipment. For example a magnetic valve can be substituted by a ball valve without the need to change any software elements if the same type of controller is used.

The actual IEC 61499 Application can be designed intuitively by simply composing a sequence of FBs that represent the Operations of the Control Recipe (see Figure 4.6). Parameters concerning the production of the Batch can easily be modified by changing the input values of the FBs.

5.3 Summary of the Results

Though certain problems still need to be solved, the introduced Distributed Control Architecture features a very intuitive engineering method and clearly structured applications that correspond to the developed recipes. The results of this work are summarized as follows:

- Consistent system structures due to modifications concerning the structural models presented in the standard IEC 61512;
- Intuitive engineering method for the development of a control system for Process Automation;
- Easy conversion of recipes into applications;
- Certain reusability of control components;
- Substitutability of lower-level control components.

6 Conclusion and Outlook

Even though certain fields of the process industries such as the petrochemical sector might be rather conservative and rely on classical control systems, other fields such as the pharmaceutical and biotechnological sectors need flexible control systems to produce a wide range of products in small amounts. These flexible control systems should enable companies to quickly react on changing market demands.

Development Activities of this Thesis

This thesis treated the development and implementation of a Modular Distributed Control Architecture to the domain of Process Automation. However, not only the resulting architecture but also the design method of an actual application were examined in regard to possible gains and problems.

The new Distributed Control Architecture featured in this thesis was based on the use of Mechatronic Control Components. For the actual software realization, the guidelines provided by the standard IEC 61499 were determined to be suitable.

To become acquainted with the field of Process Automation, the target domain was analyzed pertaining to its conditions and requirements. Not only occurring processes, but also concepts of existing engineering methods and designs were examined. The focus was laid on Batch Processes due to their relevance for several process industries. Particularly in regard to Batch Processes, the standard IEC 61512 provided structural models that could be used as a basis for any structural considerations.

An intuitive engineering method was derived by amending and combining several approaches. This engineering method comprised two parallel development activities:

- Equipment-oriented considerations: By modifying an approach of the standard IEC 61804, the functionality provided by the used Process Cell was determined. Moreover a Component Library was created by using an existing Component-based Architecture, that had already been applied to serve Discrete Processes. This architecture was analyzed in consideration of Process Automation and amended to suit the target domain's demands.

- Process-oriented considerations: To facilitate an easy understanding of the applied concepts, a rather simplistic Control Recipe was created by using the NAMUR Recommendations NE 33 and NA 46 as guidelines. However, the used nomenclature was based solely on the standard IEC 61512.

The combination of both activities led to the development of an IEC 61499 Application that was implemented on a laboratory process plant.

The laboratory process plant was represented by the FESTO Compact unit. Though the actual creation of the Component Library was faced with certain problems, an IEC 61499 Application was composed that resembled the functional structure of the physical system as well as the Process, respectively the Control Recipe.

The final task of this thesis was to present a discussion of the obtained results. This discussion included the achieved gains but also the faced problems. Possibilities for further research activities can be derived from these results and are presented in the following section.

Further Research Topics

Though the applied Modular Distributed Control Architecture provides a certain range of reusability, a research emphasis has to be laid on the concepts concerning the instance specific parameterization of the Mechatronic Control Components. Evidently this statement applies especially to lower-level components that interact directly with the hardware, as they need to be executed on certain controller types to address the sensors and actuators. However, the integration of these components could already be simplified by providing complete intelligent sensor/actuator units. A complete unit comprises the sensors and actuators as well as an integrated controller with software elements on the Step level which are already configured to handle the hardware elements. Moreover, a certain FB has to be provided in order to make the component addressable for higher-level components such as Phases or Operations.

In the absence of a certain tool for the conversion of recipes into applications, the FBDK was used to compose the actual application. Even though a development tool as the FBDK is required to create the Component Library, a further tool for the conversion of a Control Recipe into an application seems reasonable. A process engineer could then create an application by sequencing components on a certain level (for example on the Operation level) without the need to add the required lower-level components manually. Evidently this tool contains a Component Library, which can be extended by an automation engineer, in case certain functionalities are demanded but not yet provided.

As the presented concepts were only tested on a small laboratory process plant with limited functionality, an implementation on a larger Process Cell could lead to further findings. If such a Process Cell would offer several possible recipes, more results could be gathered concerning the evaluation of the control system's flexibility.

Epilogue

In respect of numerous available publications, the author hopes to have added further usable knowledge to the vast existing amount concerning automation in general and Process Automation in particular. Overall, new knowledge may be obtained not solely by starting from scratch, but also by combining various existing approaches, which is especially true nowadays. Therefore the author would like to finalize this thesis with the following quotation stated by Robert Burton in the 17th century:

> "Though there were many giants of old in physic and philosophy, yet I say with Didacus Stella, 'A dwarf standing on the shoulders of a giant may see farther than a giant himself;' I may likely add, alter, and see farther than my predecessors [...]." [Bur07]

A Appendix

The Appendix presents the recipe headers of the recipes described in Section 3.4.1 as follows:

- header of the General Recipe: Table A.1
- header of the Master Recipe: Table A.2
- header of the Control Recipe: Table A.3

General Recipe Header	
Identification	**Warm distilled water**
Purpose	**Production of warm distilled water for further use**
Version	**1.0 / Author: WL**
Reference Value for Scale-up	**1000 g**
Input Materials	**Quantity [g]**
Distilled water	1000.0
Products	**Quantity [g]**
Warm distilled water	1000.0

Table A.1: Header of the General Recipe – based on [NE 03]

Master Recipe Header	
Identification	**Warm distilled water**
Purpose	**Production of warm distilled water for further use**
Version	**1.0 / Reviewer: WL**
Variant	**1**
Status	**Released**
First Issue	**01.07.2007 / Author: WL**
Run-Time	**5 min for standard amount**
Reference Value for Scale-up	**1000 g**
Input Materials	**Quantity [g]**
Distilled water	1000.0
Products	**Quantity [g]**
Warm distilled water	1000.0

Table A.2: Header of the Master Recipe – based on [NE 03]

Control Recipe Header	
Batch Identification	**1**
Production Order	**ABCDE**
Master Recipe	**Warm distilled water**
Purpose	**Production of warm distilled water for further use**
Version	**1.0 / Reviewer: WL**
Variant	**1**
Status	**Released**
First Issue	**01.07.2007 / Author: WL**
Ordered Amount	**500 g**
Scheduled Start-Time	**02.07.2007, 10:00**
Run-Time	**2.5 min**
Allocated Units	**Reaction: Reactor tank B101**
Input Materials	**Quantity [g]**
Distilled water	500.0
Products	**Quantity [g]**
Warm distilled water	500.0

Table A.3: Header of the Control Recipe – based on [NE 03]

Bibliography

[Bec07] Beck. http://www.beck-ipc.com. 2007.

[Bur07] Robert Burton. *The Anatomy of Melancholy.* Published 1621 under the pseudonym: Democritus Junior. http://www.gutenberg.org/etext/10800, 2007.

[Dea91] G. Dean. "Meeting the demands of batch control". In *Proceedings of International Conference on Control (Control'91)*, volume 2, pages 881–886, Edinburgh, UK, 1991.

[Dut03] Christoph Dutzler. *A Modular Distributed Control Architecture for Industrial Automation Systems: Designed for Holonic Systems and Vertical Integration.* PhD thesis, Technical University of Vienna, Vienna, 2003.

[Els07] Elsist. http://www.elsist.net. 2007.

[FB04] Bernard Favre-Bulle. *Automatisierung komplexer Industrieprozesse, Systeme, Verfahren und Informationsmanagement.* SpringerWien-NewYork, Vienna, Austria, 2004.

[FB05] Bernard Favre-Bulle. *Zukunft der Forschung in den Produktionswissenschaften.* Technical report, Technical University of Vienna, 2005.

[FBZD06] Bernard Favre-Bulle, Alois Zoitl, and Christoph Dutzler. "Zukunftstrends der Automatisierungstechnik und die Bedeutung von adaptiven Produktionssystemen". *Fachzeitschrift für Elektrotechnik und Informationstechnik*, 5:172–177, 2006.

[FES07] FESTO. http://www.festo.com. 2007.

[Hel04] Jürgen Helmich. *PCS Compactunit: User Manual.* FESTO, Esslingen, Germany, 2004.

[HOL07] HOLOBLOC, Inc. FBDK – The Function Block Development Kit. http://www.holobloc.com/doc/fbdk/index.htm, 2007.

[IEC97] IEC 61512-1. *Batch control – Part 1: Models and terminology.* International Electrical Commission, Geneva, 1997. Public Available Specification.

[IEC05] IEC 61499-1. *Function blocks – Part 1: Architecture.* International Electrical Commission, Geneva, 2005. Public Available Specification.

[IEC06a] IEC 61804-1. *Function blocks (FB) for process control – Part 1: Overview of system aspects.* International Electrical Commission, Geneva, 2006. Public Available Specification.

[IEC06b] IEC 61804-2. *Function blocks (FB) for process control – Part 2: Specification of FB concept.* International Electrical Commission, Geneva, 2006. Public Available Specification.

[Ing06] Ernest Ingebrigtsen. *Upgrading Control Technology in the Lifecycle of Industrial Process Plants.* Master's thesis, Technical University of Vienna, Vienna, 2006.

[Kui99] S. Kuikka. *A batch process management framework: Domain-specific, design pattern and software component based approach.* PhD thesis, Helsinki University of Technology, Espoo, 1999.

[Lew01] Robert Lewis. *Modelling control systems using IEC 61499 – Applying function blocks to distributed systems.* The Institution of Electrical Engineers, London, UK, 2001.

[NA 03] NA 46. *Examples of Recipe-Based Operations to NAMUR Recommendation NE 33.* NAMUR-Arbeitskreis 2.3, Leverkusen, Germany, 2003.

[NE 03] NE 33. *Requirements to be Met by Systems for Recipe-Based Operations.* NAMUR-Arbeitskreis 2.3, Leverkusen, Germany, 2003.

[PCSK07] Jukka Peltola, James Christensen, Seppo Sierla, and Kari Koskinen. "A Migration Path to IEC 61499 for the Batch Process Industry". In *Proceedings of 5th IEEE International Conference on Industrial Informatics (INDIN2007)*, Vienna, Austria, 2007.

[Rai06] Michael Rainbauer. *Einheitliche Architektur für intelligente mechatronische Komponenten im steuerungstechnischen Baukastensystem.* Master's thesis, Technical University of Vienna, Vienna, 2006.

[Sem07] Dallas Semiconductor. http://www.maxim-ic.com. 2007.

[SSPK06] Mika Strömman, Seppo Sierla, Jukka Peltola, and Kari Koskinen. "Professional designers' adaptations of IEC 61499 to their individual work practices". In *IEEE Conference on Emerging Technologies and Factory Automation (ETFA'06)*, pages 743–749, Prague, Czech Republic, 2006.

[Str02] Günther Strohrmann. *Automatisierung verfahrenstechnischer Prozesse.* Oldenbourg, München, Germany, 2002.

[SV94] Masoud Soroush and Sairam Valluri. "Meeting the demands of batch control". In *American Control Conference*, volume 1, pages 490–494, Baltimore, Maryland, USA, 1994.

[SZD06] Christoph Sünder, Alois Zoitl, and Christoph Dutzler. "Functional structure-based modelling of automation systems". *International Journal Manufacturing Research*, 1:405–420, 2006.

[SZRFB06] Christoph Sünder, Alois Zoitl, Michael Rainbauer, and Bernard Favre-Bulle. "Hierarchical Control Modeling Architecture for Modular Distributed Automation Systems". In *Proceedings of 4th IEEE International Conference on Industrial Informatics (INDIN2006)*, pages 12–17, Singapore, 2006.

[SZSFB05] Christoph Sünder, Alois Zoitl, Thomas Strasser, and Bernard Favre-Bulle. "Intuitive Control Engineering for Mechatronic Components in Distributed Automation Systems based on the reference model of IEC 61499". In *Proceedings of 3rd IEEE International Conference on Industrial Informatics (INDIN2005)*, pages 50–55, Perth, Australia, 2005.

[UB95] Reiner Uhlig and Michael Bruns. *Automatisierung von Chargenprozessen.* Oldenbourg, München, Germany, 1995.

[Vya03] V. Vyatkin. "Intelligent Mechatronic Components: Control System Engineering using an Open Distributed Architecture". In *Proceedings of IEEE Conference on Emerging Technologies and Factory Automation (ETFA'03)*, volume 2, pages 277–284, Lisbon, Portugal, 2003.

[ZF00] Gerfried Zeichen and Karl Fürst. *Automatisierte Industrieprozesse.* Springer, Vienna, Austria, 2000.

[Zoi02] Alois Zoitl. *Development of an IEC 61499 based embedded control platform and integration in a distributed automation system.* Master's thesis, Technical University of Vienna, Vienna, 2002.

[Zoi07] Alois Zoitl. *Basic Real-Time Reconfiguration Services for Zero Down-Time.* PhD thesis, Technical University of Vienna, Vienna, 2007.