Proceedings of the
44th IEEE Conference on Decision and Control, and
the European Control Conference 2005
Seville, Spain, December 12-15, 2005

Application Integration the Other Way Around

Dipl.-Ing. Christina Haus, Dr.-Ing Harald Albrecht,
Lehrstuhl für Prozessleittechnik, RWTH Aachen University

***Abstract*—Complex interfaces for communication and accessing online process data are two typical plagues when integrating advanced applications into process industry plants. Examples of such advanced applications are process validation, online optimisation, advanced controllers, et cetera. Today's typical procedure is to either integrate the additional functionality directly into the particular automation system or to add it as an additional external system above the automation system level. The first procedure is often not feasible due to limited resources and restricted integration capabilitites of industrial automation systems. On the other hand, external solutions impose additional communication load and error proneness on automation systems. Also, the integration effort required increases: communication failures, such as missing setpoints, need to be handled properly within the automation system in order to avoid loosing closed loop control and to fall back to automation system-internal strategies. These barriers hinder widespread usage of advanced applications. Thus, a different integration approach provides additional application functionality and data access directly *on* the field level. Fieldbus technology is an established technology providing standardised data access. In this paper, we present a flexible solution for easily realising virtual field devices hosting diverse applications (for instance, using common PC hardware).**

I. Introduction

DEPLOYMENT of advanced applications in process industry plants today is not as widespread as it could be preferable. Two reasons are: first, industrial users are afraid of destabilizing their processes if either an advanced controller (running in closed loop operation) or communication fails. Second, application developers face large developing efforts just for closing the loop between process, automation system and their advanced controller, or for just getting process data into their applications.

Yet users in process industry are interested in optimizing their processes and plants and thus using data for supervision and process management issues. Although application integration has been discussed for quite some years, the current situation still forces developers to deal with communication issues and data acquisition nearly everytime and even before developing the actual application.

Since such advanced applications usually need to be tightly integrated with the process (especially if running in closed loop operation) also tight integration with process control systems is required. Tight system integration usually means highly specific knowledge about the particular automation systems deployed in a particular plant – and the integration possibly not easily transferable to other plants with different systems. Also, often it is not possible to integrate the desired functionalities directly into the process control system at all due to resource restrictions (storage, processing power, et cetera). A cost effective way for realizing such additional functionalities thus can be using common PC technology. In addition, fieldbus technology with a wide spectrum of available interface hardware offers access to information using convenient hardware and flexible programmability.

In the first part we analyse different strategies for implementing advanced applications in process industries, looking at both common as well as new strategies. In the second part, we describe our application device. Besides the implementation platform the solution for communication issues as well as internal data structure are described. In the third part we report experiences we gained with the developed application device so far and give a lookout to future work basing on the current development.

II. Ways of integration

The integration of advanced applications can be realised in different ways, which can be classified in different views [1]. For our further consideration a system-focused approach is chosen.

A. Superordinate to the Process Control System

So far, the often preferred way for realising advanced applications, such as optimisation or advanced control strategies, is to use an external system hosting the advanced application, which is superordinate to a process control system and usually connected to some plant bus (see also Figure 1). In consequence, developers need to tackle vertical integration issues.

Manuscript received March 7, 2005.

Dipl.-Ing. Christina Haus is with the chair of process control at RWTH Aachen, Germany (phone: +49-241-8097709; fax: +49-241-8092238; e-mail: christina.haus@ plt.rwth-aachen.de).

Dr.-Ing. Harald Albrecht, is with the Chair of process control at Aachen University, Aachen, Germany. (e-mail: harald.albrecht@plt.rwth-aachen.de).

0-7803-9568-9/05/$20.00 ©2005 IEEE

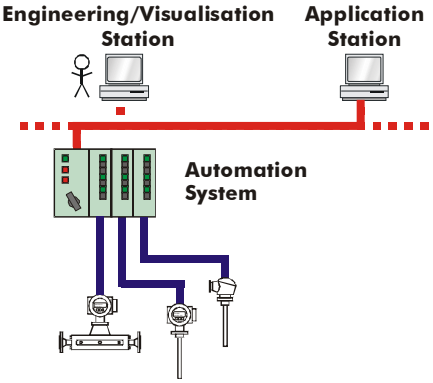

Fig. 1. Application realisation using a superordinate system.

One point when realising applications in that way is to loose direct process contact. Every online data has to go to the process control system, is handled there and only with a certain delay available for external applications. At least one time cycle passes before the information is provided at the interface. This can cause problems for time (or cycle) critical applications, such as contol methods. The same problem arises a second time when writing data back to the control system.

In addition, the external realisation of advanced applications in a closed loop can cause problems in view of communication failures. Communication failures can arise both by interfaces, the communication medium, or inside the advanced application and need to be treated properly. For instance, safety strategies are required inside the control system in order to be certain to detect such failures. A fallback strategy can be to use substitute values or switching back to conventional control strategies. However, this requires changes in the already existing and trusted control programs, as well as additional processing resources in the process control systems.

Another problem is the wide variety of available communication interfaces. These can be devided into two categories: the first category consists of manufacturer or system-specific interfaces. Several process control system manufacturers provide their own application programmer interfaces (API) on different levels of the control system. Few offer data access directly at the controller or field level. Frequently, APIs are provided at the operator station level. Sometimes, system-specific programming languages for direct integration with a particular process control system are available. However, this tightly ties application to particular systems.

The second category contains de-facto communication solutions [2], such as OPC or @aGlance/IT with standardised services. Nevertheless, data exchange and the process database has to be designed and setup individually for each application and target system, caused by special variable names and address space. Normally, a set of data access points has to be predefined and a communication server then provides access to them. Changing your informational bookkeeping usually triggers reprogramming the data exchange. In addition, most of the standardised communication interfaces for vertical integration make use of proprietary communication means, such as Microsoft DCOM, which tie them to a single operating system platform. Another problems is that usually no checks for successful transmission and receiving exist, so applications need to explicitly carry out additional checks. Also, in order to deal with such inconsistent data, the same fallback strategies are required as mentioned in the case of vendor-specific APIs.

B. Direct Integration into Process Control Systems

As described above for closed loops a safety layer has to be realised inside the control system. For that reason it can be taken into consideration to realise the complete application directly and tightly integrated in the control system (see also Figure 2). Two main disadvantages of superordinate integration case can then be avoided. Communication failures between separate systems lapse and thus no longer need to be considered. Second, because of direct data access there is usually no cycle problem (cycle desynchronization) anymore.

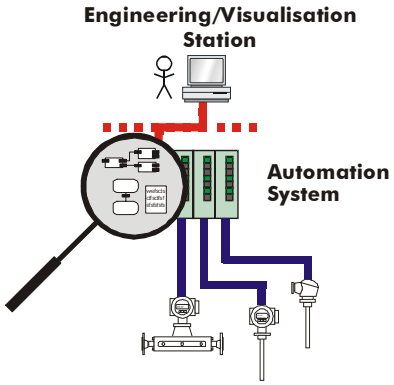

Fig. 2. Application realisation inside automation systems.

Instead of dealing with communication and database management, application developers now have to deal with manufacturer and system-specific programming enviroments. Some control systems provide programming enviroments with the ability to freely create new function blocks, while others only provide a fixed function block library.

Beside limited programming functionality, process control systems and especially programmable controllers are typically built with limited memory and processor capacity in order to reduce costs. Of course, for standard applications there is no need to provide more resources. In consequence, even if free programming is possible, when developing advanced applications the limited resources cause problems and hinder deployment of advanced applications.

Another source of problems comes in when users need to

interact with integrated applications. All handling has then to be realised either directly on the operator station (with possibly rather restricted functionality) or using an additional station external to the process control system. With an external system, the only advantage over the first scenario then would be that the important communication between automation system and advanced application could be still kept inside the process control system. However, this integration has also be backside that every change in the application requires reprogramming and therefore stopping the controller, downloading the application, followed by a new startup phase. However, such a procedure is clearly not desired and not practicable in process industry plants.

C. *Field Integration*

The highest grade of integration for applications in view of communicating process values effectively is achieved by integrating the application at the field level. The first question to tackle in this scenario is what platform(s) could be used. This kind of integration is already availbale today, albeit only in a vendor and device-specific kind: some device manufacturers have developed special applications. For instance, some actors are able to perform optimisation, diagnosis or control functions. Some sensors can perform validation methods or smoothening functions. However, such functionalities are device-specifc and only accessible through manufacturer-specific tools. It is thus not possible to freely combine functionality from different devices in order to perform higher functionalities. These applications do not support the user in realising her/his own advanced applications covering whole plant sections. Also, realising user-specific advanced applications inside field devices is not supported either and due to resource restrictions usually not possible and desirable.

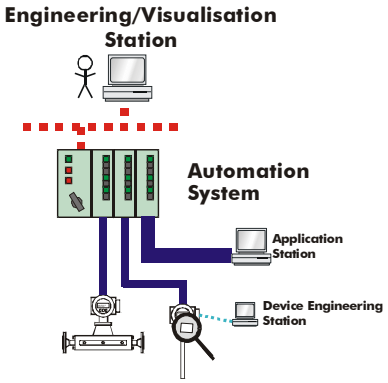

Fig. 3. Integration at the field level.

Integration of advanced applications at the field level is promising nevertheless, since it uses well-known and mastered communication techniques. In addition, support by the process control systems has already matured. Today, two classes of field devices are available. First, the conventional 4-20 mA wiring is still widespread. The disadvantage of this scheme is that only either only analogous signal transmission is available or slow HART-based communication needs to be used. In case of HART, the availability of HART modems that can operate in device mode is a problem.

However, using fieldbus technology these access problems in the field level can be easily avoided. A wide spectrum of interface hardware is available today for connecting standard PC-based systems to fieldbusses and make them act as field devices. Of course, integration in this scenario has still to cope with the various data formats, services, et cetera, which are unfortunately not the same for all the different fieldbusses. At least, the services are standardised per fieldbus.

Eventually, fieldbus technology combined with the assigment of PC technology has the potential to offer a flexible and cost effective way to get information out of or into the field layer for further use. Not only in research the use of PC technology is a cost efficient and flexible way for realizing a variety of advanced functions, ranging from process control to asset management or advanced control and loop optimizing.

Over the years, a varity of different fieldbus protocols and systems have been developed and are now in industrial use. Thus, we have to deal with a non-uniform installation base. Several standardization activities from different points of views [3]-[5] have been carried out in the past in order to deal with this situation. Every standardization activity was forced by a user organisation and so only a subset of fieldbus systems and special applications were considered. In consequence, integration of advanced applications on the field layer is challenged by the installed inhomgenous system and fieldbus specification base.

But even when focussing on one fieldbus protocol, application development has to deal with communication issues. With their primary goal of developping market-relevant applications, developers are slowed down by having to deal with recurrent problems stemming from linking fieldbus systems using manufacturer-specific hardware interfaces. Standardized services are usually not available and sometimes only implictly encapsulated in hardware-specific service models. Moreover, these service models then only realise those use cases that a particular manufacturer considered important. In consequence, the actual application code and communication code end up in a non-maintainable code hotchpotch. At the same time, this surely hinders porting applications to new platforms and plants.

From the several studies performed in the past one only dealt with pure neutralization of fieldbus protocols [6]. Others tried to create a uniform or interoperable representation of device data [7, 8] or tackled device engineering [9]. Mapping of bus specifics, integration of hardware drivers, and

representation of real devices is so far left up to users or application developers. None of the studies we are aware of tackles handling online process data and device emulation. Until now, both aspects were considered only seperately and independently – but if one wants to build an easy-to-use system, both aspects have to be taken into consideration. It is important to standardize the interface between these two aspects.

Basically, both a fieldbus-independent communicaton channel is required as well as a neutral data representation and management. "Neutral" in this case refers to device and independence. This allows the user to present data in a generic way almost without restrictions and loss of information. A suitable object oriented approach naturally supports the internal self description of the application device.

III. The Application Device

Our application device approach makes use of standardised and proven fieldbus communication technology. Application device behave the same as the other "usual" field devices (sensors, actuators) in the network and use the same standardised data access methods. Therefore, application integration into a process control system is carried out in the same way as it is done in case of a sensor. Additionally, it is possible to use so-called "user parameters" in the same way as with standard field devices for setting up parameters that do not need to be transmitted cyclically.

Therefore we reach communication safety for the device integration itself by acting as a standard device in the fieldbus network. Communication failures are detected by standard mechanisms and the automation system is able to react in the same manner it reacts to sensor and actuator communication failures.

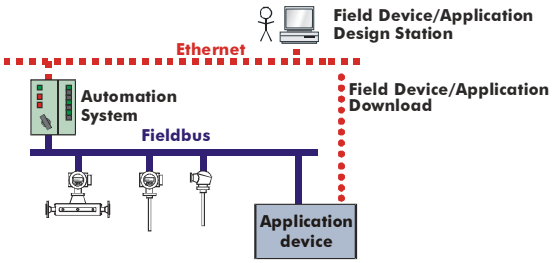

Fig. 4. Proposes approach for application device realisation and integration.

This architecture offers the possibility to integrate advanced control functions directly at the field level. This is a suitable way for implementing advanced controllers as near the process as possible without changing the control system or dealing with manufacturer-specific languages.

A second application field using our proposed platform are softsensors for generating information from other values that can not be measured direcly or to validate measured values by model prediction.

A. Implementation platform

Since operating system and hardware independence are two important aspects of our proposed platform, we rely on a portable open source object-management system called ACPLT/OV as our object-oriented application environment. ACPLT/OV [10] has been developed at our research institute in cooperation with industrial users from process industries. It is deployed, for instance, in petrochemical applications and already handles several hundreds of thousends of objects within single applications. The object management system offers a persistent object database with complete access and object management services over the network. In addition, it ensures that the object networks are always kept consistent according to the particular application model(s) – the application programmers do not need to write code for such tasks but instead the object management takes care of such issues. In our application device platform, the object management system is used (amongst other things) for loading device profiles and new applications even during full operation.

B. Fieldbus access

In order to cope with the complexities in fieldbus communication, we divide this area into two layers and thus separate interface hardware issues from fieldbus system issues. For the design, we don't go for the lowest common denominator but instead abstract from the specialised protocols to the generic communication models behind them.

First of all, we have to deal with the communication side. For real fieldbus independent communication we need to identify and analyze the communication models used in different fieldbus systems. Next, we need to find an abstracted model of communication strategies covering the specialized models. In consequence, not only the shared minimum functionality is realized. Abstraction from the specialised protocols to the communication models used provides maximum functionality by minimum specialisation.

1) Hardware abstraction

Today, a wide range of fieldbus communication hardware is available. Unfortunately, each manufacturer provides his own specific interface to grab information from the bus. In addition to a variety of protocols this causes a lot of trouble when using them. Standardized services are not accessible for every hardware but implicitly encapsulated in hardware-specific service models. These service models only realize the use cases that a particular manufacturer considered necessary. Porting applications thus becomes cumbersome or sometimes

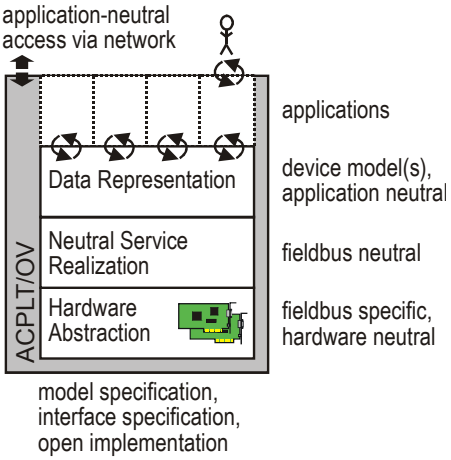

Fig. 5. Model architecture.

impossible due to budget constraints.

In the application platform architecture (see also Figure 5) we thus allocate a layer just to the integration of special hardware properties. This "hardware abstraction layer" undertakes the task of providing bus-specific functions in a hardware-independent way on its upper interface. The advantage of our approach here is that we not only describe the interface but also provide a framework with code for interfacing to different hardware in a structured way. The developer only has to parametrise but not to implement this framework. The integration of new communication hardware is structured and no longer requires a complete new development. At the top of this layer we have eventually decoupled fieldbus-specific services (which are standardized in the particular fieldbus specifications) from interface hardware.

2) Neutral service realisation

The "neutral service realisation layer" performs the task to provide services independent of particular fieldbus systems. It turned out that while the essential fieldbus services show implementaion specifics core functionalities are mostly identical. Therefore an abstraction is made from single bus-specific services to underlying communication concepts in order to build a system that provides more functionality than a reduction to a common denominator could. The aim is to provide full functionality without knowledge of bus particularities.

Every neutral service uses its own state machine which uses the hardware-neutral but bus-specific services of the underlying hardware abstraction layer. While the state machines differ for different fieldbusses they provide bus-independent services at their upper interfaces. Thus, hardware and data access over the fieldbus are decoupled. Services on this layer are completely bus-independent to upper-layer users, such as the device representation.

C. Device Self Description

After neutral communication is ensured on the field layer it is necessary to provide a structured device description on top. This is needed at two places when deploying the application device with a process control system. For application development it is advantageous to be able choose from a library of standardised data blocks, further called modules, to build up the data model of the application device. If we consider the integration of the device into fieldbus networks and the attached automation systems a description for engineering the network is also required.

For instance, the information source for the device type description for process data exchange can be a so-called "GSD" (device basic data file) in case of the PROFIBUS fieldbus. In the application device platform, the device type description is stored as an object model providing the required information for further use. It is important to note that the type description reflects all possible configurations and parameter types but not a particular configuration with its parameters. In this context, we subsume both the description of engineering parameters and online parameters under the term parameter. Both kind of parameters can be described in the same way. Application developers do not need to program new types but only need to parametrise the basic description using the existing types. Three classes of objects, device type, module type and parameter type, are sufficient to describe complex modular devices.

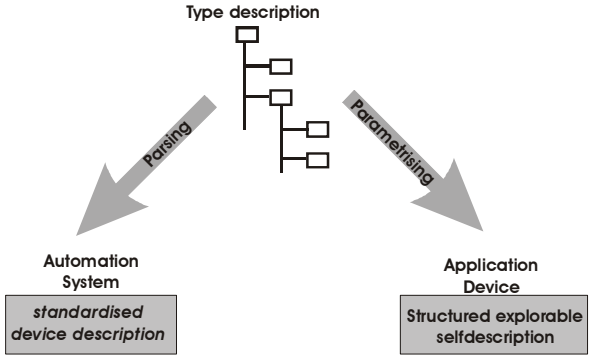

Fig. 6. Application device realisation and integration.

For Profibus devices, the GSD includes type information for devices and the module types they can contain. Unfortunately, the GSD description is only sufficient for user parameters, but not for online data / parameters. Only the overall byte length of all input and output data is declared in the GSD. In contrast, our object-oriented device model can also describe process

input and output data and the mapping of individual parameters to the overall input, output and configuration data strings. Within the model, describing input, output and configuration parameters is the same, only a property of each parameter description indicates the use for either input, output, or configuration. Users thus do not need to care about fieldbus-specific configuration issues. The application device platform takes care of these issues.

The device type description provides the required information about possible configuration items for:

- the application device itself, and
- the automatic generation of a fieldbus-specific device description.

Thus, we get a complete description of a modular field device that is suitable for both application development and engineering of the automation system (for device integration).

Beside describing device types, that is, describing all possible configurations, it is also necessary to provide a unified way for describing particular configurations and actual parameter values. For this, we developed a device instance representation that provides applications with access to parameters and process data. In combination with the associated device type description, the engineered device can then pack and unpack configuration data, as well as process input and output data according to the particular fieldbus system.

To easy deployment, our application platform comes with predefined device descriptions that are suitable for communicating certain typical data types, such as floating point values representing signals. All an application developer has to do now in order to set up fieldbus communication is to instantiate the corresponding type objects and to parameterize them in order to describe the input, output and configuration parameters for her/his particular application.

D. Application Integration

The advanced application itself can finally be integrated using function block technology. For the object management system ACPLT/OV we use a function block system that provides standardised function blocks and can be easily extended through new function blocks. Such user-specific function blocks can be loaded at runtime, so flexibility is guaranteed. Both the object management and function block systems are already in use in industrial application for several years.

IV. COMPLETED AND FURTHER WORK

Currently, the following applications have been realised using the application device: two applications have been realised in cooperation with a control system vendor. The first project resulted in a softsensor using Profibus DP for connection to the industrial process control system. It was demonstrated at Hannover Fair in 2004. The second project is currently work in progress. Here, the application platform is used by another framework for integrating advanced control [11]. In consequence, the device modules offer more advanced functionalities and use complex data structures. While development in the framework project is still going on we could already prove that our application platform is a suitable basis for connecting more easy to fieldbus systems and automation systems.

Both applications themselves were developed in MatLab Simulink and later transferred into a function block library using a standardised exporter. On the application device, it is then sufficient to load the resulting library at runtime and to connect the function block connectors of the applications with the device representation objects. On the automation system side, the application is then handled in the same way as a sensor or actuator.

Other applications to be used with the application device are, for instance, simulation and online optimisation tasks.

Besides usage as an application device, the architecture presented in this paper can also be used to support other communication roles of different devices attached to fieldbus networks [12]. Apart from implementing field devices, the architecture can also be used to provide automation system functionalities for small and pilot plants. In addition, it can also act as a “buddy implementation” in parallel to an already existing automation system. However, these other communication roles require additional services not covered in this paper, for instance, for scanning fieldbus networks for devices and for detecting the structure of devices on the basis of standardised profiles.

V. CONCLUSION

In our paper, we presented a flexible and cost effective implementation to integrate advanced applications directly on the field level. In consequence, the engineering process for integrating such advanced applications on the control system level now follows the same mechanisms and rules already established for standard field devices. Communication issues and description of the application device are based on open industrial standards.

The layered architecture in combination with a concrete implementation reduces the coding effort required for adding new fieldbus interface hardware considerably and avoids reinvention of private fieldbus service mappings and manufacturer-specific functions. The device description can be easily explored and changed at runtime.

Finally, application developers benefit from a hardware and

operating system-neutral platform with structured representation of the information contained in fieldbus instrumentation installations. Because the object-oriented data representation layer can be fully explored at runtime, applications can adapt automatically to changing installations.

The structured and device-independent representation of data gives the ability to reuse applications within different process control systems and plants. Transferring an application can now be done using limited configuration and parameterizing instead of reprogramming. Developed only once, libraries can later be loaded into multiple application devices no matter what fieldbus hardware interface or fieldbus system is deployed. Standard network interfaces provide additional data and engineering access over TCP/IP-based networks (even web-based). Application developers thus can focus on their applications instead of on communication issues.

The application platform allows to think about new use cases for PC based fieldbus devices. The applications presented show conceivable scenarios that offer new aspects in deploying fieldbus systems. New functionalities can result in increasing interest from reasearch and industrial users.

REFERENCES

[1] Münnemann and U. Enste: "Systemtechnische Integration gehobener Regelungsverfahren." *atp - Automatisierungstechnische Praxis 43,* Vol. 7, pp. 40-48, 2001

[2] Enste, U., Uecker, F.: „Standardisierte Kommunikationssysteme zur vertikalen Integration der Betriebs- und Prozessleitebene - eine Übersicht." *atp- Automatisierungstechnische Praxis 44*, Vol. 2, pp. 63-73, 2002

[3] G.H. Gürtler, "Fieldbus standardization, the european approach and experience", In D. Dietrich, H. Schweinzer ,*Feldbustechnik in Forschung, Entwicklung und Anwendung Proceedings of the Fieldbus Conference FeT'97*, Wien, Östereich, Springer Verlag, Wien New York, pp. 2-11, 1997

[4] DIN, "DIN EN 50170/2: Universelles Feldbuskommunikationssystem", Band 2/3 PROFIBUS, Beuth Verlag, Berlin, Germany, 1997

[5] IEC, "IEC 61158-1/-2/-3...6 Fieldbus for use in industrial control systems, Part 1: Introductory Guide, 1997, Part 2: Physical Layer specification and service definition, 1993, Part 3…6: Digital data communications for measurement and control, 2000"

[6] Esprit Proect 20468., " *RACKS: Uniform fieldbus interface specification version 1.4*", 1998

[7] C. Diedrich, R. Simon, "Integrierte Automation", atp – Automatisierungstechnische Praxis 43, Vol.10, pp. 26-32, 2001

[8] R. Simon, Gerätemodell zur Feldinstrumentierung von verteilten Automatisierungssystemen", *atp – Automatisierungstechnische Praxis* 50, Vol. 10, pp. 490-499

[9] C. Diedrich, Fieldbus profile harmonisation – Approach of NOAH ESPRIT 26951. FET'99, 23./24.09.19999, Magdeburg*, Proc. Of FET'99"* pp. 423-428, ISBN 3-211-83394-3, Springer Verlag Wien New York.

[10] D. Meyer, "Innovation of Software Structures in Automation Systems." *Conference Proceedings of the ISA EXPO 2000*, New Orleans. ISA, 2000.

[11] R. Jorewitz R., U. Epple: *"Zustandssteuerwerk für das Reglerrahmenkonzept eines Mehrfahrweisen-Bausteins"* Vortrag auf dem 39. Regelungstechnischen Kolloquium, Boppard, 2005.

[12] Ch. Haus, H. Albrecht: *"Neutral Data Access to Fieldbusses in Process Industries"*. In: Proceedings of the WFCS 2004 5th IEEE International Workshop on Factory Communication Systems, IEEE Catalog 04TH8777, pp. 369- 372.