Development and Evaluation of a Model for Modular Automation in Plant Manufacturing

Uwe KATZKE
Katja FISCHER

and

Birgit VOGEL-HEUSER
Automation and Process Control Engineering
Faculty of Electrical, Information, and Media Engineering, University of Wuppertal
Wuppertal, 42119, Germany

ABSTRACT

The benefit of modular concepts in plant automation is seen ambivalent. On one hand it offers advantages, on the other hand it also sets requirements on the system structure as well as discipline of designer. The main reasons to use modularity in systems design for automation applications in industry are reusability and reduction of complexity, but up to now modular concepts are rare in plant automation. This paper analyses the reasons and proposes measures and solution concepts. An analysis of the work flow and the working results of some companies in several branches show different proposals of modularity. These different proposals in production and process engineering are integrated in one model and represent different perspectives of an integrated system.

Keywords: Modularity, Reuse, Automation.

1. INTRODUCTION AND TASK ANALYSIS

In machine and plant automation for production and process engineering huge application software and hardware often is developed with much more than 3000 input/output points (process variables), which represent sensors and actuators.
The software development in process automation is still dominated by the languages for programmable logic controllers (PLCs) [1, 2] standardized by the International Electrotechnical Commission (IEC) [3]. The IEC 61131 languages are function-oriented. The designed applications consist of pieces of code, which are part of the IEC engineering concept. Usually plants are programmed in plant automation industry with IEC languages. Such a plant is unique and during design reusability is not in the first place. Nevertheless many times these implemented modules are used as draft for the next unique plant. As a result modules are often very complex. These complex modules are not subdivided into granular pieces. Consequently these modules are difficult to manage and maintain. The request to simplify complexity of plants exists not only because of maintenance problems, but this is one reason why systematic approaches and modeling are necessary.
Several projects are working on reusable modular concepts and will be summarized in the following as a basis for the introduction of this analysis. For example the idea of the project Föderal [4] is the integration of engineering disciplines (mechanical, electrical, etc.) in one modular reusable approach. The goal should be a construction kit oriented engineering process for machine tools. The organization of their so-called mechatronic components have to occur in standard libraries of the construction kit and in project-specific libraries. The basic requirements of the underlying Federal Information Architecture (FIA), which have to be considered, are among others a modular and open structure. The kernel of this initiative consists, inter alia, of three engineering companies, which are organized in the German Engineering Federation (VDMA).
MoWiMa [5] and Mova [6], two similar approaches, stand for modeling and reuse of object-oriented machine software closely connected to mechanical engineering concepts. The goal was to provide pre-conditions in order to increase the degree of reusability of control software in machine and plant manufacturing, to reduce the development costs and time, as well as to raise the software quality and flexibility. Mova was successful only with a specific development environment (ECAE system, PLC and human machine interface - HMI) and wasn't applied from any company, outside the project. Both approaches couldn't bridge the gap from university to industrial use, because there is a lack of practicability and an economic development perspective.
Another phenomena increase the problem. The Unified Modeling Language (UML) [7] as modeling standard for business processes is not accepted in this domain up to now, maybe because of missing notation constructs for modeling characteristic process automation aspects [8]. By that the appropriate modeling language for the description and documentation of such modules is not available yet.
Further approaches are following. Schnieder et al. [9] analyzed several modeling techniques and their suitability for different process characteristics. Friedrich et al. [10] worked on a comparison of modeling techniques for process control engineering. Biermann et al. [11] analyzed UML and Idiomatic Control Language (ICL) regarding decentralized systems. The results of these approaches show the lack of an appropriate accepted modeling technique for the design of plant automation integrating hardware and software as well as architectural aspects.
However aspects as for example reusability and modularity are aimed but not achieved up to now in plant automation industry. Nested structures and encapsulation are not considered in industry at all until now. These aspects are well known and adopted in computer science. Indeed computer scientists deal with the same kind of computer and profit by the

with the same kind of computers and profit by the homogeneous structure of these systems. General structures can be designed easier for uniform systems, but in process automation industry – especially in machine and plant manufacturing – the developer deals with heterogeneous systems.
Nevertheless, the solution appears to exist and it seems it only has to be mapped. But the industrial success is still missing. A first hint for these problems is, that reusability of modules failed in most cases because the following aspects have been slightly or not at all considered in industry.

- Analysis and structuring of plant requirements respective modularity of machine and plant automation
- Documentation of available already tested modules
- Acceptance of developer to use modules out of a library, which is not self-provided
- Standardization and maintenance of module libraries
- Development of concepts of new modules respective adaption concepts of modules in case of changes

To find and verify the industries aloofness of design alternatives, an analysis on the base of interviews is accomplished. In all listed approaches the development of a modular model is especially interesting in view to distributed systems. The starting point of concepts as Interface for Distributed Automation (IDA) [12] or Component Based Automation (CBA) [13] is the assignment of hardware to a software module [14].
Other approaches try to map the results of conventional object-oriented software development to process automation systems [15, 16].
Beside these points there is no special procedure for change management, for example for the integration of an added sensor, an initiator or switch, nor is a concept available how this could be maintained easily. Additionally users want a free allocation of software modules to automation devices, to use modules flexible and cost-efficient.
In this paper the results of the analysis of production applications and process engineering are discussed. Special aspects are the concept and the application of module libraries. First a draft module model is defined. This module is proofed and extended by conducted interviews with companies of different application areas and by the inspection of already existing module libraries respective application soft- and hardware.

2. REQUIREMENTS

This evaluation considers various automation systems, for example programmable logic controllers (PLC), industrial personal computers with IEC 61131-3 runtime environment [1, 2], process control systems, and its engineering tools (initially of one manufacturer).
Thus the survey handles with heterogeneous distributed real time systems. From viewpoint of system integrators and plant operating enterprises the need of standards is evident to build and to use module libraries. These standards are either reached through the independence of different manufacturers or through an orientation which is related to the technology of one major automation system as a company standard.
The adaption of module libraries requires that development, maintenance and documentation of modules can be handled easy. The project engineer shall find the appropriate modules quickly. It has to be ensured that the module will operate properly.
Designing the right size of modules and the change management (version management) belongs to the main challenges. It is just difficult to identify the right module, if numerous variants are present in the library. But an extreme restrictive behavior is not the solution. Using only few, small modules, which have to be rearranged for any new task doesn't generate all the expected benefit of modularity.
The principles of modularity are not restricted to the process of systems' design. They have to be regarded from the beginning of the project in the sales and project planning departments and keep their relevance beyond operation and maintenance. In fact a concept for reengineering has to be integrated.
The given task was to examine the concepts of different industrial users, to find similarities and differences and to develop and evaluate a common model as a conclusion.
In respect of these requirements a first basic model is built and evaluated with seven different users. These users represent sales, basic engineering, detail engineering, implementation, operating and maintenance in diverse ways. Their branches belong to production and process engineering.

3. MODULE MODEL

The model for modules is based on three levels. The class of basic modules builds the first level, the second level clusters application modules and the third level covers the so-called project or facility modules (Figure 1). The three levels differ strictly in method and grade of modules' reuse. Comparing the three levels, the occurrence of transparency is contrary to the ability to reuse modules.
Basic modules are encapsulated as black boxes. The behavior of these modules can be configured through parameters. These two aspects are special characteristics of basic modules. These application independent modules map elementary functions.
In contrast to basic modules application modules are application dependent, because of different requirements of industrial applications. They could be integrated in different applications as a result of module variation. Application modules are designed by variation of standardized drafts. Sometimes it is necessary for this reason to know the structure of modules to create new versions. Application modules can be composed of basic modules. They are also configured through parameters.
The task of project modules is to decrease complexity of plants as a result of arrangement in smaller manageable parts. Normally this draft is designed top-down until the plant is composed in application modules. Usually a plant is unique. For this reason there is less reusability of project modules but these modules are transparent.
Despite its level a module is composed of automation hardware, PLCs (functionality with operating methods and diagnosis functions), HMIs, interfaces to super ordinate systems (ERP) as well as documentation and calculation.

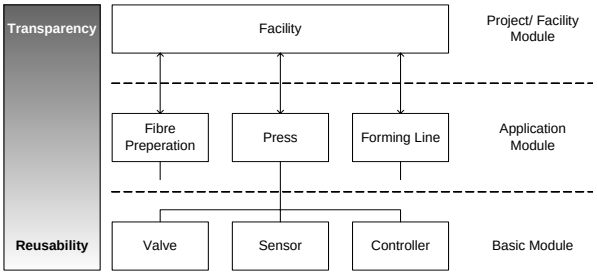

Figure 1 Three-level-module-model with an application example of a manufacturing plant of timber industry

On every level the module concept is recursive (Figure 2). Basic modules may be composed of basic modules as well as application modules may contain application modules. Basic modules, which are not composed of other basic modules, are named atomic modules. These modules are elementary and it is not reasonable to split them into smaller parts.

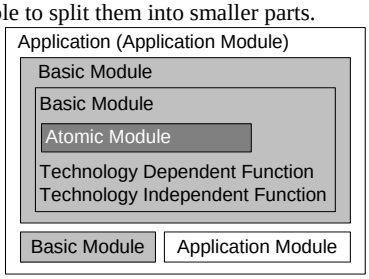

Figure 2 Recursive module structure (e.g. with application and basic modules)

4. EVALUATION

All users validated the three level model for modules. It is remarkable, that the number of basic modules is very low (less then 20), even if sophisticated and complex plants are regarded. In many cases variation and combination of modules is the preferred way of using modules. Application modules are apparently a systematic approach to close the gap between reuse and redesign of modules. They are used as patterns, which are – according to the used strategy - either restricted or supplemented.

As an interesting option application modules are divided into two classes. A so called functional module targets the primary purpose of applications. All its integral parts are designed to provide assistance for it. For example a conveyer unit consists of belts, rolls and mechanical drives as functional modules. In addition to these functional modules non operating tasks are handled by add-on-modules. Continuing this example a scanner would be an addendum to the conveyor unit. A scanner covers functions like detection and recognition, but it has nothing to do with transportation. In this context a scanner would be regarded as an add-on.

The integration of automation hardware is handled differently in module concepts of the interviewed companies. Inputs and outputs, respectively process instrumentation are represented in the module by all users, but only one user realized electrical hardware (contacts, motors) as an aspect for module design.

During the second phase of the evaluation system integrators and end customers of automation systems have been involved. The scope of these interviews was to discuss the benefit of modules developed by other companies (suppliers). The benefit of those modules is seen very critical, because communication to the module designer is disturbed. Interface definitions are regarded as very important and they are suggested as insufficient communicated. The exact meaning of such an interface may be exposed lately parallel to the progress of implementation. Misunderstandings in the interpretation of interface descriptions are regarded as a human manner. The system integrators nearly deny the opportunity of creating definite, clear descriptions. They claim, that different engineers will understand modules differently. Some developers prefer a simple step by step guidance to implement such a module and others think a detailed comprehension of a module including all its relations to the system is necessary to benefit from a module library.

One proposal to handle these interfaces is suggested by the OMAC for packaging machines. The OMAC (Open Modular Architecture Controls) [17] has proposed an approach for representing and standardizing modules in a comprehensible way. The concept reduces components to generic actions with clearly defined states. Such modules are defined as a standard, supplying defined interfaces and behavior. State machines are used to represent them definitely.

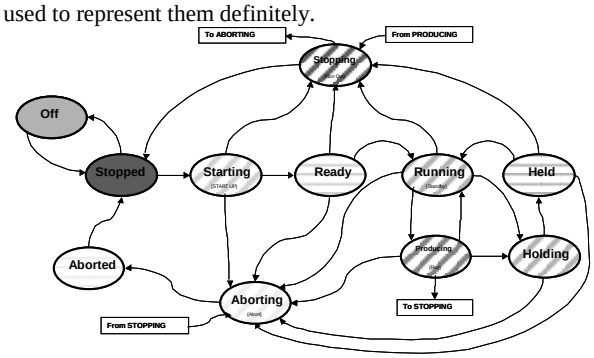

Figure 3 Automata for representation of the standardized behavior of a packaging machine by OMAC [18]

Such a description is used for modeling a packaging machine (Figure 3). Some functions could not be standardized in this way, because of lack of frequency or acuity detail. Anyway they could be described in this form.

The complexity of an interface corresponds to the capability of its module. This is one result of the interviews. Consequently it has to be deliberated up to which grade of complexity modules can be transferred to a library. The range reaches from simple basic modules up to complex modules with an all-inclusive functionality. The level of application modules offers patterns, which are qualified to be stored in libraries. So these modules suggest an agreement between the reuse of unchanged modules and the frequent design of new, individual modules. Economic aspects are the basis for the decision, for which level each module is developed. The decision respects how frequently the use of a module is expected and what effort is required to construct it. Definitive rules for this decision are rare.

Some deduced characteristics for modules are described in the following.

- **Number of basic modules**
  The average number of application independent basic modules is approximately 15. Most of them are atomic modules. Results of full reusability of modules are available, unless the functionality of these modules is not very complex. Most of the interviewed partners avoid the development of such modules. There is a lack of concepts, which compensate the development costs with reusability to be economically successful. Nevertheless one company developed these modules and realized a module library.
- **Module dependency on branches**
  Basic modules are built branch neutral. Application modules depend on the branch.
- **Granularity and complexity**
  Modules are often very complex. In industry complex modules are not be subdivided into a granular structure. Consequently these modules are difficult to manage and maintain. Nevertheless the request to simplify complexity of plants exists.

- **Configuration of modules through parameters**
  Modules should be configured through parameters independent of the module's level. In contrast to other module types basic modules cannot be adapted by other means. Parameters are just complementary to other module modifying techniques of application and project modules.
- **Interlocking**
  Some special interlocking modules exist. They are cascadable in case of very complex interlocking tasks. Often it is not seen in industry, that interlocking modules are necessary. Instead not modular solutions and auxiliary constructs are used.
- **Distributed intelligence**
  In industry it is preferred to built central automation systems. Concepts for distribution of software to hardware (mapping) hardly exist in plant automation. Only in case of encapsulated devices distribution is required.
- **Combined hardware and software modules**
  The modular approach, which bundles different requirements and techniques, is only applied on software.
- **Building of variations**
  There are different mechanisms for building variations. One approach is that standardized application modules are used as drafts for variations. Another way to build variations is using an adequate application and modifying it. Variations can be built through reduction or extension. In case of reduction the starting point is a module with maximum functionality and it will be reduced to application modules. In case of extension the starting point is a standardized module. The variation is built by adding furthermore functionality.
- **Supplier dependent implementation**
  All modules are dependent on engaged hardware with the exception of basic modules.
- **Interfaces to other systems**
  For interfaces to other systems established standards are applied. For example XML (Extension Markup Language) is used for module description, OPC [19] is used as connection to distributed control systems (DCS), and faceplates are used for the integration of individual modules to a visualization system.
- **Version management**
  There are different solutions in industry. Some companies make only a simple marking. Others have developed business processes to handle access and application rights.

From this characteristics the following criteria, which are listed in a table (Table 1), are affiliated. On the basis of these criteria the requirements of a module can be allocated concerning to the different levels of the presented module model.

5. SUMMARY AND PROSPECTS

The introduced module model is approved by typical users of plant automation engineering in Germany. The benefit of company specific module libraries is accepted as well. The benefit of general module libraries could not be evaluated because of the necessary but missing distinct documentation of modules and interface handling. The module design in diverse branches may be different. Several aspects, which are listed below, have to be considered

- which structure (relations)
- which interaction (mechanisms for building interfaces and their protocols)
- which variation (inheritance, polymorphism)
- which degree of reusability versus rapidity of design

is required. According to these aspects supporting measures will be used to create a prototypically design. The three-level-module-model offers an approach to arrange modules adequate. A further analysis to appoint the requirements of these aspects is necessary.

During the interviews it could be recognized that innovations cause acceptance problems. Developers rejected foreign modules, interfaces and new notations. Novelties can implicate fear and uncertainty [20]. Though these fears cannot be abolished completely, however it has to be considered. As a result small steps are required to introduce progressive techniques.

The aspects of hardware have been hardly integrated. In the same way the qualities of distributed intelligence have been dominated by current customer requests and the cost advantage of central PLCs. The lack of concepts for distributed software destroys the strong relation of hard- and software. Therefore a free mapping of software to corresponding hardware is required. The growing propagation of components, which are targeted by technologies like PROFInet [21] and component based automation (CBA), strengthen the assignment of hard- and software and consequently distribution is reinforced, too.

The question remains how such modules should be modeled. In the beginning we discussed several approaches. Therefore other modeling concepts are essential. Maybe the answer is an adapted UML. This contains a customization of UML for the requirements of process automation [22].

Beside the expression possibility of a model it is necessary to realize the attraction to today's mode of operation. The software development in process automation is still dominated by the IEC 61131-3. One concept to bridge the gap between UML and function oriented design in industry is an extension of the IEC 61131-3 languages to object oriented aspects. Other concepts target the mapping of a reduced UML to IEC languages [15, 16].Single extensions of PLC-toolsets, which are also function-oriented, allow extended modeling features, but they are isolated and don't actually deliver concepts for reuse [23].

The same is necessary for the design of state charts, which are recommended by the OMAC. But the idea of prototypes with generic functions extends the notation. Another approach is the adoption of the ICL [10, 11]. In contrast to UML ICL offers special notation constructs for model- ling typical process automation aspects. But ICL has other disadvantages. The comparison of these two languages, ICL and UML, is also topic of current research projects. All these approaches target gaps in the contemporary mode of work during modeling, designing and implementing projects. Further work is necessary, since new gaps are enforced through the focused view of the specific projects. Future work will take their advantages and target towards an integration of different techniques.

Table 1 Module criteria

Criteria	*Basic Modules*	*Application Modules*	*Project/ Facility Modules*
Project dependency	Modules are a source of well-engineered, branch independent functions.	Dependent on applications standardized versions of modules are created. For recurring parts, which have a related form, these modules would be adapted. Elementary functions are available as predefined elements.	These modules have got less reusability. Supplemented with application modules they reflect the modeled application. The intention of project modules is to give an overview.
Hardware dependency	Module software is not constraint by hardware. An exception is interface software.	Applications are adapted to a plant and therefore dependent on hardware.	Project modules depends on the structure of a plant, they dependent on hardware.
Dependency on branches	Basic modules are developed branch neutral.	Application modules may depend on the branch.	Project modules depend on the branch.
Encapsulation / Black Box behavior	Basic modules are encapsulated.	Encapsulation of source code and variables eliminates building of variation.	Project modules are designed top-down as white box.
Possibilities of adaption	Basic modules are adapted to standardized interfaces by parameterization.	Application modules are adapted by building of variants and standardized interfaces.	Project modules are adapted by utilizing open standards as e.g. OPC or XML.
Interlocking techniques	Interlocking is realized with special basic modules, which are standardized and fixed in a library.	Interlocking is realized with usage of basic modules.	Interlocking is realized at the application level.
Recursive structure	Basic modules can be composed of basic modules.	Application modules are composed of basic modules. The composition of different application modules is problematically. The integrity of a module has to be guaranteed, when a part of the module is modified.	Composition is possible. There are nor restrictions because of reusability.
Change Management	Changes of modules could only be realized with special change management methods because of requirements. Modified modules would be standardized, too.	Missing basic modules would be requested. New templates need acceptance procedures.	Project modules are designed top-down. Every plant is unique. Therefore the design of every plant has to be created new.
Availability	Basic modules are available by a library.	Application modules are derived from templates of a library.	Project modules are designed top-down. Every plant is unique. Therefore the design of every plant has to be created new.
Software Protection (protection against misusage)	Independency on hardware and branch requires software protection.	The need of software protection depends on the dependency on plant and configuration complexity of a module.	Software protection is guaranteed by the close relationship between the project modules and the plant.
Version management	Version management documents status and usage of basic modules.	Version management documents status and usage of drafts.	Version management documents the status of the realized system.

6. ACKNOWLEDGEMENT

The work presented in this paper based on interviews with industrial partners. We thank all participants for their patience and cooperation during the interviews and for reviewing our results.

7. REFERENCES

[1] John, K.-H.; Tiegelkamp, M.: *IEC 61131-3. Programming industrial automation systems*. Springer-Verlag, Berlin, Heidelberg, Germany, 1995.

[2] Lewis, R. W.: *Programming industrial control systems using IEC 61131-3.* IEE, Herts, 1998.

[3] www.iec.ch, September 2003.

[4] www.foederal.org, September 2003.

[5] http://www.isw.uni-stuttgart.de/projekte/mowima/index_e.htm, September 2003.

[6] Diedrich, C.; Günter, F.; Hanitsch, O.; Pastoors, N.; Petig, M.: *MOVA. Modulare Offene Verteilte Funktionsblocksysteme für die Automatisierungstechnik*. VDMA Verlag GmbH. Frankfurt, 2001.

[7] www.omg.org; September 2003.

[8] Fischer, K.; Vogel-Heuser, B.: *UML for real-time applications in automation*. In: Automatisierungstechnische Praxis (atp) 44 (2002); Heft 10, Oldenbourg, München, Germany, 2002, S. 63-69.

[9] http://www.iva.ing.tu-bs.de/

[10] Friedrich, David; Vogel-Heuser, Birgit; Bristol, Edgar: *Evaluation of Modeling Notations for Basic Software Engineering in Process Control*. In: 29th Annual Conference of the IEEE Industrial Electronics Society (IECON 03) in Roanoke, Virginia, USA, November 2003.

[11] Biermann, C.; Vogel-Heuser, B.: *Requirements of a process control description language for distributed control systems (DCS) in process industry*. In: Proceedings of IECON'02, 28th Annual Conference of the IEEE Industrial Electronics Society, Seville, November 2002.

[12] European Commission, Interchange of Data between Administrations, Architecture Guidelines. Ausgabe 4.1, Brussels June 2002.

[13] Siemens, Component based Automation, Configuring Plants, *Working with SIMATIC iMap. Manual*, 2001.

[14] Arlt, V., Funktionales Engineering: *Ein ganzheitlicher Engineeringansatz für modulare Maschinenkonzepte*. In: Tagungsband SPS/IPC/ DRIVES 2002, Hüthig Verlag Heidelberg 2002.

[15] Zhang, Wei; Diedrich, Christian: *Comparison between FB-oriented and Object-oriented Designs in Control Application*. In: IEC TC65/WG6 meeting in Bamberg, 17-21. June 2002, http://www.holobloc.com/stds/iec/tc65wg6/meetings/bambrg02/

[16] Bonfe, Marcello; Fantuzzi, Cesare: Design and Verification of Industrial Logic Controllers with UML and Statecharts. In: Proceedings CCA 2003, IEEE, 2003.

[17] *Guidelines for Packaging Machinery Automation, OMAC Packaging Workgroup*. Version 2.0, 24.04.2002, http://www.omac.org/wgs/GMC/Deliverables/Guidelines V2.03.pdf.

[18] Andrew McDonald, Collaborative Standards Development for Packing Machinery. Unilever, ARC Forum 2002.

[19] http://www.opcfoundation.org, September 2003.

[20] Bullinger, Hans-Jörg: *Socio-Economic Aspects of Human-Computer Interaction.* Invited Lecture of the International Status Conference in Saarbrücken, Germany, 26.10.2001. In cooperation with the Fraunhofer Institute for Industrial Engineering (IAO) and Institute for Human Factors and Technology Management (IAT), University of Stuttgart, 2001.

[21] http://www.profibus.com/, September 2003.

[22] www.lfa.uni-wuppertal.de/DisPA, September 2003.

[23] Siemens AG: S7-HiGraph for S7-300/400, Manual. Edition 04/2003.