INTERNATIONAL DESIGN CONFERENCE - DESIGN 2012
Dubrovnik - Croatia, May 21 - 24, 2012.

MODULARITY, VARIANT AND VERSION MANAGEMENT IN PLANT AUTOMATION – FUTURE CHALLENGES AND STATE OF THE ART

S. Feldmann, J. Fuchs and B. Vogel-Heuser

Keywords: modularity, variant and version management, concurrent engineering

1. Introduction

The automation control software created during plant engineering, which is a multi-billion Euro per year branch of industry, is very large and modified many times during the plant's operation. Certain sub-assemblies of machinery – each with its own automation control software – appear many times within a plant and across plants, and plant-specific variants of such sub-assemblies are created during plant start-up or operation for resolving process-specific issues of one particular plant. Therefore, identical reuse with subsequent modification as well as modified reuse of automation control software is ubiquitous in this domain; hence, modularity plays an enormous role for successful engineering and maintenance of plant automation control software.

Designing automation systems in machine and plant automation includes different disciplines, i.e. mechanical design, electrical design and software design. As each plant is unique, yet configured from standard machines and automation components, the plant's software is customized and constructed to a large degree from reusable modules. However, during design, reusability is often neglected in industry. Due to reduced time to market as well as high cost pressure, companies in machine and plant manufacturing are trying to find appropriate design methods for designing automation systems, but up to now there is a lack of design theory in automation compared to design in mechanical engineering.

Although the theory of modular composition in plant manufacturing is understood and the advantages and necessities for modularization are accepted, there are significant challenges in applying these concepts in industry. Extensive interconnections between software modules and their physical representations show that modularity is structured differently in the disciplines and therefore, modularity structures must be identified and considered to solve the modularity task for its application in industry. Besides providing an appropriate module composition technique, engineers must be supported in managing modules regarding variability and versioning of modules not only in single disciplines but also across disciplines.

This paper provides a survey on the state of the art, industrial practise in plant automation design and the resulting challenges and requirements focussing especially on modularity as a prerequisite for reuse to start developing an integrated or coupled approach with well-established design methods from mechanical engineering [Lindemann et al. 2009]. Two detailed analyses will show the differences between modularity structures in mechanical engineering, electrical engineering and software engineering and the resulting problems and challenges that need to be solved to prepare a solution for module, variant and version management.

The paper is structured as follows: The main requirements and challenges in plant automation, e.g. module management regarding variants and versions and the desired support for variability and change management are shown in chapter 2 using a small application scenario. The current research

focus and state of the art concerning the problems to be solved in plant automation engineering is discussed in chapter 3. Using two detailed analyses in market-leading German companies, the current industrial practise and tool application is shown to deepen the understanding of the challenges and to derive requirements for further research work in chapter 4. Finally, the results of the analyses are discussed and future work in this field is outlined.

2. Application scenario for module management in plant automation

As plants and machines in plant automation are distributed locally and as changes lack in documentation and support, a lot of problems occur concerning module, variant and version management. In the following, a small application scenario is presented to illustrate the desired support, and thus the challenges for coping with variants and versions in plant management.

2.1 Exemplary module kit

In Figure 1, a module kit of a small exemplary module, i.e. a conveyor belt, is shown that consists of basic modules being parameterized according to their specific application. The module contains a mandatory basic module, which defines the possible configurations, e.g. the locations and constraints of optional basic modules and interfaces for optional basic modules that can be added with respect to the possible configurations. In the illustrated scenario, the conveyor belt contains a light barrier, a drive and an optional light barrier that can be substituted by a bar code scanner.

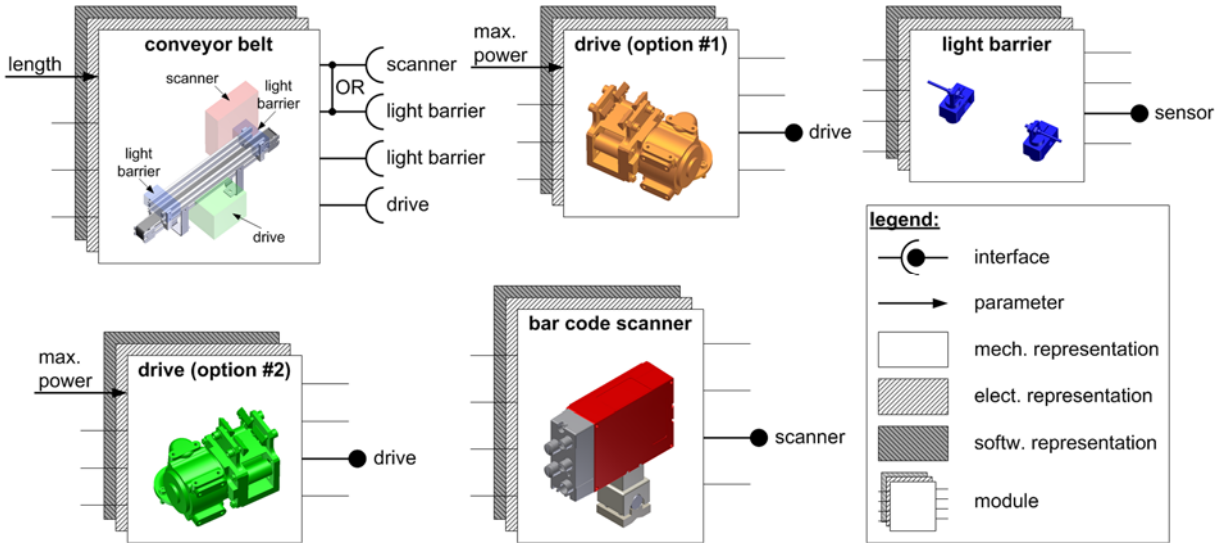

Figure 1. Module library for the conveyor belt containing the mandatory (*conveyor belt*) and optional (*drive, light barrier, bar code scanner*) basic modules with their discipline representations

Each module is represented through its discipline-specific documentations, i.e. mechanical, electrical and software representations that are subject to variant and version management. After having designed, verified and tested the conveyor belt, the mechatronic module can be used for its operation in new projects by configuring and parameterizing the mandatory and – if present – optional basic modules. In the following chapters, regarding the module's variant and version management, the conveyor belt module's software engineering representations are considered not taking the other discipline representations into account in terms of reducing the application example's complexity.

2.2 Variant and version management

As illustrated in Figure 2, three different variants ($var1_{MK}$, $var2_{MK}$ and $var3_{MK}$) have been configured from the module kit and used for different engineering projects at locations spread around the world. The figure shows the software representations of the module variants and their adjustment versions neglecting the mechanical and electrical representations as mentioned before. Having completed the

start-up phases, the current revision statuses are $var1_{Am}$, $var2_{An}$ and $var3_{Ap}$. During operation, an error in the acceleration ramp of $var3_{Ap}$ has been identified (step 1), hence, a new revision status $var3_{A(p+1)}$ has been implemented for the customer (step 2) to adjust the acceleration ramp.
Unfortunately, the programmer at the customer's location is not capable of identifying the problem's source and whether it has to be applied to existing variants as well. Therefore, the adjustment in the acceleration ramp has to be extrapolated (step 3) to eliminate the eventuality that the adjustments $var3_{A0}$ to $var3_{Ap}$ caused the problem. Afterwards, the adjustment $Ap \rightarrow A(p+1)$ is isolated and then applied to the related components in the module kit (step 4 and 5). To ensure that the acceleration ramps of the other variants are adjusted as well, the adjustment is performed on the other module variants in case the problem has not been solved during the versioning tasks $var1_{A0}$ to $var1_{Am}$ as well as $var2_{A0}$ to $var2_{An}$ (step 6).

2.3 Resulting challenges and current problems

It is desirable to trace changes in mechatronic systems to provide an integrated and correct module for each module developer, but up to now the different steps for the scenario are rarely applied in industry. As changes in plant automation occur during operation and maintenance at the customer's location, there is still a lack of change documentation in order to provide change management during late engineering phases. Furthermore, depending on the type of the system, i.e. mono-technology systems or mechatronic systems, the interconnection between a module's representations in other disciplines must be considered to be able to provide variability management; additional disciplines may add further constraints on the existing system, e.g. magnetic behaviour of the barcode scanner in Figure 2. These problems have not been solved yet.
Additionally, there are significant differences in module, variant and version management regarding engineering of mechatronic products and plant engineering. Compared to mechatronic products, plants are unique and manufactured only once; therefore, standardization potentialities are limited due to reduced time to market and high cost pressure; in consequence, appropriate modularization techniques must be developed to support an integrated module management.

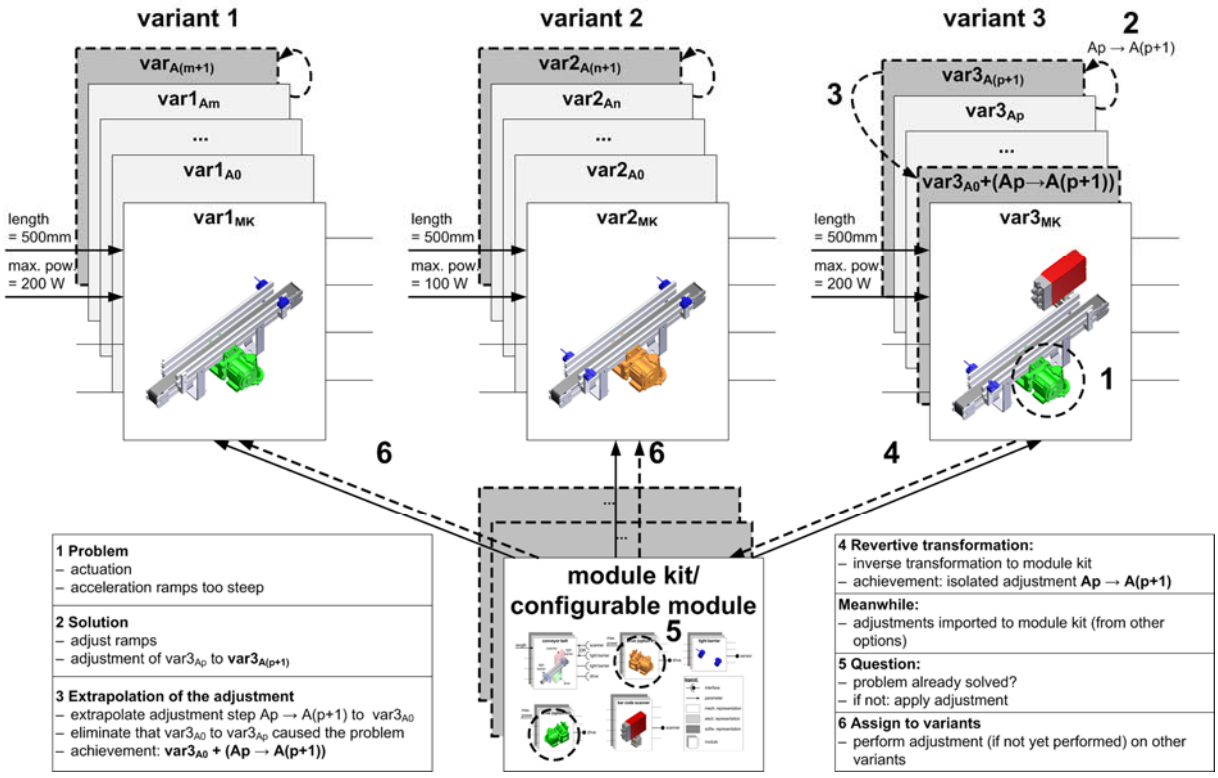

Figure 2. Variant management and tracing of changes for the conveyor belt

3. State of the art and resulting challenges in design of automation in plant manufacturing industry

Despite huge efforts spent on research initiatives in Germany, modularity is rarely fully implemented in automation control software for the machine and plant manufacturing industry in contrast to the sophisticated results in modularizing and analysing product architectures [Lindemann et al. 2009]. The importance of version and variant management was highlighted by a study with international experts [Vogel-Heuser 2009], who named modularity and reuse of variants as well as version and change management as most important criteria for automation engineering in 2020.

3.1 Plant engineering as a simultaneous process

Engineering in plant automation is a simultaneous process as engineers from different disciplines work at the same time on the same project. During engineering and planning processes, different documentation for discipline specific representations is generated, e.g. Computer Aided Design (CAD) drawings, electric circuit diagrams and software programs, that provide different views on the system and deliver different partially overlapping information to describe the whole plant.

The lifecycle of a plant in plant engineering lasts 20 – 30 years and starts with design and construction of the technical, i.e. mechanical, system. Subsequently, the electrical platform is designed and finally, the application of the plant is programed in the languages for Programmable Logic Controllers (PLC) standardized in IEC 61131-3. After the design and start-up phase is completed, the operational phase of the plant starts and in many cases, customer-specific adaptations need to be implemented locally.

Considering the simultaneous engineering process, many discipline-specific or interdisciplinary changes have to be done during a plant's design, operation and maintenance. Some adaptations during start-up and re-engineering processes can be applied locally by maintenance personnel in the customer's own factory, e.g. adjustment of software parameters; some are more complex making the support of the plant manufacturer necessary, e.g. replacing an old type of automation device (PLC) by a newer one. As the plant manufacturer often does not know the basic changes, new plant designs may contain the same deficiencies as the old design; therefore, techniques for modularization, variant and version management need to be provided.

New software tools are currently developed to support the interdisciplinary design and modularization of machines and plants, e.g. EPLAN Engineering Center (EEC). The application of these engineering tools often lacks in finding appropriate granularity levels for all disciplines participating in plant automation and is currently not state of the art in automation engineering [Maga et al. 2011].

3.2 Modularity, module size and granularity

A plant is unique in design and development and during design, reusability is scarcely considered. As a result, modules often represent whole plants or huge parts of the mechatronic system and are therefore very complex and difficult to manage and maintain.

To increase reusability and flexibility in plant automation, mechatronic systems must be subdivided into granular pieces and different representations of a system must be integrated into flexible, reusable modules. Mechatronic modules are separable units consisting of software and hardware, whereupon hardware includes both electrical and mechanical components. As described in [Katzke et al. 2004], an application module for the design of a production plant has well-defined interfaces and variation possibilities and is composed of different basic modules which can consist of a single piece of software, an electronic platform, mechanical context (mono-technology systems) or a combination of these components (mechatronic systems).

Maga et al. mention the level of granularity in automation systems as an aspect that deserves attention to avoid problems preventing reuse of modules [Maga et al. 2011]. According to the authors, conflicting interests concerning model granularity arise, namely fine-grained models enhance reusability and flexibility but implicate difficult search in specific projects, whereas coarse-grained models increase efficiency and robustness but hinder reusability because complex adaptations are necessary. Therefore, one challenge is to find the appropriate module sizes in between having many small components which make it difficult to identify all required parts and cumbersome to combine

the many parts of a solution, and few large components which constrain flexibility. To make things worse, what is small or large depends on the engineering phases suggesting that hierarchical component models may be needed [Jazdi et al. 2011]. Recently, more and more companies in machine and plant automation start analysing their module structure to find suitable module sizes and, at the same time, to cope with the problems of interlocking, error handling, and modes of operation.
Kortler et al. proposed a flexible modelling approach using matrices for mapping between different domains and to provide consistent change integration into the system [Kortler et al. 2011]. The applicability of the approach in machine and plant manufacturing must be evaluated in further steps.
These fundamental design problems are exacerbated by a lack of acceptance due to incomplete tool support or psycho-social problems, as a module used by an engineer might not be acknowledged because the module has been invented by another engineer or – even worse – by external partners and thus, might not be reliable. On a research level, UML has been evaluated as a technique for modularization of plant automation software [Secchi et al. 2007]; furthermore, a corresponding tool has been implemented in the IEC 61131-3 development environment [Vogel-Heuser et al. 2011].
Despite huge efforts spent on different research projects regarding modularity in machine and plant industry in Germany, modularity is rarely fully applied in automation software: ”Reusable artefacts are mostly fine-grained and have to be modified on different positions, so that errors are probable and important reuse potential is wasted” [Maga et al. 2011]. Reuse is mostly achieved through copy, paste and modification actions [Katzke et al. 2004].
Among the identifiable reasons for this situation are the following:

- the multitude of disciplines involved (such as software engineering, automation engineering, mechanical engineering, electrical engineering, safety engineering, perhaps also chemical engineering),
- the interdependencies of software modules with mechanical and electrical modules [Maga et al. 2011],
- the interconnectedness induced by exception handling and different modes of operation.

Maga et al. proposed the separation of Domain Engineering from Application Engineering, whereupon Domain Engineering provides suitable analysis methods, design methods, and reusable components to enhance reusability of discipline modules and Application Engineering employs these to construct the plant’s automation software more easily and specifically for customer-specific projects [Maga et al. 2011]. In order to construct Domain Engineering approaches with success, it is necessary to understand the current practise of modularity and reuse in automation software, the development processes and the interplay of the forces at work behind these processes in the automation domain.

3.3 Variant and version management

Managing variants and versions in automation engineering is an upcoming research topic. Lauder et al. consider the orthogonal problem dimensions *concurrent engineering disciplines*, *metamodelling, domain customization*, *abstraction* and *evolution* in plant automation (see Figure 3), but not in context of variant and version management [Lauder et al. 2010].
Lauder’s approach, namely Concurrent Model Driven Automation Engineering (CMDAE), uses these dimensions to identify reusable concepts and design elements addressing the integration of standard file formats and existing engineering tools instead of supporting an integrated modelling approach considering the required disciplines in a single model. The concept has been evaluated using an implementation of a bidirectional synchronisation between an ECAD tool and a PLC programming environment (SIMATIC STEP 7); however, depending on the level of detail and number of interfaces between tools in automation engineering, the application of CMDAE for plant automation can be very complex, e.g. due to continuous tool changes and updates.
Helms et al. present a modular modelling approach for the physical domain that uses the Systems Modelling Language (SysML) as a foundation containing a formal grounding based on bond graphs and dividing a model into three abstraction layers, i.e. *Function*, *Behaviour* and *Structure* [Helms et al. 2011]. In these approaches structural resources allocate physical effects as behaviours, which are assigned to for physical functions. A concept for defining interfaces between modules inside and in between these abstraction layers is introduced that uses the modelling element *port* of SysML.

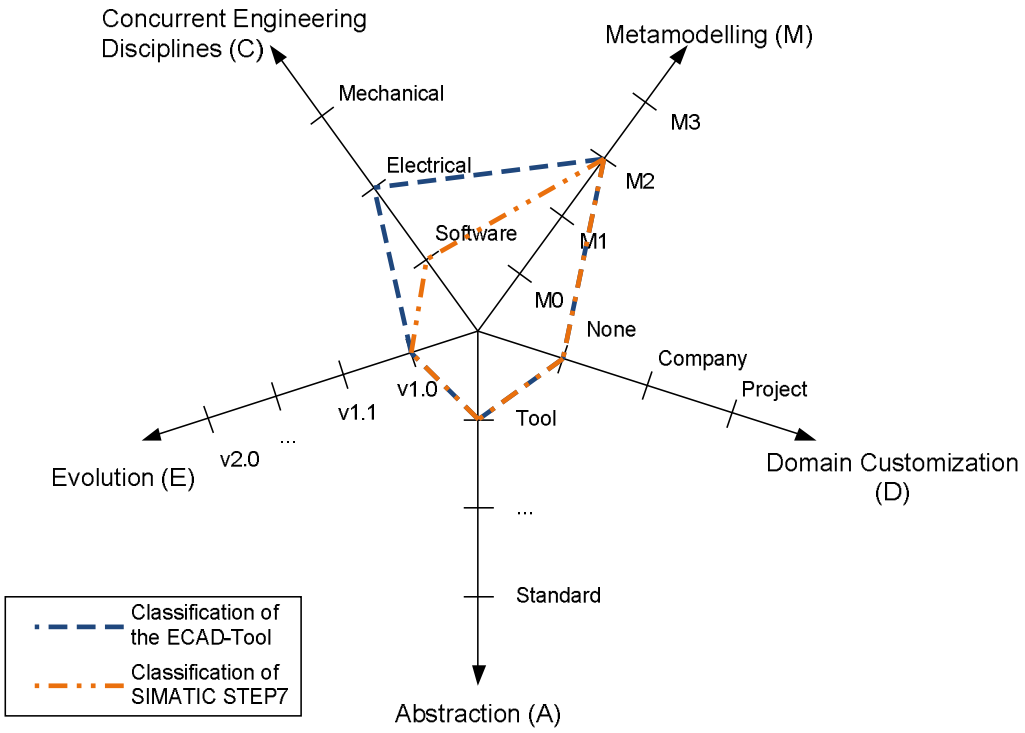

Figure 3. Dimensions of Concurrent Model Driven Automation Engineering (CMDAE) according to [Lauder et al. 2010]

4. Analyses of current industrial practise

To prepare a solution for providing module, variant and version management in automation engineering, the industrial aspects and the current industrial practise need to be considered for further work. Within two detailed analyses, the current practises in modularizing plant software implemented in IEC 61131-3 and in modularizing electric circuit diagrams have been evaluated with two different market-leading companies in the field of plant manufacturing.

4.1 Case study I – Modularity in PLC software design

Reuse of automation software in case study I is currently limited to the use of universal projects containing the maximum functionality of a machine or plant, i.e. the software is parameterized according to the customer requirements. Unnecessary software parts are disabled or negated leading to several problems, e.g. failure diagnosis, code comprehensibility, complexity and memory requirements.

The modularity structure chosen by the company has been quantified measuring the number of variables passed on between software units. Due to the sequential engineering process in plant automation, software modules are designed analogously to the module structure in mechanical engineering and the challenge in software design is to adapt the mechanical module structure to software modularity neglecting the requirements from a software point of view. The requirements on the mechanical part of a mechatronic system often differ from the software part, therefore, a function-oriented modularization of software is difficult to implement.

In this case study the interconnection between software units is currently very closely intertwined. Due to confidentiality agreements only abstract results may be published as shown for the interconnection between software modules in Figure 4. A circle in the illustration represents a function block; an arrow illustrates global data exchange. Software modules consisting of several function blocks are illustrated using different textures. The arrows' length is inversely proportional to the number of data exchanges whereas the diameter of the circles represents the complexity of the function blocks based on their pages of code. A long arrow indicates little interconnections between function blocks; a short one shows, that software units are connected strongly.

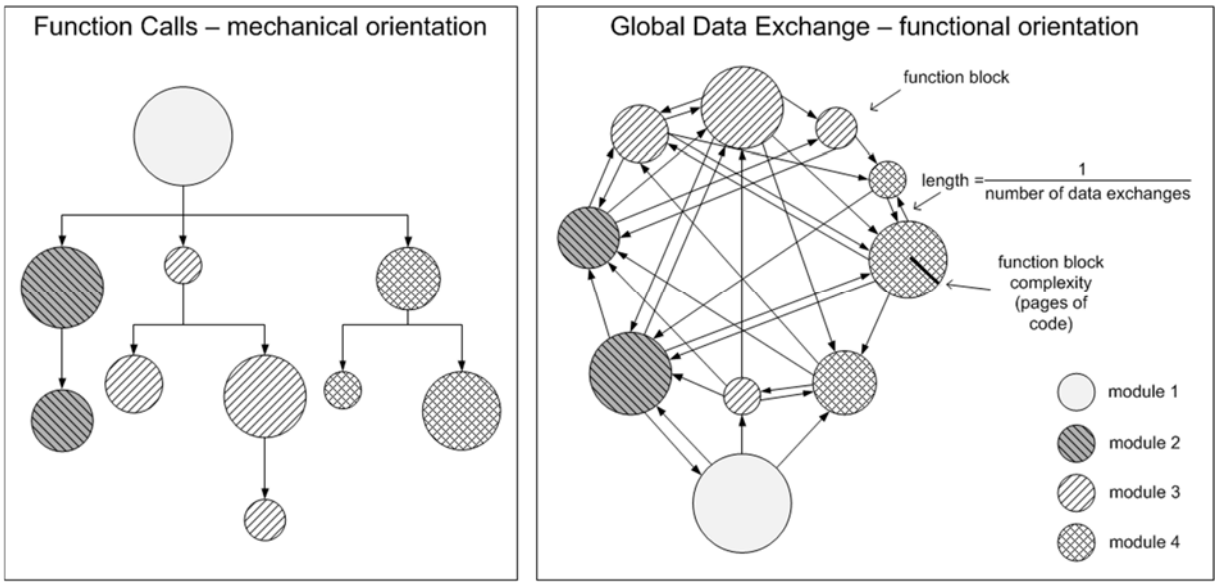

Figure 4. Schematic diagram of differences between mechanical-structured function calls and functional-structured data exchange

Currently, extensive communication, represented through short arrows, between different modules hampers modularization as software units have a huge number of connections to each other. The software is basically function-oriented according to the production process, despite having been modularized according to the mechanical modularization. Due to this a well-organized modularization is currently not really achievable; calls of software functions are structured according to the mechanical structure (see Figure 4, left), but in contradiction the exchange of global data is structured according to the functional structure (see Figure 4, right). Currently, both principles structuring the modularity exist in a single plant software, thus, the modules are connected via huge interfaces which make it hard to provide exchangeability and flexibility. For instance, if a single module is eliminated, large interfaces remain; thus, the effort to adapt the software is higher compared to applying a universal project as an alternative approach. Therefore, new approaches to provide consistent modularity structure in software design are required taking the functional character of plant automation software into account to consider the resulting function-oriented modularization criteria subsequently.

4.2 Case study II – Modularity in design of electrical engineering

In mechanical design, modules are defined according to their physically definable representation as devices, assemblies, etc.; in software design, modularity represents the functional composition of a plant. At first view, concerning modularity in electrical engineering for the design of local and central control cabinets and the corresponding circuit diagrams, there is a lack in such a simple modularity principle. For many years, modularity in electrical engineering implied reuse of circuit diagram pages, meaning that a reusable part should be on one page. These pages may be copied and then changed through eliminating or adding fragments, i.e. circuit diagram pieces, depending on the number of sensors or actuators of the devices, e.g. the necessary encoders and frequency converters. The second principle of modularity was the modularity of control cabinet arrangements, e.g. how many frequency converters may be mounted in one field of control cabinets, and the third principle was the modularity of input/ output (IO) modules for automation devices, i.e. the available granularity.

Case study II showed that more and more companies start analysing their module structure to find appropriate module sizes supporting modularity and reuse in plant engineering, especially in electrical engineering. In contrast to function-oriented module structures in software engineering, the chosen electrical module structure of the company is device-oriented. Devices must be connected to each other and are chosen by the customer; therefore, devices differ from project to project as different device manufacturers result in variability. As a result, electrical devices in different machines are not

identical in construction and functionality, hence, substituting or changing a device's functionality results necessarily in affecting the mechanical part of the system and, of course, the plant software.

4.3 Tool application

In the last few years, tools have been introduced providing an engineering framework for plant automation engineering. For instance, EPLAN Engineering Center (EEC) supports module and variant management independently from the application domain of the modelled mechatronic system and divides the engineering phases into tasks independently from projects and tasks according to precise projects. The tool integrates existing engineering tools, e.g. Electrical Computer Aided Design (ECAD) and PLC development systems and provides abstract modelling of complex mechatronic systems both interdisciplinary as well as for specific disciplines and, subsequently, the automatic generation of plant documentation.

For a deeper knowledge of already existing approaches and their quality regarding modularity the modularity and module library in the Computer Aided Engineering (CAE) system was analysed, i.e. EEC, which is one of the most advanced tools in this area [Maga et al. 2011], of a plant manufacturing company, which is one of the most advanced companies in building such modules. The following approaches for building mechatronic modules were found:

- universal project as foundation for an approach using copy, paste and modify by parameters,
- division of plants into plant units, plant areas etc. according to the structure of the technical process and the different machinery (see Figure 5).

The decomposition of the plant roughly represents the separation into *basic modules*, *application modules* and *plant modules* according to [Katzke et al. 2004], namely functions and devices are *basic modules,* whereas assemblies, machine units and machines represent *application modules*. Plant areas, plant units and plants belong to *plant modules*. The definition of basic modules is recursive, i.e. basic modules can contain basic modules.

The interdisciplinary decomposition of the mechatronic system is therefore similar to a mechanical decomposition. To provide a more flexible modelling technique and to avoid rigid structures, basic modules, i.e. functions and devices, can be used in plant modules, application modules and basic modules in the case study.

Further problems occur during the discipline-specific modelling of the system, for instance, an electrical module contains its representation through parameters, e.g. reference tags, on the one hand and its graphical representations, e.g. device symbols in an electric circuit diagram, on the other hand, and is therefore structured according to the electrical devices. Interfaces between electrical modules are defined through wirings that may contain physical information, e.g. diameter and length, in the field of the plant, but also precise electrical information, e.g. reference tags.

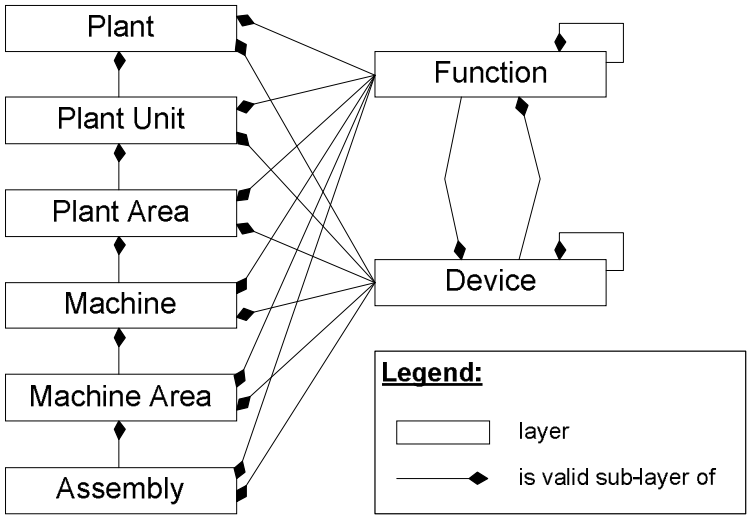

Figure 5. Layer configuration in electrical engineering framework of a plant (example) similar to the international standard ANSI/ISA-95

The engineering process is separated into discipline-specific, i.e. ECAD and PLC, and interdisciplinary, i.e. mechatronic, design. The engineer can implement the domain-specific layer configuration (see Figure 5) depending on the company's requirements. The libraries either represent the discipline modules, e.g. ECAD and PLC modules, or an additional discipline, i.e. mechatronic modules, defining the general composition of a plant (see Figure 6). Discipline-specific modules can be added to mechatronic modules and connected through parameters written in a specific language based on rules which can be filled with values by the engineer himself or using external data storages, e.g. information for circuit diagrams and information to disable or enable components. These values contain the discipline-specific information needed to generate the plant documentation, e.g. a parameter for an assembly's wire manufacturer definition can be imported from the list of parts in a database to automatically generate the reference tags for the wiring in the circuit diagrams. To instantiate and thereby use a mechatronic module and its discipline-specific representations, the engineer can drag and drop the module defined in the mechatronic library into the plant-specific project according to the layer configuration.
Module variants can be created applying two different methods:

- parameters that are filled according to the necessary configuration disable or negate specific modules to implement the concept of a universal project,
- interfaces that are used to provide instantiating necessary modules according to the module combination.

Constraints on the combination of modules or number of instances can be customized, e.g. to ensure that the conveyor belt in Figure 1 may contain a light barrier or a barcode scanner but not both at the same time.

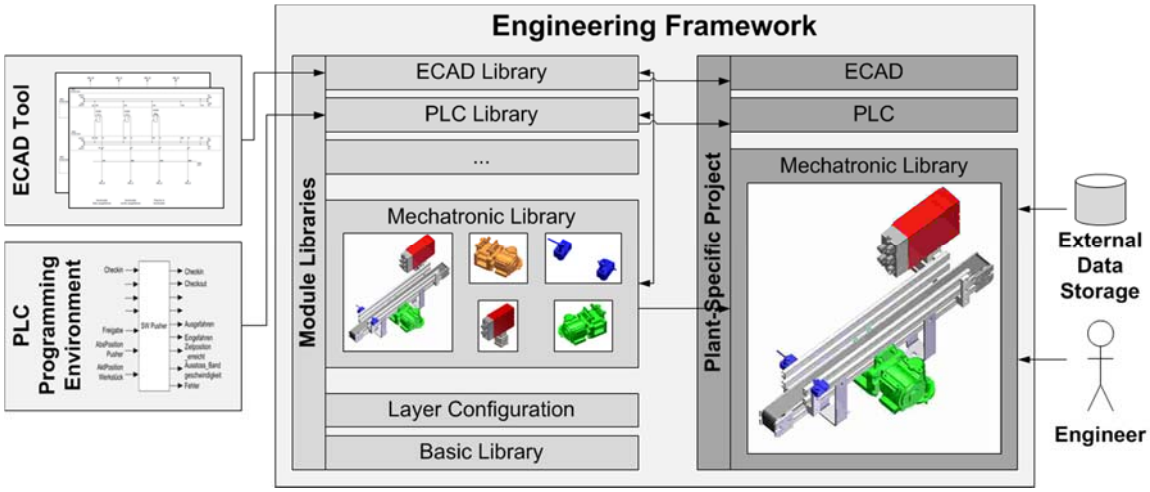

Figure 6. Exemplary information flow and data storage in an engineering framework

However, there are deficiencies concerning variability handling and experiences of engineers as defining and maintaining variants can be very complex and time-consuming when considering multiple disciplines [Maga et al. 2011]. Up to now there is a lack in version management, as tested module versions are managed using hard copies of databases. Different engineering phases from requirements' definition to the operation and maintenance phases are currently not supported.

5. Conclusion and outlook

In this paper a survey on the industrial practise in design of plants in plant manufacturing industry was given, focussing on the interdisciplinarity and the challenges in module, variant and version management. Using two detailed analyses in market-leading German plant manufacturing companies, one analysis in software engineering and one in electrical engineering, the challenges and problems to be solved were introduced.
The results show that module structures in different disciplines differ. The case studies demonstrate that modularity structures in software engineering are different from structures in electrical

engineering: Software modules are structured according to their functions whereas electrical modules are structured according to the electrical devices. Subsequently, according to the different modularity paradigms, the term *function* in software engineering refers to an action that can be called, whereas in electrical engineering a device's functionality is addressed. To solve the modularity, variant and version management tasks in automation engineering, the modularity structures of all disciplines, i.e. mechanical engineering, electrical engineering and software engineering, need to be analysed, understood and connected. Besides, challenges during runtime of industrial plants, e.g. dynamic reconfiguration and operation modes, must be considered.

Further research work will examine approaches from mechanical design methods as well as from product lines to support systematic modularization covering all disciplines and taking variant and version management into account to provide increased reuse during design. To provide an integrated module management, semantic interoperability between modules needs to be considered and formally described to support better discovering and reusing similar mechatronic modules. The system behaviour varying as a reason of the combination and configuration of different modules and disciplines is an aspect that needs to be considered to identify an automation system's functionality. Furthermore, change documentation and, thus, version management must be implemented to provide a better change management in a plant's engineering lifecycle.

References

Helms, B., Schultheiß, H., Shea, K., "Automated assignment of physical effects to functions using ports based on bond graphs", ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference (IDETC/CIE), ASME New York, USA, 2011.

Jazdi, N., Maga, C., Göhner, P., "Reusable models in industrial automation: experiences in defining appropriate levels of granularity", 18th World Congress of the International Federation of Automatic Control (IFAC), Elsevier Amsterdam, Netherlands, 2011, pp 9145–9150.

Katzke, U., Vogel-Heuser, B., Fischer, K., "Analysis and state of the art of modules in industrial automation", Automation Technology in Practice International (atpi), Vol.2, No.1, 2004, pp 23–31.

Kortler, S., Helms, B., Shea, K. and Lindemann, U., "A more flexible way of modeling structure with multiple domains", 13th Internationak Dependency and Structure Modelling Conference (DSM), Cambridge, USA, 2011, pp 19–29.

Lauder, M., Schlereth, M., Rose, S., Schürr, A., "Model-driven systems engineering: state-of-the-art and research challenges", Bulletin of the Polish Academy of Sciences: Technical Sciences, Vol.58, No.3, 2010, pp 409–421.

Lindemann, U., Maurer, M., Braun, T., "Structural complexity management: an approach for the field of product design", Springer-Verlag Berlin Heidelberg, Germany, 2009.

Maga, C., Jazdi, N., Göhner, P., "Requirements on engineering tools for increasing reuse in industrial automation", 16th IEEE Conference on Emerging Technologies and Factory Automation (ETFA), IEEE New York, USA, 2011, pp 1–7.

Secchi, C., Bonfè, M., Fantuzzi, C., Borsari, R., Borghi, D., "Object-oriented modeling of complex mechatronic components for the manufacturing industry", IEEE/ASME Transactions on Mechatronics, Vol.12, No.6, 2007, pp 696–702.

Vogel-Heuser, B., "Visions of automation engineering in 2020", Automation Technology in Practice (atp), Vol.2009, No.5, 2009, pp 49–56.

Vogel-Heuser, B., Braun, S., Kormann, B., Friedrich, D., "Implementation and evaluation of UML as modeling notation in object oriented software engineering for machine and plant automation", 18th World Congress of the International Federation of Automatic Control (IFAC), Elsevier Amsterdam, Netherlands, 2011, pp 9151–9157.

Stefan Feldmann
Institute of Automation and Information Systems, Technische Universität München
Boltzmannstr. 15, 85748 Garching near Munich, Germany
Telephone: +49 89 289-16441
Telefax: +49 89 289-16410
Email: feldmann@ais.mw.tum.de
URL: http://www.ais.mw.tum.de/