12TH INTERNATIONAL DEPENDENCY AND STRUCTURE MODELLING CONFERENCE, DSM'10
22 – 23 JULY 2010, CAMBRIDGE, UK

A FRAMEWORK FOR EVALUATING PRODUCT ARCHITECTURE OF AUTOMATION PRODUCTION FACILITIES

Maximilian P. Kissel[1], Katharina G. M. Eben[1], Steven Braun[2], Jakob Schmidt-Colinet[2], Martin Obermeier[2], Udo Lindemann[1] and Birgit Vogel-Heuser[2]

[1]Institute of Product Development, Technische Universität München
[2]Chair of Information Technology in Mechanical Engineering, Technische Universität München

Keywords: MDM, evaluation, automation systems, interdisciplinary product development

1 SILO MENTALITY IN AUTOMATION SYSTEMS DEVELOPMENT

Some interdisciplinary products are still designed with blinders. To develop automation systems, teams of each involved discipline can work on their particular subsystem (software, controller, electronic and mechanic hardware) independently by defining interfaces and requirements among the individual parts of the system. When it comes to a re-configuration of an existing subsystem, the impact on the other subsystems is ambiguous. Developers are not aware of the impact of their development decisions beyond the interfaces regarding modularization, changes in product architecture or re-shape of system borders. In this paper, we provide a framework to evaluate the impact of a re-modularization of a subsystem on the total system and give implications for further steps towards a holistic evaluation system for automation system architecture.

2 THE IMPACT OF MODULARIZATION OF SOFTWARE CODE

At the beginning of this paper, to illustrate the need for an evaluation framework, we want to present an example of a potential re-modularization scenario in the field of automation from an information technology perspective. An imminent scenario of a momentous change in one subsystem of an automation system is the introduction of object-oriented programming languages in automation industries. This change will have a noticeable impact on the IT itself and, furthermore, on the work of other disciplines involved. The application of object-oriented programming concepts to real-time automation systems is due to the fact that rising requirements in quality and security of automation production facilities cause an inflation of solicited lines of control code and therefore an increase of the complexity of the overall code. Programs with more than 200 thousand lines of code are common, most notably, to address all eventualities of error resolution. To cope with the rising complexity of the software of automation systems, a trend towards object-oriented software modularization is perceivable (Katzke, 2009; Katzke and Vogel-Heuser, 2009). Currently, the norm for controller programming (DIN EN 61131-3) is being complemented with object-oriented constructs.

Even if the idea of re-organizing the source code seems to be reasonable and smart in order to reduce the complexity of the software, the impact of this measure on hardware components and, consequently, on the total system remains yet neglected. The replacement of procedure-oriented software with object-oriented software may cause a new organisation of the IT infrastructure itself, which is supposed to have even further reciprocal effects on other independently developed subsystems. To give an example: there is no need any longer to implement all software functionality on one central real-time capable controller, because the code is no longer one big procedure, but rather a hierarchical, intricate and – possibly – even physically distributed set of code elements. New concepts for controller architecture can be developed and optimised in terms of new degrees of freedom. Re-modularization of other subsystems may become reasonable as well. Parts of the code can be located in the field on isolated micro-controllers to economise the wire harnesses to the sensors and actuators. But such a change may cause problems, if, e.g., information stored on this particular micro-controller may be needed for a surveillance routine running on a controller, which is – by mistake – not physically connected with it. Hence, there is a need for making the consequences of a change in a subsystem transparent in order to communicate it to other disciplines at the interfaces.

Another aspect to be considered in terms of evaluating the modularization of an automation system is the differing tailoring of modules by each involved discipline. As depicted in Figure 1, the modularization of – de facto – the same machine can vary in terms of the particular points of view. In this example, the relatively simple machine consists of five modules in the mechanical discipline, whereas in the software discipline of three modules. Differences result from the way of analyzing systems and creating modules. The implemented software functions may cover parts of the system functionality in a different manner as mechanical functions would be mapped to components in order to tailor hardware modules. The change of the programming paradigm, described above, might have different effects on every discipline-specific modularisation. Also these consequences have to be made transparent and communicated on a total system level.

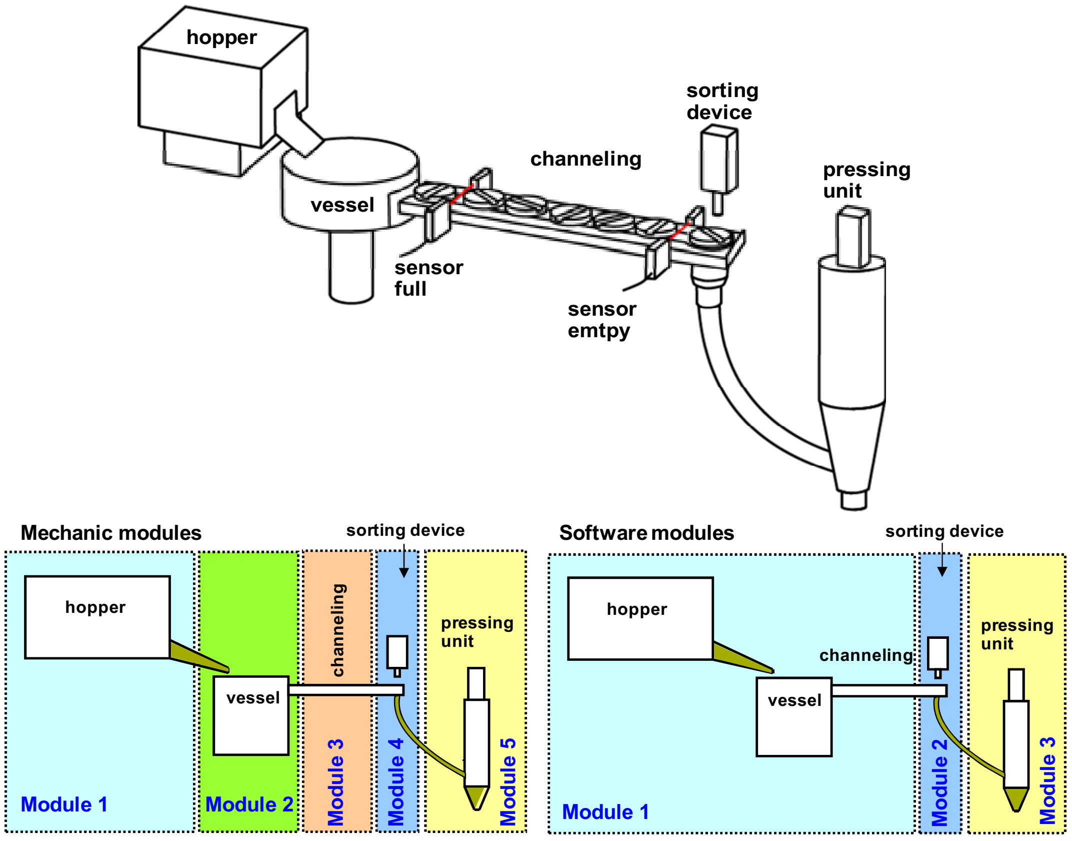


Figure 1. Exemplary modularization of the same machine in two different disciplines

While Waldman and Sangal (2009) suggest a methodology to shape and evaluate software modules in terms of software development, several other scholars are concerned with methodologies to approach modularization and evaluation of the architecture of partial aspects in each discipline involved (Ulrich, 1995; Pimmler and Eppinger 1994).

There exist also norms and checklists in practice, which evaluate automation systems as a total system. But these concepts focus on phases of the product life cycle after sales. Aspects of implementation, maintenance, operational performance of the system etc. dominate these evaluation concepts.

In order to centre an evaluation of automation systems on its total, interdisciplinary designed product architecture, a new approach has to be developed. Therefore, we propose our central research question of this paper:

How can we evaluate holistically the impact of the re-modularization of a subsystem on the total product architecture of an interdisciplinary developed system?

To address this research question, we develop a framework to understand the interrelations of independently designed subsystems. Using a Multiple Domain Matrix (MDM) (Lindemann et al., 2009), the impact of changes in a subsystem on the total system and the other particular subsystems can be made visible and evaluable – similar to the propagation of changes in Design Structure Matrices (DSM) (Clarkson et al., 2001). In the following chapter, a meta-model comprising the domains of an automation system is created and propositions for a target-oriented analysis are provided and examined. After that, implications for further steps are discussed. This paper closes with a conclusion and suggestions for further research.

3 A META-MODEL OF PRODUCT ARCHITECTURE OF AUTOMATION SYSTEMS

The goal of an evaluation framework for product architecture of automation systems is to provide transparency of mutual dependencies within the involved domains. With a profound understanding of interactions between the subsystems the additions, advancements, changes or elimination of underlying elements can be evaluated in terms of the impact on the total system. The introduction of an object-oriented programming language – as described in the introduction - is such an exemplary change of elements in the software domain. Consequently, how can the impact of such a change on the mechanical part of the system be made transparent?

Ramifications of changes – like described above - have to be expatiated in a model of the total system reflecting mutual influences of the particular subsystems developed independently by each discipline. Browning (2001) suggests Design Structure Matrices to be helpful to "display the relationships between components of a system in a compact, visual, and analytically advantageous format". In this paper, we avail ourselves of the advantages of matrix-methodologies and combine all disciplines involved in an automation system in a Multi Domain Matrix (MDM).

We designed a MDM covering all relevant aspects of an automation system. In the following, the Meta-Model of our MDM will be presented to make the derived steps for an evaluation more comprehensible. In the context of our work, the Meta-Model defines the various types of relationships between the elements of the matrix.

The automation system is structured in five domains: Mechanical components, technical functions, variables, software functions and controller components. While Ulrich (1995) suggested analysing the modularization of a mechanical product in terms of its *components* and related *technical functions*, we applied this idea to the software part of an automation system and chose *variables* and *software functions* as complementary domains. In addition to that, *controller components* realize the logical and physical connection between the software and hardware components and functions, thus, they are represented separately in a particular domain. This is advantageous to make both transparent the local and the logical allocation of components and functionality of the system. Table 1 gives an overview of the Meta-Model how the influence of each domain on the related domain is characterized and interpreted in the subsequent analysis.

The development of each subsystem is normally executed independently within the borders of a discipline. Modularization, changes in product architecture or re-shape of subsystem borders are planned and performed internally. Although, many different classifications of disciplines are possible with regards to automation systems, we determined these three to keep it simple: Mechanics, which is represented by *A+B*; Information Technology, which is represented by *C+D*, and, finally *E*, which represents interface competence between the previous two and can be referred as Electronics.

The example in the previous chapter of the replacement of one main controller with some micro-controllers to save costs for wiring, which outlined a simple effect chain of a change in a subsystem, can be made transparent and manageable in a MDM based on this framework. Especially in the DMMs, cross-disciplinary linkages can be traced logically and, therefore, be considered in the overall system architecture, when it comes to changes.

Table 1. *Meta-model for the evaluation of automation systems*

row [influences] column	*A* **Mechanical Component**	*B* **Technical Functions**	*C* **Variables**	*D* **Software Functions**	*E* **Controller Components**
A **Mechanical Components**	is connected to	performs	sends signal to	sends input for	is physically connected to
B **Technical Functions**	is performed by	is prerequisite for	is logically connected to	is implemented through	is limited by
C **Variables**	controls	is required for	is dependent of	is required for	is stored on
D **Software Functions**	controls	is required for	requires	is required for	is running on
E **Controller Components**	is physically connected to	limits	stores	runs	is connected to

In the following, we want to clarify the meanings of relevant Domain Mapping Matrices in terms of evaluating the modularization of an automation system. In these DMMs, changes of subsystems can be made transparent and effects on other subsystems can be shown. DMM *C→A* depicts relevant signals for actuators, which perform technical functions. A signal of a sensor is connected with its particular variable in DMM *A→C*. This variable is part of a software function. Thus, DMM *A→D* can be deducted logically from DMM *A→C*, and analogically, DMM *D→A* from DMM *C→A*.

The DMMs *D→B* and *B→D* imply the theoretically required interdisciplinary mapping of software functions and technical functions, which is intuitively comprehensible and could be filled easily in workshops with experts. Additionally, an analytical deduction of the DMMs *D→B* and *B→D* can be done. While the DMMs *A→B* or *B→A* match the mechanical realisation of technical functions, the DMM *A→C* in connection with DMM *C→A* describe the link to the software part of the system via the variables which - in turn - are mapped on the software functions in the intra-disciplinary DMMs *C→D* or *D→C*. This string of logical dependencies is only viable, if the physical connection is realized and modelled in the DMMs *A→E, C→E* and *D→E* and their transposed equivalents. DMM *A→E* (*E→A* analogue) describes the physical connection of the controller and the mechanical components, which could be interpreted as wiring or communication interface. The input/output-interface (I/O) of the controller is modelled in DMM *C→E* (or *E→C*). The DMMs *E→D* or *D→E* depict the implementation of assigned software functions. DMM *B→E* plays a subordinate role.

4 DISCUSSION

The framework presented in the previous section allows for an extensive description of automation systems. In order to gain benefits from these description, it is crucial to know which questions should be answered by the use of the framework. Thus, in order to enable a target oriented analysis we developed first hypotheses – which are to be amended in future work – concerning the estimated results of the former:

[H1] Changes in one domain ramify into the total system more intensively, the more interrelations with other domains exist.

These interrelations can be made transparent in the interdisciplinary DMMs. Changes and their triggered effect chains can be derived from the MDM. Changes could be, e.g. elimination of an element, substitution of an element by *n* others or the elimination or creation of coherences between two elements.

[H2] If all subsystems are modularized in terms of their mutual influences depicted in the interdisciplinary DMMs A→E, C→E and D→E or their transposed equivalents, the total system can be assumed as optimized.

This second hypothesis corresponds to the optimal modular design of the automation system architecture, as the DMMs A→E, C→E and D→E – representing the discipline electronics – can be seen as the interface between mechanics and information technology. Thereby, the question is to be answered whether differences arise, e.g. if the automated systems software functions are modularized and how this would affect mechanical view of the system. Thus, by regarding DMMs A→E, C→E and D→E the dependencies between all three disciplines are taken into account equally.

Additional work has to concentrate on the assessment of the domains of automation systems and components. Similar to work by Kreimeyer (2010) on process management and by Lindemann et al. (2009) on product re-modularization, the regarded automated systems are to be assessed by the application of structural metrics. Thereby, existing metrics are to be examined whether they are applicable to automation systems and how they can be interpreted. Hence, the sub-matrices of the MDM of automated systems are to be filled in order to allow such analysis. The data acquisition will be conducted using automation production systems available at the authors' research facility. Subsequently, single elements and modules are to be assessed concerning their relevance within the whole system. Thus, significant elements in the structure of the system are to be identified, which influence the former's modularity or the complexity of the overall system or subsystems. This should support, for example, the estimation of possible expenses for programming the software, by gaining insight on the amount of the software structure's complexity.

Eventually, the aim is to permit a holistic assessment of automated production systems concerning their complexity and optimal modularity – e.g. the extent of single modules or even the number of modules – forming the basis for the development of guidelines for re-modularisation of existing systems and also for the development of new modular automated systems.

5 CONCLUSION AND OUTLOOK

Currently, the development of new automation production facilities or optimisation of existing systems do not take into account the link between the different views of software, electronics and mechanical development. Hence, software and hardware components are developed or optimised apart from each other in separate teams. Although developers might be aware of the interfaces between both subsystems, these interfaces are not used actively in the process to gain valuable insights concerning possible optimisation of the overall system.

Therefore, a framework enabling a holistic analysis of automation production facilities is presented in this paper. The former is represented by a Multiple Domain Matrix. This MDM comprises five domains allowing for a complete description of an automation system, i.e. the mechanical components, technical functions, variables, software functions and controller components.

Based on this framework, an analysis can be conducted concerning the modularisation of the automation system. The impact of the modularisation of one domain on the whole system can be examined. Thus, in order to re-modularize an existing system, present modules can be assessed regarding the need of adjustment. Moreover, new modules can be derived to form an optimal architecture of the whole automation system.

REFERENCES

Clarkson, P. J., Simons, C., & Eckert, C. (2001). Predicting change propagation in complex design. In *ASME International Design Engineering Technical Conferences (Design Theory & Methodology Conference)*, Pittsburgh, September 2001.

DIN EN 61131, Part 3 (2003). Programmable controllers – Part 3: Programming languages (IEC 61131-3:2003); German version EN 61131-3:2003. Berlin: Beuth.

Katzke, U. (2009). *Spezifikation und Anwendung Spezifikation und Anwendung einer Modellierungssprache für die Automatisierungstechnik auf Basis der UML*. Kassel: Kassel university press.

Katzke, U.; Vogel-Heuser, B. (2009). Vergleich der Anwendbarkeit von UML und UML-PA in der anlagennahen Softwareentwicklung der Automatisierungstechnik – Beispiel einer vergleichenden empirischen Untersuchung von Modellierungssprachen. In *Automatisierungs-*

technik (at), 57, 7, pp. 332-340.

Kreimeyer, M. (2010). *A Structural Measurement System for Engineering Design Processes*. Munich: Dr. Hut (to be published).

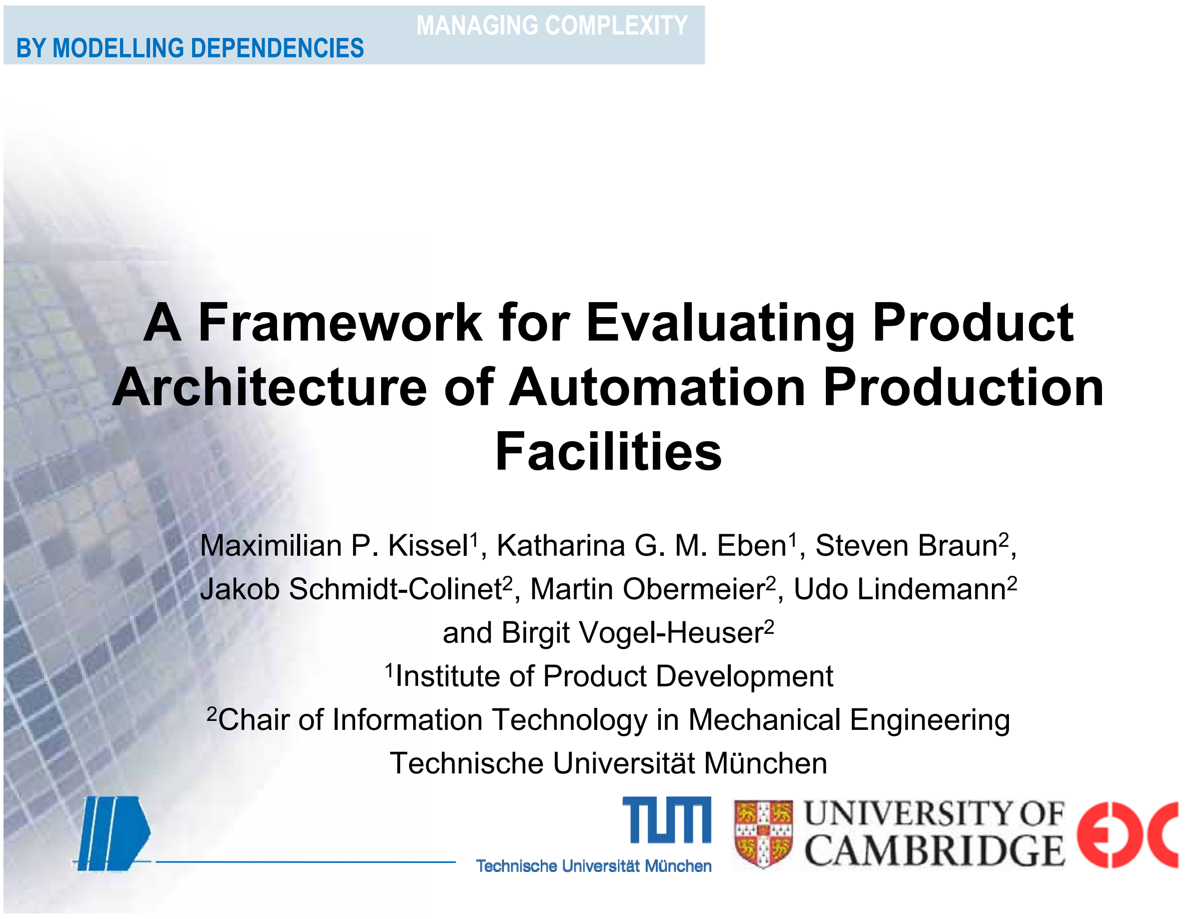
Lindemann, U., Maurer, M., & Braun, T. (2009). *Structural Complexity Management – An Approach for the Field of Product Design*. Berlin: Springer.

Pimmler, T. U., & Eppinger, S. D. (1994). Integration analysis of product decompositions. In *ASME International Design Engineering Technical Conferences (Design Theory & Methodology Conference)*, Minneapolis, September 1994.

Ulrich, K. (1995). The role of product architecture in the manufacturing firm. *Research Policy*, 24, 419 - 440.

Waldman, F., & Sangal, N. (2009). Practical uses of classification with DSM. In *Proceedings of the 11th International DSM Conference, DSM 2009*, October 2009 (pp. 119-130). Munich: Hanser.

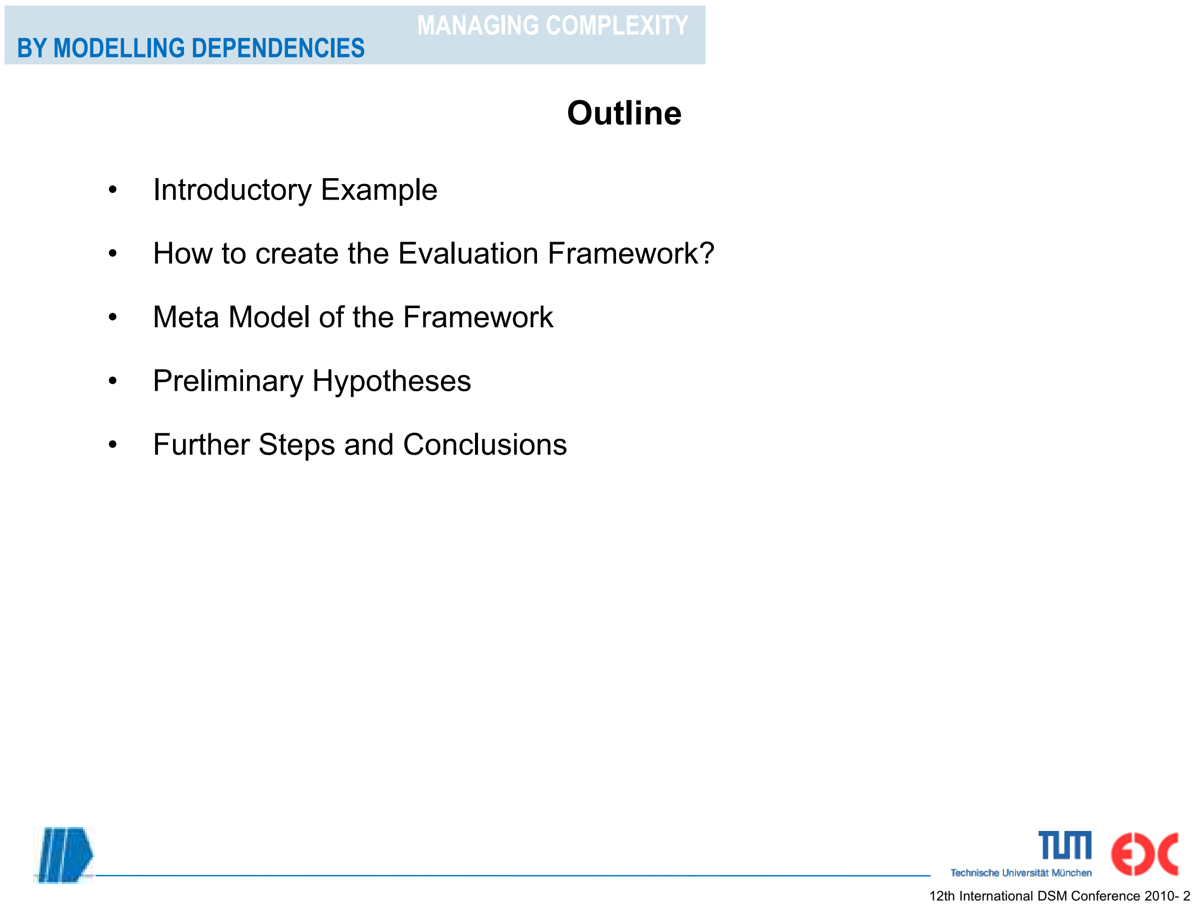
Contact: Maximilian Kissel
Technische Universität München
Institute of Product Development
Boltzmannstraße 15
85748 Garching
Germany
0049 89 289 15134
0049 89 289 15144
kissel@pe.mw.tum.de
www.pe.mw.tum.de


MANAGING COMPLEXITY
BY MODELLING DEPENDENCIES

A Framework for Evaluating Product Architecture of Automation Production Facilities

Maximilian P. Kissel[1], Katharina G. M. Eben[1], Steven Braun[2], Jakob Schmidt-Colinet[2], Martin Obermeier[2], Udo Lindemann[2] and Birgit Vogel-Heuser[2]
[1]Institute of Product Development
[2]Chair of Information Technology in Mechanical Engineering
Technische Universität München

Technische Universität München
UNIVERSITY OF CAMBRIDGE


MANAGING COMPLEXITY
BY MODELLING DEPENDENCIES

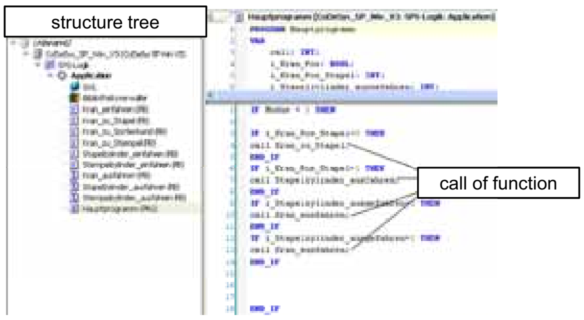
Outline

- Introductory Example
- How to create the Evaluation Framework?
- Meta Model of the Framework
- Preliminary Hypotheses
- Further Steps and Conclusions

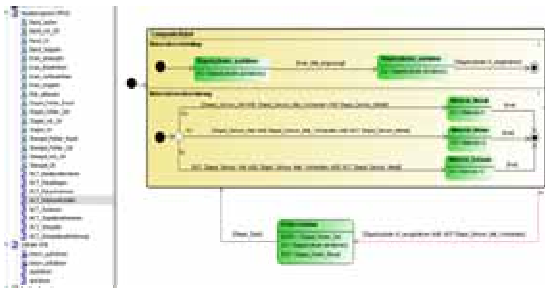
Technische Universität München
12th International DSM Conference 2010- 2

Introductory Example – Change of the programming paradigm in automation production facilities

- Process-oriented programming
  - uses logical blocks on the same hierarchical level
  - requires storage of the whole code on one real-time controller



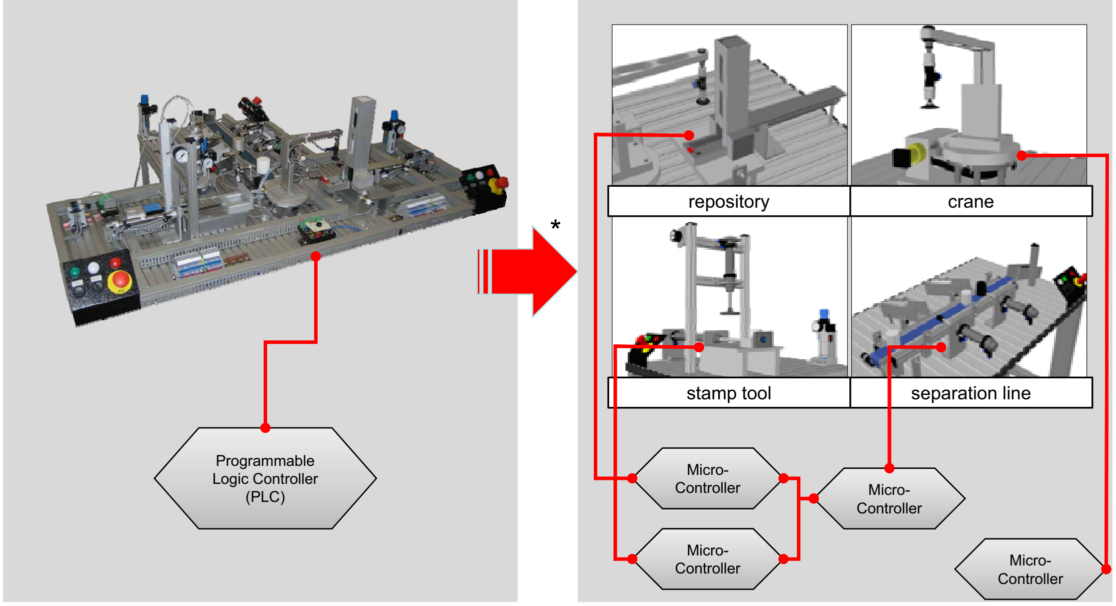
- Object-oriented programming
  - can create hierarchies, classes, children, modules etc.
  - allows development of independent, encapsulated modules within the automation system



Introductory Example – Change of the programming paradigm in automation production facilities

Process-oriented programming

Object-oriented programming



* Katzke, U. (2009). *Spezifikation und Anwendung Spezifikation und Anwendung einer Modellierungssprache für die Automatisierungstechnik auf Basis der UML*. Kassel: Kassel university press.
Katzke, U.; Vogel-Heuser, B. (2009). Vergleich der Anwendbarkeit von UML und UML-PA in der anlagennahen Softwareentwicklung der Automatisierungstechnik - Beispiel einer vergleichenden empirischen Untersuchung von Modellierungssprachen. In *Automatisierungstechnik (at)*, 57, 7, pp. 332-340.

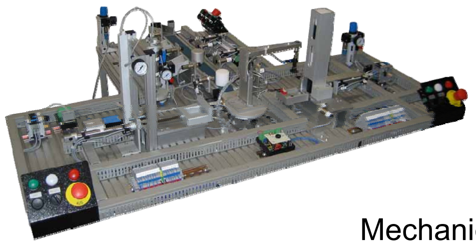
Independent Modularization in Automation

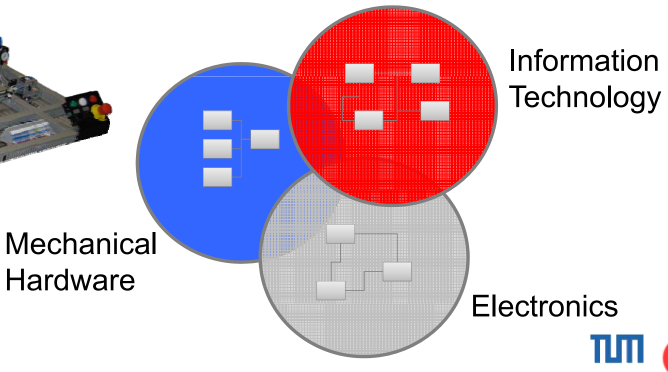
Problems

- Each discipline involved develops its subsystem independently
- The impact of re-modularization, elimination or attachment of elements of a subsystem on the other subsystems beyond its interfaces remains neglected

Challenges

- Need for a modularization approach on a total system level
- Elaboration of further potential improvements on a total system level
- Need for guidelines how to design complex, interdisciplinary developed automation systems





Literature Review

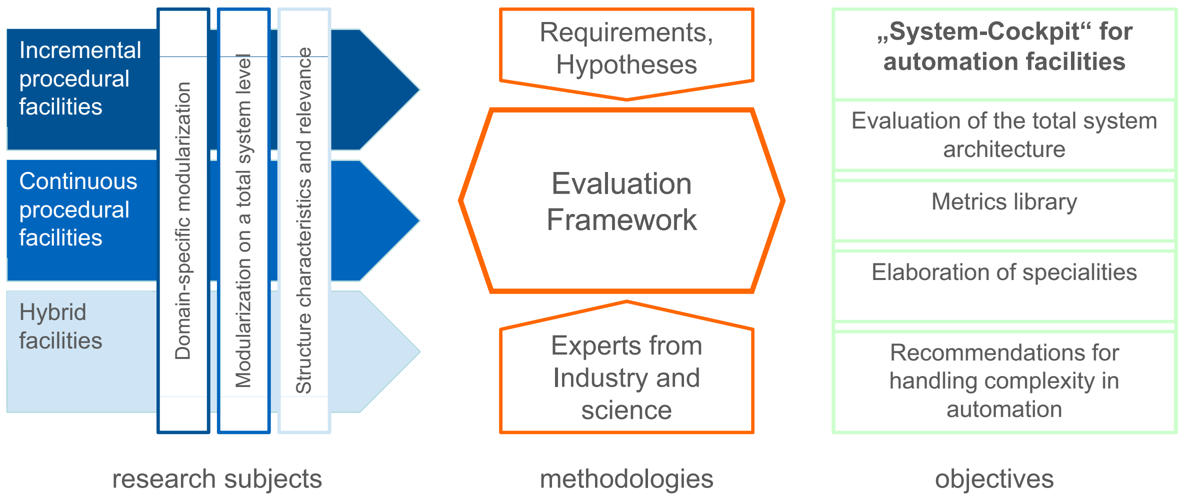
- Waldman and Sangal (2009) suggest a methodology to shape and evaluate software modules in terms of software development (Lattix)
- Methodologies to approach modularization and evaluation of product architecture were presented by Ulrich (1995) and Pimmler and Eppinger (1994)
- Using a Multiple Domain Matrix (MDM) (Lindemann et al., 2009), the impact of changes in a subsystem on the total system and the other particular subsystems can be made visible and evaluable
- Browning (2001) suggests Design Structure Matrices to be helpful to “display the relationships between components of a system in a compact, visual, and analytically advantageous format”

How can we evaluate holistically the impact of the re-modularization of a subsystem on the total product architecture of an interdisciplinary developed automation system?

Browning T. R., (2001). Applying the Design Structure Matrix to System Decomposition and Integration Problems: A Review and new Directions. *IEEE Transactions on Engineering Management, Vol. 48, No. 3, pp. 292-306*
Lindemann, U., Maurer, M., & Braun, T. (2009). *Structural Complexity Management – An Approach for the Field of Product Design*. Berlin: Springer.
Pimmler, T. U., & Eppinger, S. D. (1994). Integration Analysis of Product decompositions. In *ASME International Design Engineering Technical Conferences (Design Theory & Methodology Conference), Minneapolis, September 1994*
Ulrich, K. (1995). The role of product architecture in the manufacturing firm. In Research Policy, 24, 419 - 440.
Waldman, F., & Sangal, N. (2009). Practical Uses of Classification with DSM. In *Proceedings of the 11th International DSM Conference*, DSM 2009, October 2009 (pp. 119-130). Munich: Hanser.

How to create the Evaluation Framework?

How can the product architecture of complex, interdisciplinary developed systems be evaluated in automation technology?


Incremental procedural facilities
Continuous procedural facilities
Hybrid facilities
Domain-specific modularization
Modularization on a total system level
Structure characteristics and relevance
Requirements, Hypotheses
Evaluation Framework
Experts from Industry and science
„System-Cockpit“ for automation facilities
Evaluation of the total system architecture
Metrics library
Elaboration of specialities
Recommendations for handling complexity in automation
research subjects
methodologies
objectives

Meta Model of the framework

row [influences] column	A Mechanical Component	B Technical Functions	C Variables	D Software Functions	E Controller Components
A Mechanical Components	is connected to	performs	sends signal to	sends input for	is physically connected to
B Technical Functions	is performed by	is prerequisite for	is logically connected to	is implemented through	is limited by
C Variables	controls	is required for	is dependent of	is required for	is stored on
D Software Functions	controls	is required for	requires	is required for	is running on
E Controller Components	is physically connected to	limits	stores	runs	is connected to

The chain of cause and effects of a change can be made transparent in these DMMs, which describe relations between subsystems

Preliminary Hypotheses

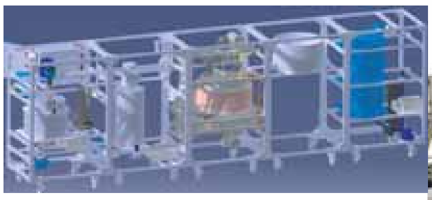
[H1] Changes in one domain ramify into the total system more intensively, the more interrelations with other domains exist.

[H2] If all subsystems are modularized in terms of their mutual influences depicted in the interdisciplinary DMMs A→E, C→E and D→E or their transposed equivalents, the total system can be assumed as optimized.

To be continued…

Further Steps

- Include feedback from scientists and experts from industry in the evaluation concept
- Fill the Meta Model with data of the research automation facilities at the institute and determine structure characteristics and their relevance
- Calculate optimal modularization on a subsystem and a total system level and perform sensitivity analyses by simulating changes of subsystems
- Deduce general assignable metrics for automation facilities
- Give recommendations for handling complexity in automation technologies



Conclusions

- Software and hardware components of an automation system are developed or optimised on a subsystem level in separate teams
- The need to re-think this design approach has been presented in this presentation
- We developed a concept to evaluate automation facilities on a total system level and deduced preliminary hypotheses
- Further steps towards an evaluation concept based on Multi-Domain-Matrices have been outlined

Technische Universität München