Preprints of the 19th World Congress
The International Federation of Automatic Control
Cape Town, South Africa. August 24-29, 2014

A SysML based design pattern for the high-level development of mechatronic systems to enhance re-usability

G. Barbieri*, K. Kernschmidt, C. Fantuzzi*, B. Vogel-Heuser****

* University of Modena and Reggio Emilia, Via Amendola 2, 42122 Reggio Emilia (Italy) (e-mail: {giacomo.barbieri, cesare.fantuzzi}@unimore.it).
** Institute of Automation and Information Systems, Technische Universität München, Boltzmannstr. 15, 85748 Garching bei München (Germany) (e-mail: {kernschmidt, vogel-heuser}@ais.mw.tum.de).

Abstract: Model driven engineering approaches can be used to handle the complexity in the development of modern mechatronic systems, containing a multitude of mechanical, electrical/electronic and software components. However, up to now SysML, as standard systems engineering language, is not wide spread in industry yet. Reasons therefore are missing adequate guidelines for the modeling process as well as an unclear benefit of the created SysML-models. A well-created system model however poses enormous time advantages during the analysis of change influences in later lifecycle phases of the system and makes an interdisciplinary reuse of modules in the development of new systems possible. A prerequisite therefore is the efficient traceability of all information within the system model. Thus, in this paper a SysML based process for the high-level development of mechatronic systems is applied, reaching from requirements specification to the detailed modeling of the element-connections (discipline specific as well as interdisciplinary). Our approach shows how the information from the different levels of abstraction and the different development phases can be connected, including a functional modularization of the mechatronic system. In this way, developers can trace change influences more easily. The functional modules can be used during the development of new systems, resulting in significant shortened development cycles. The proposed design pattern is shown at the example of a bench-scale model of a production plant.

1. INTRODUCTION

The development of modern mechatronic systems is a complex task, requiring the specific knowledge of mechanical engineers, electric/electronic (E/E) engineers and software engineers as well as an integrated view on the overall system during the development.

In order to integrate the specific knowledge of each domain, model-based systems engineering (MBSE) was introduced. It describes the application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout the development and later lifecycle phases (INCOSE, 2007). The vision implies the usage of a single model, rather than several documents, for capturing requirements, analyzing problems and designing systems. The overall objective is to replace a document-centric working style through a model-centric one.

The Systems Modeling Language (SysML) reuses a subset of the UML and adds extensions to it in order to become the standard systems engineering language. It is a graphical modeling language and can represent complex systems such as hardware, software, data, personnel, procedures, or facilities (Friendenthal et al., 2008). It is object-oriented, supports hierarchical modeling and allows the representation of these system views: structural, functional, behavioral and requirements. Moreover, it can be easily extended through the creation of profiles and can be integrated into existing tool environments (e.g. Cao et al., 2011). In spite of all these benefits, up to now SysML is still not widely spread in industry yet. Typically, the following culture and general resistances exist:

- "lack of perceived value of MBSE" (Motamedian, 2013);
- difficulties in abstracting real systems and the consequent generation of large, inflexible and hard to maintain SysML models. These aspects make the model information hard to reuse in new projects (Kasser, 2010);
- high learning efforts and the missing of modeling methods and guidelines (Albers et al., 2013);
- missing of narrative examples that describe the generation of SysML models and lack of maturity and usability of the SysML tools (Bone et al., 2009).

Due to these existing problems, we show in this paper a SysML modeling architecture and design pattern in order to build models that enhance traceability of the information, facilitating the analysis of change influences in later lifecycle phases of the system and reuse for future projects. In this way, we show a major benefit of the MBSE application, provide a narrative example and deduce guidelines for the modeling of mechatronic systems focusing particularly on

model re-usability. We illustrate the proposed approach at the example of an existing system through reverse engineering to compare the results. A development from scratch can be conducted accordingly, both in the application of the design pattern for the design of brand-new components, and the reuse of parts developed in previous projects and documented in the proposed architecture. Furthermore, most of the organizations that are using MBSE have focused on the system designing features of MBSE more than on other aspects (Motamedian, 2013). For this reason, we built the model simulating a design from scratch process of a production plant, following a general system engineering design methodology. Eventually, we demonstrated the traceability of information when changes are implemented in the model.

The rest of the paper is organized as follows: in section 2 related works are described. Section 3 illustrates the utilized design process and the use case example, while section 4 describes our proposed model architecture and design pattern. In section 5, the traceability of information in the model in case of a change is shown. Eventually, conclusions and future work are presented in section 6.

2. RELATED WORK

Various research works have been carried out on the model based design of mechatronic systems for the development of methodologies and modeling strategies that enhance the traceability of information.

Lindemann et al. (2009) propose to use a matrix-based approach at the beginning of the development process to model dependencies between system elements. However, according to the authors, these elements should be specified at an early point and later changes should be avoided. Especially in customer specific mechatronic systems, such as production plants, changes however often occur also in later lifecycle phases.

Helms and Shea (2010) extend the methods for computational design synthesis (generation of alternative solutions tailored to particular problems and to computationally describe solution spaces) integrating the graph grammar approach with concepts from object-oriented programming. They focus on the practical description of alternatives and not in design processes which lead to their generation, and they just consider the mechanical domain of mechatronic systems.

Anacker et al. (2012) illustrate an architecture for the development of software engineering solution patterns for the system design of mechatronic systems. This approach however, focuses specifically on the software parts and neglects the other mechatronic domains (mechanics and electrics/electronics).

Shah et al. (2010) propose a joint SysML model, coupled to discipline specific models. This has the advantage, in comparison to the transformation from one specific model to others, that all required information is included in one model, ensuring consistency and keeping the right level of abstraction. Nevertheless, guidelines about what information should be integrated in the SysML-model are not provided. Based on this approach, other researchers developed similar SysML based methods. Bassi et al. (2011) define a design methodology whose application generates a hierarchy of models which describe the system at different levels of abstraction. Chami et al. (2012) introduce a method for an 'Intelligent Conceptual Design Evaluation' of mechatronic systems. However, both approaches do not deal with the problem of how functional and behavioral aspects can be incorporated in the design.

Thramboulidis (2013) defines the design process as a composition of already existing mechatronic components (MTCs) that properly collaborate in order to fulfill the functionalities required by the system behavior. However, the approach does not focus on a design pattern for the development process. Furthermore, only mechatronic components are considered and not e.g. purely software modules.

Our objective is to introduce a design pattern for the design of all relevant MBSE-aspects of mechatronic systems (i.e. functional, mechanical, electrical and software), providing a SysML architecture that enhances traceability of the information and reuse for future projects.

3. ADOPTED PROCESS AND CASE STUDY DESCRIPTION

In order to deduce a method for the modeling of mechatronic systems in a way that enhances model reuse, we simulated a design from scratch process of a bench-scale model of a production plant. We followed the V model described in Biffl et al. (2006), for the design process, and the requirements writing strategy, presented in Buede (2009). The result is a hierarchical and iterative process based on different levels of abstraction: we individuated the system, the modules and the components levels. A high-level system model is designed in the first step and the modules in the module-level inherit its information. Next, the modules are defined on a high level of abstraction and are broken down into components. The components then are designed in detail or already existing (commercial) components are adopted. These steps were defined as the breakdown phase (left side of the V model). The detailed information of the lowest level is integrated then to the higher levels through a detailed development of the modules first, and finally of the system level. These steps are described as integration phase (right side of the V model) of the design process. The verification and validation phases are not in the focus of this paper.

The design steps, performed at every level of abstraction, are as following:

The higher-level requirements are broken down and, if possible to define at this level, a physical principle is associated based on these requirements. If it is not possible to define a physical principle yet, the requirements have to be broken down to the lower levels (e.g. on the system level in our case study no physical principles could be defined yet). The knowledge of the physical principle allows the

refinement of the functions that the element must perform and the subsequent generation of a functional architecture. Then, the considered element is subdivided in lower level elements and the functions are allocated to the lower level. Eventually, a high level architecture is built on the basis of the available information. The procedure is then applied accordingly to the next lower level.

After the lowest level (components level) has been designed in detail, an iterative refinement of the upper levels is conducted until the creation of an integrated detailed model of the requirements, the fulfilled functions, the behavior, and the structure of the entire system is possible.

For the highest level (the system level), the requirements are inherited from the "system context". This is composed by the system boundaries (all the actors and the quantities that interact with the system under development), the operational concept (the information exchanged with the external systems), and the stakeholder requirements (the needs of individuals or organizations that have direct interests).

As a use case example, we analyzed the development and design process for a bench-scale model of a production plant (Fig. 1). It consists of typical parts of a production system and is controlled by a single PLC. In the first part (stack) the working pieces, which arrive from an upstream system (e.g. prior production steps) are stored and then pushed separately to the handover position of a crane. The crane picks the work pieces, lifts them, and displaces them to the outlet position, which is located in an angle of 90° to the handover position. After the work pieces are placed by the crane, they are stored again in an outlet-storage (slide) from where they can be used for further processing by the downstream system. Thus, different mechanical (e.g. slide), electric/electronic (e.g. the 5/2 directional-valve of the vacuum gripper), and software components (e.g. crane control) are used, and require an integrated development, which we will analyze.

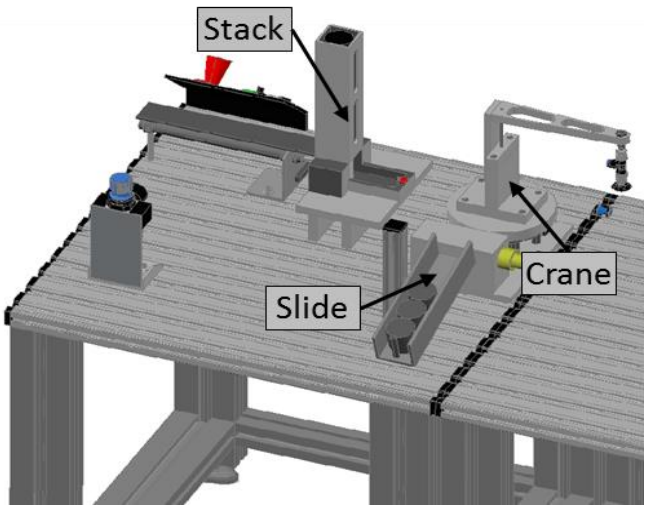

Fig. 1. CAD model of the case study (based on Legat et al., 2013).

4. MODELING APPROACH AND ARCHITECTURE

In this section, the architecture and strategies utilized to connect the information, developed during the breakdown phase ('design view' package in Fig. 2) and to represent all the system information once the integration phase is completed (the other packages in Fig. 2) are described. The architecture illustrated below, is applied to all elements of every level of abstraction.

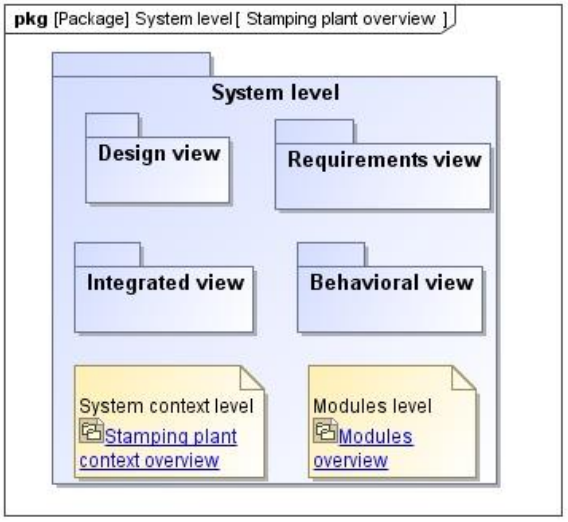

Fig. 2. Model architecture applied to the system level: hyperlinks were created to directly access to the selected view and to jump to the upper or lower level of abstraction.

4.1 Design view

This view (Fig. 3) contains the evolution of the information developed during the breakdown phase. In the integration phase, an iterative refinement will be applied to it. Nevertheless, the 'design view' package will not be modified in the later design phases, because the refined information will be represented in the other views. This is necessary to document the steps that have brought the developers to the choice of a certain design solution.

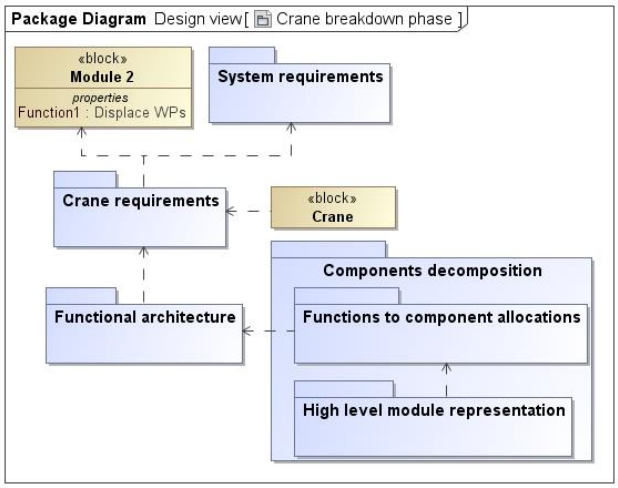

Fig. 3. Design view of the crane module; a package is created for every step of the breakdown phase

Element requirements

The element requirements (e.g. module requirements on the module level) are inherited from the upper level requirements and from the "Functions to elements allocations" of the upper level design view. If it is possible, based on the existent information, a physical principle is associated to the element (e.g. crane block in Fig. 3).

The following shortcomings about the SysML requirement block and diagram can be individuated (Ozkaya, 2006):

- additional properties should be added to the standard requirement class in order to enhance traceability;
- if one-to-many and many-to-many relations exist, the diagrams become cluttered with lines reducing legibility;
- the diagram does not offer a higher level of abstraction to support requirement navigation.

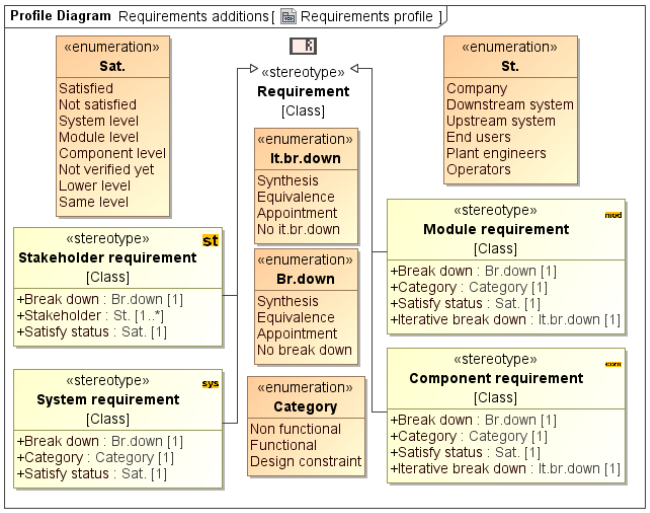

Fig. 4. Requirements profile: a stereotype is created for every level of abstraction

For these reasons, we have developed a requirements profile (Fig. 4) and propose to represent requirements in tables rather than in diagrams. Through the additional properties, it is possible to document how a requirement is broken down and, when the iterative process starts, how the requirement influences the upper levels (iterative breakdown property). Moreover, some (non functional) requirements can be just satisfied at a certain higher level, while the lower level elements have an influence on it (e.g. the total system production rate); so we integrated the satisfy status property. We then set connections among the requirements themselves and among requirements and other model elements through the standard SysML relationships:

- *derive* for organizing break down and iterative break down requirements dependencies;
- *satisfy* for connecting requirements with the design elements which fulfill them;
- *verify* for bridging the requirement with the test-case;
- *trace* for connecting elements that help on deriving requirements (e.g. simulations).

The requirement tables contain all these properties and relationships. In this way, every requirement is traced and it is possible to navigate among the different abstraction-levels of requirements and related model elements.

Functional architecture

After the requirements breakdown, the functional architecture can be generated. An activity is created for every functional requirement and all the activities are ordered and connected, through control flows in the activity diagram, for the definition of the functional architecture (Fig. 6). Further development steps e.g. discrete event simulations could be performed for refining the element requirements (e.g. time constraint for the execution of an action).

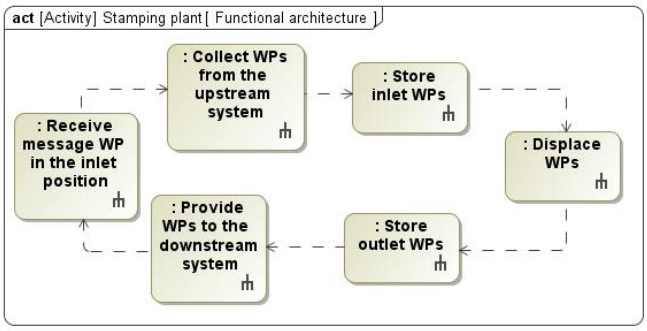

Fig. 5. Part of the system functional architecture: fault detection, error handling, communication with the system context and initialization and termination phases are not represented.

Functions to lower elements allocations

The functions are then allocated to the lower level elements: the element is represented as block with the allocated functions as attributes (e.g. Module 2 block in Fig. 3).

High level representation

Eventually, a high level representation of the system can be built considering just the physical parts and the item flows among the elements, in order to give an overall idea of its structure. The detailed design of software, electrical/electronic and mechanical parts as well as their connections cannot yet be included here, as they first can be defined specifically after the lowest level (components) has been designed.

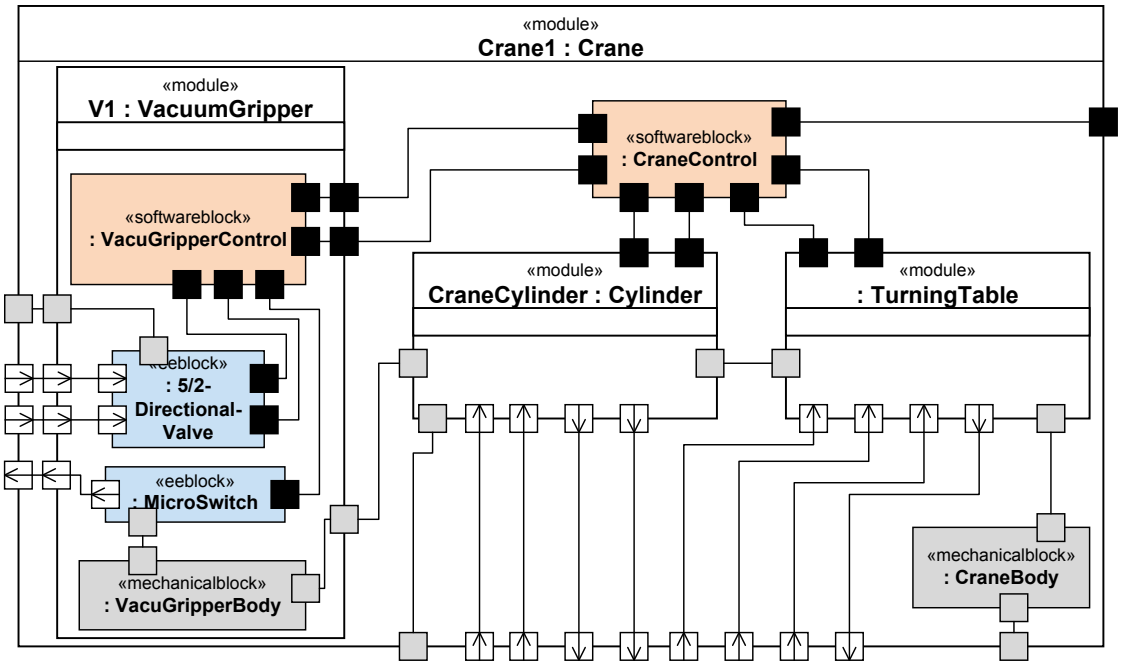

Fig. 6. Representation of the crane module in SysML4*Mechatronics*

4.2 Requirements view

This view contains all the requirements of the considered element: both the ones of the design view and the ones derived after the integration process. As described above, we utilized tables and the illustrated profile to represent the requirements. Thus, after the entire development process is finished, this package includes the complete list of all requirements concerning the respective element.

4.3 *Integrated view*

In the integrated view the concrete element-structure with all the interfaces is represented (Fig. 6). The connections among the different domains has to be taken into account; e.g. a sensor has to be considered with its electrical connection to the PLC, as well as its representation in the software, where it sets a variable if a work piece is detected. For modeling the structural view, we applied the SysML profile SysML4*Mechatronics* (Kernschmidt and Vogel-Heuser, 2013). It enables a detailed specification of the ports in the different disciplines (i.e. in Fig. 6 grey ports: mechanics, white: electrics/electronics, black: software), as well as an analysis and visualization of change influences. In this way, the system model can also be used in later lifecycle phases of the mechatronic system. Thus, if a change shall be implemented, the change influences on the system can be analyzed prior in the model before they are applied to the real system, reducing the system down-time to a minimum. Next to the change analysis, the detailed integrated model serves as linking pin to the single discipline specific developments and implementations. The software-blocks (*«SW block»*) show the real software architecture and can be used as skeletons of function blocks in the IEC61131-3 implementation. The electrical ports, which specify the communication, represent the I/O mapping of the sensors and actuators in the model and can be taken over directly in the settings of the programming environment. As described above, the system modularization during design was driven by its functionality rather than by the mechanical structure. Thus, in order to consider geometrical togetherness, the mechanical ports are used between the components within the same module, but also beyond the (functional) module borders. For example, valves are used in different modules, however, from a mechanical point of view, they are all integrated in the same valve cluster (not shown in Fig. 6) and thus, are connected mechanically to the valve cluster.

4.4 *Behavioral view*

This view contains two diagrams: an activity diagram with the functional architecture refined in a software viewpoint (Fig. 7) and the element state machine (Fig. 8). In the integration phase of the design process, the functional architecture of the design view is modified on the basis of the lower levels information; e.g. further activities could be implemented. Then, a software functional architecture is built considering just the activities that will contribute to the control system definition: the activities that are executed automatically by the mechanical parts without the necessity of a command will be neglected (e.g. in Fig. 7 'Collect, store and provide WPs' are not inserted because implicitly performed by the stack and slide modules).

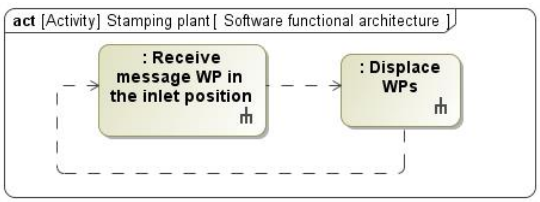

Fig. 7. Refinement of the system functional architecture represented in Fig. 5.

Eventually, the element state machine is built on the basis of the software functional architecture (Fig. 8).

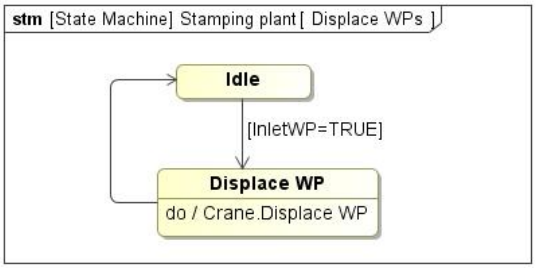

Fig. 8. Part of the stamping plant state machine. The Displace WP state machine of the crane module is called in the Displace WP state.

Following this hierarchical modeling approach, a modular control system is obtained, which, once the lower levels state machines are defined, allows the design of the control system at a higher level of abstraction. For example, in the Displace WP state (Fig. 8), a crane state machine that contains all the actions executed by the crane to displace a WP is called; this implies the actions: lift the crane, turn to the pickup position, lower, take in the WP, lift again, turn to the outlet position, lower and release WP.

5. TRACEBILITY OF INFORMATION IN THE MODEL TO ANALYZE CHANGES

In order to show how our proposed modeling approach enhances traceability, the steps that have led to the utilization of Micro-Switches as sensors for the crane positions, are illustrated in Fig. 9.

The "digital signals constraint" (stakeholder requirement), affects the derived lower level requirements and the physical parts that fulfil them. The physical parts include thereby also electrical connections as well as their representation in the software (e.g. variable-value). Through this chain all the information related to the Micro Switch adoption (i.e. requirements, mechanical, electrical/electronic and software domains) can be traced. In an appropriate software tool, e.g. MagicDraw, rational chains (as the one in Fig. 9) can be created automatically through a relational map option.

In the following the effects of a change in the stakeholder requirements is demonstrated: The new stakeholder requirement "Analogue signals have to be utilized" is broken down to the requirements on the lower levels, and finally influences the component requirement for the selection of an adequate sensor. In our case, an encoder has to be used for the crane position feedback instead of Micro Switches. This change has also influences on the electrics and the software, namely that an analogue signal between 0-10V has to be processed by the PLC and in the software the crane position feedback must be represented as 'real type' variable (These change influences on the other system elements can be analyzed in the SysML4*Mechatronics* model).

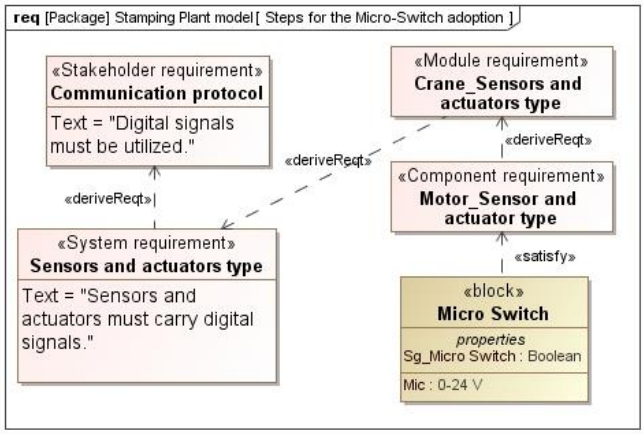

Fig. 9. Design viewpoint for the adoption of a Micro Switch. It carries a Boolean software variable, called "Sg_Micro Switch", and has an electrical "out" port (called Mic) which provides a 24V signal if the Micro Switch is on and 0V if it is off.

In an equivalent way the change can be analyzed if a component is exchanged (e.g. if during maintenance a sensor has to be exchanged that is not available anymore and thus, has to be replaced by another one). Hereby the model can show if all requirements are still fulfilled by the new component.

The example demonstrated how the proposed design pattern allows a complete traceability of the modeled information: every change can be inserted in a rational chain that contains all involved elements and requirements.

6. CONCLUSIONS AND OUTLOOK

In this paper we presented a SysML based design pattern for the high-level development of mechatronic systems, which illustrates a major benefit of model driven engineering through enhancing the reusability of models. A prerequisite therefore is the traceability of all information in the model, developed during the design, from requirements specification to the detailed modeling of the system structure and behavior.

Therefore, we defined an adequate model architecture, which can be applied on each level of abstraction (system level, module level, and component level) during the development process. In order to enable the traceability of the breakdown of requirements we introduced a specific SysML-profile, defining on which level a requirement is satisfied and from which requirements it is derived.

For modeling the integrated structural view of the system we utilized the SysML profile SysML4*Mechatronics*, which enables a detailed port-specification and analysis of the discipline-specific and interdisciplinary relationships in the mechatronic system.

Thus, by utilizing our presented methodologies, changes in later stages of the system lifecycle can be traced back to the according requirements and other effected components. In this way, the change influences can be estimated more

efficiently, leading to shorter system down times and less unexpected faults during the implementation of the change.

Through the inclusion of all relevant information in the presented modeling architecture a reuse of components, modules or entire (sub-) systems in further projects can be conducted easily by the developers, by integrating simply the prior model to the new project-model on the respective level.

Our approach was shown in this paper at the example of a bench-scale model of a production plant, as a next step the design pattern will be implemented in a real industrial use-case to show the scalability and applicability of our approach. As different approaches exist for the automatic generation of code (e.g. Vogel-Heuser et al., 2005), Fantuzzi et al., 2011), their usage in the presented approach could enhance the benefit of an integrated, model based design approach of mechatronic systems even further.

ACKNOWLEDGMENT

We thank the German Research Foundation (Deutsche Forschungsgemeinschaft – DFG) for funding this work as part of the collaborative research centre 'Sonderforschungsbereich 768 – Managing cycles in innovation processes – Integrated development of product-service-systems based on technical products' (SFB768).

REFERENCES

A. Albers and C. Zingel, "*Challenges of model-based systems engineering: A study towards unified term understanding and the state of usage of SysML*," in Smart Product Engineering: Proceedings of the 23rd CIRP Design Conference, 2013.

H. Anacker, J. Gausemeier, R. Dumitrescu, S. Dziwok and W. Schafer, "*Solution patterns of software engineering for the system design of advanced mechatronic systems*," in International Workshop on research and Education in Mechatronics, (Paris, France), November 2012.

L. Bassi, C. Secchi, M. Bonfe, and C. Fantuzzi, *"A SysML-based methodology for manufacturing machinery modeling and design,"* IEEE Trans. Mechatron., vol. 16, no. 6, pp. 1049–1062, 2011.

S. Biffl, D. Winkler, R. Höhn, H. Wetzel, "*Software Process Improvement in Europe: Potential of the new V-Modell XT and Research Issues,*" in Journal Software Process: Improvement and Practice, Vol. 11, no. 3, pp. 229-238, 2006.

M. Bone and R. Cloutier, "*The Current State of Model Based Systems Engineering: Results from the OMG SysML Request for Information 2009*," in Conference on Systems Engineering Research, CSER, (Hoboken, NJ, USA), March 2010.

D. M. Buede, "*The Engineering Design of Systems*". Wiley, 2009.

Y. Cao, Y. Liu, and C. J. Paredis, "*System-Level Model Integration of Design and Simulation for Mechatronic Systems based on SysML*," Mechatronics, September 2011.

M. Chami, H. Bou Ammar, H. Voos, K. Tuyls, and G. Weiss, *"A Nonparametric Evaluation of SysML-Based Mechatronic Conceptual Design,"* in Conference on Artificial Intelligence, 2012.

C. Fantuzzi, F. Fanfoni, C. Secchi, and M. Bonfè, "*A Design Pattern for translating UML software models into IEC 61131-3 Programming Languages,"* in 18th IFAC World Congress (pp. 9158–9163), September 2011.

S. Friendenthal, A. Moore, and R. Steiner, "*A Practical Guide to SysML: The Systems Modeling language*". Morgan Kaufmann, 2008.

B. Helms, and K. Shea, "*Object-Oriented Concepts for Computational Design Synthesis,"* in International Design Conference, Dubrovnik, Croatia, May 2010.

INCOSE, "*Systems Engineering Vision 2020*," tech. rep., International Council on Systems Engineering Seattle, USA, 2007.

J. E. Kasser, "*Seven systems engineering myths and the corresponding realities*," in Systems Engineering Test and Evaluation Conference, (Adelaide, Australia), 2010.

K. Kernschmidt, and B. Vogel-Heuser, "*An interdisciplinary SysML based modeling approach for analyzing change influences in production plants to support the engineering*," in 9th annual IEEE International Conference on Automation Science and Engineering (IEEE CASE 2013), August 17-21, 2013, Madison, WI, USA.

C. Legat, J. Folmer, and B. Vogel-Heuser, "*Evolution in Industrial Plant Automation: A Case Study*," in 39th Annual Conference of the IEEE Industrial Electronics Society (IECON 2013), Austria, Vienna, 2013.

U. Lindemann, M. Maurer and T. Braun, *"Structural Complexity Management. An Approach for the Field of Product Design,"* Berlin, Germany: Springer 2009.

B. Motamedian, "*MBSE Applicability Analysis*," International Journal of Scientific and Engineering Research, 2013.

I. Ozkaya, "*Representing requirement relationships*," in Requirements Engineering Visualization, First International Workshop on 2006.

A.A. Shah, A.A. Kerzhner, D. Schaefer, and C.J.J. Paredis, "*Multi-View Modeling to Support Embedded Systems Engineering in SysML*," in Graph transformations and model-driven engineering, Eds. 2010, G. Engels, C. Lewerentz, W. Schäfer, A. Schürr, and B. Westfechtel, Berlin, Germany: Springer, 2010, pp. 580-601.

K. Thramboulidis, *"Overcoming Mechatronic Design Challenges: the 3+1 SysML-view Model,"* Computing Science and Technology International Journal, January 2013.

B. Vogel-Heuser, D. Witsch and U. Katzke, "*Automatic Code Generation from a UML model to IEC 61131-3 and System Configuration Tools*," Int. Conf. on Control and Automation, Budapest, 27-29 June 2005, pp. 1034-1039.