INTERNATIONAL DESIGN CONFERENCE - DESIGN 2010
Dubrovnik - Croatia, May 17 - 20, 2010.

USING SYSML IN THE PRODUCT DEVELOPMENT PROCESS OF MECHATRONIC SYSTEMS

M. Follmer, P. Hehenberger, S. Punz and K. Zeman

Keywords: SysML, product development process, system-level modelling, system-level models, system models, mechatronics, systems-of-systems

1. Introduction

One of the major challenges in developing mechatronic products is the increasing complexity of the products themselves. The defining feature of mechatronic products, also known as systems-of-systems, integrated systems, or mixed systems (e.g., [De Silva 2005]), is that they merge solutions from disparate engineering disciplines. As a consequence a mechatronic design process must integrate multiple engineering disciplines. Traditional design processes do not do this, because they consider each engineering discipline more or less consecutively and focus on one dominant discipline (e.g. mechanical engineering). Although the conceptual design process is well understood in the individual engineering disciplines, and some process models even provide valuable advice for the design of mechatronic systems (e.g., [De Silva 2005], [VDI 2206]), well-established integrated approaches for mechatronic design processes are still missing in practice. The situation is similar for the various kinds of computer-aided systems (CAx-systems) used by highly skilled engineers of different disciplines [Vajna 2009]. There is a critical lack of tools supporting the interdisciplinary aspects of the development process of mechatronic products, especially in the conceptual design phase. These deficiencies make it difficult to overview the interdependencies of the involved engineering disciplines [Aberdeen 2008]. System-level models can remedy this unsatisfactory situation and allow for a holistic view on complex mechatronic systems. The graphical modelling language SysML (System Modelling Language) offers the possibility of developing useful system-level models [OMG 2008]. SysML allows engineers to express the requirements, structure, and behaviour of systems using standardized images which, however, are not executable without additional software-tools and corresponding interfaces.
This article discusses how SysML may advance system-level models necessary for modelling complex mechatronic systems, and is structured as follows: The subsequent section is dedicated to system-level models and their significance in modelling complex technical systems. Next, an introduction into SysML and an overview of related work are given. An illustrative example of a SysML model of a washing machine is presented in the forth section. A conclusion summarizes the main aspects of this article and addresses future activities.

2. System-level models

2.1 Rationale for system-level models

Several collaborative projects with industrial partners reveal a lack of models that describe features as the structure and behaviour of complex technical products in a clear and consistent way. The goal of

system-level modelling is to represent the overall system in a more comprehensible way in order to remedy this unsatisfactory situation. A number of reasons for developing system-level models are listed in [Aberdeen 2008], among them the following three: (i) to notify changes to other disciplines, (ii) to allocate design requirements to specific systems, sub-systems, and components, and (iii) to validate the system behaviour digitally by simulating integrated mechanical, electrical, and software components.
Furthermore, [Aberdeen 2008] mentions the lack of “design tools that integrate the design data of all the elements that make up the product” and the “inability to understand the impact of design changes across the disciplines”. System-level models are used to extract the main characteristics and relationships of a system with the aim of showing its requirements, structure, and behaviour and to allow a holistic view of the (overall) system under consideration.

2.2 Requirements of system-level models

Due to the increasing complexity of modern technical systems, it becomes more and more difficult even for trained engineers to master the inherent relationships and dependencies of and between complex systems. System-level models allow the description of the most important sub-systems and their relationships to and dependencies on the overall system. Systems often consist of a base-system which can be extended by optional sub-systems. System-level models should support engineers in considering both (a) base system variants and (b) sub-system options.
The basic principle of system-level modelling is to model only those data which are essential for the overall system or which are important across engineering disciplines. However, it is necessary to specify principles to decide which relationships and dependencies are to be modelled on the system level or on the level of the disciplines involved. Hence, it is important to ensure consistency in both directions - from system-level models to discipline specific models and vice versa.

2.3 Importance of system-level models to complex mechatronic systems

2.3.1 System-level models are neutral

Mechatronic systems combine elements from mechanical engineering, electronic engineering, computer engineering and control engineering into one integrated system. Each engineering discipline uses special design tools which are normally not fully integrated. System-level models are a feasible method of supporting and facilitating an interdisciplinary engineering approach. Mechatronic systems are often dominated by one engineering discipline (e.g. mechanical engineering in machine tools). System-level models promote equal treatment of all engineering disciplines involved during product development and project execution. Especially for the “non-material” components of mechatronic systems (e.g., software components) this is an important improvement.

2.3.2 Mechatronic systems are systems-of-systems

To understand complex mechatronic systems, it is convenient or even necessary to decompose them into separate sub-systems. Hence, mechatronic systems may be understood as systems-of-systems. The boundaries of these sub-systems have to be chosen such that the interfaces become clear to all engineering disciplines involved.
System-level models therefore illustrate the dependencies between the sub-systems which themselves may consist of solutions from different engineering disciplines, and provide a multi-level view of the overall system under consideration. To model these dependencies, the definition of stable interfaces between the individual systems is very important. Interfaces must support the tracking and communication not only of single values but also of more complex data types such as characteristic curves and key figures. SysML offers various ports for modelling interfaces between blocks.
Each engineering discipline involved in a mechatronic system has its own knowledge base. However, for the understanding of the overall system only the intersection (overlap) of these knowledge bases is important. System-level models offer the possibility of generating a knowledge base across disciplines and therefore support collaboration and communication between engineering disciplines on a reduced level of detail and hence on a higher level of abstraction. A further step in this direction is to consider

the design rationale, i.e., to list the main decisions made during the design process and the reasons for them. Again, it is important to identify and model those data which are necessary to understand the overall system.

2.4 How can system-level models support engineering processes?

Especially in the early phases of product development, system-level models are very helpful to show and document the main dependencies between the requirements, structure, and behaviour of the overall system. When development starts, the main requirements of the system are usually known. Depending on the maturity of the product, the abstraction level of the system-level model can decrease. Collaboration and communication during project execution can certainly be simplified by the application of already existing system-level models. Thus, engineers obtain a complete overview of the most relevant system data from the very beginning.

The system-level model should also support calculations and simulations especially in the later phases of product development such as detail design. The system-level model approach is based on integrating the existing discipline-specific development and simulation tools. This offers the advantage for the design-engineers to use familiar tools on the one hand, but implies the need for interfaces between various software tools on the other hand.

System-level models for products with high lot sizes and few variants can be developed in more detail than system-level models for products with many options for customization. For products with a very small lot size, system-level models with a higher level of abstraction can be used to support collaboration and communication.

Since approaches employing system-level models are not restricted to individual engineering disciplines, different business areas (product management, mechanical, electrical, and software engineering, sales and distribution management, etc.) may also be supported in their work. System-level models can be used to help to evaluate the properties of the system under consideration in order to ensure that the final system meets the design requirements.

The present paper is intended to examine the ability of SysML to describe system level models.

3. SysML

3.1 Introduction to SysML

SysML is a graphical modelling language based on UML (Unified Modelling Language) and was adopted by the OMG (Object Management Group) in 2006. Since November 2008, version 1.1 has been available [OMG 2008]. The SysML taxonomy is presented in Figure 1.

The word system is used in a general meaning and may represent a consumer product or an industrial product. As a rule, such systems consist of further systems called sub-systems or elements. By means of the various diagram types shown above, SysML allows modelling the requirements, structure, and behaviour of a system. SysML provides *allocations* to generate relationships between the different *model elements* (e.g. *diagrams, blocks, parts, activities*). Thus, the relationships and dependencies between the requirements, structure and behaviour of a technical system can also be modelled. SysML includes the following types of standardized diagrams (Figure 1). Additional software tools and corresponding interfaces are required to render them executable.

- *Requirement diagrams* are used to model the *requirements* and their relationships, i.e. the requirement structure.
- *Activity diagrams* allow modelling the chronological order of *activities.* Additionally, the input and output data of activities are visualized.
- *Sequence diagrams* are used to model the flow of control between *actors* and systems (*blocks*) or between *parts* of a system.
- *State machine diagrams* describe the *states* of a system and their changes in response to *events*.
- *Use case diagrams* describe the usage of a system by its *actors* (environment), irrespectively of any technical realization (solution) of the system's functions.

- *Block definition diagrams* represent the structure of the system and provide an option for modelling the system hierarchy. The system consists of various *blocks* (modular units of the system), which are interconnected by *connectors,* which specify relationships between model elements, both within and across the boundary of the system.
- *Internal block diagrams* describe the internal structure of one *block* in terms of its *parts*, *ports* (input and output interfaces), and *connectors*.
- *Parametric diagrams* represent physical aspects of the system, using *constraint blocks* in a specialized variant of an *internal block diagram*. *Constraint blocks* include a constraint (mathematical equation) and the parameters the constraint requires.
- *Package diagrams* are used to organize the model (e.g. for visualisation and evaluation purposes).

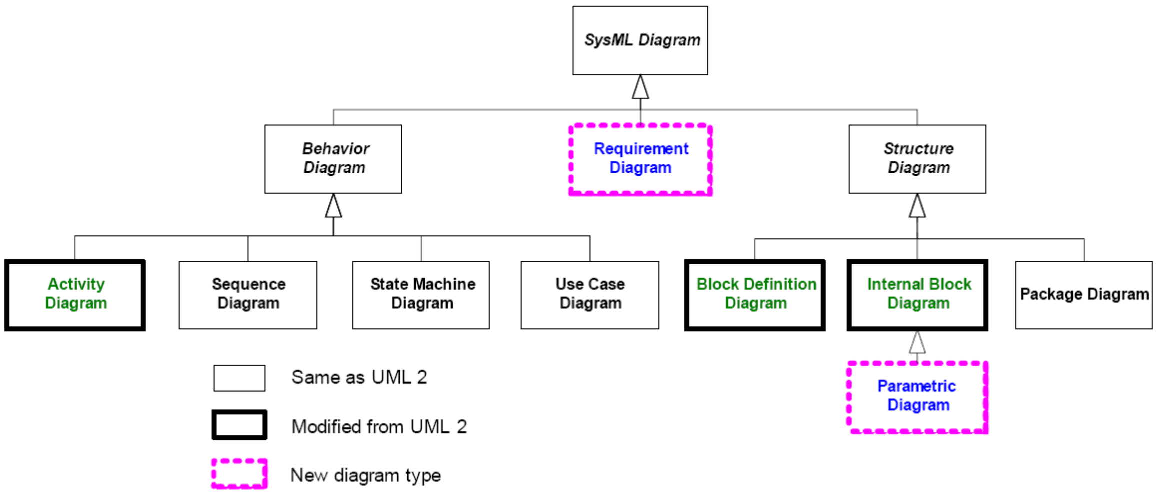

Figure 1. Diagram of the SysML taxonomy [OMG 2008]

For more information about SysML see for example [OMG 2008] and [Weilkiens 2006].

3.2 Related Works

A UML profile (based on SysML) for Modelica, named ModelicaML, was presented in [Pop 2007-1] and [Pop 2007-2]. It extends the diagram types (requirement, structure, behaviour) of SysML by two new ones: The authors added the *simulation diagram* type and extended the group of behaviour diagrams by an *equation diagram*. *Simulation diagrams* model and document simulation parameters and results, and *equation diagrams* allow the modelling of more complex equations and logical operations (e.g., if, for, and when). In [Pop 2007-2], the implementation of the ModelicaML UML profile for Modelica in Eclipse was presented. The Eclipse open source framework is used for creating extensible integrated development environments.
In [Peak 2007], a means of executing SysML parametric models with the help of *Composable Objects* (COBs) was shown. The *COB* representation is based on object and constraint graph concepts to support modularity of *COBs* and multi-directional capabilities. *COBs* consist of both lexical and graphical formulations, and can also be represented using SysML *constraint blocks* and *parametric diagrams*. The models become executable with the XaiTools toolkit which enables links, for example to Mathematica, Abaqus and Ansys.
In [Johnson 2008], the use of triple graph grammars (TGGs) was introduced in order to specify transformations between Modelica and SysML models. SysML was used to model the structure and requirements of a system, whereas Modelica was employed to handle differential algebraic equations (DAE-systems). The authors established a bidirectional mapping between SysML and Modelica.

4. SysML model of a washing machine

The following SysML model of a washing machine (WM) is used to explain the meaning and use of the various diagram types. From this example, general observations are derived that are also valid for the modelling of other systems.

4.1 Requirement Diagram

The “price range” is the main requirement appearing in the *requirement diagram* of the “washing machine”, as shown in Figure 2. Other requirements, such as “specifications” are derived from the requirement “price range” via a *deriveReqt* relationship. According to [OMG 2008] a *deriveReqt* relationship is a dependency between two requirements in which a “client requirement” can be derived from the “supplier requirement”. The requirement “specifications” is connected to the *block* “controller” via a *satisfy* relationship, thus representing a dependency between a requirement and a model element that is to fulfil the requirement. The relationship *trace* between requirements provides a general-purpose relationship between a requirement and any other model element. For example, in Figure 2 a dependency between the requirements “type of constructions” and “specification” is modelled by a *trace* relationship. To illustrate the hierarchy of requirements, *namespace containment* mechanisms are used (indicated by crosslines in Figure 2). This relationship enables a complex requirement to be decomposed into its constituent child requirements, [OMG 2008]. Figure 2 shows the hierarchical structure of the requirement “type of construction” and the related child requirements “door alignment”, “dimensions” and “dryer”.

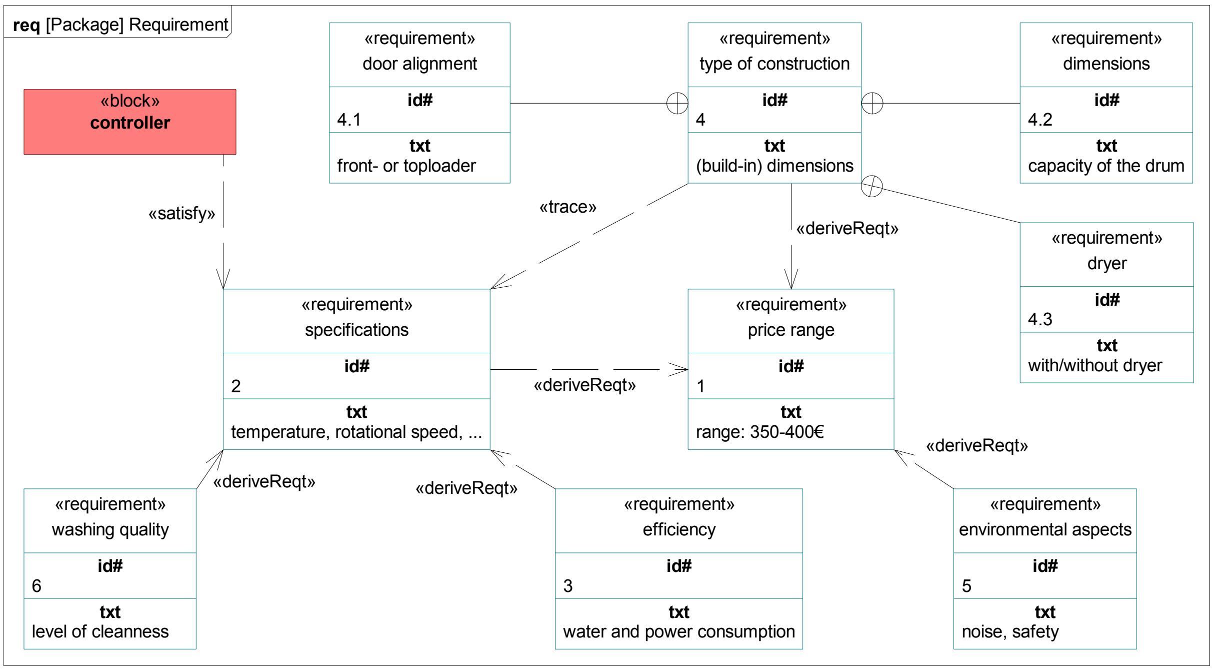

Figure 2. Requirement Diagram of the washing machine

4.2 Block Definition Diagram

The *block definition diagram* is used to model the system structure of the “washing machine” and shows the definition of and relationships between different *blocks* (see Figure 3). The washing machine comprises five main *blocks:* “frame”, “controller”, “drivetrain”, “washing parts” and “water circulation and heating”.

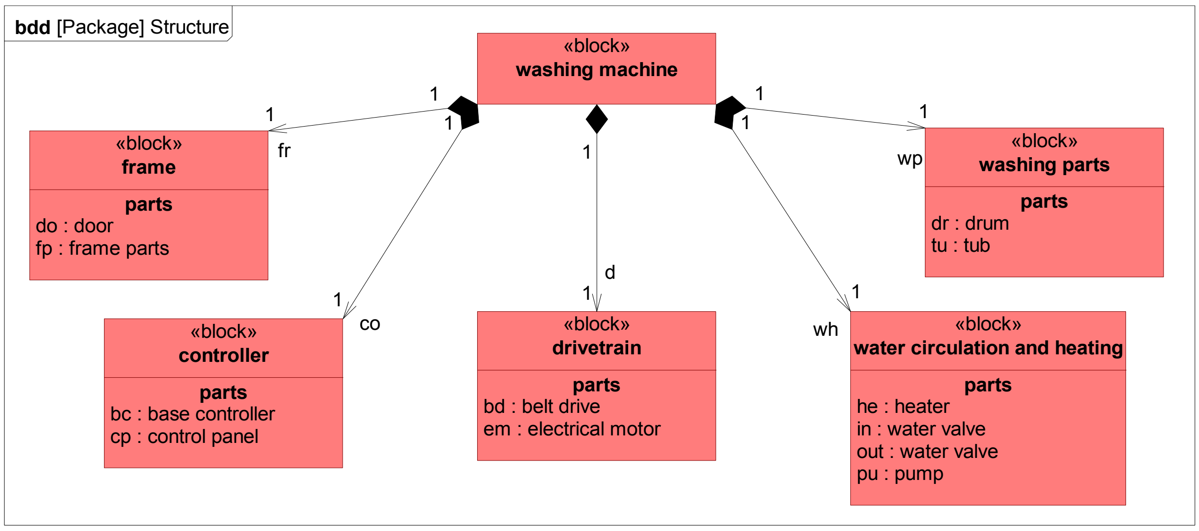

Figure 3. Block Definition Diagram of the washing machine

These *blocks* can also be understood as sub-systems of a super-ordinate system. The *parts* of the *blocks* are described in more detail in the following section.

4.3 Internal Block Diagram

The *internal block diagram* in Figure 4 shows the internals of the *block* "washing machine". This diagram describes the flow of e.g. "Information", as well as the relations between the different parts.

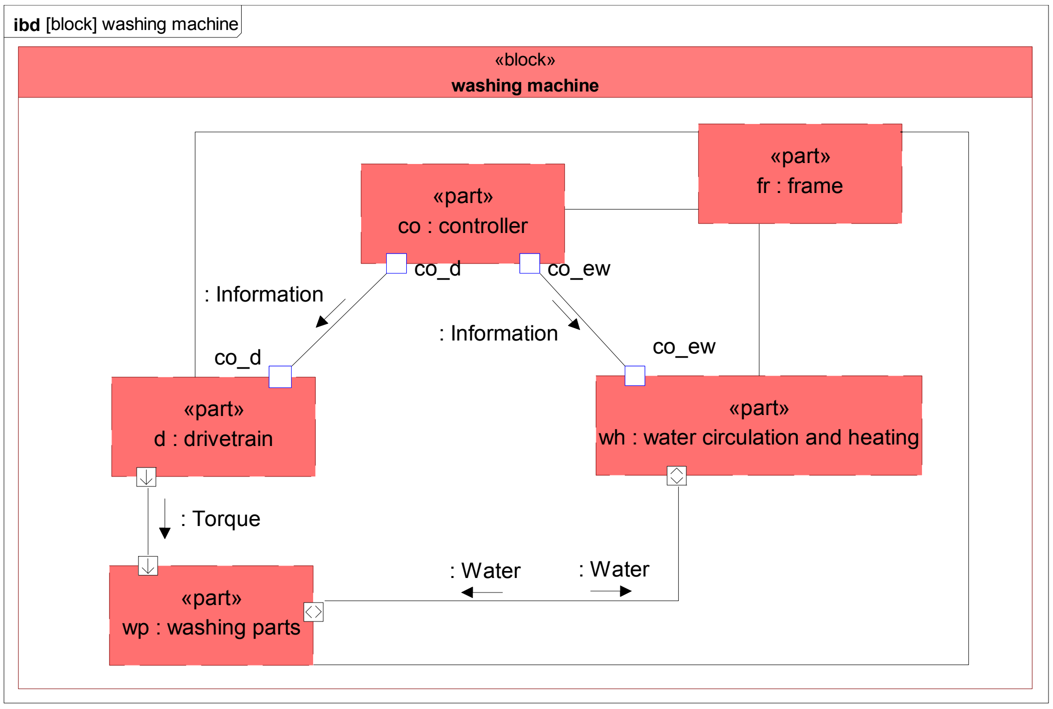

Figure 4. Internal Block Diagram of the block "washing machine"

4.4 Parametric Diagram

The *block definition diagram* in Figure 5 shows the relationships between various *constraint blocks* and the *block* "washing machine". A *constraint block* represents mathematical expressions (so called *constraints*), such as {E=P*t}, and the parameters used, such as E, P, and t.

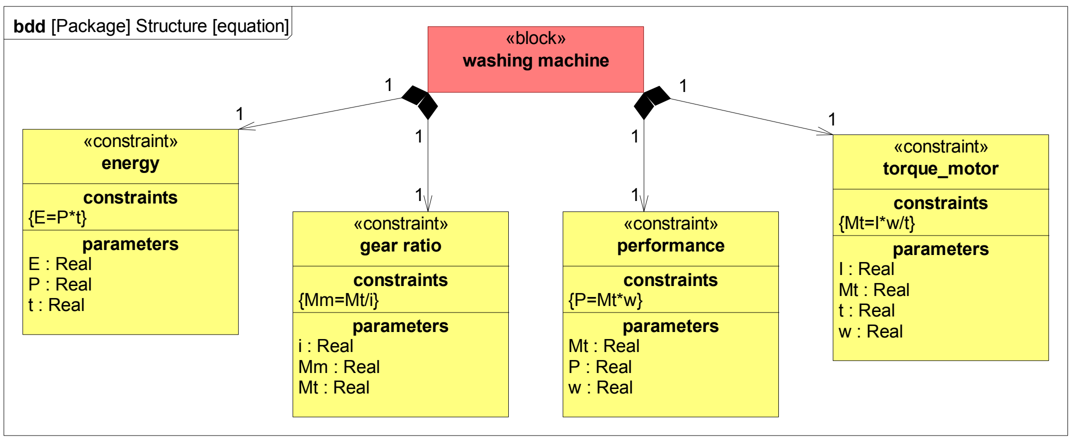

Figure 5. Block Definition Diagram with constraint blocks

The *parametric diagram* in Figure 6, derived from the *block definition diagram* in Figure 5, shows the relations for the evaluation of the motor torque. The parameters in the diagram are non-directional and can also be connected, for example with *blocks (parts)* and *notes*.
The diagrams shown in Figures 5 and 6 are merely representations of mathematical relations according to SysML, which are per se not executable.

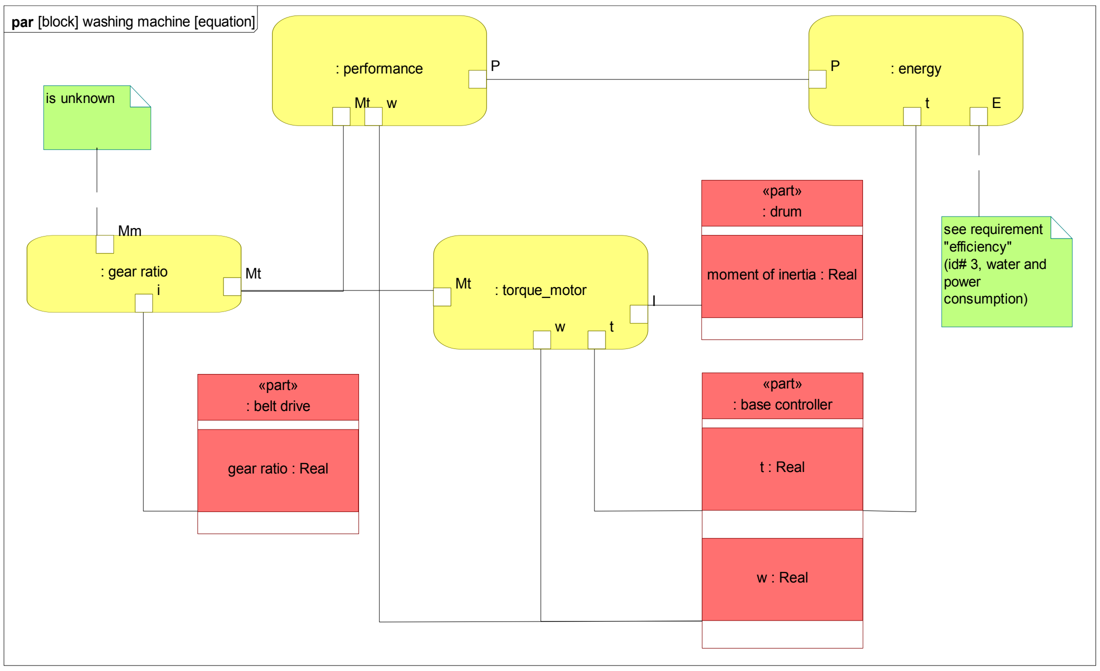

Figure 6. Parametric diagram for the evaluation of the motor torque (Mt)

The following four behaviour diagrams provide a detailed technical description of the system. Such specifications are currently not standard in the engineering disciplines considered. Table 1 summarizes the different aspects that the four diagrams describe by listing the main questions they address.

Table 1. Behaviour diagrams in SysML

Diagram type	Main questions
Use Case Diagram	Which use cases must be covered?
Activity Diagram	What has to happen in which order? Which data are input, which data are output?
Sequence Diagram	Which entity will call another entity when and how?
State Machine Diagram	How must objects react to events?

4.5 Use Case Diagram

Figure 7 illustrates the use of the “washing machine” by its *actor* (user) and some possible *use cases*, but without any technical details. For a correct understanding it is important to read the diagram in the direction of the arrows. This means, for example that “Washing laundry” is a kind of “Using WM”. (WM stands for washing machine)

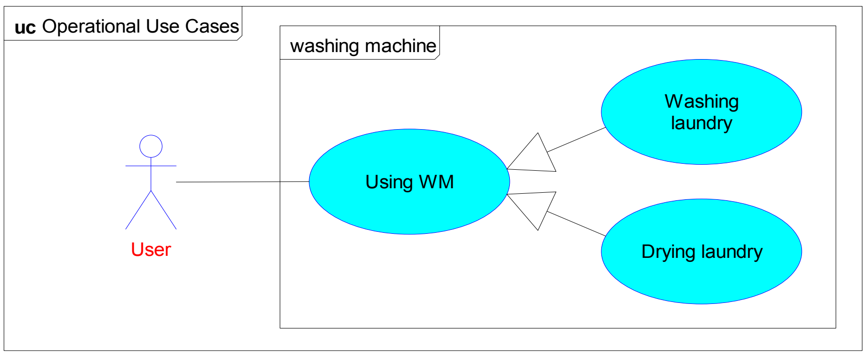

Figure 7. Use Case Diagram of “Using WM”

Use case diagrams offer a new approach to developing modern technical systems. They are common in software engineering but rarely used in other engineering disciplines.

4.6 Activity Diagram

The *activity diagram* in Figure 8 shows the chronological order of the activities necessary for “Doing laundry”.

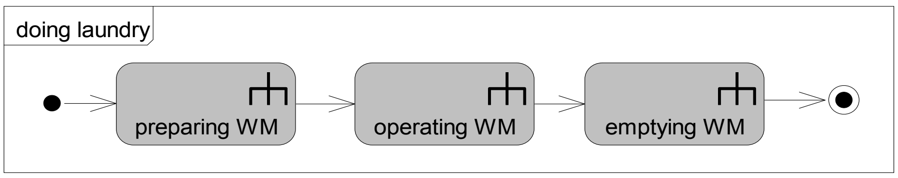

Figure 8. Activity Diagram of “Doing laundry”

4.7 Sequence Diagram

The *sequence diagram* in Figure 9 shows a more detailed description of the use case “Washing laundry”. It includes the interactions between *parts* (form left to right) and the chronological order (top-down) of *messages* (communication between the interacting entities).

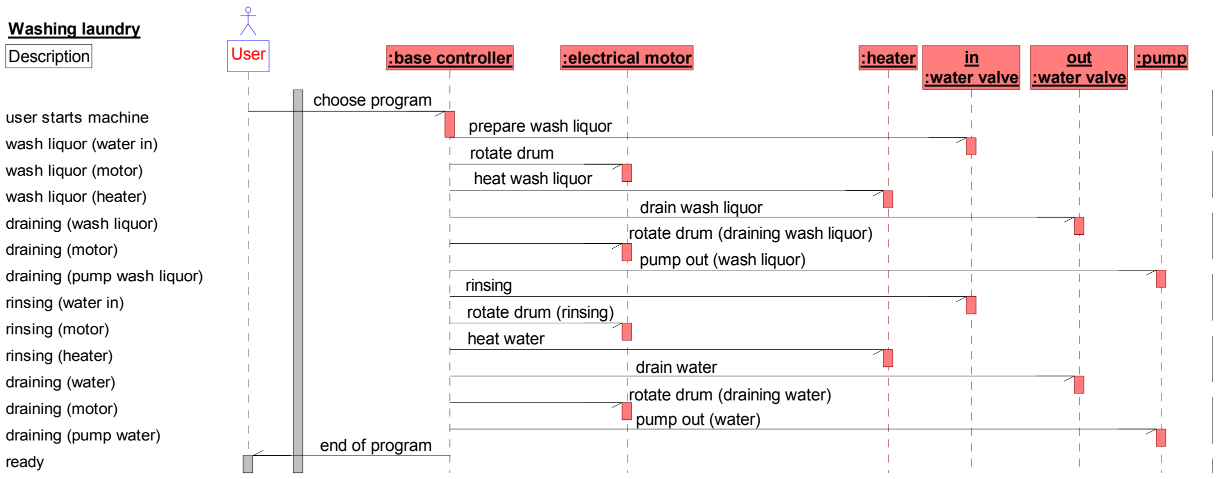

Figure 9. Sequence Diagram of “Washing laundry”

4.8 State Machine Diagram

The possible *states* of the "pump" are shown in Figure 10. The *state machine diagram* describes the relations between events such as "rotational speed" and "emergency stops" and the behaviour of the *part* ("pump").

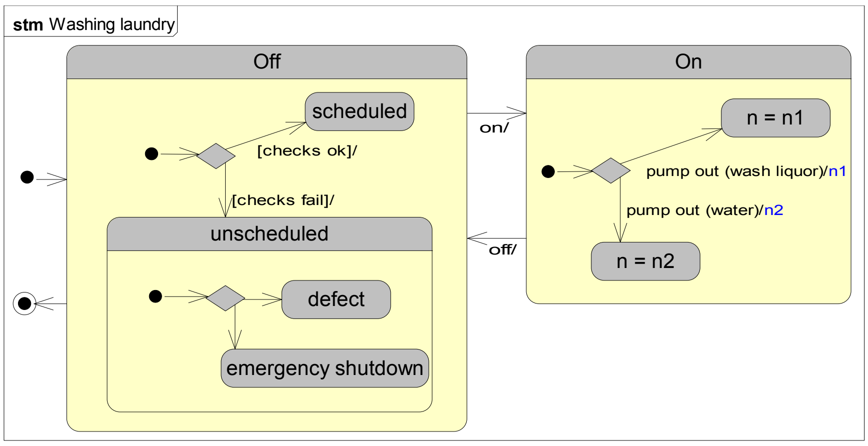

Figure 10. State Machine Diagram of the "pump"

4.9 Potential of SysML

The approach presented in this article shows that SysML provides a possibility to create discipline-neutral system-level models. The standardized diagram types of SysML allow engineers to model not only the requirements, structure, and behaviour of a system, but also the relationships and dependencies between them, as illustrated in Table 2. Depending on the views of the system under consideration, the relevance of the different SysML-diagrams may vary. In total, SysML seems to be a proper means of modelling complex mechatronic systems and supports an interdisciplinary engineering approach. Although SysML diagrams cannot directly be used as executable simulation tools, this becomes possible with additional software.

5. Conclusion and further activities

We have outlined the significance of system-level modelling especially in the design of multi-disciplinary and multi-level systems. Furthermore, we have shown how SysML can advance the modelling of complex (mechatronic) systems. The potential of SysML has been demonstrated by means of a simple example of a washing machine. Our next step is to apply this approach to a product development project in cooperation with an industrial partner. This will provide the opportunity to gather practical experience, particularly in the early phases of product development.

Table 2. Diagram types in SysML

	Modelling of		
Diagram type	requirements	behaviour	structure
Requirement Diagram	X		
Activity Diagram		X	
Sequence Diagram		X	
State Machine Diagram		X	
Use Case Diagram		X	
Block Definition Diagram			X
Internal Block Diagram			X
Parametric Diagram			X
Package Diagram			X

Acknowledgments

This work was kindly supported by the Austrian Center of Competence in Mechatronics (ACCM), a K2-Center of the COMET/K2 program, which is aided by funds of the Austrian Republic and the Provincial Government of Upper Austria. The authors thank all involved partners for their support.

References

Aberdeen Group, "System Design: New Product Development for Mechatronics", 2008.

De Silva, C. W., "Mechatronics – an integrated approach", CRC Press Boca Raton, London, New York, Washington DC, 2005.

OMG, "Systems Modeling Language V1.1", OMG Available Specification, 2008

Peak, R. S., Burkhart, R. M., Friedenthal, S. A., Wilson, M. W., Bajaj, M., Kim, I., "Simulation-Based Design Using SysML" Part 1 and Part 2, INCOSE Intl. Symposium, San Diego, USA, 2007.

Pop, A., Akhvlediani, D., Fritzson, P., "Towards Unified System Modeling with the ModelicaML UML Profile", 1st International Workshop on Equation-Based Object-Oriented Languages and Tools, Berlin, Germany, 2007.

Pop A., Băluţă V., Fritzson, P., "Eclipse Support for Design and Requirements Engineering Based on ModelicaML", 48th Scandinavian Conference on Simulation and Modeling, Göteborg, Sweden, 2007.

Johnson, T., Paredis, C. J. J., Burkhart, R., "Integrating Models and Simulations of Continuous Dynamics into SysML", 6th International Modelica Conference, Bielefeld, Germany, 2008.

Vajna, S., Weber, C., Bley, H., Zeman, K., Hehenberger, P., "CAx für Ingenieure: Eine praxisbezogene Einführung", Springer Verlag Berlin Heidelberg Germany, 2009.

VDI-Richtlinie, VDI 2206, "Entwicklungsmethodik für mechatronische Systeme", Beuth Verlag Germany, 2003.

Weilkiens, T., "Systems Engineering mit SysML / UML: Modellierung, Analyse, Design", Dpunkt Verlag Heidelberg Germany, 2006.

Martin Follmer
University Assistant
Johannes Kepler University, Institute of Computer-Aided Methods in Mechanical Engineering
Altenberger Straße 69, 4040 Linz, Austria
Telephone: ++43/732 2468 6555
Telefax: ++43/732 2468 6542
Email: martin.follmer@jku.at
URL: http://came.mechatronik.uni-linz.ac.at