Mechanical concept development using principle solution models

Mechanical concept development using principle solution models

Thilo Zerwas[1], Georg Jacobs[1], Kathrin Spütz[1], Gregor Höpfner[1], Imke Drave[2], Joerg Berroth[1], Christian Guist[3], Christian Konrad[1], Bernhard Rumpe[2] and Jens Kohl[3]

[1] Institute for Machine Elements and Systems Engineering,
RWTH Aachen University, Templergraben 55, 52062 Aachen, Germany

[2] Software Engineering, RWTH Aachen University,
Templergraben 55, 52062 Aachen, Germany

[3] BMW Group AG, Petuelring 130, 80788 München, Germany

thilo.zerwas@imse.rwth-aachen.de

Abstract. Cyber-physical systems (CPS) are characterized by the interaction of mechanical, electronic and information technology subsystems. Model-Based Systems Engineering (MBSE) is an approach for the cross-domain development of CPS and requires compatible methods and models for a function-oriented collaboration of the domains. However, the mechanics operate mainly component-oriented and thus cannot participate in the function-oriented development process. We propose a new modeling method that allows mechanics to develop a consistent and function-oriented system model from requirements and functions to principle solutions. The principle solution formally specifies the physical effect, active surfaces and material through which a function is fulfilled. Since principle solutions are consistently parameter-based, they can be calculated by linked external models and checked against requirements. This enables to examine possible solutions for functions at an early stage without having to develop components. Since principle solutions consist of recurring elements, we also propose a modeling concept for a solution libraryso that proven models can be efficiently reused and the modeling effort is reduced. Modeling method, test and solution library are explained using the example of an electric water pump of an automotive cooling circuit.

1. Introduction

The objective of product development is to transform the customer's requirements and wishes as efficiently as possible into a functionally fulfilling product. In order to meet today's customer requirements, more and more cyber-physical systems (CPS) are being developed. These are characterized by the interaction of mechanical, electronic and information technology subsystems [1,2]. Model-Based Systems Engineering (MBSE) is an approach for the cross-domain development of CPS. It is based on a common, parameter-based system architecture, which structures the system to be developed in a function-oriented way. In order to successfully develop CPS in the context of MBSE, the methods and models of all participating domains must be compatible and suitable for collaboration [1,3,4]. While the other domains operate function-oriented, mechanical engineers usually develop component-oriented: Requirements are translated directly into components without any detours. Since

the development of components is very time-consuming compared to the small-step procedure of the other domains, the mechanics have to invest a lot of effort in order to show first development results. Thus, parallel and agile developing is not possible. Instead, the mechanics must also start to develop function-oriented so that it can collaborate adequately with the other domains.
For this purpose, the mechanics must be able to derive a functional architecture from its requirements that is solution- and domain-neutral and can thus structure the development in a function-oriented way. A function can often be technically realized with different solutions from different domains. Therefore it is important for the mechanics to be able to define their possible solutions with little effort and test them as early as possible against requirements. One possible approach from design methodology are principle solutions. These describe the essential properties of a possible solution by specifying a physical effect, effective surfaces and material [5].
However, up to now there is no suitable way to describe principle solutions in the sense of [5] in a system architecture and test them against requirements. Hence, the contributions of this paper are

1. a modeling method for the function-oriented and continuous development of requirements, functions and principle solutions,
2. a formalization of principle solutions that allows testing against requirements, and
3. a modeling concept for a solution library for efficiently reusing principle solutions.

The paper is structured as follows: Section 2 provides an overview of the state of research concerning function-oriented development with principle solutions in the mechanical domain. Section 3 defines the research question and hypothesis of the paper at hand. Section 4 presents a method for the function-oriented and continuous modeling of requirements, functions and principle solutions, while Section 5 illustrates how principle solutions can be tested against requirements. Section 6 focuses on the efficient storage and reuse of principle solutions with a solution library before Section 7 concludes.

2. State of research

The Systems Modeling Language (SysML) has become established for modeling CPS in the context of MBSE. SysML is a language family that extends a subset of the Unified Modeling Language for the integrated representation of several domains in systems engineering [6]. For this purpose, the SysML comprises multiple language elements and diagrams for modeling behavior, structure and requirements [6]. Since these elements often result in several options for modeling the same information in the mechanics, it is important to define which SysML elements are used for which development models.

For the domain of mechanics it was noted in the introduction (cf. section 1) that principle solutions are essential for a function-oriented development of CPS, since they efficiently describe a possible solution by specifying effect, active surfaces and material without losing the functional orientation [5,7]. The basic idea is to derive possible physical effects for the realization of a function from the function itself. For this purpose, the functional architecture must have so-called elementary functions on its leaves, which describe a concrete physical relationship between the incoming and outgoing function flows (e.g. increase torque). The set of elementary functions is finite and allows the description of any overall technical function by its combinatorics. For each of these elementary functions the Koller catalog documents which physical effects are basically suitable for their realization [8]. If the selected physical effect is supplemented by active surfaces and material specifications, a principle solution is obtained. In the state of research of [5,8] this principle solution is only represented by a sketch and description. This representation does not fulfill the demands of developing CPS with SysML in the mechanics: it is neither formal or parameter-based, nor testable or efficiently reusable.

There are several research approaches to describe principle solutions or mechanical realizations of a function with SysML. One approach is the description of concepts by manual sketches [9]. Although this approach allows the integration of sketches into SysML, it is not a parameter-based description of a physical behavior that can be tested against requirements. Other approaches describe the realization

of a function by components [1,3,10]. Thus, components have to be elaborately detailed before the realization can be tested against requirements and the mechanical domain can only collaborate with a time delay. Another proposal for modeling principle solutions in the sense of [5] principle solution is provided by [11]. However, the parameters are neither linked within a principle solution nor externally. Hence, the principle solution cannot be tested directly with CAD or simulation models [12–15]. All in all, there is no integrated approach to model, test and efficiently reuse principle solutions in the sense of [5] in a function-oriented development process with SysML.

3. Research question and hypotheses

In the introduction (cf. section 1), the challenges for mechanics in the development of CPS were described, which have not yet been overcome by the current state of research (cf. section 2). Therefore, the research question of this publication is:

How can principle solutions be described in a function-oriented, testable and efficient way?

To answer this question, three research hypotheses are formulated:

1. *Principle solutions can be formalized as SysML principle solution models in an object-oriented way.*
2. *SysML principle solution models can be tested due to their formalization.*
3. *SysML principle solution models can be captured in SysML libraries and efficiently reused.*

The following three chapters each address one of the three hypotheses. In section 4 we present a method for modeling requirements, functions and principle solutions, which allows a seamless connection of the individual artifacts and parameters. Section 5 illustrates how principle solution models can be tested against requirements and thus allow early, function-oriented validation independent of components. Finally, section 6 shows how functions and principle solution models can be stored in a model-based solution library and can be efficiently reused.

4. Modeling method for function-oriented development

In this section a modeling method is presented, which allows to describe requirements, functions and principle solutions of a development process in a system model. An essential characteristic is the consistent linking of artifacts and parameters. In this way, the function orientation can also be consistently maintained for the mechanical domain and cross-domain collaboration is improved. The modeling method is based on the SysML profile SysML4FMArch, which was developed as a linguistic basis for modeling functional architectures in the mechanical domain [16].

The modeling method is explained on an automotive cooling system, which will be briefly introduced at this point. The main function of a vehicle is locomotion. For this purpose, the drive system provides mechanical energy that is conducted to the wheels and then transferred to the road. In vehicles with combustion engines, mechanical energy is obtained from the chemical energy of a fuel. For this purpose, the physical effect of combustion is used in the cylinders of the engine, resulting in a thermal expansion of the fuel-air mixture. The sudden increase in pressure accelerates the piston, which transfers the mechanical energy to the rest of the drive system. During combustion of the fuel-air mixture, not all the chemical energy is converted into mechanical energy for propulsion: A part of the energy is conducted out of the system via the escaping exhaust gas and another part is induced into the engine components as thermal energy. Since the engine is often unable to release all of this thermal energy via its outer surfaces, its internal energy and temperature rise.

The rising component temperature is becoming increasingly critical for the component material as well as the combustion process and endangers the functional reliability of the engine. For this reason, combustion engines are usually kept within an optimum temperature window by a liquid-based cooling system. A cooling medium circulates in this cooling system, which absorbs heat from the engine and releases it to the cooler. At the cooler the thermal energy is emitted to the environment. The cooling medium is accelerated by a pump so that it can absorb and release sufficient heat by convection and remains in motion despite the pressure losses. In our example system, the coolant pump is not operated mechanically but electrically and can thus be set to a certain rotational speed by

a control unit and based on the current temperatures of the engine. In addition, the engine is simplified and consists of the components cylinder head (CH) and crankcase (CC), which both require different target temperatures.

The following subchapters each describe the methodical modeling of this example system with regard to its requirements, functions and principle solutions.

4.1. Requirements

Requirements are demands and wishes that customers, manufacturers, legislators and many other stakeholders have for the product to be developed. A successful product must not only fulfill the wishes of the customer, but must also be able to be produced efficiently (e.g. factory standards) and comply with legal requirements. Therefore it makes sense to document these requirements and to continuously check their compliance during the development process [5]. Until today, requirements in many companies are still formulated in textual sentences (unstructured text) and stored as a requirements list in unlinked documents or software tools. Furthermore, it is not uncommon to try to define the requirements finally at the beginning of a project and not to change them afterwards. This endeavor is understandable, but usually not compatible with the dynamic development processes of CPS. The essential disadvantages are the often-ambiguous formulation of requirements and the missing link to the development models based on them, which are thus cut off from requirement changes.

Instead, for some time now, there have been a wide range of suggestions on how requirements can be modeled using MBSE approaches. Many of them have two common features that make a significant difference to document-based and informal requirements. One is a clear formalization and explicit description that leaves no room for interpretation. On the other hand, the requirements are expressed as far as possible by (physical) quantities with concrete values. These can be linked to other development models so that changes in requirements directly reach all relevant models.

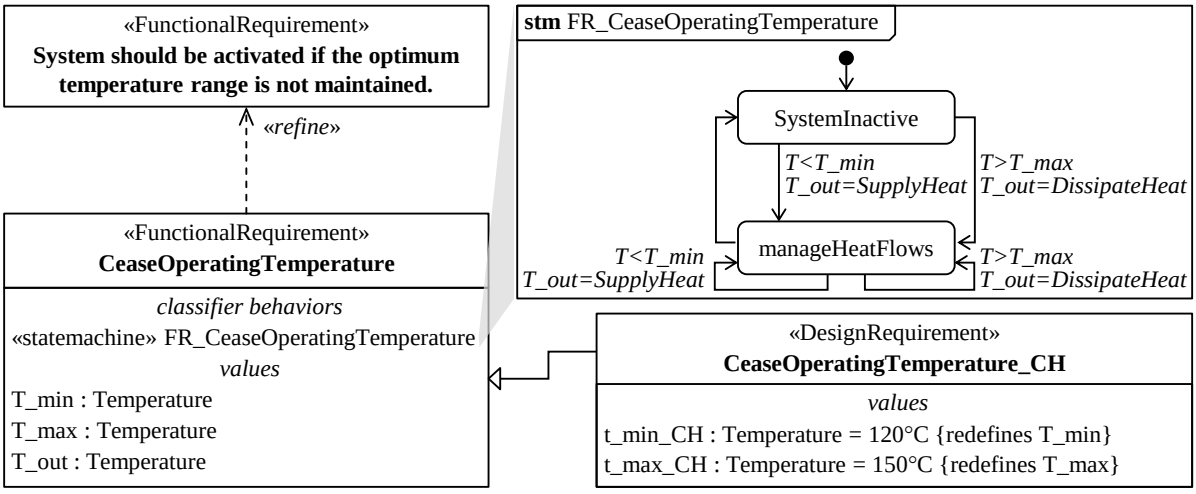

Figure 1. Requirements of the example system.

The requirements are differentiated into two categories. A «FunctionalRequirement» specifies the desired functionality of a technical system and is formalized as state machine or activity diagram. In the example system the superior behaviour of the whole cooling system is described as a state machine, which is always in one of two states: Either it is idle or active (figure 1). The transitions between the states depend on the temperature states of cylinder head and crankcase. If, e.g., the cylinder head exceeds its permissible maximum temperature, the system switches to the active state. This modeling allows for example the automated generation of test cases. This way it can be checked whether the system fulfills the prescriptive behavior and is always in the expected state [17,18].

The second category are design requirements («DesignRequirement»), which limit the value range of a parameter occurring in the system. For example, the optimum range for the operating temperature of the cylinder head can be specified as 120 °C to 130 °C. Since this temperature range is decisive for the described transition in the state machine, this design requirement is modeled as a specialization of the functional requirement. Thus, the modeling of behavior and the restriction of parameter values are clearly separated, but can be used for common statements.

4.2. Functional architecture

Functions describe the specific behavior of a product without specifying which components, effects, etc. physically implement this behavior. The concept of functions is based on the idea that physical flows enter and leave a system over a given system boundary. These *function flows* are quantified by concrete parameters values and can be categorized as flows of energy, material, and signal. Functions describe not only which function flows enter and exit, but also which *operation* takes place [5]. The decomposition of the overall function into subfunctions results in a functional architecture [7]. According to the SysML profile SysML4FMArch, the functions can be distinguished into decomposed functions («Architecture») and elementary functions («ElementaryFunction») [16]. Each elementary function describes an elementary mathematical relationship between the input and output flows [5]. Functions can be linked to the requirements they fulfill through function calls or satisfy relationships.

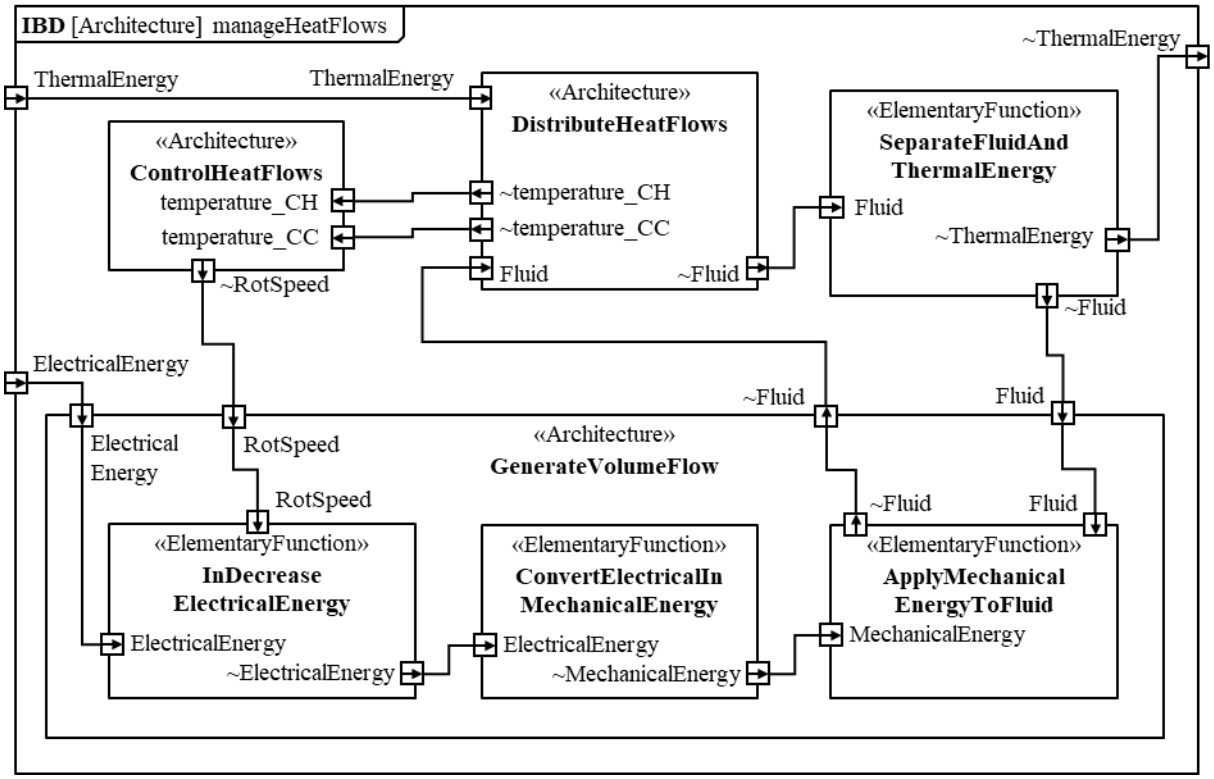

Figure 2. Functional architecture of the example system.

In the example system, the function manageHeatFlows is decomposed into four functions (figure 2). The function *generateVolumeFlow* generates a volume flow of the coolant according to the specified signal from the function *controlHeatFlows*. This volume flow is directed to the function *DistributeHeatFlows*, where heat is absorbed (at the cylinder head and crankcase). The heated coolant leaves this function and releases the thermal energy to the environment in the elementary function *SeparateFluidAndThermalEnergy*, before it circulates back into the *GenerateVolumeFlow* function. This function can be divided into three elementary functions: *InDecreaseElectricalEnergy* transforms

the incoming electrical energy so that the following function *ConvertElectricalInMechanicalEnergy* generates mechanical power according to the specified rotational speed. This mechanical power is used in the subsequent function *ApplyMechanicalEnergyToFluid* to pressurize and accelerate the coolant.

The realization of an elementary function is often possible in multiple domains [19]. For example, cooling circuits can be controlled by mechanical thermostats or software-based controllers. Often the flows of a function can give a hint, in which domain this function can be realized. At the latest by defining a physical effect, the further development of this elementary function is assigned to a certain domain. Therefore, the transition from functions to principle solutions also entails a shift from cross-domain to domain-specific development.

4.3. Principle solution models

While functions only describe the desired changes of function flows, so-called principle solutions concretize how this change is physically realized. Therefore, elementary functions and principle solution models are linked with a generalization relationship. Thus, the principle solution inherits all functional flows as ports from the elementary function and can use them to describe more precisely how the incoming flows are transformed into the outgoing flows using a physical effect, active surfaces and material. As elements of the principle solution, physical effect, active surfaces and material can be modeled with the corresponding stereotypes defined in [16] in the internal block diagram of the principle solution. Physical effects can usually be described with mathematical equations which are modeled as constraints. The parameters of such equations can depend on function flows, active surfaces and material and are linked to them accordingly. Physical quantities, which refer to function flows (e.g. volume flow, pressure, torque), are linked to the corresponding values of the incoming and outgoing function flows. Parameters that refer to geometric or material-related quantities are linked to the value properties of an «ActiveSurface» or a «Material». If the equation contains natural constants, these are modeled as value properties directly into the principle solution and linked to the constraint parameters.

The upper section of figure 3 shows the continuous modeled path from a requirement to the associated function and its principle solution for the example system. The requirement that a volume flow of 7 l/s should be generated is met by the elementary function *ApplyMechanicalEnergyToFluid*. This elementary function is now specialized by the principle solution *CentrifugalPumpWheel* that describes how mechanical energy is applied to the fluid.

The lower section of figure 3 illustrates the internal block diagram of the principal solution. Several possible physical effects for the elementary function *ApplyMechanicalEnergyToFluid* can be found in the Koller catalog [8]: conservation of momentum, friction, Boyle's law, adhesion, Coulomb’s law, Bernoulli’s principle and others. Here the physical effect *CentrifugalForce* is chosen, which is modeled as «PrincipleEffect» with all relevant parameters in the principle solution model. The active surfaces are selected to match this effect: The *PumpWheel* rotates and conveys the fluid outwards against the *Cylinder*, where the induced kinetic energy is converted into static pressure and the fluid can exit through a radial opening. These active surfaces are described by a few parameters for their geometry (e.g. outer diameter of the *PumpWheel*) and design parameters (e.g. optimal volume flow), which are essential for their physical behavior. In the example shown, it is not the material parameters of the active surfaces that are relevant for the modeled physical effect, but the material parameters of the fluid flow. Therefore, the density parameters of the incoming and outgoing fluid flow are linked to the density parameter of the physical effect. This is also the reason why *PumpWheel* and *Cylinder* do not contain any material parameters here, as it is basically enabled by [16]. In addition to the essential physical effect, the principle solution contains another «EffectElement» representing the pressure difference. Finally, the parameters of the «PrincipleEffect» are linked to the corresponding counterparts of the active surfaces and function flows.

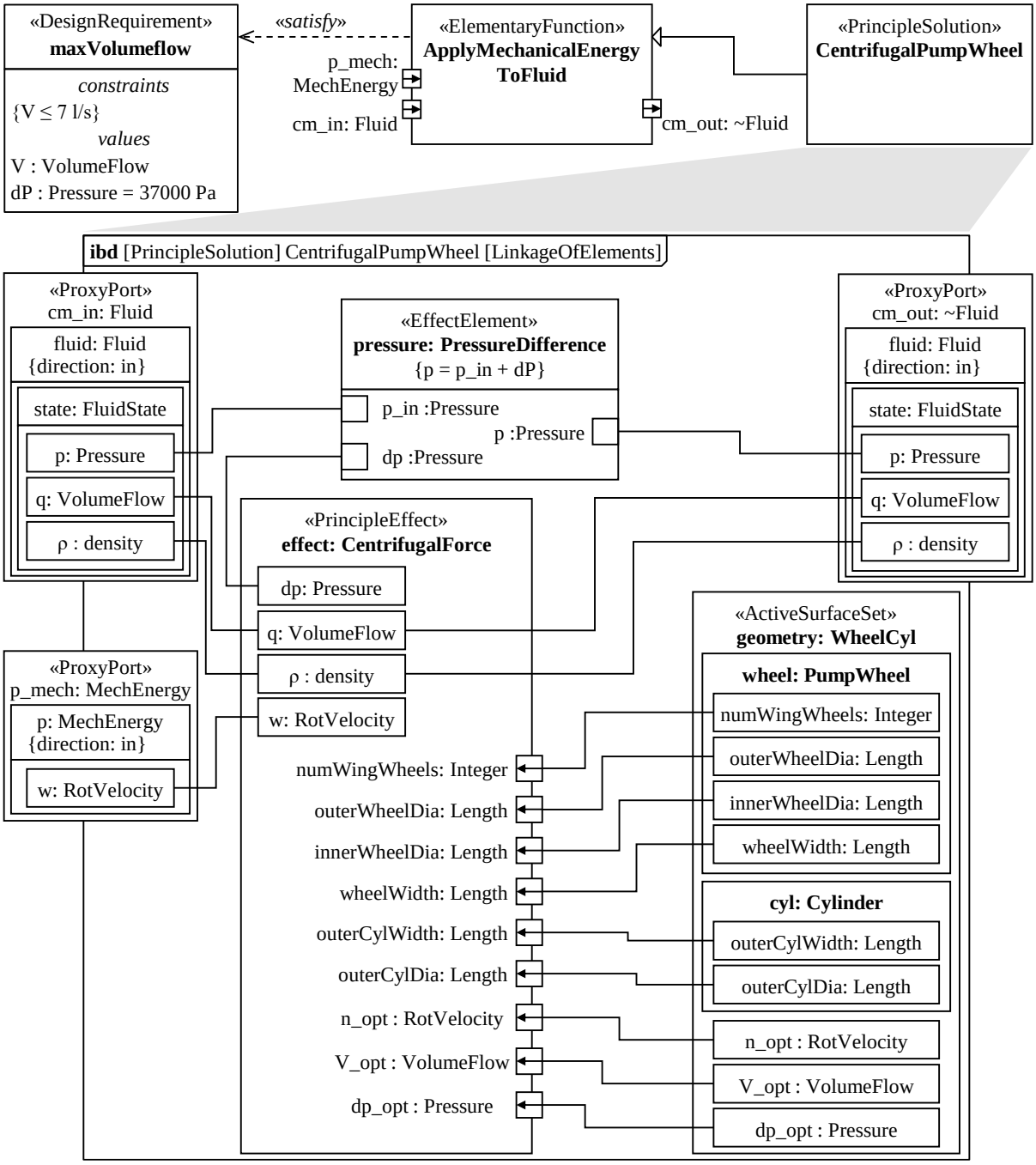

Figure 3. Requirements are satisfied by elementary functions and their principle solutions whose elements, parameters and relationships are visible in the internal block diagram.

4.4. Interim conclusion

The modeling method presented demonstrates the function-oriented development in mechanics by consistently linking requirements via functions to principle solutions. The modeling of the latter transfers the concept of Koller [5] into a formalization for SysML and, in contrast to other research approaches [20], uses a consistently parameter-based representation of effect, geometry and material [16]. This is a key advancement that enables initial performance testing of principle solutions (cf. section 5) and linkage to more detailed behavior models (cf. section 6). And this confirms the first

research hypothesis “Principle solutions can be formalized as SysML principle solution models in an object-oriented way”.

5. Test of principle solution models

The basis for testing a principle solution is the previously presented formalization (cf. section 4). Due to the parameter-based representation, physical effect and active surfaces can be linked to external models, see figure 13.7. In our example system, the *CentrifugalForce* of the principle solution *CentrifugalPumpWheel* is linked to a MATLAB model that calculates the hydrodynamic behavior of a pump wheel. Similarly, the parameters of the «ActiveSurfaces» are linked via an Excel table to the corresponding CAD models of these active surfaces. With these links the described principle solution (figure 3) can be executed by an engine that handles the execution of the single solvers. In our case, we used the integrated simulation engine of a system modeler (e.g. Cameo Systems Modeler). For this purpose, parameter values are applied externally via function flows and read from other external models (e.g. the CAD model). All values are passed to the MATLAB function for effect calculation, which returns the calculated results. Those can be further processed in the principle solution or passed on to the subsequent principle solution via a function flow.

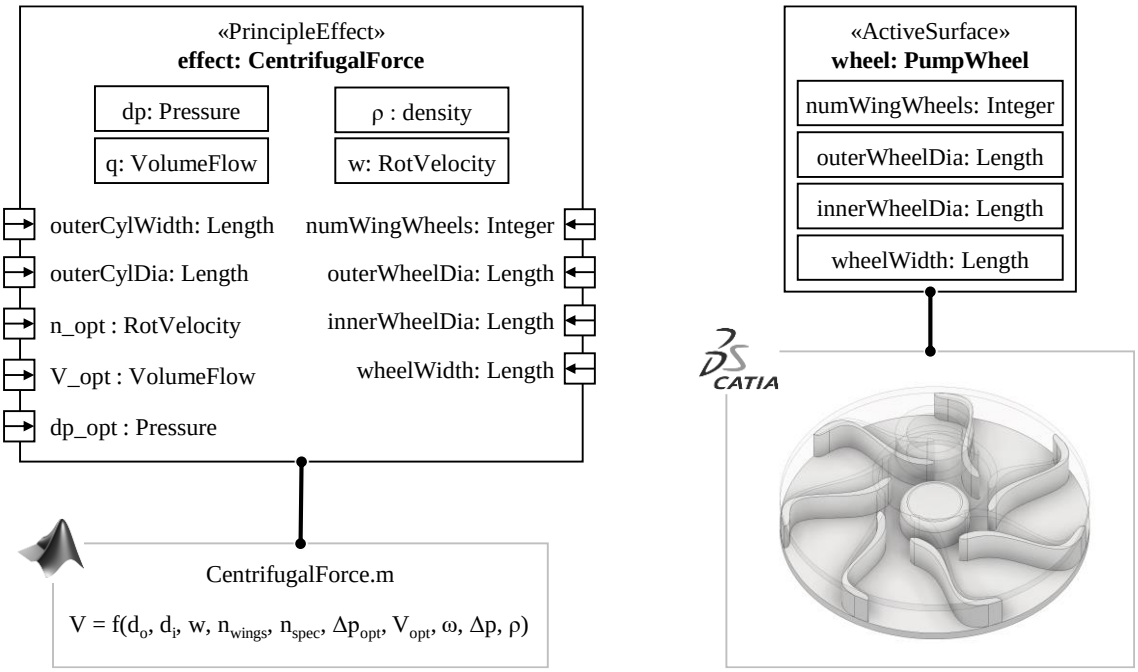

Figure 4. The «PrincipleEffect» *CentrifugalForce* is connected to an external MATLAB model and the «ActiveSurface» *PumpWheel* is connected to a corresponding CAD model.

In our example system it was required that the defined principle solution must overcome a pressure difference of 37 kPa (from the cooling circuit) and generate a volume flow of at least 0.007 m^3/s to ensure sufficient cooling of the motor. The execution of the principle solution results in a volume flow of 0.0074 m^3/s and confirms that the requirement can be met with this principle solution (table 1).

This confirms the second research hypothesis. With the described procedure, principle solutions can be tested without having to develop complex components. This means that the mechanics can collaborate with the other domains about the function realization earlier than before and rely on objective test results. In accordance with the described procedure, not only individual but also several principle solutions can be interconnected to create system tests. In addition, it is possible to define design processes as activities, for example to parameterize principle solutions for optimal functional fulfillment [21].

Table 1. Parameter values for the calculation of the principle solution *CentrifugalPumpWheel*.

Parameters of the functional flows	Parameters of the principle solution	Resulting parameters of the effect calculation
ω = 15 1/s ρ = 1000 kg/m^3 Δp = 37000 Pa	V_{opt} = 0.00606 m^3/s Δp_{opt} = 40000 Pa n_{spec} = 40,7 n_{wings} = 7 d_o = 115 mm d_i = 57,5 mm w = 32,1 mm	V = 0.0074 m^3/s

6. Solution Library

Besides the described advantages of parameter-based modelling of principle solutions, it is unmistakable that this involves a certain modelling effort. One approach to reduce the modelling effort is to reuse the models created once. Since elementary functions and physical effects are not only a finite but also a known quantity, their reuse offers high potential.

Koller has structured elementary functions and physical effects with non-formalized descriptions in a document-based catalog [5]. Since, according to Roth [22], catalogs should basically fit the method used and enable efficient use, it is necessary to develop a new concept for a digital library. Therefore, we use SysML as conceptual design language to develop the *Solution Library* and the interfaces to the system architecture.

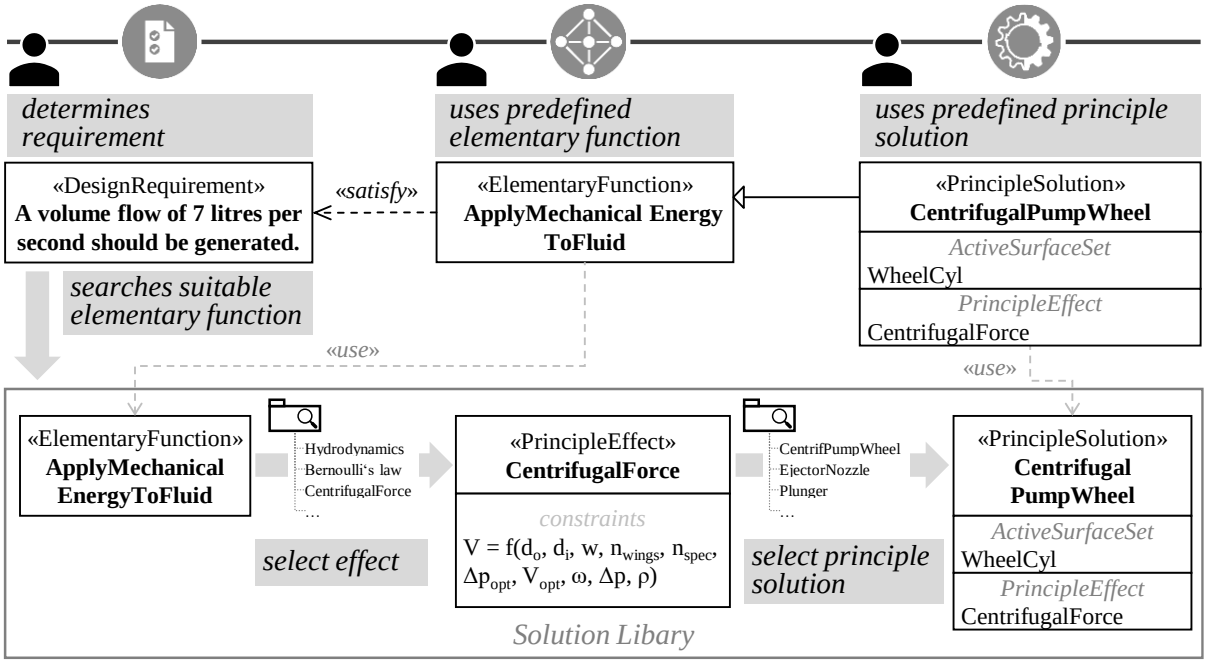

Figure 5. The solution library (lower part) supports the development process (upper part) by identifying possible physical effects and principle solutions for a selected elementary function.

The first use of the solution library takes place during the modelling of the functional architecture. Here, the user can reuse exactly those elements from the finite and predefined pool of elementary functions that he needs to fulfill the requirements (figure 5 left). By selecting the elementary function (here: *ApplyMechanicalEnergyToFluid*) the solution library is automatically filtered and only those physical effects are displayed, which can realize the chosen elementary function (figure 5 center). After a physical effect is also selected (here: *CentrifugalForce*), the set of stored principle solutions is

filtered so that only those containing the selected physical effect are listed (figure 5 right). All these listed principle solutions are suitable as technical realization of the elementary function and are able to fulfill the initial requirement (chosen here: *CentrifugalPumpWheel*).
The described approach is feasible because we have extended the well-known Koller catalog [8] by central elements: Our solution library contains not only elementary functions and physical effects (like Koller), but complete principle solutions including frequent active surfaces and materials (figure 6 left). Thus, our principle solutions can be varied not only with regard to the physical effect, but also with other active surfaces and materials.

With principle solutions, initial functional tests (cf. section 5) can be carried out, but more specialized models often have to be used in order to validate further requirements (e.g. service life or noise propagation). Therefore, as a second major enhancement compared to Koller, we store each principle solution together with suitable behavior models and workflows in a so-called solution element (figure 6 right). In this way, behavior models are clearly assigned to concrete principle solutions in our solution library and can be used efficiently for virtual behavior testing of evolving solutions based on the chosen purpose [21]. This confirms the third research hypothesis.

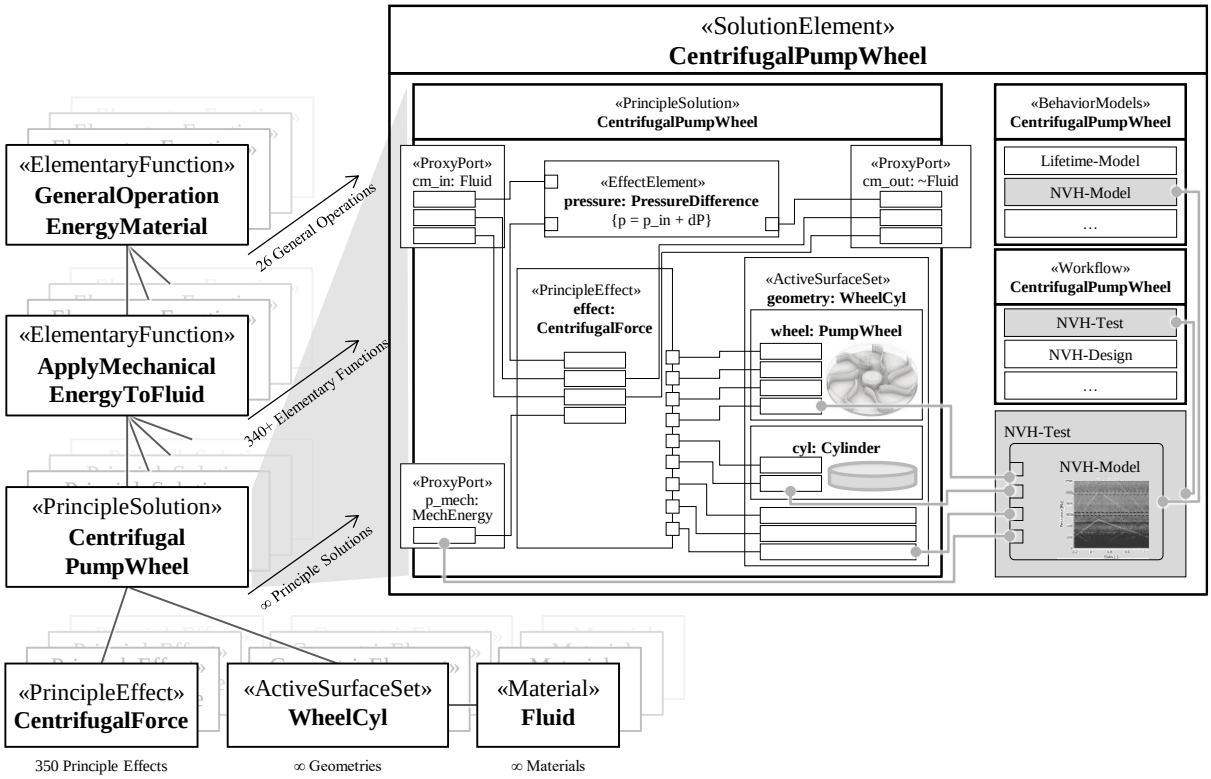

Figure 6. In addition to elementary functions, physical effects, active surfaces and materials, the solution library primarily contains a set of predefined principle solutions. These principle solutions are stored together with more detailed behavior models and workflows in a so-called solution element. Here it is shown for the example system how the principle solution *CentrifugalPumpWheel* is classified in the variant structure of the solution library.

7. Conclusion

In this paper a SysML-based modeling method for the mechanical domain in the development of CPS was presented. This method enables the mechanical domain to consistently link its requirements, functions and principle solutions with each other, so that the function orientation and thus the possibility of collaboration with other domains is maintained. The principle solution was formalized based on the concept of Koller [8] for SysML and consists of physical effect, active surfaces and

material, which are linked to each other via their parameters. This parameter linking makes the principle solution executable so that it can be calculated by external models and the results can be compared against requirements. In contrast to the Koller catalog [8], the modeling concept for the solution library contains not only elementary functions and effects, but entire principle solutions with more sophisticated behavior models and their corresponding test workflows. The solution library thus makes principle solutions efficiently reusable and paves the way for virtual behavior testing of evolving solutions based on the chosen purpose.

References

[1] Eigner M, Gilz T and Zafirov R 2012 Proposal for functional product description as part of a PLM solution in interdisciplinary product development *DESIGN 2012* ed Marjanović D, Storga M, Pavkovic N, Bojcetic N (Zagreb: UNIZAG-FSB) pp. 1667–76.
[2] Alur R 2015 *Principles of cyber-physical systems* (Cambridge: The MIT Press).
[3] Gausemeier J, Dorociak R, Pook S, Nyßen A and Terfloth A 2010 Computer-aided cross-domain modeling of mechatronic systems *Design 2010* ed Marjanović D, Storga M, Pavkovic N, Bojcetic N (Zagreb) pp. 723–32.
[4] Broy M 2010 *Cyber-Physical Systems* (Berlin Heidelberg: Springer).
[5] Koller R 1998 *Konstruktionslehre für den Maschinenbau* (Berlin: Springer).
[6] OMG OMG Systems Modeling Language Version 1.6, https://www.omg.org/spec/SysML/1.6/PDF.
[7] Feldhusen J and Grote K-H (eds.) 2013 Pahl/Beitz Konstruktionslehre Methoden und Anwendung erfolgreicher Produktentwicklung 8th ed. (Berlin: Springer).
[8] Koller R and Kastrup N 1994 *Prinziplösungen zur Konstruktion technischer Produkte* (Berlin: Springer).
[9] Moeser G, Albers A and Kumpel S 2015 Usage of Free Sketches in MBSE *2015 IEEE International Symposium on Systems Engineering (ISSE)* ed IEEE) pp. 50–55.
[10] Moeser G, Kramer C, Grundel M, Neubert M, Kümpel S and Scheithauer A et al. 2015 Fortschrittsbericht zur modellbasierten Unterstützung der Konstrukteurstätigkeit durch FAS4M *Tag des Systems Engineering* ed Schulze S-O, Muggeo C (München: Hanser) pp. 69–78.
[11] Wölkl S and Shea K 2009 A Computational Product Model for Conceptual Design Using SysML *Volume 2: 29th Computers and Information in Engineering Conference* ed ASMEDC pp. 635–45.
[12] Pasch G, Jacobs G, Höpfner G and Berroth JK 2019 *Multi-Domain Simulation for the Assessment of the NVH Behaviour of a Tractor with Hydrostatic-Mechanical Power Split Transmission*: RWTH Aachen University.
[13] Golafshan R, Jacobs G, Wegerhoff M, Drichel P and Berroth JK 2018 Investigation on the Effects of Structural Dynamics on Rolling Bearing Fault Diagnosis by Means of Multibody Simulation.
[14] Andary FS, Berroth JK and Jacobs G 2019 An Energy-Based Load Distribution Approach for the Application of Gear Mesh Stiffness on Elastic Bodies *Journal of mechanical design* **141** 95001.
[15] Berroth J, Jacobs G, Kroll T and Schelenz R 2016 Investigation on pitch system loads by means of an integral multi body simulation approach.
[16] Drave I, Rumpe B, Wortmann A, Berroth J, Höpfner G and Jacobs G et al. Modeling Mechanical Functional Architectures in SysML *MODELS '20: Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems* ed pp. 79–89.
[17] Rumpe B 2012 *Agile Modellierung mit UML* (Berlin, Heidelberg: Springer).
[18] Drave I, Hillemacher S, Greifenberg T, Kriebel S, Kusmenko E and Markthaler M et al. 2019 SMArDT modeling for automotive software testing *Softw: Pract Exper* **49** pp. 301–28.

[19] Albers A and Zingel C 2011 Interdisciplinary Systems Modeling using the Contact and Channel-Model for SYSML *Impacting society through engineering design* ed Culley SJ (Glasgow: Design Society) pp. 196–207.
[20] Munker F and Albers A 2015 SystemSketcher – Entstehung eines anwenderorientierten Ansatzes zur interdisziplinären Systemmodellierung *Tag des Systems Engineering* ed Schulze S-O, Muggeo C (München: Hanser) pp. 291–300.
[21] Höpfner G, Jacobs G, Zerwas T, Drave I, Berroth J and Guist C et al. Model-Based Design Workflows for Cyber-Physical Systems Applied to an Electric-Mechanical Coolant Pump *Antriebstechnisches Kolloquium 2021*
[22] Roth K 1994 *Konstruieren mit Konstruktionskatalogen* (Berlin: Springer).