Model-Based Design Workflows for Cyber-Physical Systems Applied to an Electric-Mechanical Coolant Pump

Model-Based Design Workflows for Cyber-Physical Systems Applied to an Electric-Mechanical Coolant Pump

Gregor Höpfner[1,4], Georg Jacobs[1], Thilo Zerwas[1], Imke Drave[2], Joerg Berroth[1], Christian Guist[3], Bernhard Rumpe[2] and Jens Kohl[3]

[1]Institute for Machine Elements and Systems Engineering, RWTH Aachen University, Schinkelstraße 10, 52062 Aachen, Germany

[2]Chair of Software Engineering, RWTH Aachen University, Ahornstraße 55, 52074 Aachen, Germany

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

[4]gregor.hoepfner@imse.rwth-aachen.de

Abstract. Cyber-Physical Systems (CPS) connect mechanics, electrics and electronics as well as software. Considering the interactions between these domains is a major challenge in product development. Model-Based Systems Engineering (MBSE) enables cross-domain system development, based on commonly understood system architectures. However, there is still a gap in using MBSE for development, especially in mechanics. Systematic evaluation of design decisions based on system architecture models is not done. The strong simulation methods predicting physical product behaviour are not well connected to the evolving system architecture. We propose a new approach for connecting simulation and design models in a System Modelling Language (SysML) system model using the example of a combustion engine's cooling circuit. The electric-mechanical coolant pump is chosen for design. The design processes of the hydrodynamic pump wheel and the volume flow controller are modelled as workflows in SysML. Here, we integrate design models for the pump wheel and the controller into the workflows. Thereby, parameters of the pump wheel and controller are calculated. Design results are automatically transferred to CAD. A workflow is created, testing the cooling circuit's behaviour against requirements, combining a Simulink control model and a Simcenter Amesim heat transfer model. The approach results in a framework for automated design and test processes.

1. Introduction

Cyber-Physical Systems (CPS) connect mechanics, electrics and electronics as well as software solutions. Considering the interactions between these different domains is one of the major challenges in modern product development. Model-Based Systems Engineering (MBSE) is a well-known approach, that enables cross-domain system development by modelling commonly understandable system architectures, e.g. for functional and product architectures, including interactions between sub functions of systems. Architecture models describe the system elements from requirement over functions to products. They may be used to link the artifacts that are created during system design. [1]

However, there is still a gap in using MBSE methods for product development, especially in the mechanical domain. MBSE often is only used up to abstract architecture definition. Design decisions

are not evaluated systematically based on the system's functional architecture, especially in the mechanical domain. Instead, mechanical development is nowadays mainly driven by experience. Mechanical engineers know about their domain and how to design mechanical products. Strong simulation methods are used for product design and testing of component's physical behavior virtually using simulation models [2–6]. CPS are not component but function oriented. The mechanical domain has not yet developed a method to test the products with regard to functional requirements. A link between functional architecture and physical behavior models is not given.

Model-based design and test procedures, i.e. executable workflows allow for automated designing and testing of elements in the functional architecture at different stages of the design process. Such model-based workflows link the physical behavior models to the system's parameters in the functional architecture. By modelling the order of design and test steps in executable workflows, the parameters describing the system's geometrical, physical, material, and logical properties can be calculated automatically and the system can be tested at any system state. This may close the gap between functional architecture modeling and physical behavior models and lead to functional testing in mechanical engineering. In order to reach for such a modeling approach, we state the following research hypotheses:

- Structured, hierarchical workflows can link simulation models efficiently and independent of a simulation tool.
- Workflows efficiently parametrize the (principle) solutions in a solution architecture.
- Workflows help to redesign and test system solutions when requirements change (engineering change requests)

We derive a new modeling approach, connecting simulation and design models in a SysML system model, modelled in Cameo Systems Modeler (CSM) [7]. Based on the hypotheses, we analyze the functional architecture of a mechatronic system. To create design and behavior prediction models we use external tools and integrate the models within executable workflows using SysML activity diagrams and CSM. We propose two kinds of Activity Diagrams for modeling design workflows and test workflows, respectively. In order to show efficiency of the modeling technique, we chose the example of an engineering change request for a combustion engine's cooling circuit.

The article is structured as follows: section 2 presents the state of research regarding architecture and workflow modelling. Section 3 describes the system of interest and its functional architecture focusing on the solutions and principle solutions relevant for this paper. Section 4 derives a modelling technique for design workflows used to find parameter values for principle solutions using SysML Activity Diagrams. Section 5 integrates these detailed design workflows into top level design processes. Section 6 adds testing workflows, testing the system behavior. Section 7 gives the application case of the workflows, an artificial engineering change request. Section 8 gives a discussion on the presented work and section 9 concludes.

2. State of Research

In MBSE, the modelling language family SysML has become widely used. SysML is a general-purpose modeling language family for systems engineering [8]. It extends the Unified Modeling Language (UML) 2.5 [9] to represent aspects of both, software and hardware elements. Currently, SysML provides modeling languages for behavior, structure, and requirement modeling. The structure modeling languages are often used for architecture description. Internal Block Diagrams (IBDs) therein can be used to describe the internal structure of a block.

There are various methods for CPS architecture development in SysML. Their aim is to link requirements, functions and product components. Some methods coming from informatics model this link focusing on software system elements and software behavior [10–15]. However, they do not provide a link between function and product for mechanical system elements. For modeling the functional architecture of mechanical systems, the methods have been extended, e.g. in [16,17]. These techniques do not integrate the product describing parameters on the detailed levels of function and product description. In [18], the authors propose a technique that uses informal sketches which describe the

mechanical solution but are not usable for executable design workflows integrating simulations, as they do not contain parameters for mathematical calculations.

The seamless approach provided in [19] allows for product description up to detailed parameter level. Modeling languages that support this methodology in MBSE are specified in [20], which also provides a SysML profile as a specific language. These languages do not yet integrate simulation workflows for product design or testing.

Activity Diagrams are a commonly used language for modeling sequential behaviors of a system [21,22], functional requirements of CPS [23–25], as well as processes and workflows [26]. In [27] the process of developing a system model in SysML is described in an Activity Diagram. In [28] the authors use SysML Activity Diagrams to execute work processes in context of grid computing in software. To the best of our knowledge, Activity Diagrams have not yet been used to describe design and test workflows and link the activities to simulation models for mechanical engineering in an executable manner.

Workflows for simulations, however, are used in mechanical engineering to model and predict the physical behavior of a product. They combine simulation models to provide reusable and automated test procedures. Examples for tools providing such workflows are given in [29,30]. Workflows are then executable but not linked to the parameters in functional architecture and often limited to specific tools.

In total, there are methods to describe the functional architecture of CPSs that include the mechanical domain in SysML. However, these do not integrate simulation workflows and use these workflows to parametrize and test system architecture.

3. System of Interest: Cooling Circuit and Electric Mechanical Coolant Pump

To illustrate modelling concepts, we use the cooling circuit of a combustion engine from automotive engineering as a running example throughout the paper. The system describes the following physical behavior: Combustion drives convert the chemical energy held by fuel into mechanical energy. Due to the physical effect of combustion to serve this function, a portion of the chemical energy is converted into thermal energy, i.e. heat, during a chemical reaction, which causes the pressure in the combustion chamber to increase. The increasing pressure acts on the surface of the engine's piston causing the piston to move. The heat is partly removed from the engine by exhaust gas. However, some amount of heat causes the engine's temperature to increase by being transferred to the engine structure. Thus, the engines temperature increases. Due to material limitations, the engine stops functioning once a maximum temperature is reached. To prevent the engine from overheating, the thermal energy has to be removed from the engine. Often, water-cooling circuits take on this task. In our example, we use a water-cooling circuit with an electric mechanical cooling medium pump.

The overall function of a water-cooling circuit is to manage the engines heat flows (*manage heat flows*) in order to set a specific temperature of the engine. This function can be achieved by combining subfunctions: the heat flows from the two engine components crank case (CC) and cylinder head (CH) are considered as inputs for the function. The heat flows are removed from the engine by two volume flows (*distribute heat*) using the physical effect of convection, the surrounding air and an internally circulating cooling medium. The air is removed to the environment. The internal cooling medium is transferred to a cooler, where the heat is also released to the air using convection (*release heat*). The cooling medium has to keep circulating in order to keep heat transfer active, i.e. a volume flow is necessary, which is generated from electrical energy (*generate volume flow*). For setting a specific engine temperature, the amount of removed heat has to be controlled. In our system, this is done by controlling the volume flow of the cooling medium. Hence, the function *generate volume flow* can get a control signal, setting the required volume flow, from a controller (*control heat flows*), which evaluates the engine components temperature signals. Figure 1, top, shows the described internal structure of *manage heat flows* in an IBD.

In order to find a solution for the function *generate volume flow*, we have to continue the functional decomposition, Figure 1, bottom. To generate a volume flow, we apply mechanical energy to the fluid (*apply fluid with mechanical energy*). Hence, we have to get mechanical energy. In an electric

mechanical coolant pump, the mechanical energy is converted from electrical energy (*convert el to mech energy*). In order to set specific volume flows based on a control signal, the electrical energy is increased or decreased before being converted (*increase/decrease electrical energy*).

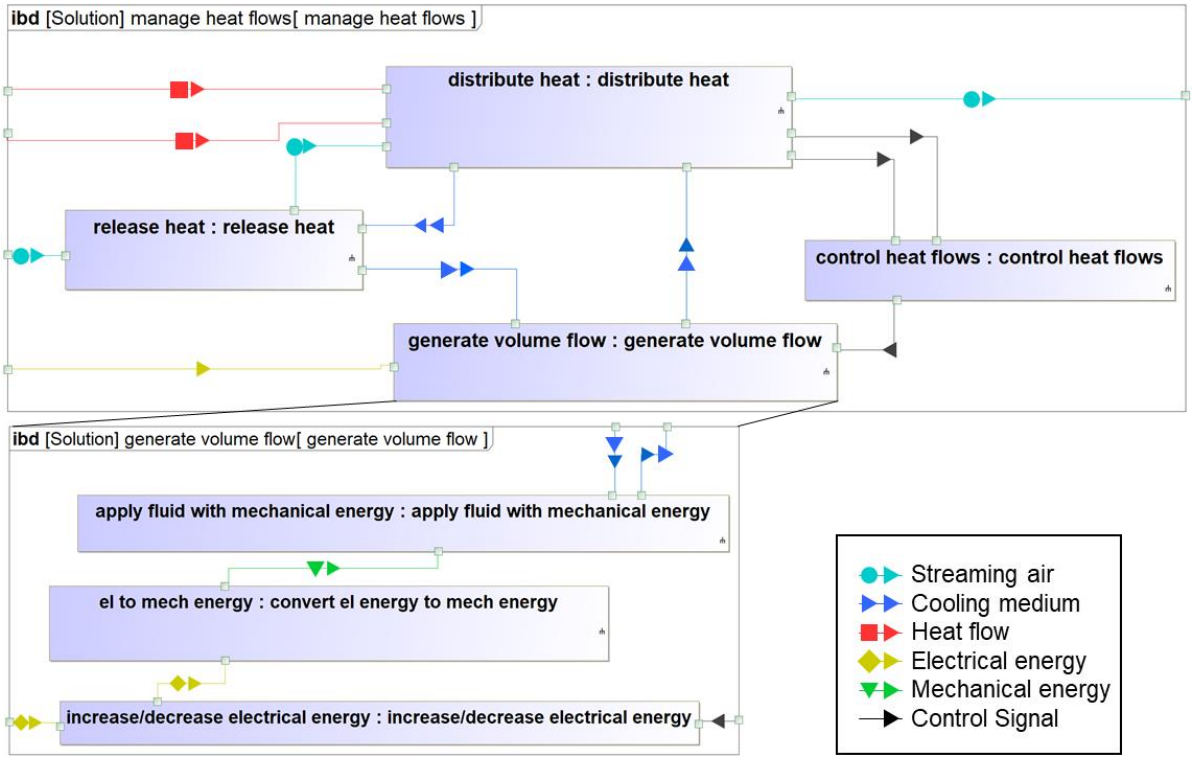

Figure 1. Functional architecture for *manage heat flows* and *generate volume flow*

The functional architecture of the cooling circuit has been modelled following the ideas of Koller, who described a system by elementary functions realized by principle solutions [19]. The described modeling follows the language profile SysML4FMArch, proposed in [20]. The functional architecture of the system has been developed in [31]. The functional decompositions of *generateVolumeFlow* and *distributeHeat* are discussed in [20] and [31], respectively. For each elementary function, a principle solution has been derived and modelled. Principle solutions realize elementary functions using physical effects or logics. Physical effects act between active surfaces from specific materials. The principle solution for *apply fluid with mechanical energy* is a hydrodynamic pump wheel. The physical effect is hydrodynamics, which accelerates an input fluid due to mechanical energy. The active surface required for this is a pump wheel. This active surface can be described by characteristic parameters, its inner and outer diameter, width, inlet and outlet angle and the number of wings in the hydrodynamic pump wheel. Based on these geometric parameters, the physical effect hydrodynamics is realized in a quantitative manner. The SysML4FMArch representation of the principle solution is shown in Figure 2. Parameters are modeled as values in SysML. From external requirements, there is a limitation for the pump wheels rotational velocity. The connection between the external requirement and parameter *rotational velocity* is modeled using a SysML-satisfy-connection.

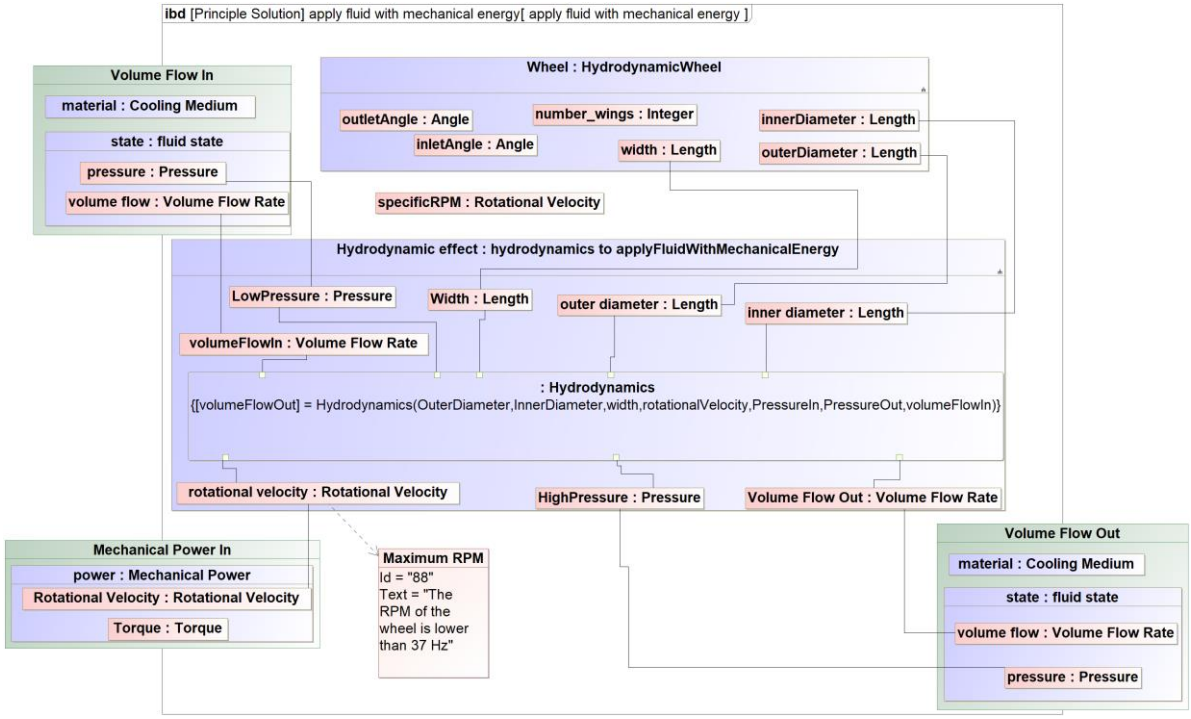

Figure 2. IBD for the principle solutions apply fluid with mechanical energy

Table 1 gives an overview about the further principle solutions, the realizing physical effects, geometric elements and relevant parameters with a description. The hydrodynamic pump wheel is driven by an electric drive providing rotational speed and torque. The volume flow is transferred to the engine and takes up an amount of heat by convection. The heat is released at the cooler by convection, reducing the pressure. To set volume flow rate based on heat, a control element is introduced, changing rotational speed of the pump.

Table 1. Selected functions, their physical effects, important parameters and parameter descriptions

Function	Solution/Principle Solution	Physical effect	Parameters	Parameter Description
distribute heat	Solution	Convection	maxVolFlow	Maximum volume flow required to satisfy the tempe-rature requirement
			minVolFlow	Minimum volume flow required to satisfy the tempe-rature requirement
release heat	Principle Solution	convection	deltaPOpt	Pressure difference of cooling medium between input and output port
control heat flows	Principle Solution	-	controlParameter	Control parameter of the P control
			controllVolumeFlow	Set point of volume flow in control
convert electrical to mechanical energy	Principle Solution	biot-savart law	RPM	Optimum rotational velocity
manage heat flows	Solution	-	RhoCooling	Density of the chosen cooling medium

The parameters describing the principle solution are given in the architecture. When investigating Figure 2, it becomes clear, that the values of the principle solution parameters, e.g. geometric parameters inner and outer diameter, have influence on how the physical effect fulfills the function, e.g. what quantity the volume flow rate is. The mechanical design process puts a lot of effort in finding the suitable parameter value set, in order to get the required behavior. In the following chapter, we describe a way of modeling the parameter estimation in the mechanical design process in design workflows and test the functional architecture with regard to requirement satisfaction.

4. Design workflow for the hydrodynamic pump wheel

Parametrization of principle solutions means finding a suitable set of values for the geometric, material and physical parameters in order to satisfy the requirements. Mechanical engineering has developed design procedures for standardized principle solutions in the past decades, which are given in norms, standards and design guidelines. Here, critical use cases are considered as input, standard parameters are declared, and their values are calculated so that the principle solution satisfies the use case. In order to find suitable parameter values for the geometric and physical parameters for the hydrodynamic pump wheel, we set up an automatized, executable workflow based on a given design guideline. In the following section, the design process of a hydrodynamic pump wheel is explained and a procedure to integrate it into the system model in an executable manner is proposed.

4.1. Design process of the hydrodynamic pump wheel

The design process of the hydrodynamic pump wheel aims to find a suitable value set for the parameters describing the hydrodynamic pump wheel's active surfaces, which are modelled in Figure 2. By fixing these geometric parameters, the hydrodynamic characteristic function is specified, i.e. the physical behaviour of the principle solution. We set up a parametrizable CAD model, which is adaptable based on the modelled geometric parameters.**Fehler! Verweisquelle konnte nicht gefunden werden.**

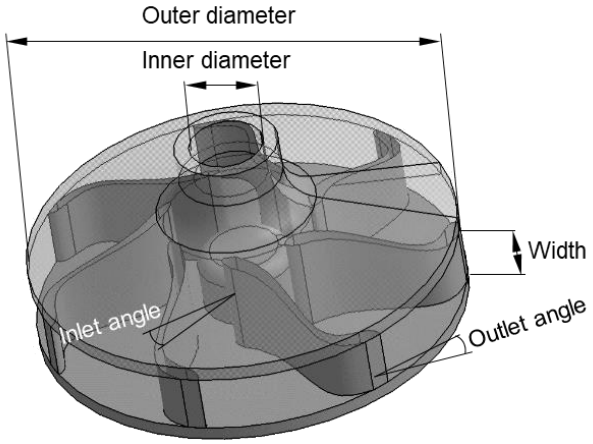

Figure 3. Geometric parameters of pump wheel

The aforementioned parameters describing a hydrodynamic pump wheel and a design guideline to calculate their values for a specific system are given in literature [32]. The guideline calculates the geometry of the pump wheel for a required operating point. The parameter values are calculated based on empirical equations and diagrams. The required inputs for the guideline are:

- Pressure delta, which the pump has to provide between input volume flow and output volume flow at optimum working point
- Rotational speed of the pump at optimum working point
- Density of the transported cooling medium at optimum working point
- Required volume flow at optimum working point

We implement the calculation steps of the guideline in an external software. Multiple tools may be chosen. As those guidelines in companies are often implemented in MS Excel, we choose this tool.

4.2. Concept for connecting functional architecture, workflow and simulation model

The functional architecture described in section 3 provides a structure for principle solutions and their describing parameters. The design guidelines calculate the values of principle solution parameters. They are not modelled directly in the functional architecture, as they do not serve a functional mean during product lifetime, but are only used during design. We model guidelines and other workflows due to their sequential character as SysML Activity Diagrams. A connection between the principle solution architecture and the Activity Diagram is needed. The parameters are the obvious connection, as they occur in both, IBD and Activity Diagram. While the IBD models the parameter structure and interdependencies, the Activity Diagram uses specific parameter values and calculates other parameter values from them, e.g. outer diameter of the hydrodynamic pump wheel from required volume flow. As storing parameter values is not the purpose of a systems modeller, especially for large systems, we create a separate parameter database and link IBD and Activity Diagram via this database, Figure 4. The structure of the database (2) is given by the principle solution IBD (1). In the IBD, parameters are modelled and organized. When a new parameter is added to the principle solution IBD, a database entry is created. Therefore, the database contains all product describing parameters from IBD and their system specific values.

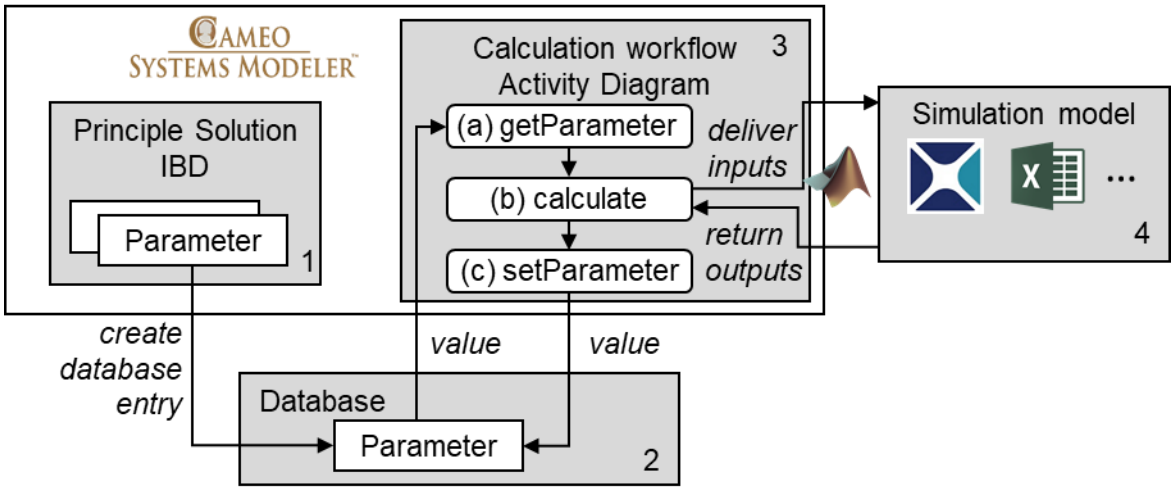

Figure 4. Concept of connecting Activity Diagram and architecture in CSM and integrating an external database and simulation models

Design workflows described in Activity Diagrams (3) need the current set of specified parameters in order to run calculations. Hence, the Activity Diagram needs to be able to load parameters from the database. For loading parameters, a standardized activity in the Activity Diagram is defined, named getParameter (a). The activity takes the value of a parameter from the external database and provides it for further calculations in the Activity Diagram.

After calculating parameter values in external tools, these parameter values have to be transferred to the database again, so that they are usable for further design and test steps. The parameter values are then saved in the database and can be loaded and used by further workflows. For saving a parameter value in a database, another standardized activity is defined, named setParameter (c). This activity takes the value, which has been calculated and stores it in the database at the position of its corresponding parameter.

Calculating values for principle solution parameters needs external calculation models, which are usually implemented in domain specific tools. External calculations or simulations are integrated in Activity Diagrams using a standardized interface activity calculate (b). The interface activity defines standard inputs for the calculation model and delivers them to the model in an external tool (4), starts the model and takes the result parameters from it. In the Activity Diagram, the calculate activity provides various input ports, to which the values from getParameter activities are delivered, and various output

ports, from which further calculations or setParameter activities may take values in order to process the results.

In a prototypical implementation for the given use case, we used MATLAB data files (.mat) instead of a full database. The concepts are transferable to other database types. All external interfaces to both, database and simulation models where implemented in Matlab functions using CSM's Matlab interface.

4.3. Design workflow for the hydrodynamic pump wheel

The concept described in 4.2 is applied to the design workflow of the hydrodynamic pump wheel from 4.1. We model an Activity Diagram, which loads the required parameter inputs from the external database, calculates the geometric and physical parameter values and stores the calculation results in the database. Since the calculation has impact on geometric and physical behaviour of the principle solution, in a last step, further describing models as CAD models and physical characteristic curves are updated based on the new parameter value set, Figure 5.

The required values *deltaPOpt*, *RhoCooling*, *RPM* and *maxVolFlow* are defined by further principle solutions, which interact with the hydrodynamic pump wheel according to the functional architecture, Figure 1. The listed parameters are hence modelled in their respective principle solutions, Table 1:

- *deltaPOpt*, the pressure delta is defined by the principle solution *release heat*
- *RhoCooling* is the desnsity of the chosen cooling medium for *manage heat flows*
- *RPM* is the optimal rotational velocity for the principle solution of *convert electrical to mechanical energy*
- *maxVolFlow*, the required volume flow for the optimal working point is defined by the necessary volume flow of *distributeHeat*

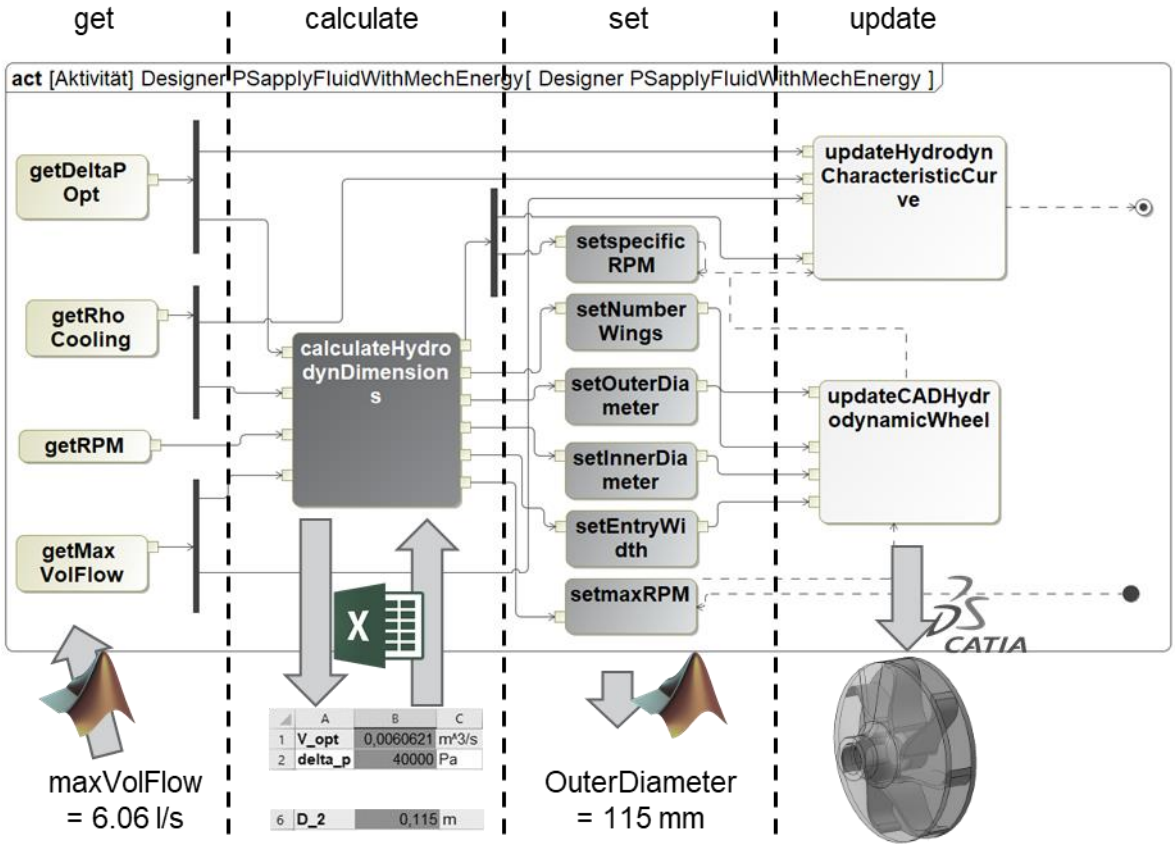

Figure 5. Design workflow for the hydrodynamic pump wheel in SysML

The calculation model implementing the design guideline is integrated in the activity *calculateHydrodynamicDimensions*. It estimates the previously described values of the hydrodynamic

pump wheel geometric element. Additionally, two physical parameters are calculated, which result from the geometric parameter set: *specificRPM* and *maxRPM* are performance indicators describing the principle solution hydrodynamic pump wheel and its limitations and are used for estimating the characteristic curve of the principle solution. The estimated values are saved using individual setParameter-activities.

After parametrizing the principle solution, describing models have to be updated, so that they contain the new parameter set. In the case of the hydrodynamic pump wheel, describing models are the CAD model of the hydrodynamic wheel's active surfaces and the characteristic curve describing the behavior between pressure, volume flow and rotational velocity. The CAD model is prepared in a parametrized way: The parameters outer and inner diameter, outlet and inlet angle, and width are set as parameters in the CAD software Catia and exported to a parameter table. Changing the value in this parameter table automatically changes the value in the CAD model. The activity *updateCADHydrodynamicWheel* takes the calculated values and writes them into the aforementioned table. The activity *updateHydrodynCharacteristicCurve* recalculates the characteristic curve in MATLAB and stores it.

We have modelled an Activity Diagram which parametrizes the principle solution based on given requirements using design guidelines. As this describes the step to design the principle solution, we call this kind of Activity Diagram a *designer*. Designer Activity Diagrams are permitted to change geometric values of their principle solution in the database based on defined calculation models. They ensure that design decisions are made based on calculation models and document the calculations. As they are executable, design becomes automated.

5. Top level design workflow for the cooling circuit

In section 4, the principle solution of the hydrodynamic pump wheel is parametrized using a designer Activity Diagram. As described in 4.3, this designer requires input from other principle solutions. For example, the physical behaviour and parameter values from *distribute heat* define the required volume flow, as *distribute heat* describes the heat transfer functionally. Hence, the design of the principle solution hydrodynamic pump wheel cannot be done independently of the surrounding solutions. Further designers for other principle solutions are required. The designers have to be executed in an order, so that values can be estimated based on already calculated values from other principle solutions. Exemplary, we set up a second designer for *distribute heat*.

In distribute heat, the incoming heat from the engine is transferred to surrounding air and cooling medium and thereby removed from the system. Depending on the quantity of volume flow of the cooling medium, a different amount of heat is removed, resulting in different engine temperatures. We assume the geometric parameter values in distribute heat and the engine components as fixed. To estimate the required volume flow of the solution *distribute heat* we set up a physical model, describing the heat transfer behaviour in a physical simulation in Simcenter Amesim, Figure 6, a. The engine consists of two parts, the cylinder head and the crank case. Both may have different temperatures and are modelled as thermal capacities interacting with each other. The heat exchange with air is modelled. Heat introduced by engine and removed from cooling medium are thermal sources and sinks, introducing and removing an amount of heat from the capacities. As the engine components and the active surfaces are fixed, the heat transfer from engine and to air are fully described. The engine components temperatures are external requirements. The model can be used to estimate the required heat released to cooling medium in order to set the engine temperature. The heat released to cooling medium is varied until the temperatures fit the external requirements. Then from the required heat, the required volume flow is estimated. This is done for both, minimum and maximum required temperature, resulting in minimum and maximum required volume flow. The described calculation is implemented in the activity *calculateMinMaxVolumeFlow*. This activity takes the fixed values from engine geometry and *distribute heat* and the required temperatures and gives minimum and maximum volume flows from it. The maximum volume flow can then be used in the designer of hydrodynamic pump wheel. The full Activity Diagram for *distribute heat* is modelled in Figure 6 b).

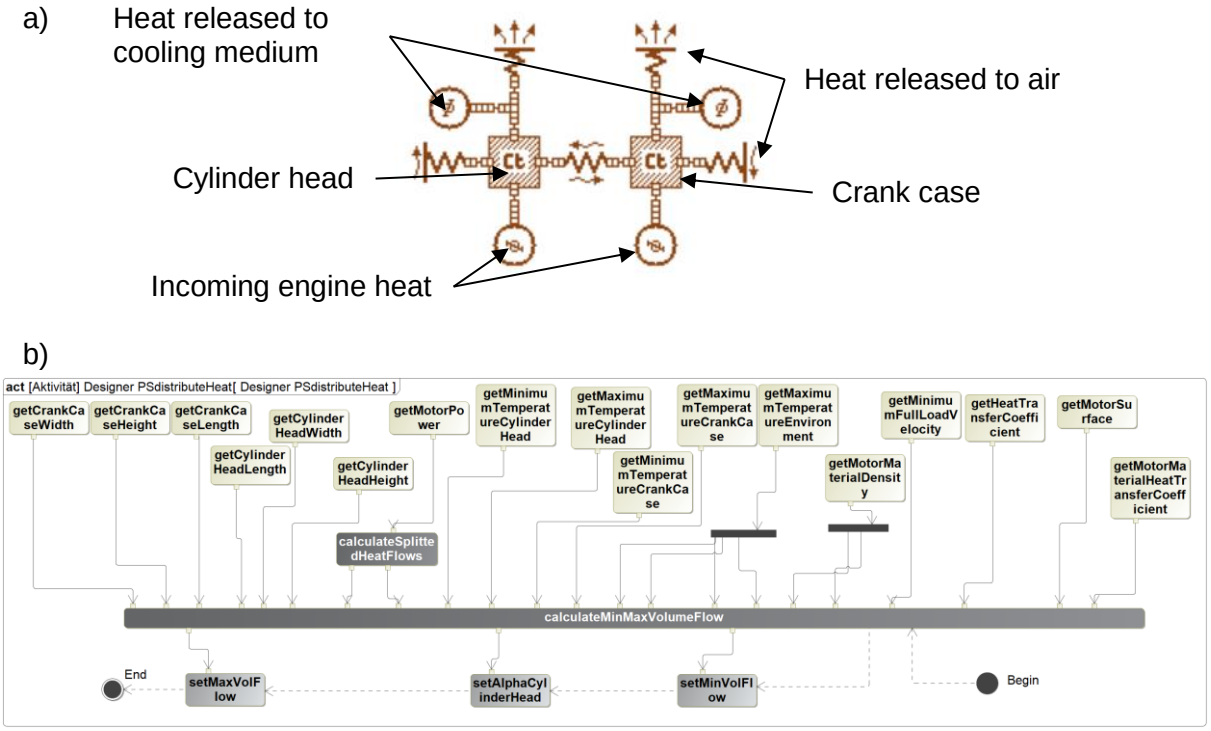

Figure 6. a) Heat transfer model b) workflow for calculating minimal and maximal necessary volume flow using the model from a)

As the designer applyFluidWithMechanicalEnergy, i.e. the hydrodynamic pump wheel from section 4.3 is only usable, when the value for *maxVolFlow* is set, an order for executing designers is necessary. This order describes the design activities necessary for the top-level solution *manage heat flows*. The top-level solution designer models the order in which principle solution designers are executed. The single designers are considered as one activity each in the overall designer for the full solution *manageHeatFlows*, Figure 7. This executable workflow can be used to redesign every principle solution one after another based on the formalized design process.

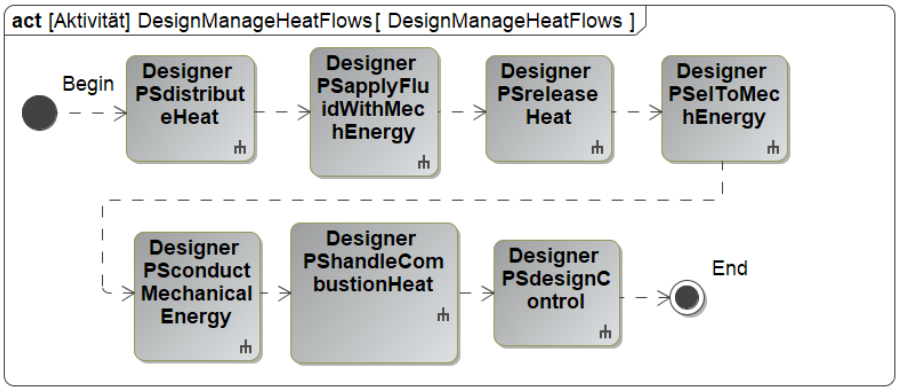

Figure 7. Overall design process of the solution *manageHeatFlows*, including the previously presented designers: Designer PSdistributeHeat from Figure 6 b) and Designer PSapplyFluidWithMechEnergy from Figure 5

6. Solutions Testing

The described design process uses explicit design cases to calculate principle solution parameters, e.g. the geometric values of the hydrodynamic pump wheel have been calculated for maximum volume flow.

Besides, not all requirements or use cases attached to the system are considered. Testing, if all requirements are satisfied for all use cases of the system, has to be done in a separate way. This is essential to deliver a fully working product and in case of requirement changes, in order to answer the question if a redesign is necessary.

Our system has three requirements to satisfy: temperature value ranges, the two engine components have to keep, and a maximum rotational speed for the hydrodynamic pump wheel. Testing the system behaviour has to regard the overall requirement fulfilment. We use two different simulation models to test the requirements. A Simulink model calculates the rotational speed of the hydrodynamic pump wheel and the volume flow created due to this rotational speed based on the hydrodynamic characteristic curve of the hydrodynamic pump wheel and the parameter values from *control heat flows* and *convert electrical to mechanical energy* (Table 1). The Simcenter Amesim model from Figure 6 a) is used to calculate the engine component temperatures, which set due to the calculated volume flow.

The previously named designers are used to set solution parameters, e.g. geometric parameters. However, they do not test further requirements and system interactions. To integrate the described tests into our system modeling approach, we introduce another workflow model. Tests are also modelled as Activity Diagrams; however, their purpose is different. A test does not manipulate values of system parameters, but uses the values to calculate physical states by executing physical models, e.g. the engine temperatures using the heat transfer model. The resulting physical states are then compared to values given as requirements. Test Activity Diagrams load solution parameters using getParameter activities as defined for designers before. They execute external simulations and calculations using calculation activities. The calculated parameter values are stored and compared to the requirements, which are linked by a *satisfy*-connection between the parameter and a design constrained.
The described testing procedure including both simulation models is modelled as a test Activity Diagram. The tests load values from all principle solutions which are contained in *manageHeatFlows*. The test then triggers the two simulation models: At first the Simulink model is started (*testRPMAndVolFlow*), which calculates the rotational speed due to the control parameter values and the volume flow rate resulting from rotational speed. The volume flow rate is then transferred to the Amesim model describing heat flows and temperatures in the engine components (*calculateTemperature*). Results of rotational speed and temperatures are compared to the requirements. The workflow including both models is shown in Figure 8.

7. Use case: Changing the engine power

The described workflow modelling allows for straightforward system testing and redesign when dealing with changed requirements. In an artificial scenario from automotive engineering, we show the usage of the functional architecture modelled in SysML4FMArch using the executable Activity Diagrams linked to principle solutions. The cooling circuit has been developed for a 135kW engine. The temperature requirements are set for temperature of cylinder head between 120°C and 150 °C and for crank case between 100°C and 130°C. The maximum speed of the pump is set to 2500 rpm. The system setup satisfies these requirements.

In a first change, the cooling circuit is tested for a higher engine class. The new engine has a power of 185 kW. Due to higher power, the heat production of the engine increases, which implies the need to retest whether the cooling system still fulfils all requirements. The test workflow from section 6 fails, showing, that the temperature of both engine components is now much to high (270°C for crank case and 347°C for cylinder head). Executing the Activity Diagram that designs the cooling circuit function *manage heat flows* yields two possibilities to change the system: Manipulating only the rotational speed in order to create a higher volume flow rate or to change geometric values of the hydrodynamic pump wheel. It is assumed, that changing a control parameter value to increase rotational speed is less effort than changing the geometry.

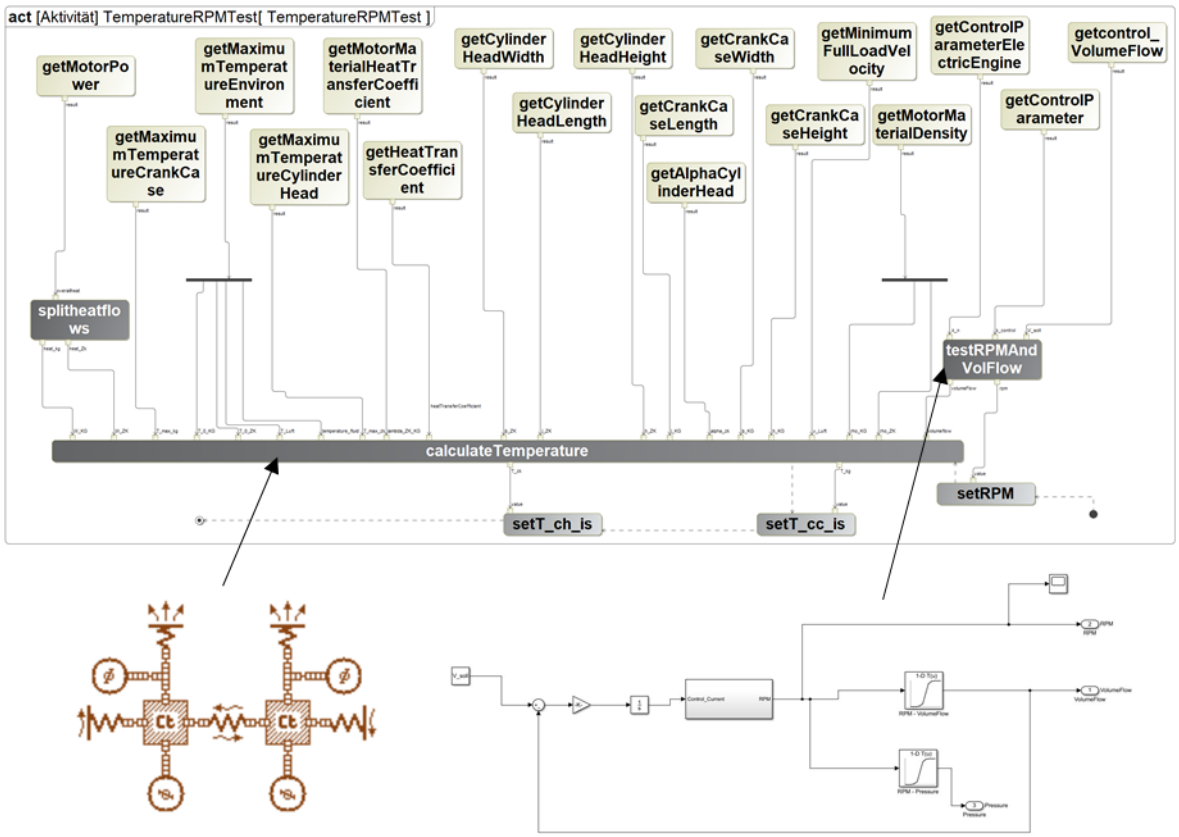

Figure 8. Test workflow to verify the engine components temperatures and the rotational speed of the pump wheel

Using the design Activity Diagram for control heat flows gives the new parameter value. Re-executing the functional tests reveals, that the temperature requirements are now satisfied. Rotational speed has increased to 2011 rpm.

In a second change, the engine power is raised to 250 kW. Controller redesign increases rotational speed once again, so temperatures are satisfied again. This time, rotational speed of the pump wheel becomes too high (2700 rpm). Therefore, the full redesign process is repeated. The pump wheel geometry is redesigned and the control parameter is adapted. The final repetition of the tests shows, that both, temperatures and rpm are now in range of the requirements (120°C crank case, 136 °C cylinder head and 1500 rpm). To achieve this, the pump wheel width has been increased by 5 mm. Modelling the design process in reusable design workflows and development of tests for all system requirements allows for automatic redesign and the information whether system requirements are fulfilled at any time.

8. Discussion

Integrating simulation models into architecture development using executable workflows provides the possibility to reproducible design the system and test it regarding requirement satisfaction. We have shown, that it is possible to model mechanical design processes in executable Activity Diagrams. Especially guidelines for single principle solutions, e.g. the pump wheel, can be modelled that way. Such guidelines are often given in mechanical engineering (i.e. norms and standards). Our approach makes such norms reusable in an easy way. Designers handle external requirement values and other boundary conditions as well as requirements coming from other internal system solutions. Design changes can be seamlessly proceeded and corresponding CAD- and other describing models can be updated automatically.

Executable models of test workflows enable agile development and allow to use strong simulations for virtual testing. Following the thoughts of continuous integration in software engineering, it is possible to run all test workflows overnight in order to prove system validity regarding requirements.

However, tool integration of such procedures is not optimal. Currently the get and set functions have to be written manually for every parameter. Storing parameter values in a mat-file does not scale, if the number of parameters exceeds a certain number and does not allow for role definitions. However, there are various data management techniques, which could be integrated in our approach to provide efficient, safe and distributed data handling.

A bigger disadvantage of the graphical Activity Diagrams is, that they get crowded from get and set functions, when many parameters are given for a model. Here a textual alternative for connecting get- and set-elements with models would be a solution.

9. Summary and Outlook

In the present study, we proposed an approach for modeling mechanical design and test processes in executable workflows using SysML Activity Diagrams and external tools. We developed two kinds of Activity Diagram standards for dimensioning (designers) and testing (testers), respectively. Designers load values of solution parameters, run simulations and calculations based on these values, and calculate values of other solutions' parameters. The parameter values are stored and describing models as CAD are updated automatically. Tests run simulation models based on the current values of solution parameters and calculate the values of physical states that way. Afterwards, the values of physical states are compared against the requirements to prove requirement satisfactions.

The concepts were explained by the design of an electric mechanical coolant pump from automotive industry. In the use case, the power of the combustion engine was adapted, and the cooling system was able to automatically adapt to the new requirements. The use case reveals the benefits of automated redesign and testing as a reaction to requirement changes.

Future work aims to standardize and classify design and test workflows. Simulation models have varying detail level. Higher detail levels usually require more parameters to be specified. This also has influence on the workflow detail level. Classification is here a step to have suitable libraries for both simulation models and design and test workflows using the models. Engineers can then use the best suitable workflow for their current system status.

Another step for automatization is ontology development. Developing an ontology for principle solutions, models and workflows allows for a stronger automatization and closes the gap between physical modelling and architecture. Hereby, strong design and model catalogues can be created, making agile development in mechanical engineering easier.

References

[1] International Council on Systems Engineering 2007 *Systems Engineering Vision 2020* (Seattle: International Council on Systems Engineering)

[2] Andary F, Berroth J and Jacobs G 2019 An Energy-Based Load Distribution Approach for the Application of Gear Mesh Stiffness on Elastic Bodies *Journal of Mechanical Design* 2019 **141**

[3] Berroth J, Jacobs G, Kroll T and Schelenz R 2016 Investigation on pitch system loads by means of an integral multi body simulation approach *Journal of Physics: Conference Series* **753**

[4] Golafshan R, Jacobs G, Wegerhoff M, Drichel P and Berroth J 2018 Investigation on the Effects of Structural Dynamics on Rolling Bearing Fault Diagnosis by Means of Multibody Simulation *International journal of rotating machinery* **2018**

[5] Pasch G, Jacobs G, Höpfner G and Berroth J 2019 Multi-Domain Simulation for the Assessment of the NVH Behaviour of a Tractor with Hydrostatic-Mechanical Power Split Transmission *Land.Technik AgEng* **2019**

[6] Wegerhoff M, Jacobs G and Drichel P 2019 Noise, vibration and harshness validation methodology for complex elastic multibody simulation models: With application to an electrified drive train *Journal of Vibration and Control* **25**

[7] No Magic Inc 2020 *MagicDraw 19.0 LTR SP4 User Manual*
[8] Object Management Group 2019 *OMG System Modeling Language (OMG SysML) Version 1.6*
[9] Object Management Group 2015 *OMG Unified Modeling Language (OMG UML) Version 2.5*
[10] Dänzer M, Kleiner S, Lamm J G, Moeser G, Morant F, Munker F and Weilkiens T 2014 Funktionale Systemmodellierung nach der FAS-Methode: Auswertung von vier Industrieprojekten *Tag des Systems Engineering* **2014**
[11] Pearce P and Hause M 2012 ISO-15288, Oosem and Model-Based Submarine Design
[12] Weilkiens T 2016 *SYSMOD-The Systems Modeling Toolbox-Pragmatic MBSE with SysML* (Fredesdorf: MBSE4U)
[13] Pohl K, Broy M, Daembkes H and Hönninger H 2016 *Advanced Model-Based Engineering of Embedded Systems* (Berlin: Springer)
[14] Alt O 2012 *Modell-basierte Systementwicklung mit SysML* (München: Hanser)
[15] Schlecht S and Alt O 2007 Strategien zur Testfallgenerierung aus SysML-Modellen *Software Engineering*
[16] Gausemeier J, Dorociak R and Nyßen A 2010 The Mechatronic Modeller: A Software Tool for Computer-Aided Modeling of the Principle Solution of an Advanced Mechatronic System *11th International Workshop on Research and Education in Mechatronics*
[17] Eigner M, Koch W and Muggeo C 2017 *Modellbasierter Entwicklungsprozess cybertronischer Systeme* (Berlin: Springer)
[18] Moeser G, Kramer C, Grundel M, Neubert M, Kümpel S, Scheithauer A, Kleiner S and Albers A 2015 Fortschrittsbericht zur modellbasierten Unterstützung der Konstrukteurstätigkeit durch FAS4M *Tag des Systems Engineering* **2015**
[19] Koller R and Kastrup N 1994 *Prinziplösungen zur Konstruktion technischer Produkte* (Berlin, Heidelberg: Springer)
[20] Drave I, Rumpe B, Wortmann A, Berroth J, Hoepfner G, Jacobs G, Spuetz K, Zerwas T, Guist C and Kohl J 2020 Modeling Mechanical Functional Architectures in SysML *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*
[21] Hernandez C, Rodriguez M, Diaz I and Sanz R 2016 Model Based Engineering of Process Plants using SysML *26th European Symposium on Computer Aided Process Engineering*
[22] Biswas N, Chattopadhyay S, Mahapatra G, Chatterjee S and Mondal K C 2017 SysML Based Conceptual ETL Process Modeling *Communications in Computer and Information Science* **776**
[23] Markthaler M, Kriebel S, Salman K S, Greifenberg T, Hillemacher S, Rumpe B, Schulze C, Wortmann A, Orth P and Richenhagen J 2018 Improving model-based testing in automotive software engineering *IEEE/ACM 40th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP)*
[24] Drave I, Hillemacher S, Greifenberg T, Kriebel S, Kusmenko E, Markthaler M, Orth P, Salman K S, Richenhagen J, Rumpe B, et al. 2019 SMArDT modeling for automotive software testing *Software: Practice and Experience* **49** 2
[25] Drave I, Hillemacher S, Greifenberg T, Rumpe B, Wortmann A, Markthaler M and Kriebel S 2018 Model-Based Testing of Software-Based System Functions *44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*
[26] Reiß D and Rumpe B 2012 Using Lightweight Activity Diagrams for Modeling and Generation of Web Information Systems *International United Information Systems Conference*
[27] Schonherr O and Rose O 2009 First steps towards a general SysML model for discrete processes in production systems *Proceedings of the 2009 Winter Simulation Conference (WSC)*
[28] Patel V V, McGregor J D, Goasguen S 2010 SysML-Based Domain-Specific Executable Workflows *IEEE International Systems Conference*
[29] MathWorks 2020. *Matlab 2020b - Manual*
[30] Siemens PLM Software 2018 *Teamcenter Rapid Start 11.6 - Setting Up Workflows for Product*

Development
[31] Zerwas T, Jacobs G, Höpfner G, Drave I, Berroth J, Guist C, Spuetz K, Konrad C, Rumpe B and Kohl J 2021 Mechanical Concept Development using Principle Solution Models *Antriebstechnisches Kolloquium 2021*
[32] Wesche W 2016 Radiale Kreiselpumpen: *Berechnung und Konstruktion der Hydrodynamischen Komponenten* (Berlin: Springer Vieweg)