




# Function-Oriented Model-Based Product Development

# 13

Georg Jacobs , Christian Konrad, Joerg Berroth, Thilo Zerwas, Gregor Höpfner, and Kathrin Spütz

## Abstract

The innovative strength and competitiveness of a company depends on mastering the growing complexity of digitally networked products in an efficient way. The complexity is driven by increasing interactions among the different domains, like mechanical, electrical or software engineering on all system levels. The interdependencies require modelling approaches, that allow to explicitly and transparently reveal those interdependencies on requirements, functional architectures and solution level over all phases of the development. The increasing interdependencies and the need for more efficiency forces a change from component oriented, document-based product development to a function-oriented, model-based product development with consistently linked models across all participating domains. We propose a system architecture that describes the system in a comprehensible way across domains. The domains are able to connect their models to the architecture and link them down to the parameter level over requirements, functional architecture to the solution layer. The resulting system model allows a transparent, cross-domain mapping of functional interactions. Principle solution models close the gap between the functional and the solution layer, especially in mechanical engineering. The efficiency in development processes can be significantly increased by using model libraries to assign functions to solution models and by building ontologies to structure domain-specific models.

---

G. Jacobs (✉) · C. Konrad · J. Berroth · T. Zerwas · G. Höpfner · K. Spütz  
RWTH Aachen University, Institute for Machine Elements and Systems Engineering (MSE),  
Aachen, Germany  
e-mail: [georg.jacobs@imse.rwth-aachen.de](mailto:georg.jacobs@imse.rwth-aachen.de); [gregor.hoepfner@imse.rwth-aachen.de](mailto:gregor.hoepfner@imse.rwth-aachen.de);  
[kathrin.spuetz@imse.rwth-aachen.de](mailto:kathrin.spuetz@imse.rwth-aachen.de)

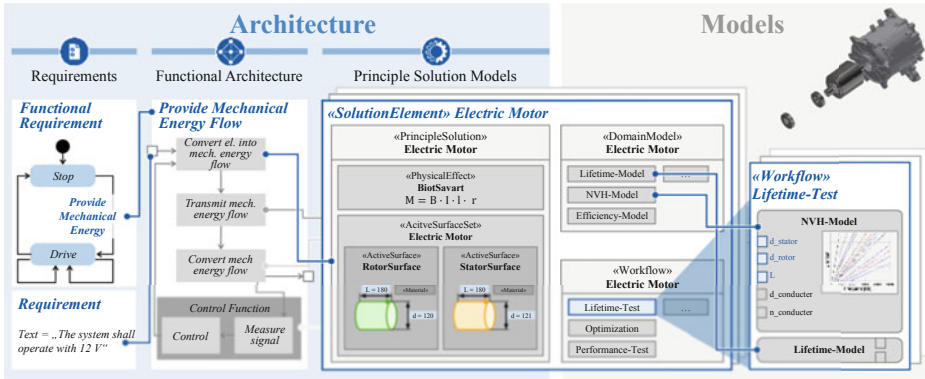
### 13.1 Introduction

In order to meet today's customer requirements, cyber-physical systems (CPS) are being developed. CPS are interconnected mechatronic systems, which in turn consist of mechanical, electrical, electronic and software components. Mastering the dependencies between the domains is one of the major challenges in product development today (Alur 2015; Eigner et al. 2012; Graessler and Hentze 2020).

The growing challenges in the development, production and operation of digitally networked technical systems can't be overcome with conventional methods of product development in mechanical engineering. Model-Based Systems Engineering (MBSE) is an approach for the cross-domain development (Eigner et al. 2012; Gausemeier et al. 2010; Broy 2010). Generic guidelines for the development of technical systems provide the methodological framework for a function-oriented product development process (VDI 2221a, b). While other domains are developing in a strongly function-oriented manner, for mechanical engineering this is more difficult due to the methodical breaks in the transfer of functions into components. Since the development of components is time-consuming compared to the small-step procedure of other domains, the mechanics have to invest a lot of effort in order to show first development results (Graessler and Hentze 2020). Thus, parallel and agile developing is more difficult. To overcome this obstacle, the mechanical domain needs methods that allow for faster functional testing and virtual validation of the system. Therefore, behavioral models must be integrated into function-oriented development processes in a standardized way. This enables model-based design decisions and collaboration with other domains in agile development processes.

The Institute for Machine Elements and Systems Engineering (MSE) at RWTH Aachen University has identified key success factors for the use of MBSE as part of research activities at the Center for Systems Engineering together with leading companies, see Fig. 13.1. The success of MBSE in industrial practice requires standards for setting up a function-oriented and model-based system architecture, the classification of existing expert models to describe the system behavior and the consistent linking of the architecture with models of the involved domains.

This chapter presents a method for function-oriented and model-based system development. The method proposed, inherits the formalization of system requirements in form of requirement models and the derivation of functional architectures based on the modelled requirements. A decomposition of the functions of the functional architecture down to so called elementary functions as described in (Koller and Kastrup 1994) is introduced as a tool to structure the solution approach. The central element of the method is the introduction of principle solution models that are linked to elementary functions via physical effects. The relation between elementary functions and function fulfilling physical effects are the basis for the clustering of principle solutions in solution libraries, see Sect. 13.2. Besides a principle solution model, the solution libraries carry additional behavior and test models that are evolved from the principle solution models. The solution models enable



**Fig. 13.1** Function oriented and model-based Architecture, classified expert models and seamless linking of architecture with expert models as key success factors

iterative functional testing and virtual validation in mechatronic system development, see Sect. 13.3.

Adding additional behavior models to the solution elements within the solution libraries with different fidelity levels for different purposes calls for necessity to structure, classify and standardize the models in organizing ontologies. On the basis of these ontologies and the SE architecture, interfaces for the corresponding model classes can then be systematically developed and used for the seamless linking of the models within the product development process. This enables the efficient reuse of expert models and forms an excellent basis for the introduction of digital seamless and more agile model-based product development processes, see Sect. 13.4.

## 13.2 Basic Architecture for Model-Based Systems Development

As a basis for cross-domain agile development processes, functional tests and virtual validation of system properties (e.g. dimensions, physical behavior or costs), a methodical foundation for function-oriented and model-based system architectures is needed. Thus, current methods have to be further developed, in order to enable the full description of the functional layer and its linkage to physical solution models (Eigner et al. 2012; Drave et al. 2020; Zerwas et al. 2021).

As a foundation, engineers must be able to derive a functional architecture from product requirements that is solution- and domain-neutral and can thus structure the development in a function-oriented way (Suh 1998, VDI 2221 Part 1). A function can often be realized with different solutions out of one or more domains. To support design decisions it is important for engineers to be able to define their possible solutions with little effort and test them as early as possible against requirements.

In research projects (e. g. FAS4M), first approaches have been developed to enable a transition from functional system description to the physical geometry of components (Möser et al. 2016; Grundel 2017). Another approach from design methodology are principle solutions. “Principle Solutions” describe a possible solution principle by specifying a physical effect, active surfaces and material (Roth 1994; Koller 1998; Feldhusen and Grote 2013). The concept of principle solutions is already well known in design methodology and there are several concepts based on it with regard to design catalogues and the description of components by active surfaces and guiding support structures.

As an architectural basis, the concept of using principle solutions offers an ideal starting point for the enhancement of model-based systems engineering within mechanical engineering. The approach according to Koller has been revised and further developed into a function-oriented and model-based method. MBSE modelling languages and tools are used to transfer requirements via functions into principle solutions (Drave et al. 2020; Zerwas et al. 2021; Höpfner et al. 2021).

The underlying idea of the modelling method is that properties of physical systems rely on clearly definable behavior descriptions that incorporate parameters. There are already extensive parameter classifications for technical systems in design methodology (Hubka and Eder 1988; Patzak 1982; Weber 2014). In this chapter, parameters are understood as relevant inputs and outputs of models. As models we understand representations and abstractions from real systems as 3D-representations, numerical differential equations in CAE or simple analytical equations describing physical effects. Parameters are of different types and quantifiable. However, they are not always of scalar nature but may be matrices, tuples or objects, which store scalar parameters by themselves. Examples of parameters include target values from requirements, variables and constants in physical laws, property values such as weight, volume, stiffness and post-processed outputs from models such as natural frequencies and sound pressure level maps. Therefore, an essential idea of the system architecture approach presented in this chapter is the consistent and seamless linking of parameters.

The presented modelling method ensures the function orientation for the mechanical domain similar to existing software domain approaches and thus enables the cross-domain collaboration required in today’s development processes. For modeling purposes, the SysML profile SysML4FMArch is used. SysML4FMArch was developed as a basis for modelling functional architectures in all domains, with a strong focus on the improvement of the mechanical domain’s particularities (Drave et al. 2020).

The modeling Method is demonstrated 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 a mechanical energy flow 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.

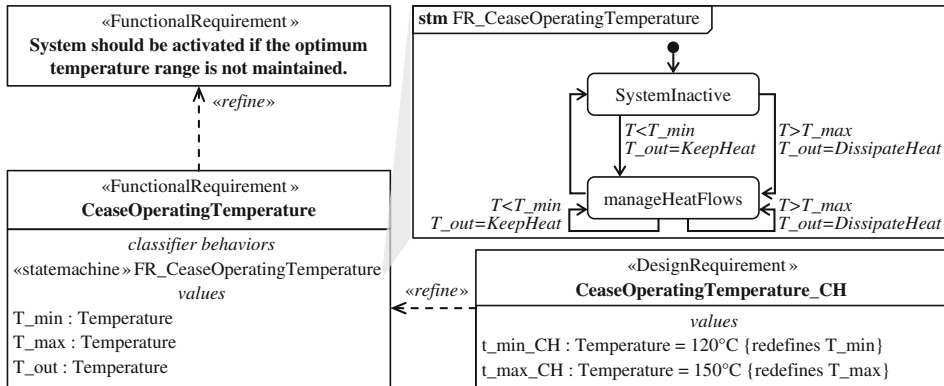
The rising component temperature is becoming increasingly critical for the component material as well as the combustion process and endangers the functional reliability of the engine. For this reason, combustion engines are usually kept within an optimum temperature window by a liquid-based cooling system. A cooling medium circulates in this cooling system, which absorbs heat from the engine and releases it to the radiator (heat exchanger). At the radiator the thermal energy flow 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, 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 modelling of this example system with regard to its requirements and functions, as well as the evolution of the solution models starting from a selection of principle solution models.

### 13.2.1 Modelling of Requirements

Requirements are demands and wishes that customers, users, 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 is important to document these requirements and to continuously check their compliance during the development process (Koller 1998). There are many approaches to classifying and formulating requirements in product development (Ross and Schoman 1977; Göhlich and Fay 2021). Until today, requirements in many companies are still formulated in textual sentences and stored as a list in unlinked documents or software tools. Same applies to e.g. guidelines, norms, etc. Even if requirements engineering and -management is supported by good templates (Rupp 2014), the disadvantages are often the ambiguous formulation of requirements and the missing link to the models based on them, which are thus cut off from requirement changes (Konrad et al. 2019; Graessler et al. 2020).

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



**Fig. 13.2** Requirements of the example system

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, see Fig. 13.2. We differentiate requirements into two categories. Functional requirements (`«FunctionalRequirement»`) specify the desired functionality of a technical system. Functional Requirements are modeled in requirement diagrams and are formalized in state machine diagrams (stm), activity diagrams (act) or sequence diagram (sd).

In the example system, the superior behavior of the complete cooling system is described as a state machine, which is always in one of two states: Either it is idle or active. The transitions between the states depend on the temperature states of cylinder head and crankcase. If, for example, the cylinder head exceeds its permissible maximum temperature, the system switches to the active state. This modelling 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 (Drave et al. 2019).

The second category are restrictions or 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 refines the functional requirement. Thus, the modelling of behavior and the restriction of parameter values are clearly separated, but can be used for common statements.

### 13.2.2 Functional Architecture

There are different types and views of functions in literature where in this paper the focus is on system environment functions according to (Srinivasan et al. 2012).

Accordingly, Functions describe the specific behavior of a product without specifying the solution, e.g. which domain, components, effects, etc. implement this behavior. The concept of functions is based on the idea that function flows enter and leave a system over a given system boundary. These function flows are quantified by parameter values and can be categorized as flows of energy, material, or signal. Functions describe not only which function flows enter and exit, but also which operation takes place (Koller 1998). The decomposition of the overall function into subfunctions results in a functional architecture (Feldhusen and Grote 2013). The SysML4FMArch profile defines that functions can be divided into decomposed functions («Architecture») and elementary functions («ElementaryFunction»).

Each elementary function describes an elementary mathematical relationship between the input and output flows (Koller 1998). Ideally, functions can directly be derived from requirements (customer functions). In any case, functions can be linked to the requirements they fulfill through function calls or satisfy relationships.

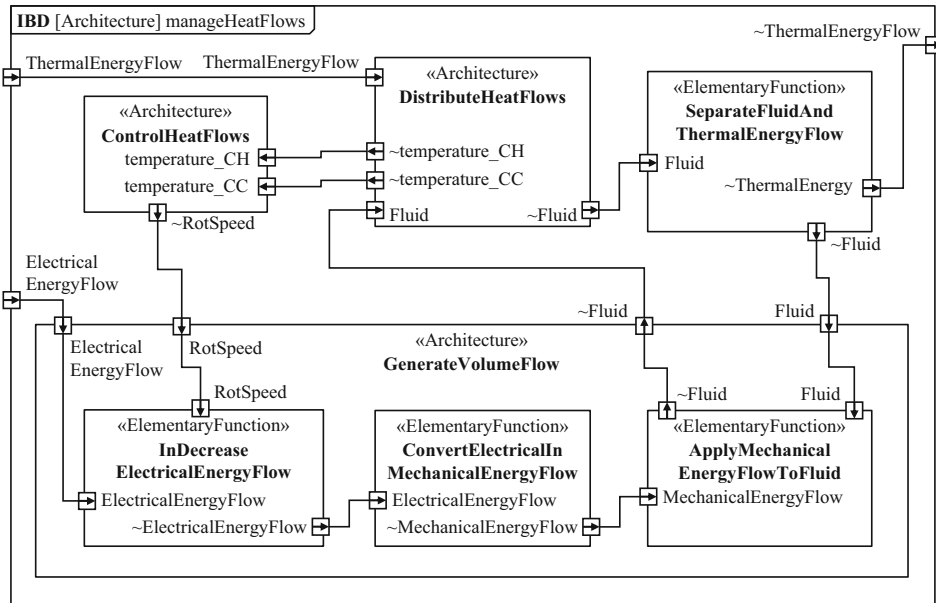
In the example system, the function *ManageHeatFlows* is decomposed into four functions, see Fig. 13.3. The function *GenerateVolumeFlow* generates a volume flow of the coolant according to the specified signal from the function *ControlHeatFlows*. This volume flow is directed to the function *DistributeHeatFlows*, where heat is absorbed (at the cylinder head and crankcase). The heated coolant flow leaves this function and releases a thermal energy flow 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 flow 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 flow. States of a system, as introduced in (Ponn and Lindemann 2011) are modeled via the ports of the functions.

The realization of an elementary function is often possible in multiple domains. 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.

### 13.2.3 Principle Solution Models

While functions describe the changes of function flows, principle solutions concretize how this change is physically realized. Therefore, elementary functions and principle solution models are linked with a generalization relationship. The principle solution inherits all

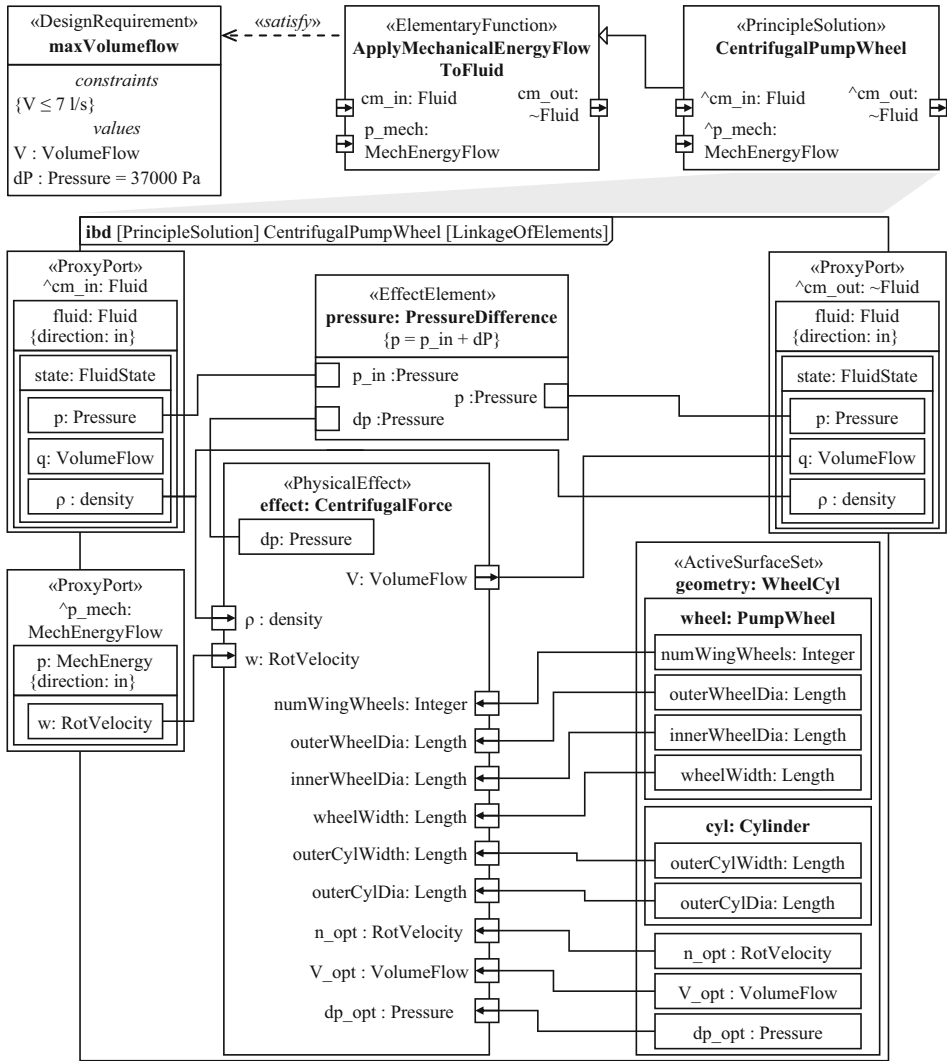


**Fig. 13.3** Functional architecture of the example system

functional flows as ports from the elementary function and can use them to describe more precisely how incoming flows are transformed into 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 (Drave et al. 2020) 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), are linked to the corresponding values of 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 Fig. 13.4 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 at a rotational speed of 15 s<sup>-1</sup> and against a pressure difference of 37 kPa 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 Fig. 13.4 illustrates the internal block diagram of the principal solution. Several possible physical effects for the elementary function





**Fig. 13.4** Requirements are satisfied by elementary functions and their principle solutions whose elements, parameters and relationships are visible in the internal block diagram

*ApplyMechanicalEnergyToFluid* can be found in the Koller catalog (Koller and Kastrup 1994): Boyle’s law, adhesion, Coulomb’s law 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 (Drave et al. 2020). In addition to the physical effect, the principle solution contains another «EffectElement» representing the pressure difference. Finally, the parameters of the «PrincipleEffect» are linked to the counterparts of the active surfaces and function flows.

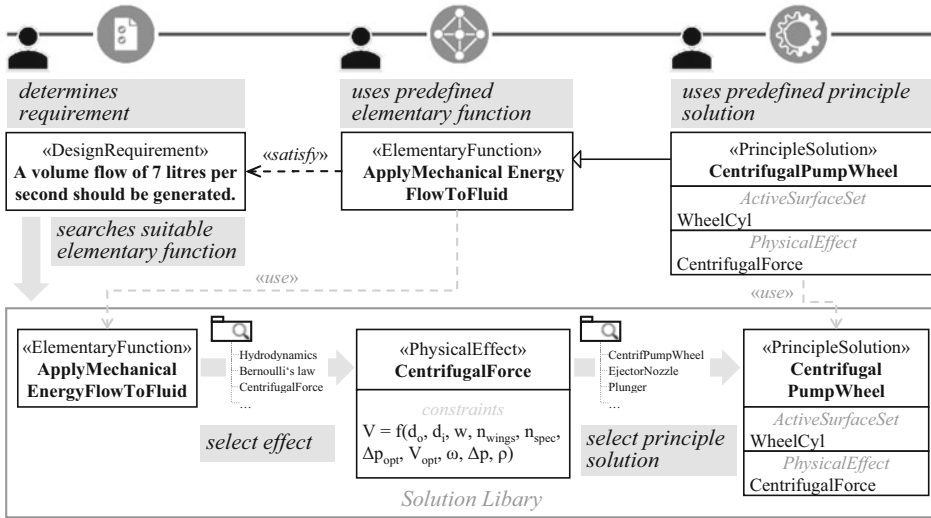
The modelling of the latter transfers the concept of Koller (Koller 1998) into a formalization for SysML and, in contrast to other approaches (Möser et al. 2016; Weilkiens 2016; Albers and Zingel 2011; Lamm and Weilkiens 2010), uses a consistently parameter-based representation of effect, geometry and material (Drave et al. 2020). This key advancement enables initial performance testing of principle solutions.

### 13.2.4 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 (Koller 1998). Since, according to Roth (Roth 1994), 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.

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, see Fig. 13.5 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, see Fig. 13.5 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, see Fig. 13.5 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 extends the well-known Koller catalog (Koller and Kastrup 1994) by central elements: The solution library contains not only elementary functions and physical effects (like Koller), but complete principle solutions including frequent active

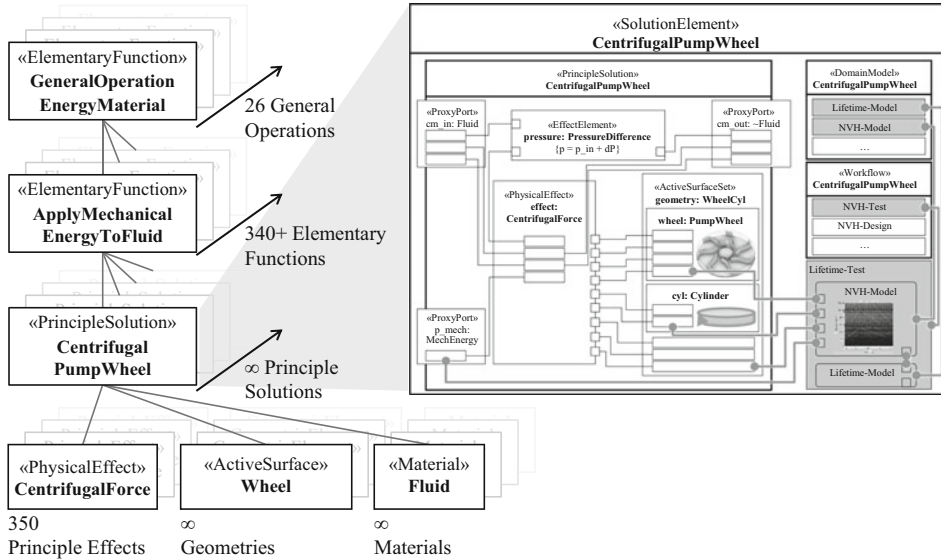


**Fig. 13.5** The solution library (lower part) supports the development process (upper part) by identifying possible physical effects and principle solutions for a selected elementary function

surfaces and materials, see Fig. 13.6 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. Sect. 13.2.5) 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, see Fig. 13.6 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 (cf. Sect. 13.3).

### 13.2.5 Initial Performance Testing of Principle Solutions

The basis for testing a principle solution is the previously presented formalization. Due to the parameter-based representation, physical effect and active surfaces can be linked to external models, see Fig. 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 functional behavior of the modeled principle solution (Fig. 13.4) can be calculated and validated by a software engine that handles the execution of the single solvers. In our case, we used the integrated simulation engine of a

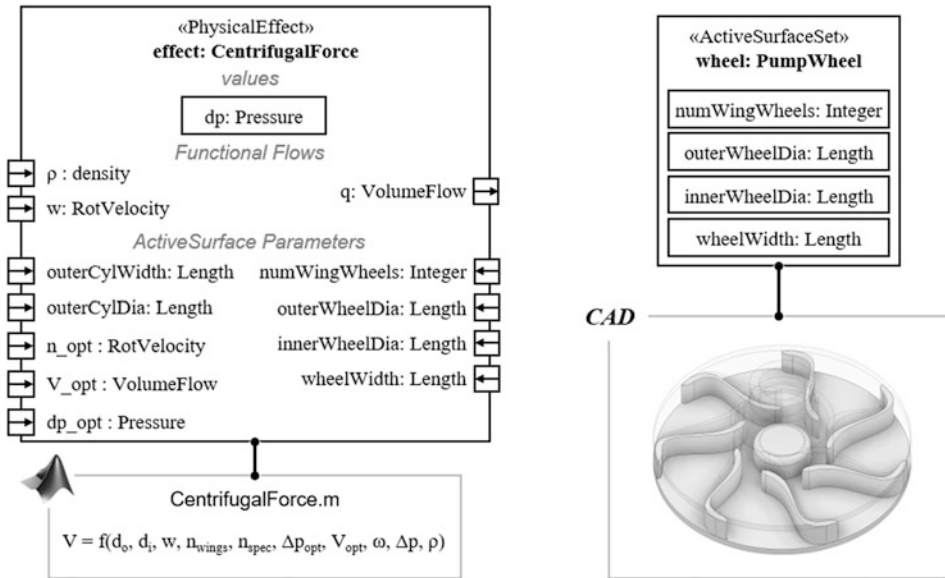


**Fig. 13.6** In addition to elementary functions, physical effects, active surfaces and materials, the solution library primarily contains a set of predefined principle solutions. These principle solutions are stored together with more detailed behavior models and workflows in a so-called solution element

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.

With the procedure described, principle solutions can be tested functionally based on the basic physical parameters that define the active surfaces without having to design complex components first. As a result, the mechanical domain can collaborate with the other domains on the function realization earlier than before and rely on objective test results on basis of a common architecture. 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 (Höpfner et al. 2021).

As the verification of the fulfillment of functional requirements using analytical equations is only a first step in the conceptual design, the discussion of the further design process using more sophisticated models and verifying further constraints is necessary. In addition, the principle solution and its parameters can be supplemented in the further course of development by more detailed models that allow to test phenomena such as cavitation or acoustics.



**Fig. 13.7** The «PrincipleEffect» CentrifugalForce is connected to an external MATLAB model and «ActiveSurface» of the PumpWheel is connected to a CAD model (Zerwas et al. 2021)

### 13.3 Virtual Testing of the Behavior of Evolving Solutions

The functional testing and virtual verification of technical systems' behavior with regard to further restrictions as lifetime, efficiency or acoustics requires the use of a complex set of physical behavior models (Andary et al. 2019; Pasch et al. 2019).

These models require a specific amount of input parameters from the principle solution. Mandatory prerequisite for the efficient (re-)use of physical behavior models, and also data, economics or business models in product development, is the seamless linking of the higher fidelity behavior models with the architecture given by the principle solution model to ensure the traceability of changes down to the behavior models.

A structural framework for linking principle solutions, physical behavior models and design processes is introduced in Sect. 13.3. We propose the solution element as structuring element. The solution element contains the principle solution as described in 13.2 as architectural element. Additionally, behavior models are integrated in the solution element and connected to the principle solution parameters. By using higher-fidelity behavior models, new parameters describing the solution are added. The whole solution element evolves with new parameters and models during product development.

Furthermore, in Sect. 13.4, a suitable classification method is proposed, that can be seen as the starting point for standardization of expert models to be used to verify functions against corresponding failure modes.

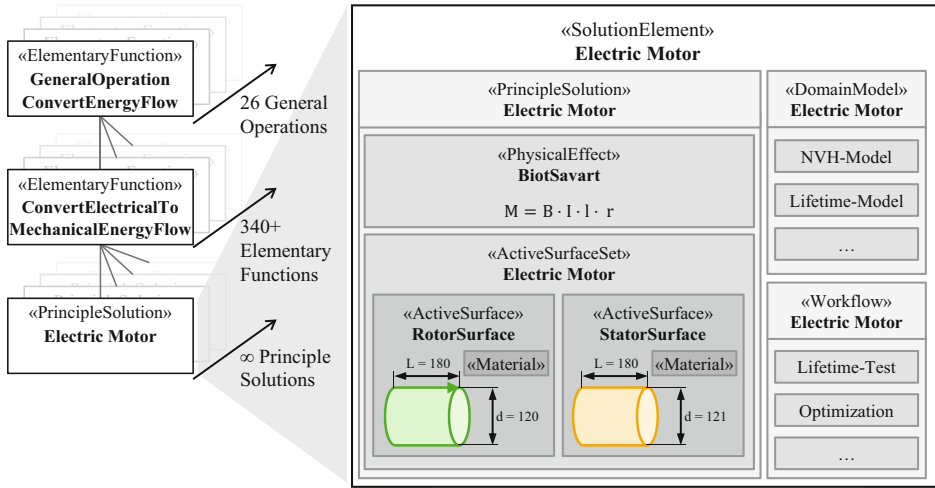
### 13.3.1 Framework for Solution Libraries Based on Behavior Models

The function-oriented architecture described in Sect. 13.2 provides a structure for principle solutions and their corresponding parameters as key element.

The principle solution describes the transformation of different input and output flows via a physical effect, active surfaces and material. An initial performance test as in 13.2.5 verifies, whether the functional requirements regarding the desired transformation are satisfied and uses simple analytical equations. However, the initial performance test neglects further physical effects between the active surfaces and interactions with other (principle) solutions, which may affect the function fulfillment. In addition, there are multiple design constraints to be tested besides the functional requirements, which are not directly related to function fulfillment, but linked to further restrictions, such as lifetime, acoustics or efficiency.

The principle solution in mechanical engineering is typically verified against functional requirements and design constraints using various kinds of physical behavior models. The models describe the physical behavior of a principal solution regarding function fulfillment or design requirements. For each principle solution, we need to organize the corresponding behavior models and link them to the parameters of the principle solution. By connecting models to the principle solution, we can trace requirement changes down to the behavior models via function and solution. The interconnected models allow us to react to changes of requirements in later design steps, as they allow for automated repetition of simulation procedures; as well as impact analysis regarding changes. Further, the information of when to use and how to combine the physical models in order to verify requirements is needed. Therefore, we propose a solution element that combines the principle solution model, higher fidelity behavior models and testing and optimization workflows. Behavior models use principle solution parameters e.g. to predict physical behavior; verify requirements or automate design. For each solution element it is necessary to consider different models and verify different requirements. The propagated workflows allow for sequential behavior model execution in order to verify specific requirements. For virtual verification of requirements, usually a combination of multiple behavior models is necessary. Modelling the order of behavior models is done using workflows. They make the requirement verification repeatable and automatable. In workflows, we organize the model and architecture evolution as well as testing and optimization of solutions. An example for a solution element is discussed in Fig. 13.8 using the example of the coolant pump's electric motor. The motor uses the physical effect of Biot Savart's law between the two active surfaces rotor and stator. Each active surface is described via width and diameter.

The functional test for the principle solution electric motor verifies, whether the motor transforms the electrical power into mechanical power in a sufficient manner. However, there are further requirements related to the principle solution *ElectricMotor*. E.g., the principle solution has to be available for a required lifetime, efficiency might have to be as high as possible and the behavior regarding acoustics has to be pleasing. These requirements are not verified using analytical equations of physical effects. In virtual



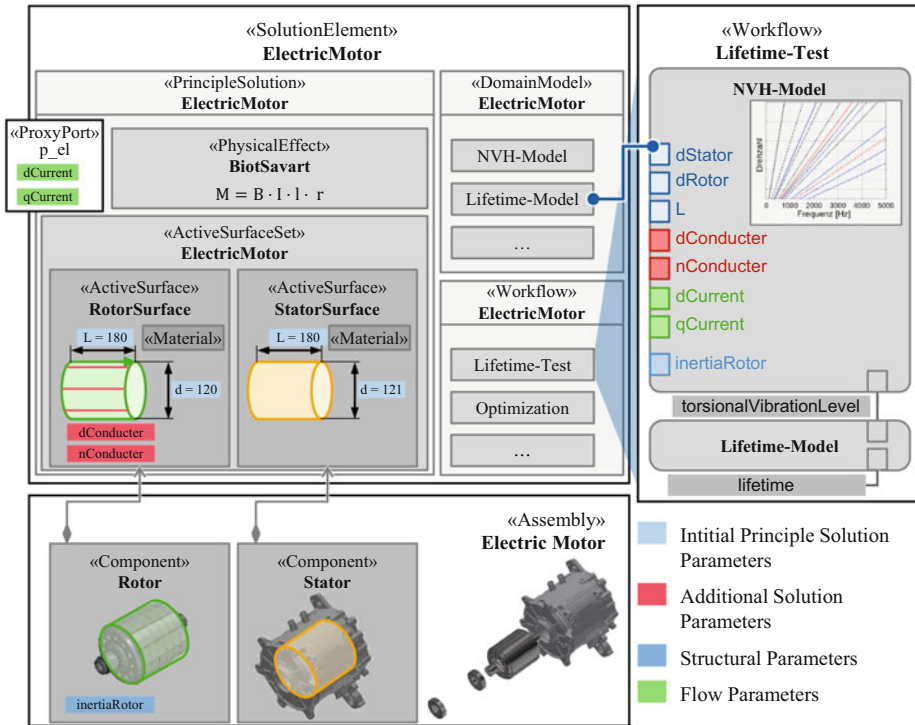
**Fig. 13.8** Solution elements integrate principle solutions, their describing behavior models, as well as design and testing workflows. Workflows use the principle solution’s parameters and behavior models to design and verify the principle solution

product development, specific behavior models predict the behavior for the principle solution regarding each of these requirements, and we can develop tests regarding the aforementioned requirements using combinations of these models. In the next section, we describe how to use existing models for the evolution of single solution elements as well as for the evolution of entire system models. System models consists of multiple solutions. These solutions either also consist of further solutions or refer to their solution parameters.

### 13.3.2 Evolving the Solution Using Physical Behavior Models

Models and workflows, which are added to the principle solution in Fig. 13.8 allow for virtual verification of the requirements, the principle solution has to fulfill. When using behavior models, we find that each model has a specific demand on parameters from the principle solution. Some models need less parameters and may be used at an early stage of design, while other models represent a high-fidelity solution requiring lot more parameters (Weber 2014). When integrating these models into our solution model we have to link the model’s input parameters to the already given ones and may add new parameters. Further, models can consist of model chains that might be linked, too. In that case, outputs from one model serve as inputs of another one. Using the example of the electric engine, we derive parameter groups to be linked to the models.

As an example, Fig. 13.9 shows a still relatively simple lifetime-test workflow. The workflow combines two behavior models of the solution model electric engine, one NVH model (NVH, noise, vibration and harshness) in order to predict the torsional vibration



**Fig. 13.9** The use of physical behavior models in test workflows during development evolves the principle solution by adding new parameters, refining geometry and physics, as well as developing components from principle solution active surfaces

level of the electric engine and one life time prediction model that uses the result in order to predict the lifetime of the electric engine. The workflow models the execution order of the behavior models and the parameter interdependencies. As the parameter need of some models is satisfied by other models, the execution order has to be modelled in the workflow to ensure parameter availability for the later ones.

In Fig. 13.9, the input parameters required for the usage of the NVH-model are shown in detail. Every model demands a specific set of parameters in order to be executable. Thereby, we differentiate four basic types of parameters:

- **Principle solution parameters** are already given in the principle solution in Fig. 13.8 and hence determined (in the example:  $dStator$ ,  $dRotor$  and  $L$ , initial principal solution parameters). They can directly be linked to the model.
- For some additional parameters, we need to have more detailed information on the principle solution's active surfaces or physics. We have to continue development of the solution element, refine the active surfaces, material or physics (we add conductor elements to the rotor's active surface) and set further **solution parameters**



(e.g. *dConductor* and *nConductor*). These parameters describe the solution itself in more detail.

- Some parameters require a further level of detail regarding the functional flow descriptions, which enter the solution via ports. For the NVH model, we need information not only about current as part of electric power, but also about the separation in d- and q-current. We add these **flow parameters** (e.g. *dCurrent*, *qCurrent*) to the functional flow, as they are inputs to the principle solution.
- Some parameters depend on the components in which the active surfaces are integrated. Components store multiple active surfaces and connect them by connecting structures. The connecting structure carries physical parameters as mass and inertia. These **structural parameters** can be obtained from CAD (e.g. *inertiaRotor*). As shown in Fig. 13.9, the solution element's active surfaces are the link to CAD-components, which can then be combined in assemblies and finally be aggregated to the component view of the system under development, while the system's solution elements provide a link to the functional view.

Continuing development, we are able to verify requirements using more and more sophisticated behavior models. More complex behavior models ask for more parameters and we continue evolving the solution elements and their attached components by integrating required parameters into the architecture, as the system under development increases maturity.

At a certain point of development, certain models may need a more detailed description of further, surrounding principle solutions in order to increase the detail level of behavior prediction. E.g. the stiffness of the supporting bearing system has influence on the electric motor's air gap and resulting electromagnetic forces. In this case we need a behavior model, which includes the bearing stiffness. We then increase the system scope by combining principle solution elements in a system solution element, which aggregates them. This system solution element itself does again contain behavior models and workflows. However, it does not only contain one principle solution but the solution elements of its sub solutions with connections between them. The behavior models stored in solution elements of system solutions are usually per principle solution less detailed but do have a wider system scope and consider interactions between multiple principle solutions.

Storing the whole set of principle solution, behavior models and workflows enables automated solution design of solutions from libraries. In order to set up these libraries, model classification becomes crucial. The list of behavior models needs to be classified and parameter dependencies need to be modelled. At this point, we propose model ontologies as the required next step. Therefore, an outlook on ontology development for physical behavior models is given in Sect. 13.4.

### 13.4 Model Frameworks and Ontologies for Efficient Model Re-Use

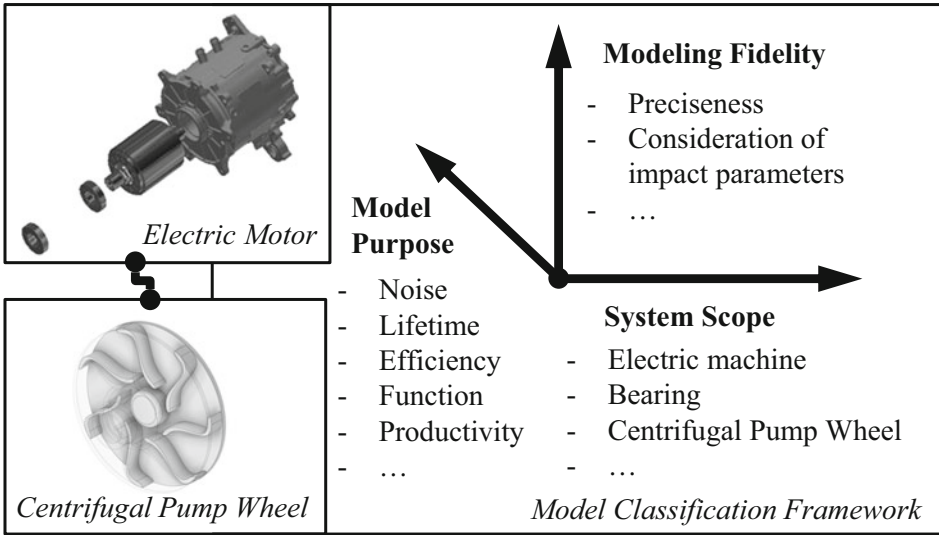
The solution library derived in 13.2 and 13.3 allows for a reuse of (principle) solution elements with their corresponding models in the enterprise context and classified behavior models allow the functional verification and the virtual validation of the system behavior. Expert models thus hold a key position in system development. The efficient use of expert models allows for agile development and continuous system testing. However, the solution element can only be used efficiently with a strong reusability of behavior models. In preparation for or as part of the introduction of a seamless function-oriented model-based product development, the existing behavior models have to be classified, characterized, restructured and thus prepared for efficient reuse in product development processes.

For integrating behavior models in solution elements as proposed in 13.3, behavior models have to be set up in a function oriented, modular way. In order to classify the modular behavior models, we propose a framework consisting of the three axes system scope, model purpose and model fidelity, see Fig. 13.10.

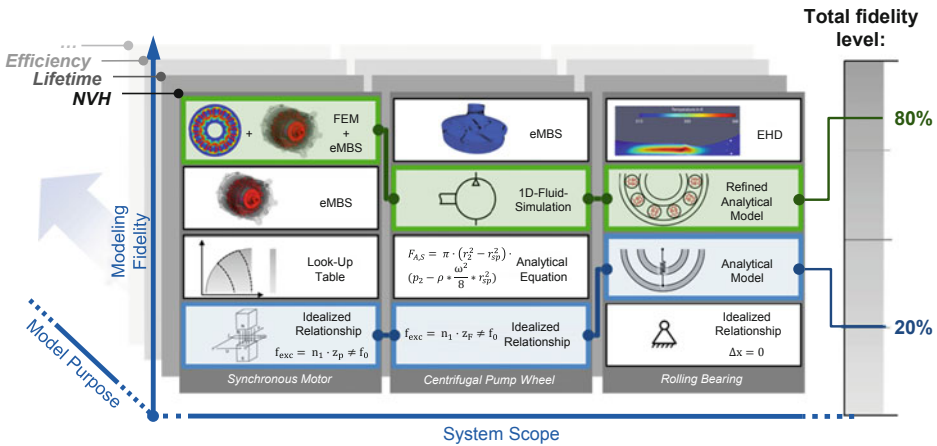
Structuring models in such a framework results in a multidimensional classification matrix for behavior models. Per model purpose, we can describe the solutions in scope and per solution identify multiple modelling fidelities with increasing detail level and parameter request. The overall behavior model for a system solution is then one possible combination of principle solution behavior models in the matrix. When using many model elements with high fidelity, we increase the total fidelity level, but also the required parameters and calculation effort, Fig. 13.11.

The classification of expert models in the coordinate system of Model Fidelity, Model Purpose and System Scope results in a classification scheme for expert models. However, models are also characterized by the input parameters they use, the output parameters they produce and the connection of their sub-models. Classification is only a first step for model reusability. By establishing standardized interfaces, expert models can be connected efficiently to handle parameter connections. Characterizing models and their corresponding parameters is required. Ontologies are a way to characterize models in that regards. Ontologies in this context describe the parameters that belong to a model and their interdependencies with each other in a formal way. Building ontologies allows for formalization and identification of dependencies and relationships between the models. That enables efficient reuse of the models to validate specific issues in development processes. Model ontologies and standardized workflows allow semi-automatic up-load into solution models and further into entire system models as discussed in 13.3.

By consistently organizing product parameters in a function-oriented architecture and linking classified behavior models to it, the presented method represents an excellent starting point for participation in highly relevant and current international research priorities such as the IEA Wind TCP Task 37, which is coordinating international research activities to analyze wind power plants as holistic systems. Within the scope of these research activities, the development of a Systems Modelling Framework and Ontology for Wind Turbines plays a key role (Dykes et al. 2017).



**Fig. 13.10** Framework to classify expert models



**Fig. 13.11** Order matrix of expert models consisting of framework and ontologies

### 13.5 Summary and Conclusion

In summary, the presented method allows for consequent use of virtual behavior prediction using models throughout product development, beginning in early concept phase. It closes the gap between top level function development and expert component design and allows for reliable design decisions based on virtual models for CPS in industrial practice. Agile

development with changing requirements is supported using structured functional verification processes in virtual product development. Repetitive design steps can be partially automatized and are accessible in company specific solution libraries, allowing for efficient reuse, especially in change processes. To enable the future use of expert models in solution models and MBSE, behavior models need to be organized in a clear structure with well described interfaces. Classification framework and model characterizing ontologies hence are enablers for future's model-based design process.

---

## References

- Albers A, Zingel C (2011) Interdisciplinary systems modelling using the contact and channel-model for SYSML. In: Culley SJ (ed) ICED 11: impacting society through engineering design, København, the 18th international conference on engineering design. Design Society, Glasgow, pp 196–207
- Alur R (2015) Principles of cyber-physical systems. MIT-Press, Massachusetts
- Andary F, Berroth J, Jacobs G (2019) An energy-based load distribution approach for the application of gear mesh stiffness on elastic bodies. *J Mech Des* 141(9)
- Broy M (2010) Cyber-physical systems innovation Durch software-intensive Eingebettete Systeme. Springer, Berlin, Heidelberg
- Drave I, Hillemacher S, Greifenberg T et al (2019) SMArDT modelling for automotive software testing. *Softw Pract Exp* 49(2):301–328
- Drave I, Rumpe B, Wortmann A et al (2020) Modelling mechanical functional architectures in SysML. In: Proceedings of the 23rd ACM/IEEE international conference on model driven engineering languages and systems. Association for Computing Machinery, New York, pp 79–89
- Dykes K L, Zahle F, Merz K, et al (2017) IEA wind task 37: systems modelling framework and ontology for wind turbines and plants. National Renewable Energy lab. (NREL), golden
- Eigner M, Gilz T, Zafirov R (2012) Proposal for functional product de-scription as part of a PLM solution in interdisciplinary product development. In: Marjanović D, Storga M, Pavkovic N, Bojčetić N (eds) DESIGN 2012 proceedings of the 12th international design conference, Dubrovnik, May 2012, UNIZAG-FSB 2012, pp 1667–76
- Feldhusen J, Grote K-H (eds) (2013) Pahl/Beitz Konstruktionslehre Methoden und Anwendung erfolgreicher Produktentwicklung, vol 8. Springer, Berlin
- Gausemeier J, Dorociak R, Pook S, et al (2010) A computer-aided cross-domain modelling of mechatronic systems. In: Marjanović D, Storga M, Pavkovic N, Bojčetić N (eds) DESIGN 2010 proceedings of the 11th international design conference, Dubrovnik, May 2010, pp 723–32
- Göhlich D, Fay TA (2021) Arbeiten mit Anforderungen: Requirements Management. In: Bender B, Gericke K (eds) Pahl/Beitz Konstruktionslehre. Springer Vieweg, Berlin, Heidelberg
- Graessler I, Hentze J (2020) The new V-model of VDI 2206 and its validation. *Automatisierungstechnik* 68(5):312–324
- Graessler I, Oleff C, Scholle P (2020) Method for systematic assessment of – requirement change risk in industrial practice. *Appl Sci* 10:8697–8725
- Grundel M (2017) Ein Modell zur Überbrückung der derzeitigen Lücke in der modellbasierten Konzeption mechanischer Systeme. Institut für Konstruktions und Fertigungstechnik, vol 45
- Höpfner G, Jacobs G, Zerwas T, et al (2021) Model-based design workflows for cyber-physical systems applied to an electric-mechanical coolant pump. In: IOP Conference Series: Materials Science and Engineering 1097, 1

- Hubka V, Eder WE (1988) Theory of technical systems. A total concept theory for engineering design. Springer, Berlin, Heidelberg
- Koller R (1998) Konstruktionslehre für den Maschinenbau Grundlagen zur Neu- und Weiterentwicklung technischer Produkte mit Beispielen, vol 4. Springer, Berlin
- Koller R, Kastrup N (1994) Prinziplösungen zur Konstruktion technischer Produkte. Springer, Berlin
- Konrad C, Riedel R, Rasor R et al (2019) Enabling complexity management through merging business process modelling with MBSE. In: Procedia CIRP. 29th CIRP design conference, vol 84. Elsevier, Amsterdam, pp 451–456
- Lamm JG, Weilkiens T (2010) Functional architectures in SysML. Proceedings of the TdSE
- Möser G, Kramer C, Grundel M et al (2016) Fortschrittsbericht zur modellbasierten Unterstützung der Konstrukteurstätigkeit durch FAS4M. In: Schulze S-O, Muggeo C (eds) Tag des systems engineering. Hanser, München, pp 69–78
- Pasch G, Jacobs G, Höpfner G et al (2019) Multi-domain simulation for the assessment of the NVH behaviour of a tractor with hydrostatic-mechanical power Split transmission. LandTechnik AgEng:19–27
- Patzak G (1982) Systemtechnik - Planung komplexer innovativer Systeme. Grundlagen, Methoden, Techniken. Springer, Berlin, Heidelberg
- Ponn J, Lindemann U (2011) Funktionen. In: Konzeptentwicklung und Gestaltung technischer Produkte. VDI-Buch. Springer, Berlin, Heidelberg
- Ross DT, Schoman KE (1977) Structured analysis for requirements definition. IEEE Trans Softw Eng SE-3(1):6–15
- Roth K (1994) Konstruieren mit Konstruktionskatalogen Band 1: Konstruktionslehre, 2nd edn. Springer, Berlin
- Rupp C (2014) Requirements-Engineering und –Management: Aus der Praxis von klassisch bis agil, 6th edn. Carl Hanser Verlag, München
- Srinivasan V, Chakrabarti A, Lindemann U (2012) A framework for describing functions in design. In: Marjanović D, Storga M, Pavkovic N, Bojcetic N (eds) DESIGN 2012 proceedings of the 12th international design conference, Dubrovnik, May 2012, UNIZAG-FSB 2012, pp 1111–1122
- Suh N (1998) Axiomatic design theory for systems. Res Eng Des 10:189–209
- VDI 2221 Blatt 1:2019–11 Entwicklung technischer Produkte und Systeme - Modell der Produktentwicklung, 2019–11
- VDI 2221 Blatt 2:2019–11 Entwicklung technischer Produkte und Systeme - Gestaltung individueller Produktentwicklungsprozesse, 2019–11
- Weber C (2014) Modelling products and product development based on characteristics and properties. In: Chakrabarti A, Blessing LTM (eds) An anthology of theories and models of design. Springer, London, pp 327–352
- Weilkiens T (2016) SYSMOD – the systems modelling toolbox. Pragmatic MBSE with SysML. MBSE4U
- Zerwas T, Jacobs G, Spütz K et al (2021) Mechanical concept development using principle solution models. In: IOP conference series: materials science and engineering 1097, 1