Forschung im Ingenieurwesen (2025)89:60
https://doi.org/10.1007/s10010-025-00827-w

ORIGINALARBEITEN/ORIGINALS

SysML v2 based modelling guidelines for mechanical system elements

Kathrin Boelsen[1] · **Matthias May**[1] · **Georg Jacobs**[1] · **Manuel Mennicken**[1] · **Frederik Moers**[1] · **Thilo Zerwas**[1] · **Gregor Höpfner**[1]

Received: 6 September 2024 / Accepted: 4 March 2025

Abstract

One approach to tackle the complexity in the development of cyber-physical systems is system decomposition and modelling in a central system model with model-based system engineering (MBSE). However, MBSE also requires increased development effort due to the high degree of formalisation of the development results using the Systems Modelling Language (SysML). One solution is reusing system elements from model libraries, which requires formalization that allows them to be networked into consistent models. Due to the abstract nature of SysML, the options for formalizing system elements are extremely diverse, meaning that even identical technical system elements are modelled with a different structure by engineers. As a consequence, no standardized structure is maintained and the reuse of system elements from model libraries is very unlikely. This leads to high, redundant modelling efforts as well as a considerable error potential and significantly hinders the economic application of MBSE in mechanical engineering. A standardized structure of the system elements can be ensured through MBSE methodology specific modelling guidelines. Current modelling guidelines for MBSE are based on the SysMLv1 language standard and cannot be used unchanged in SysML v2. Nevertheless, SysML v2 offers great potential for modelling reusable system elements, especially with its synonymous textual and graphical modelling capabilities, enabling the easy linkage of knowledge from various domain-specific tools. Consequently, modelling guidelines based on SysML v2, which guarantee a standardized structure, are required to fully exploit the potential of model libraries in MBSE. To solve this deficit, this paper presents modelling guidelines based on SysML v2 and their application for building reusable system elements.

1 Introduction

Technical systems are becoming increasingly more complex as they integrate multiple domains such as electronics or software [1, 2]. Technical systems which combine mechanical, electronic and software subsystems, which are interconnected with the internet of things and services, are referred to as cyber-physical systems (CPS) [3]. A particular challenge in the development of CPS is mastering the complexity resulting in development, which results from considering the CPS as a whole with its discipline-specific mechanical, electrical and software subsystems and their interactions [4–7]. An approach for mastering this complexity is Model-Based Systems Engineering (MBSE) [8–11]. MBSE extends the classic systems engineering approach by a centralized system model, in which the CPS as a whole with its discipline-specific subsystems is clearly described and all development parameters are interconnected to define a single point of truth (SPOT) [12]. To form the system model, the modelling language provides a system of symbols and characters (notation) and rules for linking theses (syntax) with which the CPS is modelled [13, 14].

✉ Kathrin Boelsen
kathrin.boelsen@imse.rwth-aachen.de

[1] Institute for Machine Elements and Systems Engineering, Schinkelstr. 10, 52062 Aachen, Germany

2 State of the art and research need

For modelling CPS in system models in the context of MBSE, the graphical modelling language SysML (Systems Modelling Language) v1 is established [15]. A significant disadvantage of SysML v1 is the lack of interoperability between different modelling environments, i.e. between different software platform in which SysML v1 is implemented [15]. To improve this disadvantage of SysML v1, the revised SysML v2 contains a standardised, synonymous textual notation and syntax in addition to the graphical notation and syntax [15, 16]. As the textual notation does not

Published online: 01 April 2025

dependent on the environment-specific file format of the system model, interoperability is improved. The modelling of CPS with SysML v2 requires a high degree of formalization. This results in an increased effort in MBSE compared to classic, document-based development approaches [17]. One approach to reduce these barriers in the application of MBSE in industrial practice is to increase the reuse of modelled system elements [18, 19]. To maximize reuse of system elements, system elements must be modelled in a way that they occur unmodified in as many CPS as possible and must be formalized in a way that they can be combined with each other [5, 20]. However, the general-purpose character of SysML v2 currently allows the same system elements to be modelled differently by different engineers. This leads to a high risk that system elements are modelled redundantly and not reused and are also formalized in different ways so that they cannot be aggregated into a consistent overall model. In order to minimize this risk and thus promote the reuse of system elements, modelling guidelines are required that provide a standardised SysML v2 notation and syntax for modelling system elements of CPS.

In MBSE, modelling guidelines are usually assigned to a specific MBSE methodology. The MBSE methodology describes the process for the virtual modelling of the system elements of CPS, i.e. how CPS are developed. In addition, modelling guidelines describe which notation and syntax of a modelling language should be used in the individual steps of the process. Irrespective of the associated SysML v1-based modelling guidelines, the common MBSE methodologies can be used for the development of CPS with SysML v2. However, the SysML v2 has a generic character, implying that a different notation and syntax can be chosen for modelling system elements in the process steps of the MBSE methodologies.

As modelling with abstract modelling languages such as SysML v2 in particular is new to the discipline of mechanics and not yet widely established, the selection of a suitable notation and syntax is particularly challenging for engineers when modelling mechanical system elements. For this reason, SysML v2-based modelling guidelines for mechanical system elements are introduced in this paper as a first step. The extension of the modelling guidelines for the electronics and software discipline is ensured by considering the requirements of these disciplines when selecting the MBSE methodology.

In summary, SysML v2-based modelling guidelines are needed. However, currently there are no modelling guidelines for mechanical system elements based on the modelling language SysML v2. This results in the following research questions:

- Which requirements must be fulfilled by the modelling guidelines for reusable mechanical system elements?
- Which SysML v2 notation and syntax are best suited for modelling reusable mechanical system elements under consideration of the requirements?

3 Method

The goal of this paper is the development of SysML v2-based modelling guidelines for reusable mechanical system elements. The method for developing the modelling guidelines is divided into four steps: To ensure the reuse of modelled system elements in a wide variety of CPS, the modelling guidelines need to fulfil certain requirements. These requirements are determined in the first step of the method. Since modelling guidelines in MBSE are always integral to an MBSE methodology and the essential properties of the modelling guidelines depend largely on the selected MBSE methodology, an MBSE methodology is selected in the second step based on the identified requirements. In order to capture the influences of the MBSE methodology on the modelling guidelines, the requirements for the modelling guidelines in SysML v2 are specified in the third step. In the final step of the method, the usable notation and syntax of the SysML v2 for the modelling of mechanical elements is identified and the most suitable is selected based on the specified requirements.

3.1 Requirements for modelling reusable mechanical system elements

In MBSE of CPS, the system elements of the mechanical, electronic and software disciplines must be harmonized during the entire development process so that they can be aggregated to form the overall CPS [1, 3]. To achieve this, a well-established approach in design methodology is the solution-neutral description of the desired system behavior with functions and their decomposition into sub-functions [3]. In order to avoid failures in the subsequent aggregation of the partial solution into the overall system, a detailed and precise description of the interfaces to each other is required when defining the sub-functions [21]. This results in the first requirement for the modelling guidelines and the associated MBSE methodology:

Requirement 1 The specification of the mechanical system elements must be function-oriented and include a detailed definition of the subfunctions and their interfaces.

While the functions are used to describe the desired behavior of the subsystems and their interfaces in a solution-neutral way, the logic level is used in the development of CPS to describe conceptually how the desired behavior is realized by discipline-specific system elements, i.e. how the incoming function flows via the interfaces of the sub-functions are transformed into the outgoing function flows [22]. In order for the system elements and the associated functions to be used as extensively as possible in the development of various CPS, they must describe a suitable scope of the system. This scope must be defined in such a way that the system elements occur repeatedly in as many different systems as possible, have a meaningful information content and the modelling is economically viable [23]. This results in the second requirement for the modelling guidelines and the associated MBSE methodology:

Requirement 2 The scope of the mechanical system elements must be broad enough to contain meaningful information, but small enough so that the system elements can be reused in system models for different purposes and CPS.

To ensure the SPOT of the system model, a detailed modelling of the relationships between the system elements on the individual abstraction levels is required [24, 25]. Fundamentally, this can mean exclusively modeling traceability relationships. However, this qualitative modeling of dependencies does not ensure that changes to individual parameters have an automated effect on all dependent areas of the system model. Consequently, only by quantifying the dependencies the system model as a SPOT is ensured at any time without manual effort. In the case of mechanical system elements, when linking the system elements at the functional and logic levels, it must be described how the parameters of the interfaces of the functions are related to the geometric and physical parameters of the mechanical, electrical or software system elements [26]. This results in the third requirement for the modelling guidelines and the associated MBSE methodology:

Requirement 3 The specification of the mechanical system elements must include a parametric description of the solution concept, which enables a consistent linking of the mechanical system elements with other system elements.

3.2 Selection of the MBSE methodology

Currently various MBSE methodologies exist that pursue different approaches to the development of CPS. In particular, the MBSE methodologies *motego* [5, 27, 28], *Functional Product Description/mecpro²* [29], *Mechatronic Modeller* [30], *SPES* [31], *FAS4M* and System Sketcher [32, 33], *Integrated Product Model for Conceptual Design* [34], *SYSMOD* [35], *Digital Function Modeling in DC43* [36], *MagicGrid* [37] and *OpenMBEE* [38] need to be mentioned here. The following section describes how far these MBSE methodologies fulfil the previously defined requirements for reusable mechanical system elements.

Requirement 1 While all MBSE methodologies considered pursue a function-oriented approach, the integration of interfaces to connect elements across different domains varies. To connect between all three domains, the information flows between the functions should be differentiated into the three types energy, signal and material [39]. Some methodologies (e.g. *OpenSE/OpenMBEE* [38]) focus on the modeling of signal flows. The *Digital Functional Modeling in DC43* [36] does not specify the usage of functional flows at all. *MagicGrid* [37] does not explicitly mandate the differentiation of function flows and leaves categories to be defined by users. Interfaces to connect elements to other elements of the mechanical, electrical or software domains are only found in the methodologies *Functional Product Description/mecpro²* [29], *motego* [27], *FAS4M* [32], *Integrated product Model for Conceptual Design* [34] and *SYSMOD* [35].

Requirement 2 As in mechanical engineering, the interactions of components occur in their physical contact, this modelling approach is the best suited to define a scope of system elements within the logical abstraction, which ensures a high degree of reusability. Only the methodologies *FAS4M* [32] and *motego* [27] explicitly define the lowest reusable mechanical system element on the physical contact level. Beyond that, *motego* [27, 28] allows the extensive integration of simulation models from a range of software environments for virtual verification (see [40–43]). The methodology *FAS4M* [32] also allows for the modelling of physical contacts but does so using sketches thus not allowing the integration of virtual testing.

Requirement 3 There are generally two different approaches to define the relation between the concept level and other abstraction levels in the considered methodologies. The first approach is the creation of a qualitative association between the levels. This is used in the methodologies *FAS4M* [32], *Integrated Product Model for Conceptual Design* [34] and *SYSMOD* [35]. For *FAS4M* for example, this is made visible by the choice of connection in the SysML v1 application, where the connection type *trace* is chosen, which does not enable the transfer of parameters. The second approach is used by *motego* [27]. Here, a consistent connection of parameters on the concept level with parameters on the product level is established. In its SysMLv1-application it uses *ports* which allow for a full transfer of all information. This allows the quantification of the dependencies between the system elements.

Table 1 summarizes the described degree of fulfilment of the requirements by the mentioned MBSE methodologies. As the *motego* methodology is the only MBSE methodology which fulfill all requirements for the development of CPS with reusable, mechanic system elements, it is cho-

Table 1 Qualitative overview of the analysed state-of-the-art MBSE methodologies

Approach	Requirement 1	Requirement 2	Requirement 3
Motego [5, 27, 28]	●	●	●
Functional Product Description/mecpro² [29]	●	○	○
Mechatronic Modeller [30]	◐	○	○
SPES [31]	◐	○	○
FAS4M and System Sketcher [32, 33]	●	◐	◐
Integrated Product Model for Conceptual Design [34]	●	○	◐
SYSMOD [35]	●	○	◐
Digital Function Modeling in DC43 [36]	◐	○	○
MagicGrid [37]	◐	○	○
OpenMBEE [38]	◐	○	○

sen for establishing the modelling guidelines for reusable mechanical system elements based on SysML v2.

3.3 Specification of the requirements for mechanical system elements

Although the chosen methodology, i.e. the description of the processes involved in the development of CPS, is described in current publications exclusively for modelling with SysML v1 and a domain-specific modelling language based on SysML v1 (cf. Spütz et al. [27]), it is also generally applicable to other modelling languages such as SysML v2. In order to consider the essential characteristics of the *motego* methodology for modelling guidelines based on SysML v2, the main features of the *motego* methodology are presented below and based on these, the requirements for the modelling guidelines are specified.

Figure 1 shows the main steps of the *motego* methodology that are relevant for modelling reusable mechanical system elements: the functional decomposition of the system and the development of mechanical system elements, which realizes the decomposed functions.

The scope of the CPS subsystems and thus its elements is already defined at the functional level of the *motego* methodology through functional decomposition. Based on Koller's design methodology, the functions are decomposed down to their most elementary level [26]. These elementary functions form the lowest functional level in *motego*. Functions that can be decomposed are referred to as functional architecture. According to Koller, elementary functions are defined by incoming and outgoing function flows and an operator that transforms these flows into each other [39]. Based on this, requirement 1 for the modelling guidelines (see Sect. 4.1) can be refined as follows:

Requirement 4 The modelling guidelines must include a notation and syntax for the decomposition of functional architectures into further functional architectures and elementary functions.

Based on the same approach by Koller, the interfaces between functions can be defined as well. Koller differentiates the type of interfaces between the functions into material, energy and signal flows, which are further described using parameters in *motego* [26]. For example, the rotational energy flow is described by torque, rotational speed and the resulting power. The following requirement for the modelling guidelines is derived therefrom as a refinement of requirement 1:

Requirement 5 The modelling guidelines must include a notation and syntax for the definition of material, energy and signal flows and their parameters as interfaces of the functions.

The functional architecture, including function decomposition and interface descriptions, forms the basis for defining system elements in the logical level of the *motego* methodology. This logical level is referred to as the solution level in the *motego* methodology. The realization of elementary functions is described by solution elements, which are mechanical system components defined by a physical effect and the location at which the effect occurs (cf. principle solution definition according to Koller [26]). The location is in turn defined by a pair of active surfaces with a specified geometry and a material. In mechanics, the solution elements represent a meaningful section of the system, as they occur in a finite number in different CPS (cf. Jagla et al. [44]). For example, the solution element *LubricatedMechanicalLineRollingContact* shown in Fig. 1 can be used in both gear pairings and cylindrical roller bearings and can be combined with other solution elements to create system solutions. This leads to the following requirement for the modelling guidelines as a refinement of requirement 2:

Requirement 6 The modelling guidelines must include a notation and syntax for the definition of solution elements consisting of a physical effect and a set of active

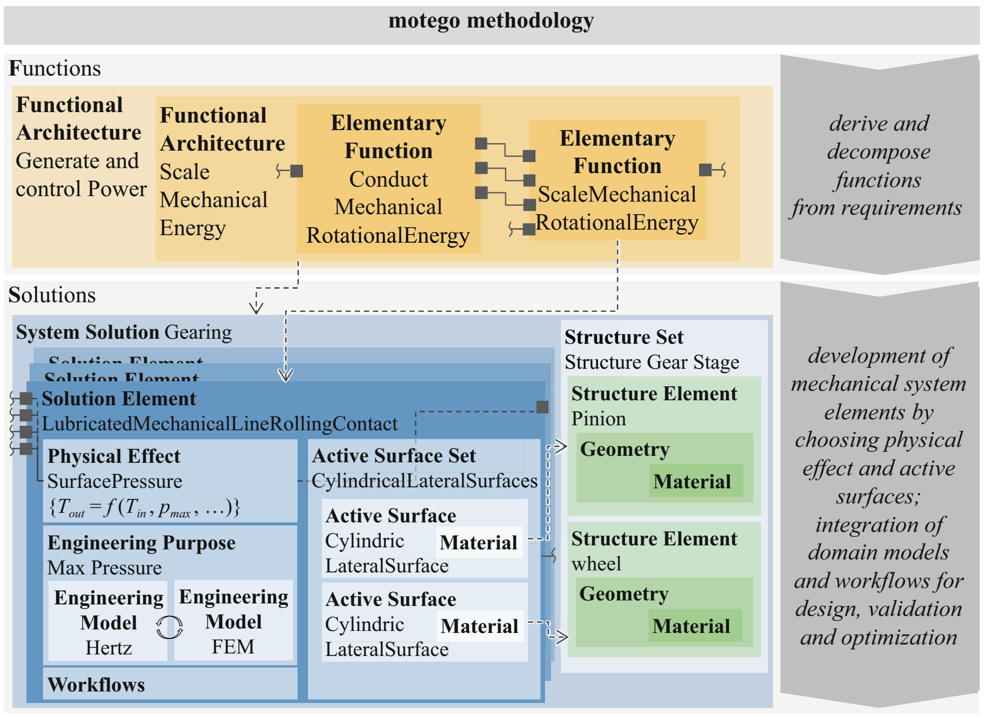

motego methodology
Functions
Functional Architecture Generate and control Power
Functional Architecture Scale Mechanical Energy
Elementary Function Conduct Mechanical RotationalEnergy
Elementary Function ScaleMechanical RotationalEnergy
derive and decompose functions from requirements
Solutions
System Solution Gearing
Solution Element
Solution Element LubricatedMechanicalLineRollingContact
Physical Effect SurfacePressure $\{T_{out}=f(T_{in}, p_{max}, \dots)\}$
Engineering Purpose Max Pressure
Engineering Model Hertz
Engineering Model FEM
Workflows
Active Surface Set CylindricLateralSurfaces
Active Surface Cylindric LateralSurface Material
Active Surface Cylindric LateralSurface Material
Structure Set Structure Gear Stage
Structure Element Pinion
Geometry
Material
Structure Element wheel
Geometry
Material
development of mechanical system elements by choosing physical effect and active surfaces; integration of domain models and workflows for design, validation and optimization

Fig. 1 Excerpt of the process for developing CPS according to the motego methodology according to [27]

surfaces, which is described by the geometry and the material of the active surfaces.

The physical effect is described as an executable simulation model that defines the physical relationship between the input and output flows of the elementary function to be realized. To quantify functional flows, the location of the effect is also specified via the geometry and material parameters of the active surfaces. Therefore, the following requirement for the modelling guidelines is derived as a refinement of requirement 3:

Requirement 7 The modelling guidelines must include a notation and syntax for linking elementary functions and mechanical system elements, which makes it possible to describe the relationships between the physical effect and incoming and outgoing functional flows.

In addition to the physical effect and the active surface set, the solution element contains domain models and workflows for the design, verification and optimization of the solution elements (cf. [45–47]). This is crucial to maximize the reuse of company knowledge across various domains and tools by the system elements. Integrating domain models and their knowledge enriches the solution elements' information, leading to a modelling guidelines requirement as a refinement of requirement 2:

Requirement 8 The modelling guidelines must include a notation and syntax enabling the integration of domain models for the design, verification and optimization of mechanical system elements.

3.4 SysML v2-based modelling guidelines for the application of the *motego* methodology

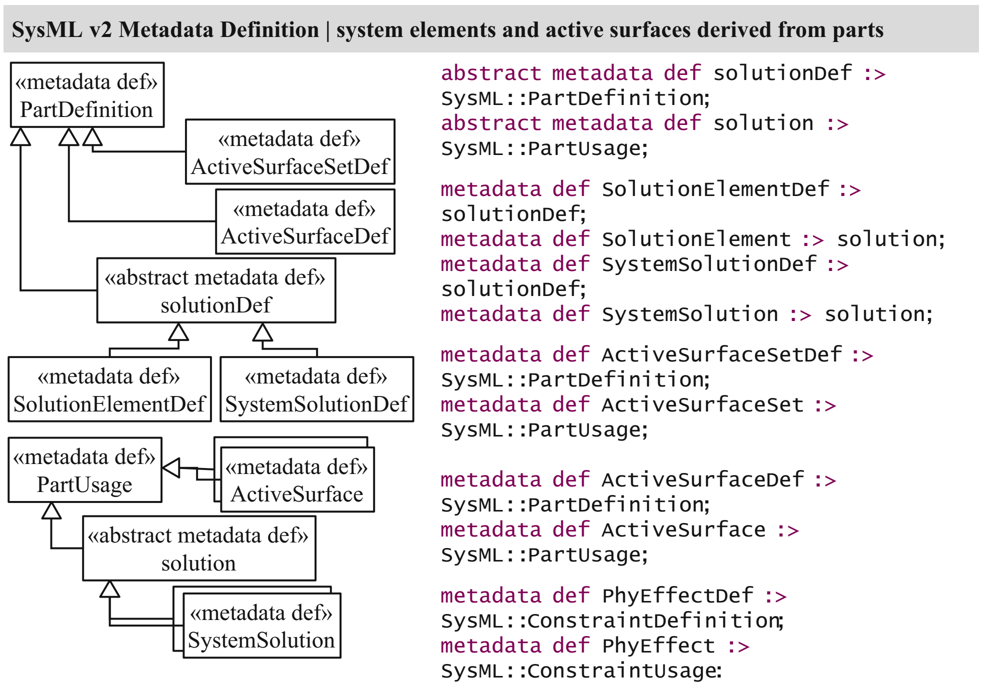
In this chapter, the SysML v2 modelling guidelines for the function and logic layer are specified. These are needed to model mechanical system elements in the modelling language with the *motego* methodology.

3.4.1 SysML v2-based modelling guidelines for mechanical system elements

To model the solution elements of the *motego* methodology, requirement 6 applies. Solution elements consist mainly of the physical effect and the surfaces on which these effects take place. Physical effects define the relationships between the input and output of solution elements and are represented by equations. These are generally modelled in the SysML v2 either via *calculations* or *constraints*. Both can be specified with *input parameters*. Beyond that, *constraints* can return implicit *Boolean values* which will help fulfilling requirements in the future. *PhysicalEffects* are therefore created as an element of the type *constraint*. The output of a *constraint* can be created as an *interior attribute* and called from the superordinate element (see example in Fig. 5). *ActiveSurfaces* store parameters about the surfaces on which the effects work. They represent a modular unit of structure and are therefore modelled as a *part*.

Requirement 8 also needs to be considered. The integration of simulation models (*DomainModels*) within solution elements must be enabled. These are integrated as

Fig. 2 Metadata implementation of the motego modelling guidelines for mechanical system elements


SysML v2 Metadata Definition | system elements and active surfaces derived from parts
«metadata def» PartDefinition
«metadata def» ActiveSurfaceSetDef
«metadata def» ActiveSurfaceDef
«abstract metadata def» solutionDef
«metadata def» SolutionElementDef
«metadata def» SystemSolutionDef
«metadata def» PartUsage
«metadata def» ActiveSurface
«abstract metadata def» solution
«metadata def» SystemSolution

```
abstract metadata def solutionDef :>
SysML::PartDefinition;
abstract metadata def solution :>
SysML::PartUsage;

metadata def SolutionElementDef :>
solutionDef;
metadata def SolutionElement :> solution;
metadata def SystemSolutionDef :>
solutionDef;
metadata def SystemSolution :> solution;

metadata def ActiveSurfaceSetDef :>
SysML::PartDefinition;
metadata def ActiveSurfaceSet :>
SysML::PartUsage;

metadata def ActiveSurfaceDef :>
SysML::PartDefinition;
metadata def ActiveSurface :>
SysML::PartUsage;

metadata def PhyEffectDef :>
SysML::ConstraintDefinition;
metadata def PhyEffect :>
SysML::ConstraintUsage;
```

constraints for the same reasons as for the *PhysicalEffect*. *Ports* allow for the transfer of *attributes* and *parts* such as active surfaces. To transfer large amounts of information in one element (e.g. for feeding complex domain models), this is beneficial.

In summary, solution elements include *attributes, parts, ports* and *constraints*. To represent these properties of *SolutionElements*, the element *part* is chosen to represent itself as it enables the integration of all necessary elements.

In SysML v2 customized model elements can be created by using metadata definitions. By defining new elements and creating relations between them subsequently a metamodel is built up. The according metamodel for the previously described modelling elements of the *motego* methodology can be seen in Fig. 2. *Parts* are defined with the *PartDefinition* metadata definition and used or reused with the help of the *PartUsage* metadata definition. In first instance, an abstract metadata definition for the general element *solution* is set up as a specialization of the *PartDefinition* metadata definition. From this general element, the specialized cases *SolutionElement* and *SystemSolution* are derived. The *ActiveSurface* and *ActiveSurfaceSet* are also specializations of the *PartDefinition* metadata definition. Similarly, equal metadata definitions are derived from the *PartUsage* metadata definition to enable the usage of the defined solution elements.

As describes, the *PhysicalEffect* is a specialization of the *constraint* meta type. The *Material* is specified as part of the *ActiveSurfaces* and integrated as a reusable element within them. As a structural type, like the active surface, it is defined with the metadata type *part.*

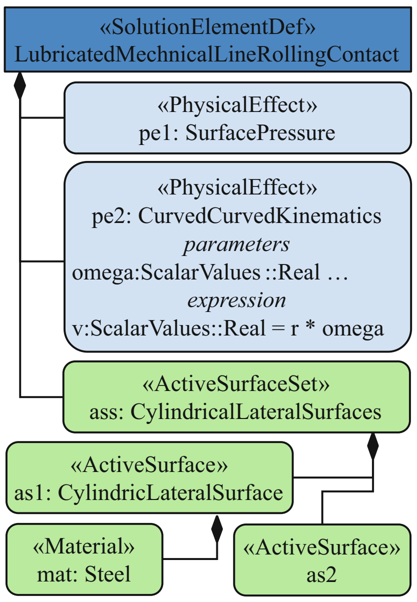
As an exemplary implementation, the solution element for a lubricated mechanical rolling contact in linear direction is considered (see Sect. 4.3 and Fig. 3). The element is initiated with the *#SolutionElementDef* metadata definition from the metadata definitions of the modelling guidelines. It uses the previously defined physical effects and an active surface set which in turn uses the previously defined active surfaces *CylindricalLateralSurface*. The included elements of the solution element which have been described before (e.g. *PhysicalEffect, ActiveSurfaceSet*, etc.) are called explicitly with their corresponding metadata definition (see Fig. 3). The active surface *CylindricalLateralSurface,* integrates the parameters by the definition of *attributes* and the material *steel,* which is defined beforehand and used in the active surface with the *#Material* command. The physical effect gains *attributes* as in- and outputs and *expressions* as calculation specifications are added to complete the effect definition.

3.4.2 Modelling guidelines for functions

For the definition of functions with the SysML v2 modelling guidelines, requirements 4 and 5 are considered. In function-oriented development, functions depict the systems behaviour that is to be realized by solutions in the logical level. Functions must be described hierarchically and in terms of their input-output-relationships. There are two possible modelling variants for the definition of a function with input and output flows and hierarchical orders in which the use of the analogue SysML v1 language elements is already proven: *action* and *part* (see [28, 37, 48]). *Actions* have the

Fig. 3 exemplary implementation of a solution element using the motego SysML v2 modelling guidelines

Graphical Modelling Guidelines


«SolutionElementDef»
LubricatedMechnicalLineRollingContact
«PhysicalEffect»
pe1: SurfacePressure
«PhysicalEffect»
pe2: CurvedCurvedKinematics
parameters
omega:ScalarValues ::Real …
expression
v:ScalarValues::Real = r * omega
«ActiveSurfaceSet»
ass: CylindricalLateralSurfaces
«ActiveSurface»
as1: CylindricLateralSurface
«Material»
mat: Steel
«ActiveSurface»
as2

Textual Modelling Guidelines

```
#SolutionElementDef def
LubricatedMechnicalLineRollingContact{
   #PhyEffect pe1:SurfacePressure;
   #PhyEffect pe2: CurvedCurvedKinematics;
   #ActiveSurfaceSet ass:
CylindricalLateralSurfaces;
   …
}

#ActiveSurfaceSetDef def
CylindricalLateralSurfaces{
   #ActiveSurface as1:
CylindricLateralSurface;
   #ActiveSurface as2:
CylindricLateralSurface{
    #Material mat:Steel; }
   …
}

#PhyEffectDef def CurvedCurvedKinematics{
   in omega:ScalarValues::Real;
   v_out:ScalarValues::Real = r * omega;
   …
}
```

advantage, that besides the hierarchical decomposition and integration of input and outputs to functional flows, they also allow the modelling of time executed sequences. As functions depict a behaviour, the ability to describe the sequence of execution is an advantage. Therefore, compared to the structure modelling element *part, actions* are more suited for modelling the function-level of systems and thus used in the suggested modelling guidelines.

As above for solution elements, metadata types are defined for the elementary function (*ElFunction*/*ElFunctionDef*) and functional architecture (*ArchFunction*/*ArchFunctionDef*). For functional flows *MaterialFlow, EnergyFlow* and *SignalFlow* are defined as *attribute definitions*.

In the example of Fig. 4, an excerpt of the functional architecture *ScaleMechanicalEnergy* is shown. The superordinate function is implemented using the *ArchFunctionDef* metadata definition. It uses the previously defined energy flow *mechRotEF* as an input. The subordinate elementary functions *sf1* and *sf2* are inserted to the functional architecture by specifying the functions as previously defined elementary functions.

The elementary functions can be connected in two ways relevant: in regard to their input-output-relationships and their timed execution sequence. The *flow attributes* are connected by the *bind* and *flow* commands, while the sequence is defined by the *first* and *then* commands.

3.4.3 Modelling guidelines for linking elementary functions and mechanical system elements

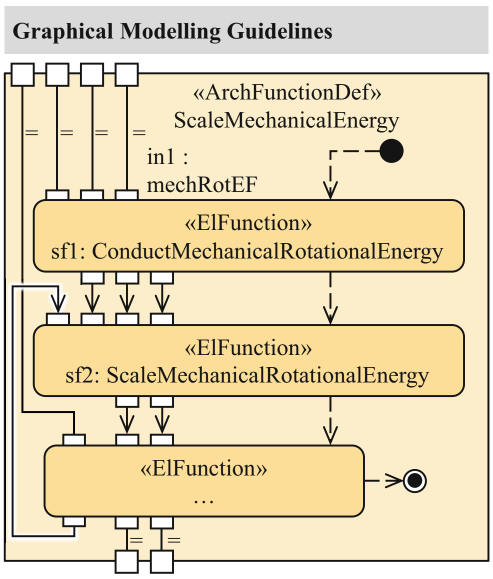
One benefit of modelling functions as *actions* and solution elements as *parts* is the ability of *parts* to *perform actions.* This fulfils requirement 7. The sequence of execution established in the functional level (see Fig. 4) can thus be transferred to the logical level. Figure 5 shows the exemplary solution element definition. The function it realises is integrated with the *perform* command and connected to the corresponding *attributes* of the solution element.

3.4.4 Modelling guidelines for the integration of domain models into mechanical system elements

For the integration of domain models, requirement 8 is considered. The methodology defines the superordinate element *Purpose* that collects all models for a specific simulation purpose to enable swapping between domain models of different fidelity. By accessing only specific purposes, it is enabled that the same solution element can be used for different modelling purposes in system models. To transfer this concept to the modelling guidelines, a metadata types for the *Purpose* and the *DomainModel* level each are defined. Like *PhysicalEffects*, the element *constraint* is used.

To connect models to the other elements in solution elements, *ModelParameters* are defined based on *attributes*. Domain models receive their necessary input parameters from active surfaces, physical effect, in-outputs of the solution element and other models via their *ModelParameters*. Figure 6 shows the implementation of an *EngineeringPurpose* to integrate models calculating the maximal permissible pressure on the surface contact. *GeometryParameters (GeomPar)* are used to transfer information with the *ActiveSurfaces* while an *EngineeringModelParameter (EngModelPar)* is used to connect to the physical effect. Corresponding, the same types are used in the connected ele-

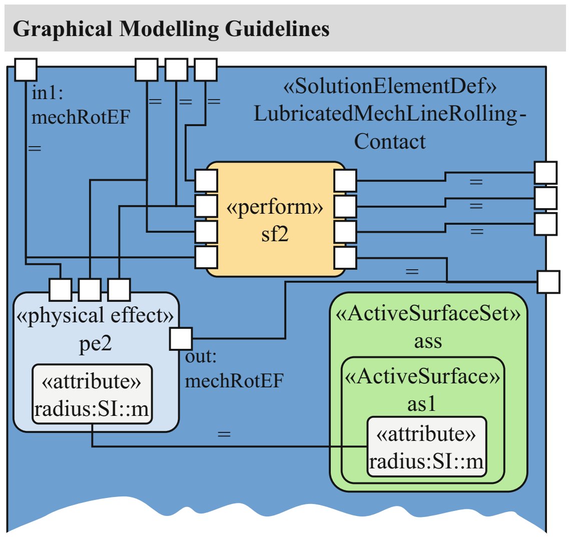
Fig. 4 Application of the SysML v2 based motego modelling guidelines for modelling the functional architecture *ScaleMechanicalEnergy*


Graphical Modelling Guidelines
«ArchFunctionDef»
ScaleMechanicalEnergy
in1 :
mechRotEF
«ElFunction»
sf1: ConductMechanicalRotationalEnergy
«ElFunction»
sf2: ScaleMechanicalRotationalEnergy
«ElFunction»
…

Textual Modelling Guidelines

```
#EnergyFlowDef def mechRotEF{
   attribute P:>ISQ::power;
…}
#ArchFunctionDef def
ScaleMechanicalEnergy {
   in in1: mechRotEF;
   in …
   first start;
   then #ElFunction sf1:
ConductMechanicalRotationalEnergy;
   then #ElFunction sf2:
ScaleMechanicalRotationalEnergy;
   then #ElFunction …;
   then done;
   bind in1=sf1.in1;
   flow from sf1.out1 to sf2.in1;
   …
}
```

Fig. 5 Excerpt from exemplary solution element: linking solution element with function it realises and connecting elements inside the solution element



Textual Modelling Guidelines

```
#SolutionElementDef def
LubricatedMechnicalLineRollingCo
ntact{
   perform sf2;
   #PhyEffect pe2:
   CurvedCurvedKinematics;
   #ActiveSurfaceSet ass:
   CylindricalLateralSurfaces;
   bind in1 = sf2.in1;
   bind sf2.out4 = out4;
   bind pe2.out = out4;
   bind pe2.radius =
   ass.as1.radius;
   …
}
```

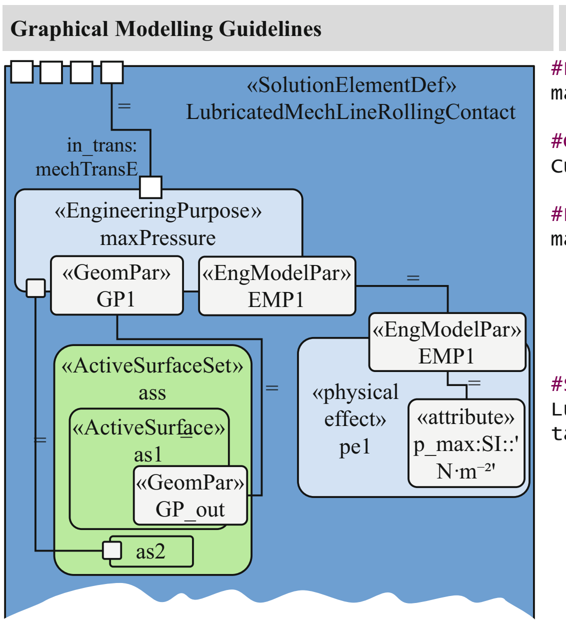
ments. One *EnergyFlow* port feeds the information of the incoming translatory energy in the contact to the model.

4 Conclusion and outlook

MBSE can help handling the complexity and interdisciplinary nature of CPS. The reuse of elements of MBSE system models prevents high individual modelling efforts with considerable error potential and promotes the economic use of MBSE in engineering. To ensure consistency, interoperability, and reusability of system elements, a standardized modelling approach is necessary. The new modelling language SysML v2 promises to enhance the interoperability between different modelling tools and the connection of the MBSE models with complex simulation models. However, currently there are no modelling guidelines for mechanical system elements based on the modelling language SysML v2. Therefore, the goal of this research was the development of modelling guidelines for reusable mechanical system elements based on SysML v2.

In a first step, requirements for a suitable function-oriented methodology were defined. Following an evaluation of various MBSE methodologies, the *motego* methodology was selected as the most suitable, as it fully meets all requirements and enables maximum reusability, the seamless function-orientated development and the integration of simulation models. Requirements for the SysML v2 modelling guidelines are derived from the methodology. Afterwards, the modelling guidelines implementation is discussed.

Functions are modelled as *actions* to take advantage of their ability to establish execution sequences. Logical com-



Textual Modelling Guidelines

```
#EngineeringModelParameterDef def
maxPressure{
   attribute maxP:SI::m;}
#GeometryParameterDef def
Curved_Surf{
   attribute radius:SI::m;}
#EngineeringPurposeDef def
maxP_Fatigue_Line{
      in in_trans:mechTransE;
      out EMP1:maxPressure;
      in GP1:Curved_Surf;
      in GP2:Curved_Surf;}

#SolutionElementDef def
LubricatedMechnicalLineRollingCon
tact{
   #EngineeringPurpose
   maxPressure:maxP_Fatigue_Line;
   bind maxPressure.GP1 =
   ass.as1.GP_out;
   bind maxPressure.EMP1 =
   pe1.p_max_in;
   …}
```

Fig. 6 Exemplary implementation of an *EngineeringPurpose* in the solution element

ponents are modelled as *parts* to enable full integration of calculations, parameter storage and information transfer between the components.

The application of the *motego* methodology with the new SysML v2 language showed that this methodology can be used language-independently by defining suitable modelling guidelines for the required elements of the methodology. With these SysML v2-based modelling guidelines, a reduction of modelling efforts for the engineer and improvement of the consistency of models is targeted. As a first step, the modelling guidelines focus on the fundamental Structure of the functional and logical elements. In future, the modelling guidelines should be expanded for the requirements and physical level. In addition the integration of simulation models from different authoring tools into the shown SysML v2 solution elements and their connection to the SysML v2-API should be explored in detail. To facilitate the application of the presented SysML v2-based modelling guidelines for the motego methodology, metadata definitions were implemented. These can be used for example in the eclipse modelling environment for modelling mechanical system elements. As SysML v1 is currently still frequently used for MBSE, a domain-specific modelling language for modelling mechanical system elements with SysML v1 is also available for the Cameo Systems Modeler modelling environment (see [49]), which ensures the application of SysML v1-based modelling guidelines.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

1. Isermann R (ed) (2010) Elektronisches Management motorischer Fahrzeugantriebe: Elektronik, Modellbildung, Regelung und Diagnose für Verbrennungsmotoren, Getriebe und Elektroantriebe, 1st edn. Vieweg + Teubner, Wiesbaden (mit 24 Tabellen)
2. Eigner M (2021) System Lifecycle Management: Digitalisierung des Engineering. Springer Vieweg, Berlin, Heidelberg
3. Verein Deutscher Ingenieure e. V. (2021) Entwicklung mechatronischer und cyber-physischer Systeme (VDI/VDE 2206)
4. Bradley D, Russell D, Ferguson I et al (2015) The Internet of Things—The future or the end of mechatronics. Mechatronics 27:57–74. https://doi.org/10.1016/j.mechatronics.2015.02.005
5. Jacobs G, Konrad C, Berroth J et al (2022) Function-Oriented Model-Based Product Development. In: Design Methodology for Future Products. Springer, Cham, pp 243–263
6. Masior J, Schneider B, Kürümlüoglu M et al (2020) Beyond Model-Based Systems Engineering towards Managing Complexity. Procedia Cirp 91:325–329. https://doi.org/10.1016/j.procir.2020.02.183
7. VDI Verein Deutscher Ingenieure e. V. (2019) Entwicklung technischer Produkte und Systeme Modell der Produktentwicklung (VDI 2221 Blatt 1)
8. Morkevicius A, Aleksandraviciene A, Mazeika D et al (2017) MBSE Grid: A Simplified SysML-Based Approach for Modeling

Complex Systems. Incose Int Symp 27:136–150. https://doi.org/10.1002/j.2334-5837.2017.00350.x
9. Qamar A, Törngren M, Wikander J et al (2010) Integrating multidomain models for the design and development of mechatronic systems. In: 7th European Systems Engineering Conference EuSEC 2010
10. Vazquez-Santacruz JA, Portillo-Velez R, Torres-Figueroa J et al (2023) Towards an integrated design methodology for mechatronic systems. Res Eng Design 34:497–512. https://doi.org/10.1007/s00163-023-00416-4
11. Di Maio M, Weilkiens T, Hussein O et al (2021) Evaluating MBSE Methodologies Using the FEMMP Framework. In: 2021 IEEE International Symposium on Systems Engineering (ISSE). IEEE, Piscataway, NJ, pp 1–8
12. Estefan JA (2008) Jet Propulsion Laboratory, California Institute of Technology. Survey of Model-Based Systems Engineering (MBSE) Methodologies, Pasadena, California, USA (Rev. B)
13. Duden (2018) Notation. Duden.de
14. Lackes R, Siepermann M (2018) Definition: Syntax einer Programmiersprache. Springer Fachmedien Wiesbaden GmbH
15. Friedenthal S (2018) Requirements for the next generation systems modeling language (SysML® v2). Insight 21:21–25. https://doi.org/10.1002/inst.12186
16. Jansen N, Pfeiffe J, Rumpe B et al (2022) The Language of SysML v2 under the Magnifying Glass. JOT 21(3):1. https://doi.org/10.5381/jot.2022.21.3.a11
17. Wilking F, Schleich B, Wartzack S (2020) MBSE along the Value Chain—An Approach for the Compensation of additional Effort. In: 2020 IEEE 15th International Conference of System of Systems Engineering (SoSE). IEEE, pp 61–66
18. Wu Q, Gouyon D, Hubert P et al (2017) Towards model-based systems engineering (MBSE) patterns to efficiently reuse know-how. Insight 20:31–33. https://doi.org/10.1002/inst.12178
19. Anacker H, Dumitrescu R, Kharatyan A et al (2020) Pattern based systems engineering—application of solution patterns in the design of intelligent technical systems. Proc Des Soc: Des Conf 1:1195–1204. https://doi.org/10.1017/dsd.2020.107
20. Mahboob A, Husung S (2022) A Modelling Method for Describing and Facilitating the Reuse of SysML Models during Design Process. Proc Des Soc 2:1925–1934. https://doi.org/10.1017/pds.2022.195
21. Hybertson DW (2009) Model-oriented systems engineering science: A unifying framework for traditional and complex systems. Complex and enterprise systems engineering. Auerbach, Boca Raton, FL
22. Chauvin F, Fanmuy G (2014) System Engineering on 3DEXPERIENCE Platform—UAS Use Case. CSDM 2014:113–126. https://ceur-ws.org/Vol-1234/paper-11.pdf
23. Thomas O, Scheer A-W (2017) Verfahren und Werkzeuge zur Informationsmodellierung. In: Spath D, Westkämper E, Bullinger H-J, al (eds) Neue Entwicklungen in der Unternehmensorganisation. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 429–457
24. Friedenthal S, Moore A, Steiner R (2015) A practical guide to SysML: The systems modeling language, Third edn. The MK/OMG Press, Elsevier/Morgan Kaufmann, Waltham, MA
25. Alt O (2012) Modellbasierte Systementwicklung mit SysML. Hanser, München
26. Koller R, Kastrup N (1998) Prinziplösungen zur Konstruktion technischer Produkte, 2nd edn. Springer eBook Collection Computer Science and Engineering. Springer Berlin Heidelberg, Berlin, Heidelberg
27. Spütz K, Jacobs G, Zerwas T et al (2023) Modeling language for the function-oriented development of mechatronic systems with motego. Forsch Ingenieurwes. https://doi.org/10.1007/s10010-023-00623-4
28. Wyrwich C, Boelsen K, Jacobs G et al (2024) Seamless Function-Oriented Mechanical System Architectures and Models. Eng 5:301–318. https://doi.org/10.3390/eng5010016
29. Eigner M (2017) Modellbasierter Entwicklungsprozess cybertronischer Systeme. Springer Berlin Heidelberg, Berlin, Heidelberg
30. Gausemeier J, Dorociak R, Pook S et al (2010) Computer-aided cross-domain modeling of mechatronic systems. In: Marjanovic D, Storga M, Pavkovic N, Bojcetic N (eds) DS 60: Proceedings of DESIGN 2010, the 11th International Design Conference. Dubrovnik, Croatia, Seattlee, WA, USA, pp 723–732
31. Pohl K (2012) Model-Based Engineering of Embedded Systems: The SPES 2020 Methodology. Springer Berlin Heidelberg, Berlin, Heidelberg
32. Moeser G, Grundel M, Weilkiens T et al (2017) Modellbasierter mechanischer Konzeptentwurf: Ergebnisse des FAS4M-Projektes. In: Schulze S-O, Tschirner C, Kaffenberger R et al (eds) Tag des Systems Engineering, vol 25.–27. Hanser, München, pp 417–428
33. Moeser G, Kramer C, Grundel M et al (2015) Fortschrittsbericht zur modellbasierten Unterstützung der Konstrukteurstätigkeit durch FAS4M. Tag des. Syst Eng:69–78 https://doi.org/10.3139/9783446447288.008
34. Wölkl S, Shea K (2010) A Computational Product Model for Conceptual Design Using SysML. In: Proceedings of the ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference—2009: Presented at 2009 ASME International Design Engineering Technical Conferences and Computers and Information in. American Society of Mechanical Engineers, New York, N.Y (San Diego, California, USA)
35. Weilkiens T (2016) SYSMOD—the systems modeling toolbox: Pragmatic MBSE with SysML, 2nd edition, version 4.1. MBSE4U booklet series. MBSE4U. Fredesdorf ISBN: 978-3-9817875-8-0
36. Elwert M, Ramsaier M, Eisenbart B et al (2022) Digital Function Modeling in Graph-Based Design Languages. Appl Sci 12:5301. https://doi.org/10.3390/app12115301
37. Aleksandraviˇciene A, Morkeviˇcius A (2021) MagicGrid®Book of knowledge: A Practical Guide to Systems Modeling Using MagicGrid from Dassault Systèmes, 2nd edn. Dassault Systèmes, Vélizy-Villacoublay, France
38. Karban R, Crawford A, Trancho G et al (2018) The OpenSE Cookbook: A practical, recipe based collection of patterns, procedures, and best practices for executable systems engineering for the Thirty Meter Telescope. In: Angeli GZ, Dierickx P (eds) Modeling, Systems Engineering, and Project Management for Astronomy VIII. United States. SPIE, Bellingham, Washington, Austin, Texas, pp 10–12
39. Koller R, Kastrup N (eds) (1994) Prinziplösungen zur Konstruktion technischer Produkte. Springer eBook Collection Computer Science and Engineering. Springer Berlin Heidelberg, Berlin, Heidelberg
40. Habermehl C, Höpfner G, Berroth J et al (2022) Optimization workflows for linking model-based systems engineering (MBSE) and multidisciplinary analysis and optimization (MDAO). Appl Sci 12:5316. https://doi.org/10.3390/app12115316
41. Mennicken M, Jacobs G, Jagla P et al (2024) (2024) Efficient HiL-Testing for Electric Heavy-Duty Drivetrains using Model-Based Systems Engineering. In: Berns K, Dreßler K, Kalmar R, al (eds) Commercial Vehicle Technology. Springer, Wiesbaden, pp 222–236
42. Berges JM (2024) Systemmodell zur virtuellen Auslegung und Optimierung von thermisch gefügten laserstrukturierten Kunststoff-Metall-Verbindungen. RWTH Aachen University
43. Zhang Y (2023) Development of hierarchical testing workflows to support virtual verification of technical systems. RWTH Aachen University

44. Jagla P, Jacobs G, Spütz K et al (2023) Classification of function-oriented solution elements for MBSE. Forsch Ingenieurwes 87:469–477. https://doi.org/10.1007/s10010-023-00651-0
45. Zerwas T, Jacobs G, Kowalski J et al (2022) Model signatures for the integration of simulation models into system models. Systems 10:199. https://doi.org/10.3390/systems10060199
46. Spütz K, Jacobs G, Konrad C et al (2021) Integration of production and cost models in model-based product development. JSS 09:53–64. https://doi.org/10.4236/jss.2021.912004
47. Spütz K, Berges J, Jacobs G et al (2022) Classification of simulation models for the model-based design of plastic-metal hybrid joints. Procedia CIRP 109:37–42. https://doi.org/10.1016/j.procir.2022.05.211
48. Lamm JG, Weilkiens T (2014) Method for Deriving Functional Architectures from Use Cases. Syst Eng 17:225–236. https://doi.org/10.1002/sys.21265
49. motego (2024) motego. https://www.motego.info/. Accessed 6 Sept 2024