






Model Signatures for Design and Usage of Simulation-Capable Model Networks in MBSE

Stephan Husung¹ , Detlef Gerhard² , Georg Jacobs³ , Julia Kowalski³ ,
Bernhard Rumpe³ , Klaus Zeman⁴ , and Thilo Zerwas³ 

¹ Technische Universität Ilmenau, Max-Planck-Ring 12, 98693 Ilmenau, Germany
stephan.husung@tu-ilmenau.de

² Ruhr-University Bochum, Universitaetsstr. 150, 44801 Bochum, Germany

³ RWTH Aachen University, Eilfschornsteinstr. 18, 52062 Aachen, Germany

⁴ Johannes Kepler University Linz, Altenbergerstraße 69, 4040 Linz, Austria

Abstract. Product development is characterised by numerous synthesis and analysis loops. Analysis provides information on the fulfillment of the required properties of the system under development. Analysis results therefore are an important basis for further synthesis steps. In the context of Model-Based Systems Engineering (MBSE), different types of simulation models play an important role. The overall system (product) can be broken down to system elements. For each system element a set of models has to be established, that provides the required degrees of fidelity, representations of system properties, flows, etc. reflecting the variety of modelling purposes. These models have to be integrated horizontally (same system level along the relevant flows of material, energy, and information) and vertically (aggregation from subsystem level to the overall system level, refinement in the opposite direction) to create a holistic model-based system representation. An important and challenging task is to identify and shape relevant subsystem models. In order to define an appropriate structure of these models, model developers may utilize criteria like selected properties of system elements and interrelations, their degree of detail or modelling assumptions. The relevant criteria have to be made transparent. For this purpose, the paper discusses the concept of model signatures that contain relevant meta information about each single model of all system elements, subsystems up to models of the overall system. This standardized meta information enables an identification and selection of those models and the decision on the necessary model integration. The concept is discussed on the basis of a roller bearing as an example out of an electro-mechanical drivetrain. A potential analysis provides information about the possible usage of the model signatures concept.

Keywords: MBSE · Product development · System models · Model signatures · Meta data · Model networks

1 Introduction

Product development is characterised by numerous synthesis and analysis loops to reduce the delta between the required and as-is properties of a product [1, 2]. Analysis provides information on the fulfillment of the required properties of the product. Therefore, analysis results are an important basis for further synthesis steps. The analysis can be carried out by means of physical tests, calculations or simulations. In the context of model-based development, different types of simulation models are required.

For the development of mechatronic products, their description using Systems Theory is recommended [3]. Based on this, products are described via models of so-called mechatronic systems (systems are themselves models of the product). An essential technique to master the complexity of systems is their decomposition into system elements. Decomposition is, in principle, arbitrary, and different forms of decomposition can be chosen for a product depending on the objectives [4]. Decomposition is often driven by organizational structures or even better by focusing on specific (groups of) properties, thus leading to e.g. functional structures [5, 6]. In any case, the decomposition of a system into system elements results in hierarchically structured levels and leads to a desired modularity of a system, such that reuse of already existing system element libraries becomes feasible. Common system elements can then be reused in their given qualities for building up products as overall systems.

Decomposing a system constitutes a hierarchically structured **system architecture** which has to be defined at the beginning of the design phase [7]. Given the system architecture, the implemented system elements have to be **composed** accordingly, which works best, when a relevant set of properties is visible on the **system element interface and inside the system element**.

As representations of complex systems, models themselves exhibit complex structures, accordingly. Modelling works best, when the models are decomposed in accordance with a relevant (e.g. physical) structure of the system. Thus, system architecture typically coincides with the **model architecture**.

During the composition of a system model, for each system element an appropriate set of models has to be established, that provides the required degrees of fidelity, representations of system properties, flows, etc. reflecting the variety of modelling purposes. These models have to be integrated horizontally (same system level along the relevant flows of material, energy, and information) and vertically (aggregation from subsystem level to the overall system level, refinement in the opposite direction) to create a holistic model-based system representation. An important and challenging task is to identify and shape the relevant set of models. In order to define an appropriate structure of these models, model developers may utilize criteria like selected (groups of) properties of system elements and interrelations, their degree of detail or modelling assumptions. The relevant criteria have to be made transparent, especially in heterogeneous simulation environments. Naturally, models describing individual system elements may encapsulate internal parts as well. They should exhibit an explicit interface description, precisely specifying the ports and interactions of models during subsystem operation (also called runtime), the parameters of the model that allow to configure a model network for the system element before the simulation, and further semantic information about the model (describing objective of the model or model assumptions).

For this purpose, the concept of **model signatures** for the development, specification and compositional integration of models is introduced in this paper. The **objective** of the paper is to discuss the requirements for model signatures and first realisation concepts. First, the requirements and the state of the art are elaborated (cf. Sects. 2 and 3). Based on this, the concept of model signatures is explained (cf. Sect. 4), which is then illustrated by means of an example (cf. Sect. 5). The paper concludes with an analysis of potentials and further research questions (cf. Sect. 6).

2 Requirements for Model Signatures

The requirements for model signatures can be divided into requirements for supporting specific use cases, requirements for usability, and requirements for implementation. The relevant stakeholders for the model signatures are engineers who want to select the appropriate models for model networks (use case 1) and plan their interconnections and thus composition (use case 2). Necessary requirements for model signatures can be derived from these use cases:

For use case 1, engineers need relevant information on the model purpose as well as information on the specific model. This information includes the interfaces provided by the model to the outside and the flows available at the interfaces during operation (e.g., force, velocity, current, information etc.). Furthermore, it must be known which state parameters (e.g. temperature) the model exhibits during operation. For the flow variables and state parameters, it must be possible to distinguish whether they are static or dynamic during operation and, if dynamic, whether they are influenced from outside or controlled by the system element. In mathematical terms, a static parameter is just a constant value, while a dynamic parameter is a discrete or continuous function in a possible value range. Important for selecting a specific model is its level of idealization determined by model assumptions (e.g., linear stress-strain behaviour). Assumptions determine the scope of the model and necessarily lead to limitations in its validity.

For use case 2, the interconnectivity of models must be verified, quite like type checking in programming languages. As described in Sect. 1, it must be possible to network and link models horizontally and vertically. For the evaluation of the networking capability, it must be possible to provide relevant information via the model signature. For horizontal networking, the information about the interfaces and flows as well as the associated compatibility are required above all. Compatibility depends, among other things, on whether the flows fit together in terms of type, granularity, physical unit, direction, temporal behaviour, etc., as well as the modelling properties (e.g., data types and potentially given additional value ranges).

Regarding usability, it must be ensured that the relevant information is explicitly and unambiguously represented in the model signature.

Model signatures have to reflect (1) the interfaces of the system, sub-system, system element that they describe, and (2) the configuration and dimensioning parameters that engineers can decide upon during development. Furthermore, because of the need for reduction of complexity through abstraction, e.g., by omission of relevant flows (or their dynamics), and due to the need for discretization of continuous processes and distributed parameters, models exhibit restrictions in their connectivity. Such restrictions

are typically not driven by the original system itself, but may be rooted in the design of the model, the underlying timing model, the form of simulation execution, etc. These forms of restrictions also have to be taken into consideration in model signatures.

The implementation of model signatures is subject to requirements regarding the definition of templates of model signatures, their instantiation and coupling with the executable simulation models. The implementation of these requirements is not described in concrete terms in this paper.

3 State of the Art

3.1 System Development

An established method for virtual development of mechatronic systems including their interacting subsystems is *Model-Based Systems Engineering (MBSE)* [6, 8, 9]. Thereby, the system to be developed is represented by a *system model* that is continuously both further specified (synthesis) and used for virtual behaviour verification (analysis) by all domains involved in the product development process. In order to control the complexity in the system model, the entire system is decomposed several times into *system elements*, so that a system architecture may span across several hierarchy levels (see e.g. Fig. 1) [6, 10]. Due to the encapsulation, these system elements have a handy complexity, interact with each other via interfaces and jointly represent the behaviour of the superordinate system.

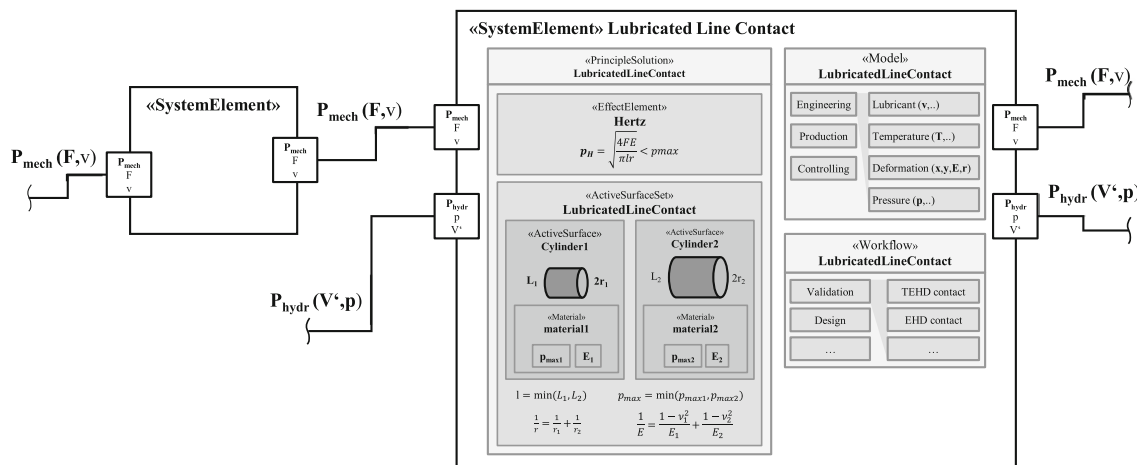


Fig. 1. Functional system architecture of system elements (according to [6, 11])

Using analysis based on defined characteristics of a product, its properties can be determined or – if the product does not yet exist – predicted [1]. In principle, analyses can be carried out using physical tests or simulation that can be based on proprietary and problem-specific simulation models. The reuse of models (product description [12] and simulation) is becoming increasingly important in order to reduce the effort required for model creation and to transfer existing knowledge between simulations. Standards such as FMI [13] have become established for the reuse of existing simulations, even in heterogeneous simulation environments.

3.2 Model Management

There is a variety of MBSE modelling tools to support engineers creating system models on different abstraction and fidelity levels for different purposes and views. The compatibility of those tools is limited; data exchange between different tools requires well-suited interfaces, model transformations and mappings. The limits in collaboration lead to a very restricted ability of composing and using model networks.

Most MBSE tools already enable some amount of analysis whereas deeper analysis or simulation requires software for simulation or custom analysis [14]. There are some approaches for integrating SysML and PLM systems for data and model management purposes. Heber and Groll [15] introduce a meta model to connect MBSE with PLM. Kirsch et al. [16] present an approach for SysML model management within PLM. PLM tools focus on data management in hierarchical (product) structures (BOM) but MBSE leads to data networks instead of hierarchies.

Wang [17] introduces an MBSE-Compliant Product Lifecycle Model Management (PLMM) as a methodology tailored to industrial application of MBSE, which implements a system of systems methodology in managing multitudinous models throughout a product lifecycle. One key aspect is the use of the SysML language as the unified top-level modelling method for building product meta models representing a set of specific product domain models (sub-models). The aim is to realize trans-phase and trans-domain model integration and synchronization, which tackle various challenges encountered in complex product developments.

Parrott and Spayd [18] focus in their research on the configuration management aspects of MBSE model management, in particular the change management of configuration controlled items. It was investigated, how changes to the models could be implemented with regard to how base content is affected and how the model versioning could be controlled. Hu et al. [19] propose a simulation models' meta model and ontology for their universal description. The meta model reflects the most essential features of simulation models without the consideration of a specific implementation method. Allen et al. [20] developed the Model-Driven Development Process (MDDP) methodology, which extends modelling to combine MBSE with Lean Information Management (LIM). The result is a fully consistent model for all relevant engineering items (EI) using only real-world semantics. Particular focus was put on model consistency and alignment to ISO 15288 to ensure that all information created during the process is automatically added to the model in its most appropriate form. In the context of mechatronic product development, Friedl et al. [21] propose the method of a Model Dependency Map (MDM) to disclose and describe the complex interrelations (e.g., inputs, outputs and their stakeholders) between different models of existing model networks. This approach may serve as a first step towards model signatures.

The management of a model network reflecting different abstraction and fidelity levels as well as the management of model transformations and mappings within the product development process is a challenge in PLM system environments. The model signature approach is a basis for tackling this issue on meta-model level.

4 Model Signature Approach

The requirements described in Sect. 2 illustrate that a formalised description is needed for the model signatures, which comprises the relevant information of a simulation model and can be read by the model user and, if necessary, be coupled without the specific executable simulation model (see Fig. 2). A model signature is necessary for each executable simulation model.

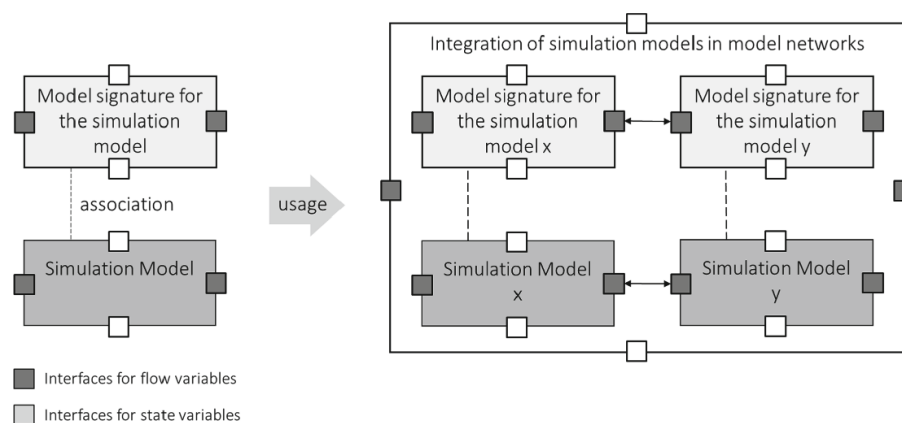


Fig. 2. Definition and usage of model signatures (with interfaces for variable exchange during runtime)

The analysis of different examples of simulation models (such as the lubricated line contact, see Sect. 5) shows that different types of simulation models can be found in the application. The possible types of simulation models are not considered in detail in this paper and are subject of further research activities. For the model signatures, this differentiation of the simulation models means that there can also be different classes of model signatures. A useful classification of model signatures supports the search for appropriate simulation models for specific applications.

If appropriate model signatures have been selected for specific use cases, their interconnectivity has to be ensured. Therefore, the model signatures must be represented in such a way that they can be used as a basis for checking whether the interfaces and flows or state variables are compatible in terms of content. Furthermore, due to the potential heterogeneous executable simulation models, it must be possible to ensure model compatibility on the basis of model signatures. Therefore, a model signature has to describe at least (1) the flows of the model of the system element for physical compatibility checking, and (2) the forms of encoding in the model, that unfortunately often also comprise tool specific properties such as data types, meshing parameters, time increments etc., for simulation compatibility.

In software development, object-orientation (OO) is a powerful concept to describe interfaces through their signatures and to encapsulate internal details. Domain Specific Languages (DSL) [22] tend to be defined from scratch and for a specific small purpose only. Composition, however, enforces them to provide explicit model signatures [23]. The concrete form of a model signature is rather dependent on the kind of model, but a common pattern of model signatures is an explicit naming of symbols that can be

accessed or even influenced from outside [24]. Therefore, in this paper, the authors propose an approach based on semi-formal modelling languages such as SysML (see Fig. 3). They allow the representation of the relevant information in the model signatures as well as the interfaces including the flows or state variables when using the language in a clearly defined methodological form as discussed before. With such a sound use of SysML, semi-formal description languages for the model signatures can be used to determine their interconnectivity and various other models of consistency issues. In particular, it would allow to carry over the idea of a strong type system from programming to systems engineering models.

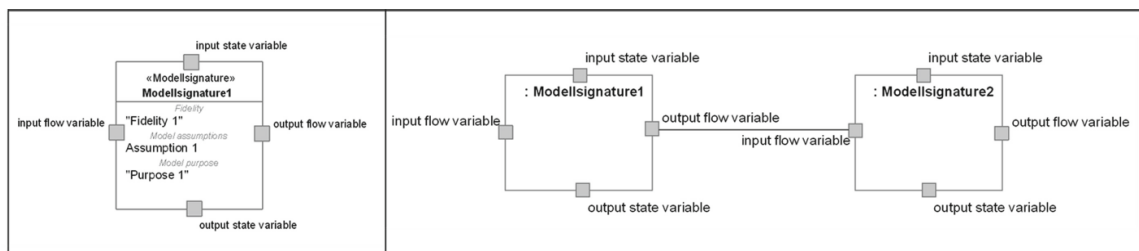


Fig. 3. Representation of model signatures using SysML

5 Example

The system element “lubricated line contact” as it appears in bearings and gear contacts, as well as the models belonging to it are well-known by literature. Therefore, it is selected as example for the intended model signature.

Figure 1 shows the system element “lubricated line contact” existing out of principle solution [11], a set of models and workflows [6]. The principle solution as a well-known element of design theory contains physical effects and acting surfaces [25]. The model set of the system element describes behaviours of the system element in different domains. Within this publication, we focus on the engineering domain, which includes models of analytical and numerical nature. With the intended model signature, the selection of appropriate models for specific purposes and fidelities is supported. Result is a case specific selection of models out of the system element’s full model set. Those selected models are connected by workflows which allow the execution of the model chain.

The engineering models that are linked to the system element “lubricated line contact” are depicted in Fig. 4. They can be differentiated by purpose and fidelity [6]. For example, three models of different fidelity can be distinguished for the purpose “temperature calculation”. In dependency of the required purpose of the system element (e.g. TEHD-calculation), a meaningful combination of multiple models as indicated by the black line has to be found. The decision on possible combinations of models needs deep insight into each model, which the authors aim to formalize by an appropriate model signature in order to support the compositional integration of models.

Figure 5 shows on the left hand side the parameters that are exchanged between the models when a TEHD calculation is executed. Obviously, state parameters, design

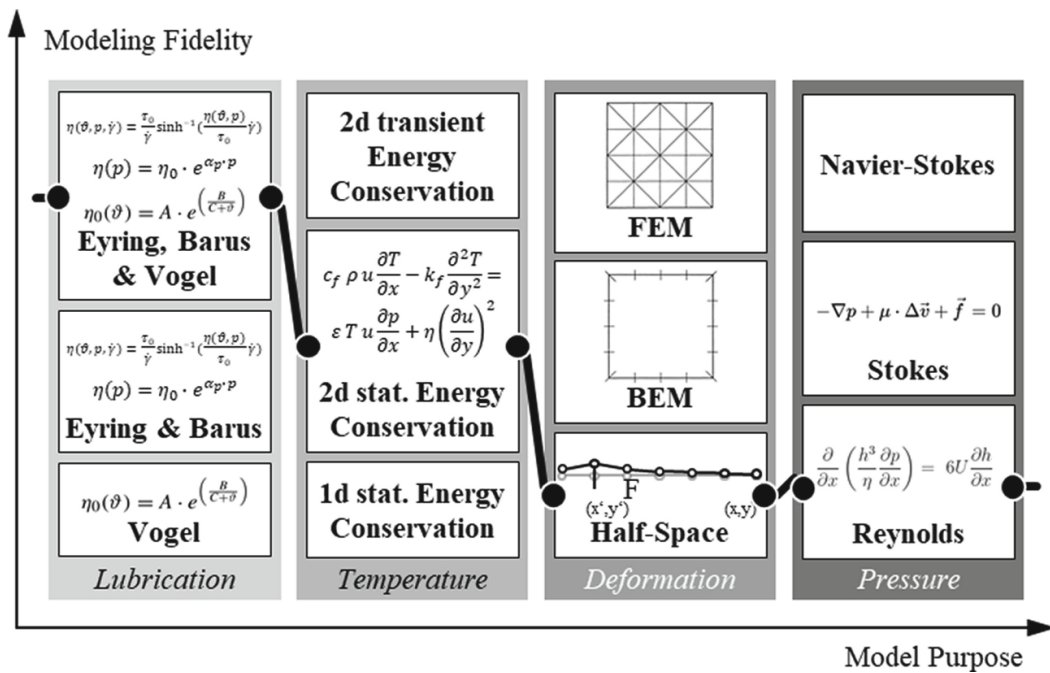


Fig. 4. Selection of engineering models for the scope “lubricated line contact”

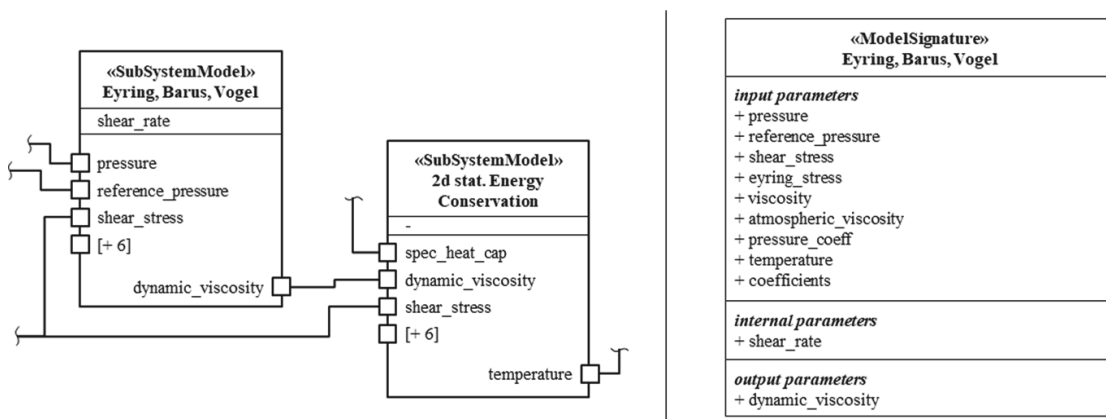


Fig. 5. Model chain of the lubrication model “Eyring, Barus, Vogel” and “2d stationary Energy Conservation” with their parameters (left side) as well as the proposed model signature of the lubrication model “Eyring, Barus, Vogel” (right side)

parameters and functional flows can occur as input, output or internal parameters. Each parameter can be of static or dynamic nature. A dynamic parameter can be driven externally or by the system element itself. The decision which parameter is input, output or internal depends on the purpose the model is used for. Especially, the lack of access to the instance of an internal parameter often leads to contradictions between the instances of the same parameter in other models. Models with inconsistent parameter instances are still a common problem in system modelling. Therefore, the model signature comprises input, output and internal parameters (Fig. 5, right hand side).

6 Relevant Research Questions for Further Steps

The discussions in this paper point out key messages and research questions derived therefrom:

- Model selection for analysis, and assurance of interconnectivity require explicit descriptions of model contents, model assumptions as well as modelling specifications in the form of model signatures in addition to executable models → derived research question: How must the model signatures (contents and formalization) be described (part of the paper)? Can the usual typing techniques known from software programming be adapted to these model signatures?
- The models of the system elements must be interconnected for composition. → derived research question: How is the interconnection of the models performed based on the model signatures? (research question is only partly addressed in this paper)
- In the context of today's modelling, different model types exist. For the different model types different model signatures are necessary → derived research question: Which model types exist and which model signatures are therefore necessary? Which technical requirements are necessary to implement the signatures (SysML or an extension)? (research question is only partly addressed in this paper)
- For compositional integration of models at different hierarchical levels, the signatures must reflect different hierarchical levels as well → derived research question: Which properties of the models have to be included in model signatures to ensure vertical compatibility and to enable hierarchical integration of models across different levels? (research question will be concretized in further publications)

The research questions will be investigated interdisciplinary in further activities.

References

1. Weber, C., Husung, S.: Virtualisation of product development/design - seen from design theory and methodology. In: 18th International Conference on Engineering Design (ICED 2011), pp. 226–235 (2011)
2. VDI: Entwicklung technischer Produkte und Systeme/Design of technical products and systems. Blatt 2 (VDI 2221:2019) (2019)
3. Ropohl, G.: Systemtechnik. Grundlagen und Anwendung, Hanser, München (1975)
4. Ariyo, O.O., Eckert, C.M., Clarkson, P.J.: Hierarchical decompositions for complex product representation. In: 10th International Design Conference, pp. 737–744 (2008)
5. Browning, T.R.: Applying the design structure matrix to system decomposition and integration problems: a review and new directions. *IEEE Trans. Eng. Manag.* **48**(3), 292–306 (2001). <https://doi.org/10.1109/17.946528>
6. Jacobs, G., Konrad, C., Berroth, J., Zerwas, T., Höpfner, G., Spütz, K.: Function-oriented model-based product development. In: Krause, D., Heyden, E. (eds.) *Design Methodology for Future Products*, pp. 243–263. Springer, Cham (2022). https://doi.org/10.1007/978-3-030-78368-6_13
7. VDI: Entwicklungsmethodik für mechatronische Systeme/Design methodology for mechatronic systems (VDI 2206:2004) (2004)

8. Rumpe, B.: Modeling with UML. Language, Concepts, Methods. Springer eBook Collection Computer Science. Springer, Cham (2016). <https://doi.org/10.1007/978-3-319-33933-7>
9. Friedenthal, S., Moore, A., Steiner, R.: A Practical Guide to SysML: The Systems Modeling Language, 3rd edn. The MK/OMG Press, Burlington (2015)
10. Husung, S., Weber, C., Mahboob, A.: Model-based systems engineering: a new way for function-driven product development. In: Krause, D., Heyden, E. (eds.) Design Methodology for Future Products, pp. 221–241. Springer, Cham (2022). https://doi.org/10.1007/978-3-030-78368-6_12
11. Zerwas, T., et al.: Mechanical concept development using principle solution models. In: IOP Conference Series: Materials Science and Engineering, p. 012001 (2021). <https://doi.org/10.1088/1757-899X/1097/1/012001>
12. Hick, H., Bajzek, M., Faustmann, C.: Definition of a system model for model-based development. SN Appl. Sci. **1**(9), 1–15 (2019). <https://doi.org/10.1007/s42452-019-1069-0>
13. Blochwitz, T., et al.: The functional mockup interface for tool independent exchange of simulation models. In: Proceedings of the 8th International Modelica Conference, pp. 105–114 (2011)
14. Bretz, L., Tschirner, C., Dumitrescu, R.: A concept for managing information in early stages of product engineering by integrating MBSE and workflow management systems. In: IEEE International Symposium on Systems Engineering (ISSE), pp. 1–8 (2016)
15. Heber, D.T., Groll, M.W.: A meta-model to connect model-based systems engineering with product data management by dint of the blockchain. In: IEEE International Conference on Intelligent Systems (IS), pp. 280–287 (2018). <https://doi.org/10.1109/IS.2018.8710527>
16. Kirsch, L., Müller, P., Eigner, M., Muggeo, C.: SysML-Modellverwaltung im PDM/PLM-Umfeld. In: Tag des Systems Engineering, pp. 333–342. Carl Hanser (2016)
17. Wang, C.: MBSE-compliant product lifecycle model management. In: 14th Annual Conference System of Systems Engineering (SoSE), pp. 248–253. IEEE (2019). <https://doi.org/10.1109/SYSOSE.2019.8753869>
18. Parrott, E.L., Spayd, L.C.: Configuration and data management of the NASA power and propulsion element MBSE model(s). In: 2020 IEEE Aerospace Conference, pp. 1–11. IEEE (2020). <https://doi.org/10.1109/AERO47225.2020.9172375>
19. Hu, C., Xu, C., Fan, G., Li, H., Song, D.: A simulation model design method for cloud-based simulation environment. Adv. Mech. Eng. **5**, 932684 (2013)
20. Allen, C., Di Maio, M., Kapos, G.-D., Klusmann, N.: MDDP: a pragmatic approach to managing complex and complicated MBSE models. In: IEEE International Symposium on Systems Engineering (ISSE), pp. 1–8 (2016). <https://doi.org/10.1109/SysEng.2016.7753165>
21. Friedl, M., Weingartner, L., Hehenberger, P., Scheidl, R.: Model dependency maps for transparent concurrent engineering processes. In: 14th Mechatronics Forum International Conference (Mechatronics 2014), pp. 614–621 (2014)
22. Fowler, M., Parsons, R.: Domain-Specific Languages. Addison-Wesley (2011)
23. Clark, T., van den Brand, M., Combemale, B., Rumpe, B.: Conceptual model of the globalization for domain-specific languages. In: Cheng, B.H.C., Combemale, B., France(†), R.B., Jézéquel, J.-M., Rumpe, B. (eds.) Globalizing Domain-Specific Languages. LNCS, vol. 9400, pp. 7–20. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-26172-0_2
24. Butting, A., Hölldobler, K., Rumpe, B., Wortmann, A.: Compositional modelling languages with analytics and construction infrastructures based on object-oriented techniques—the MontiCore approach. In: Heinrich, R., Durán, F., Talcott, C., Zschaler, S. (eds) Composing Model-Based Analysis Tools, pp. 217–234. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-81915-6_10
25. Koller, R.: Konstruktionslehre für den Maschinenbau. Grundlagen zur Neu- und Weiterentwicklung technischer Produkte mit Beispielen. Springer, Heidelberg (1994). <https://doi.org/10.1007/978-3-662-08165-5>