

35th CIRP Design 2025

Automated Derivation of MBS Models from SysML System Architecture Models

Manuel Mennicken^{a,*}, Gregor Hoepfner^a, Georg Jacobs^a

^a*Institute for Machine Elements and Systems Engineering, RWTH Aachen University, Eilfschornsteinstraße 18, 52062 Aachen, Germany*

* Corresponding author. Tel.: +49-241-8090245; E-mail address: manuel.mennicken@imse.rwth-aachen.de

Abstract

Increasing complexity of technical systems and shorter time-to-market demand virtual validation. Model-based systems Engineering enables seamless connection of systems through all development stages utilizing a system model. A system model stores various system information, from requirements to system architecture and design. However, to validate design changes, simulation models as MBS models are required, which are often created manually and separate from the system model. Therefore, this work aims to connect system and simulation models, enabling an automated MBS model generation from a SysML architecture. With this method the manual effort required for system and design validation can be minimized.

© 2025 The Authors. Published by Elsevier B.V.

This is an open access article under the CC BY-NC-ND license (<https://creativecommons.org/licenses/by-nc-nd/4.0>)

Peer-review under responsibility of the scientific committee of the 35th CIRP Design 2025

Keywords: MBSE, MBS, design verification, model generation

1. Introduction

Modern products demand higher quality and more features leading to larger and more complex systems. To manage this complexity, virtual representation, or virtual twins, are essential for validating and simulating the architecture and behaviour of systems before building costly prototypes. To drive efficiency in creating virtual twins, Model-Based System Engineering (MBSE) is a well-established method which connects models of different domains inside a virtual twin [1]. However, to verify the systems requirements and functionality, various simulation models are required each serving a defined purpose as for example Finite Element Method (FEM) for structural analysis, Computational Fluid Dynamics (CFD) for the prediction fluid dynamics as pressure and leakage or Multibody Simulation (MBS) for the analysis of motion and forces in technical systems. The primary challenge in simulation is the creation of simulation model for such complex systems since it requires a good amount of data and effort. An MBSE system model can provide the required data since it

holds the systems architecture and defines its parameters. The aim is therefore to connect system models and simulation models in a matter, that it is possible to generate simulation models from system models. With this it can be possible to minimize the modelling effort and provide data consistency between system model and simulation.

This paper presents a method for automatically generating simulation models using the case of MBS models. This method employs a formal system architecture of a system modelled in SysML as a basis for creating a model. The state of research demonstrates various methods for formally describing a systems architecture and first approaches for automated model generation.

2. State of Research

MBSE is a modern approach for product development. It utilizes models from various domains to describe a systems requirements, architecture, and behaviour. Typically, MBSE revolves around a central system model that connects various

domains such as requirements, functional modelling, system architecture and simulation. For this the system model describes the system formally utilizing a suitable language. Example for those languages are the Unified Modeling Language (UML) for software systems or the AutomationML (AML) for production systems [3, 4]. The systems modeling language (SysML) is widely used for general system description of technical systems including hardware and software components [5, 6]. However, SysML does not provide guidance on how to build or derive a systems architecture. Therefore, methods such as SysMOD [7], Harmony [8], or motego [2] must be employed. The motego method, used in this paper, is particularly well-suited for modelling cyber-mechatronic systems focusing the integration of domain-specific models [2].

Initial approaches, such as those proposed by Groß et al. have explored the automated model generation (AMG) of simulation models (e.g. MBS) from SysML. However, to do so the required MBS model elements are created as blocks in SysML, so that the SysML model mirrors the simulation model. This reduces the degree of reusability and stays in contrast with a functional modelling approach [6]. Concerning model transformation with SysML other approaches have proposed methods which allow the transformation from SysML to Simulink or Modelica. In those cases however the simulation models are less complex (1D-Simulation) and do not require information about geometry [9–11].

3. Problem Formulation

Current research shows no suitable method to generate a simulation model from a system model in which the system model has not been especially customized or created for the model generation. Thus, simulation models have to be manually created by replicating information from the system architecture, which is time-consuming and prone to errors and inconsistencies. The following research questions (Q) and hypotheses (H) aim to address this issue:

- Q1 What data is necessary for the construction of an MBS model within the system model?
- H1 Solution Elements and System Solutions can be used to model the physical connection of geometries which is required for MBS models.
- Q2 How can an MBS model be created from a SysML system model?
- H2 Solution Elements and System Solutions allow to derive transformation rules between system architecture and system behaviour for automated model generation

In this research MBS models are analyzed to identify the information that is required to be given by an MBSE system architecture model and the information which can not be provided in the motego method yet is identified. Then the meta model of the MBSE modelling method motego is extended with required stereotypes which allow to hold the missing information. After that, transformation rules between motego-Stereotypes and MBS model elements are identified and implemented into an interface, that allows an automated generation of an MBS model from a SysML systems model. Fig. 1 shows the method to be developed in which the system

model serves as the data basis for the model generator which allows to automatically create the MBS model which can be then integrated into the system model for requirements verification.

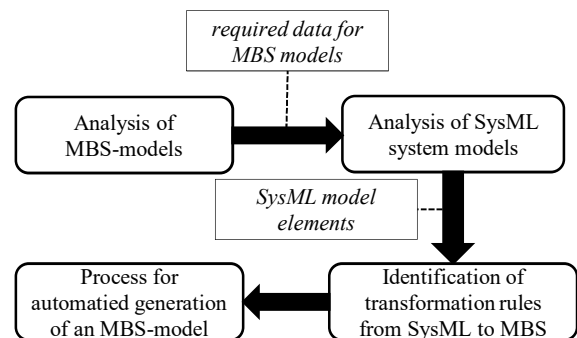


Fig. 1: Procedure for the development of the methodology

4. Methodology / Approach

4.1 Identification of required model elements for MBS

In general, an MBS-Model consists of geometry models (**bodies**) and the physical connections between each other. These **connections** or **joints** are created between two coordinate systems (**marker**) on different bodies. The connection then limits the degrees of freedom for the movement of those bodies. Physical constraints of a connection like friction or damping can be specified in **force-elements**. MBS models can be organized using **subsystems**, which can be integrated into a larger model with connections between them [12].

4.2 Model elements in motego

The motego method enables the description of a system throughout the entire development process, encompassing the stages from requirements to the formulation of functions, the derivation of solution concepts and eventually the realization of the physical product. For this purpose, the SysML is employed as a modeling language. However, the SysML lacks in stereotypes, which serve to distinguish between different system elements. Therefore, a SysML-based meta model is created within the motego framework, which introduces additional stereotypes for various stages of the development process [2, 13]. Fig. 2 provides a brief illustration of this approach.

Within the motego method the requirements describe customer needs and wishes for the product concerning functionality and design. The function layer describes the behavior of a product without specifying the solution whilst the solution layer describes the physical realization of a function. At last the product layer implements geometry and manufacturing of the product itself [2, 13].

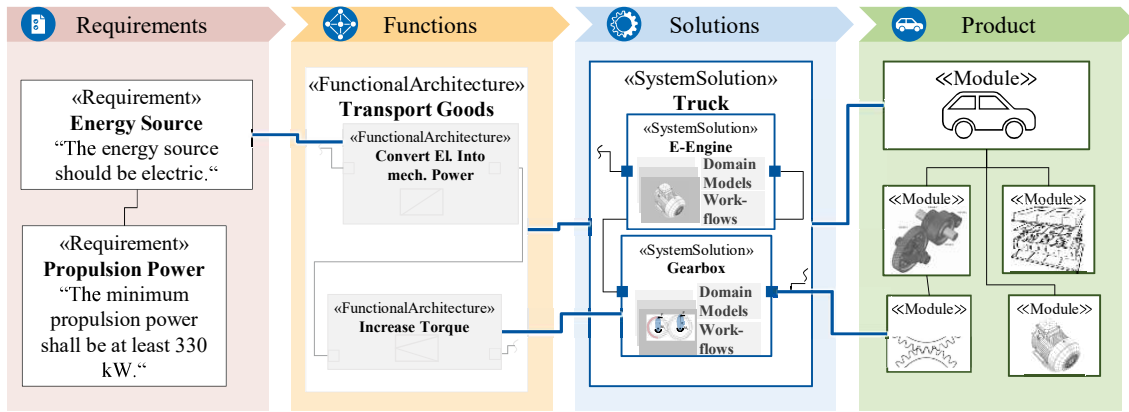


Fig. 2: Function-oriented model based development with the motego [2]

Since simulation models in the engineering domain are used to simulate the physical behavior of a product the solution layer is suitable to derive a simulation model from since it should hold the relevant system information. The most important element of the solution layer are [2]:

- **SystemSolution:** Structures elementary Solutions into subsystem for function realization
- **SolutionElement:** elementary solution concepts which realize an elementary function
- **ActiveSurfaces:** Describe the geometrical surfaces which are connected with a physical effect within a SolutionElement
- **StructureElement:** Structure elements define active surfaces realization within one physical structure
- **DomainModels:** simulation models are integrated into the system solutions and solution elements as so-called domain models

Fig. 3 shows the structure of a SystemSolution Roller Bearing including the above described model elements of motego.

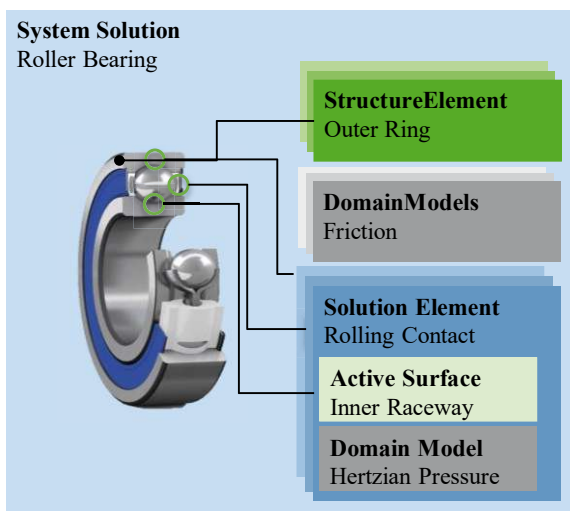
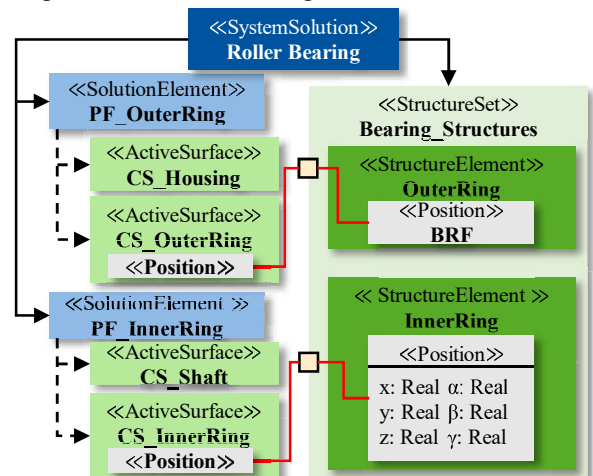


Fig. 3: Structure of a SystemSolution for a Roller Bearing

As described in section 4.1 an MBS model consists of bodies, markers, joint or connections and force elements. In this case, the force element describes the physical behavior of the connection, and thus can be directly assigned to a SolutionElement, given that the SolutionElement describes the physical connection of two surfaces. Markers apply forces on

bodies, thereby representing points of force application. They can thus be interpreted as reduced active surfaces, whereby ActiveSurfaces represent the corresponding SysML element. However, in order to implement a Marker in the MBS, it is necessary to define its position on a body, which requires the use of coordinate systems. A Body itself is a geometric representation and is implemented as a StructureElement in the SysML. At this stage a StructureElement serves only as the collection of surfaces., which can then be connected to a volumetric geometry. The implementation of a model element that stores the information on the geometry model is still pending.

Above it is stated that the motego metamodel includes stereotypes useful for system modelling, but still lacks of stereotypes required for an MBS model describing coordinate systems and geometry models, leading to the need of an extension to incorporate these elements. To map coordinate systems in a system model, a stereotype called Position is created. A Position has six value properties for each coordinate (x-, y-, z-Translation and α -, β -, γ -Rotation) and can be assigned to surfaces and components. Within a component, a Position specifies a surfaces location, a surface holds a Position to be connected with a component. The connection of a surface to a component can be seen in Fig. 4.



PF:Press Fit; CS: Contact Surface; BRF: Body Reference Frame

Fig. 4: Connection of surfaces and components for a rolling bearing

The second stereotype which is created is called *DesignModel* and is used for the storage of information concerning a geometry model. The *DesignModel* represents a special *DomainModel* which encapsulated a geometrical model (CAD- or FE-model) and is capable of storing paths to e.g. a CAD-File. Subsequently, a *DesignModel* can be assigned to a surface as a surface model (2D) or to a component as a volumetric model (3D).

4.3 Transformation from system model to an MBS-Model

Fig. 5 shows the general approach of the model generation process which utilizes a transformation from a SysML system model to an MBS model.

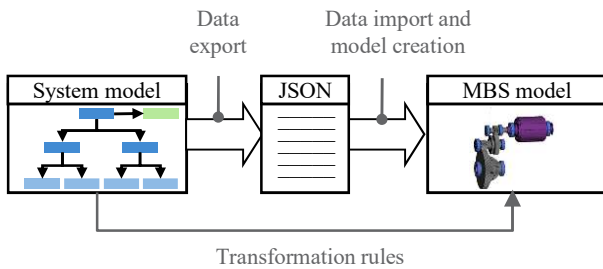


Fig. 5: Model transformation from system model to MBS model

Transformation Rules

In order to transform a SysML system model into an MBS, it is necessary to identify transformation rules. In general, transformation rules describe the process of transforming a model element from a source language into a model element in a source language. Fig. 6 illustrates the concept of model transformation.

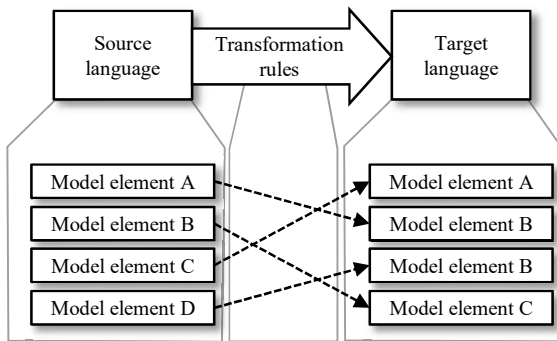
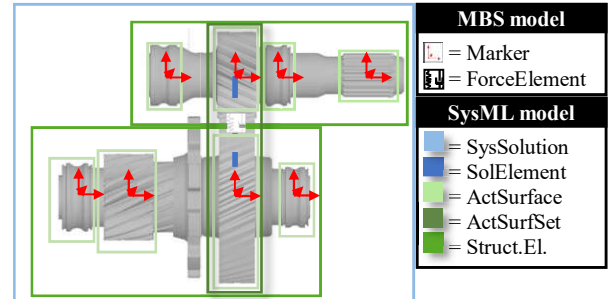


Fig. 6: Model transformation concept

In this case the transformation rules define the correspondence between SysML and MBS element. An MBS **subsystem** is created for each *SystemSolution*, whilst a *SolutionElement*, which describes the physical connection between two or more surfaces, is represented in the MBS model as both a **connection** and a **force element**. The **markers**, constructed from the *Position*, that are assigned to *ActiveSurfaces* in the system model, implement these connections in the MBS. The markers are placed on **bodies** which are transformed from *StructureElement*. A Body contains the *Positions* from the system model as **markers** and a geometrical model (**primitive**) which uses the CAD file which has been stored in a *designModel*. By doing so an

ActiveSurface is implemented abstractly as a **marker** in the MBS. To define the type of **force element** in the MBS a *SolutionElement* also uses a general type chosen from a SysML model library. Fig. 7 shows the correlation between an MBS model and the stereotypes of the system model as well as some identified transformation rules.



Motego Stereotype	MBS model element
SystemSolution	Subsystem
SolutionElement	Connection + Force Element
ActiveSurface + Position	Marker
StructuralComponent	Body

Fig. 7: Correlation between MBS-Elements and SysML-Elements for a gearbox

In addition, the SysML allows to create model libraries with predefined model elements. This allows an easier aggregation process when modelling system architectures. Furthermore this allows to draw additional transformation rules between element from the SysML model library to an MBS model library. Examples for element of the SysML library can be already seen in this paper (Fig. 3 & Fig. 4) and include *SystemSolutions* as Roller Bearings, journal bearing, gear stages as well as *SolutionElements* as press fits, rolling contacts, gear contacts and many more. For each element in the SysML library, one or more suitable MBS-model representations can be selected. Table 1 illustrates the relations between the SysML library and the SIMPACK model elements library.

Table 1: Correlation between SysML Elements and MBS Elements

SysML		MBS	
Stereotype	System element	Model element	Element type
SystemSolution	Gearbox	-	Subsystem
SolutionElement	Gear contact	FE 225: Gear Pair	Force Element
SolutionElement	Press fit	CTN2: Rigid	Connection
SolutionElement	Rolling Contact	FE88: Rolling Bearing	Force Element
		CTN: Revolute	Connection
ActiveSurface	Gear	Gear	Primitive

Data export

In the scope of this work the Cameo Systems Modeler (CSM) has been used to employed to utilize the system model in accordance with the SysML standards. In CSM the system model is stored as an mdzip-file, which contains the system information as well as diagram views, block positioning and other relevant data. However, the file is rather unsorted and encapsulated, containing more data than necessary for the

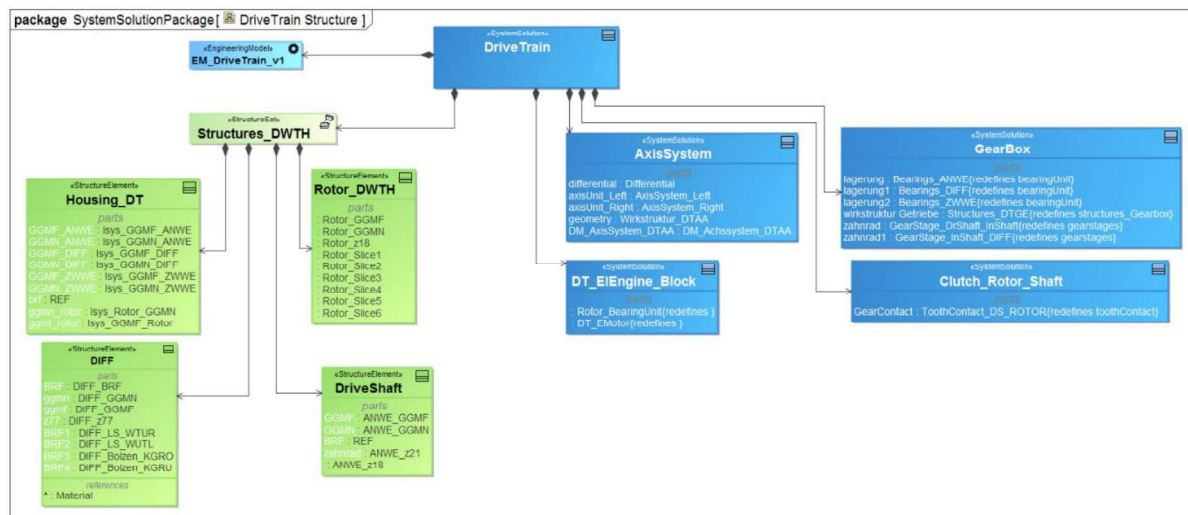


Fig. 8: Extract of the systems model for the electric drive train

simulation models. Therefore, an export tool was developed to extract only the relevant system information from the system model and present it in a machine-readable format. In the course of this process an exchange format – in this case Javascript Object Notation (JSON) – is provided along the system model to store relevant data. This data can be used for the generation of multiple other simulation models - not only MBS models.

Data import and model creation

In the subsequent phase, the data must be rendered suitable for processing by the MBS tool. This is achieved by importing the model data from the JSON file. The model data is imported to the MBS-tool SIMPACK, which has been utilized in the context of this research, through SIMPACK Scripts which facilitate the creation and manipulation of MBS-Models. Following the import of the data from the JSON-File for each *SystemSolution* iteratively:

- a) an MBS model is created
- b) all bodies and their markers are created from the components and position list
- c) a connection and force elements for each *SolutionElement* are created as well as parameters are assigned
- d) all subsystems associated with subordinated *SystemSolutions* are integrated

5. Use case

This work focuses on an electrified drivetrain previously used in research projects FVA 682 I and II to develop modeling methods for predicting the acoustics of electromechanical drives. The powertrain consists of a permanent magnet synchronous machine (PMSM) for converting electrical into mechanical power, a two-stage spur gear for adjusting speed and torque, and a differential to distribute mechanical power to the side shafts connected to the wheels. A control system is employed to adjust the motor currents via power electronics and pulse width modulation, optimizing operational points.

The according system model of the drivetrain consists of 35 *SystemSolutions*, 50 *SolutionElements* and 35 *StructuralComponents*. The model is divided into the

submodels of engine system, gearbox, axis system and clutch.

The main *SolutionElements* are rolling contacts for the roller bearings, gear contacts for the gear stages in the gearbox and differential and press fits for the connection of bearing rings. The *SolutionsElements* are drawn from the model library and mostly connect two different cylindrical surfaces which are placed on the bearing rings, housing or shafts. The gear contacts are created between two gear surfaces. The system model contains several *StructureElement* including bearing rings, housing, gears and shafts. The geometry models of these components are stored with their respective file paths as *DesignModels*. Additionally, surfaces are placed at their specific *Positions* which can be read from the geometric models. An extract of the model is shown in Fig. 8.

The MBS-model is created by initializing the plugin for any *SystemSolution*. The plugin will then automatically export all relevant model data for the chosen *SystemSolution* and all its subordinates and start the scripts in the MBS-tool which manage the import and generation of the MBS-model. The resulting MBS model and an example for a roller bearing in SysML and MBS can be found in Fig. 9. Required parameters – concerning markers coordinates or for force elements – are read from the properties in the system model into the JSON format and are assigned to the according parameter of an MBS model element within the transformation rules.

6. Summary and outlook

The paper presents a methodology for the generation of simulation models. The method employs a formal description of the system, utilizing the SysML, to provide the data necessary for the simulation model. The transfer of model data from an architectural modelling tool to a simulation tool is demonstrated on the example of an MBS model. Furthermore, the identification and application of transformation rules

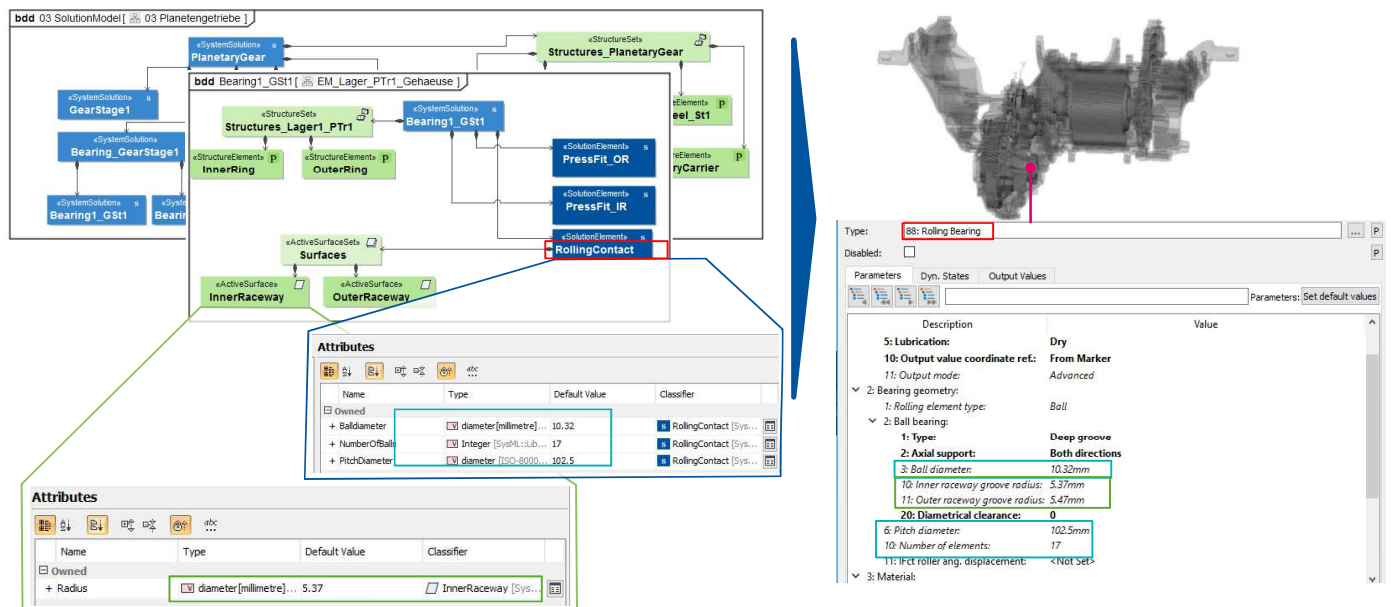


Fig. 9: SysML to MBS for a roller bearing

between SysML and MBS models is presented. For this, the structure of the simulation model – in this case, MBS – is analyzed to identify the data required for its creation. Subsequently, a system model utilizing the motego method is analyzed to identify whether and where the relevant information already exists and if not, how it can be integrated into the modelling method. This then allows the identification of transformation rules between the relevant modelling elements.

Future work will include automating the generation of geometry models, incorporating flexible bodies, and adding MBS-relevant forces and constraints into the SysML model. Additionally, the model library will be expanded to support more elements from MBS and other simulation models like CFD and FEM, enabling more comprehensive systems models.

References

[1] Madni AM, Sievers M (2018) Model-based systems engineering: Motivation, current status, and research opportunities. *Systems Engineering* 21:172–190. <https://doi.org/10.1002/sys.21438>

[2] Spütz K, Jacobs G, Zerwas T et al. (2023) Modeling language for the function-oriented development of mechatronic systems with motego. *Forsch Ingenieurwes* 87:387–398. <https://doi.org/10.1007/s10010-023-00623-4>

[3] Lüder A, Schmidt N (2016) AutomationML in a Nutshell. In: Vogel-Heuser B, Bauernhansl T, Hompel M ten (eds) *Handbuch Industrie 4.0*. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 1–46

[4] Czuchra W (2010) *UML in logistischen Prozessen: Graphische Sprache zur Modellierung der Systeme ; mit 4 Tabellen ; [mit Online-Service, 1. Aufl. Studium*. Vieweg + Teubner, Wiesbaden

[5] Friedenthal S, Moore A, Steiner R (2015) *A practical guide to SysML: The systems modeling language*, Third edition. MK/OMG Press. Elsevier, Inc, Waltham, Mass.

[6] Gross J, Mukherjee R (2016) Integrating Multibody Simulations With SysML. In: 11th International Conference on Multibody Systems, Nonlinear Dynamics, and Control: Presented at ASME 2015 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, August 2-5, 2015, Boston, Massachusetts, USA. The American Society of Mechanical Engineers, New York, N.Y.

[7] Weikiens T (2016) *SYSMOD - the systems modeling toolbox: Pragmatic MBSE with SysML*, 2nd edition, version 4.1. MBSE4U booklet series. MBSE4U, [Fredesdorf]

[8] Douglass BP (2015) *Harmony MBSE Modeling Standards for use with UML, SysML, and Rhapsody*

[9] Reichwein A, Paredis CJJ, Canedo A et al. (102012) Maintaining consistency between system architecture and dynamic system models with SysML4Modelica. In: Hardebolle C, Syriani E, Sprinkle J et al. (eds) *Proceedings of the 6th International Workshop on Multi-Paradigm Modeling*. ACM, New York, NY, USA, pp 43–48

[10] Chabibi B, Douche A, Anwar A et al. (62016) Integrating SysML with Simulation Environments (Simulink) by Model Transformation Approach. In: 2016 IEEE 25th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE). IEEE, pp 148–150

[11] Paredis CJ, Bernard Y, Burkhart RM et al. (2010) 5.5.1 An Overview of the SysML-Modelica Transformation Specification. *INCOSE International Symp* 20:709–722. <https://doi.org/10.1002/j.2334-5837.2010.tb01099.x>

[12] Dassault Systèmes (2019) *SimpacK User Assistance 2020.1*, Gilching

[13] Jacobs G, Konrad C, Bertho J et al. (2022) *Function-Oriented Model-Based Product Development*. In: Krause D, Heyden E (eds) *Design Methodology for Future Products: Data Driven, Agile and Flexible*, 1st ed. 2022. Springer International Publishing; Springer, Cham, pp 243–263