INTERNATIONAL DESIGN CONFERENCE - DESIGN 2018
https://doi.org/10.21278/idc.2018.0146

IMPLEMENTATION OF SYSTEMS MODELING LANGUAGE (SYSML) IN CONSIDERATION OF THE CONSENS APPROACH

V. Salehi, G. Florian and J. Taha

Abstract

The following paper, which is based on the design research methodology (DRM) according to Blessing and Chakrabarti, will first make a literature survey related to MBSE approaches in general. Based on the literature survey this paper will demonstrate the combination of SysML and MBSE in an industrial context by means of Cameo Systems as a MBSE-Modeler. Especially at the use of modeling tools, like Cameo Systems Modeler, users often have to create a completely new model draft.

Keywords: complex systems, cyber physical systems, systems engineering (SE)

1. Introduction

The graphical modeling language SysML helps users with specifying, analysing, designing, and verifying complex systems that may include hardware, software, information, personnel, procedures, and facilities (Object Management Group, 2017b). The main application field is supporting development processes of mechatronic systems. The language which was released in 2007, by the Object Management Group (OMG), an international technology standards consortium (Object Management Group, 2017a), is based on UML. Many parts of UML have been adopted to SysML, but also some concepts have been added and omitted (Dori, 2016). According to Weilkiens, the most important extensions of SysML are (Weilkiens and Soley, 2014):

- Adding the requirement diagram to enable requirement definition
- Supporting the data type ISO AP-233 to enable data exchange between different tools
- Renaming and upgrading some diagram types

In the section of software engineering, the developers of SysML have cut out many specific parts, because the located area of application is mechatronic systems (Hampson, 2015). SysML helps engineers to describe and connect the structure, the behavior and the requirements of these systems. Thus there are two areas of application (Alt, 2012, p. 30):

- As a documentary language
- As a development supporting tool

In the case of this paper, it is used from requirement management up to physical engineering to support the process of system development. The purpose is mainly to improve the information flow between the team members. The Cameo Systems Modeler will be the tool to realize the modelling of the system. This MBSE environment of No Magic Inc. supports engineers with engineering analyses for design decisions, evaluation and requirements verification as well as checking model consistency and tracking design progress with metrics (No Magic Inc., 2017). The chapter “Implementation” contains screenshots which shall further improve the understanding of the tool.

The goal of the project behind this paper is to develop and build up a robotic system, as seen in Figure 1, which enables an autonomous transportation of components. The current situation is a production line with no connection to a warehouse. The robot shall be able to recognize a lack of parts and manage the compensation of the shortage as a result. At this point the exact implementation or rather the solution will not be further discussed. Also, when creating the system, less attention was paid to creating an innovative product; rather, a project topic as suitable as possible was defined to verify the implemented SysML approach.

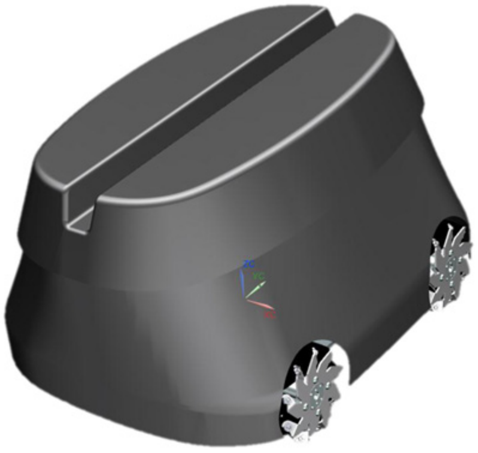


Figure 1. Autonomous transportation robot (Autobot)

The focus will be to present a simple and efficient method that includes a sequence of steps which support users with implementing mechatronic systems into the Cameo Systems Modeler. The method which is considered in this paper will be the specification technique CONSENS (CONceptual design Specification technique for the ENgineering of complex Systems) (Gausemeier and Behmann, 2012).

2. Literature survey

Many methodologies have been developed in the field of MBSE. In this context methodology is defined as a collection of related processes, methods and tools (Martin, 1997). The purpose of this section is to give a short insight into the most widely used methodologies, which have been summarized by J. A. Estefan (2007):

- Telelogic Harmony-SE: The tool- and vendor-neutral Harmony process mirrors the classical "Vee" lifecycle development model of system design. The task flow and work products in the Harmony-SE process include the three top-level process elements: requirements analysis, system functional analysis and architectural design (Estefan, 2007, pp. 15–19). It uses a "service request-driven" modeling approach along with OMG SysML artifacts (Hoffmann, 2006).
- INCOSE Object-Oriented Systems Engineering Method (OOSEM): "OOSEM integrates a top-down, model-based approach that uses OMG SysML to support the specification, analysis, design, and verification of systems." (Estefan, 2007, p. 19) Thereby it helps to architect more flexible and extensible systems that can accommodate evolving technology and changing requirements. OOSEM has the following development activities which are also consistent with the "Vee" process: analyzing stakeholder needs, defining system requirements and logical architecture, optimizing and evaluating alternatives and validating/verifying the system.
- IBM Rational Unified Process for Systems Engineering (RUP SE) for Model-Driven Systems Development (MDSD): RUP SE was created to specifically address the needs of systems engineering projects. The objective of its creation was to apply the discipline and best practices of the RUP for software development to the challenges of systems specification, analysis, design, and development." (Estefan, 2007, p. 20)
- Vitech Model-Based Systems Engineering Methodology: This Methodology includes a set of tutorials, developed and offered by Vitech CEO and Chief Methodologist James E. Long (Long,

2000). The approach shall be spread by multi-day courses through the Vitech training services. The Vitech MBSE Methodology is based on four primary system engineering activities: Functional/Behavior analysis, Source Requirements Analysis, Architecture Synthesis as well as Design Validation and Verification. It is stressed that a MBSE System Definition Language is needed to set up and manage this approach.

- State Analysis (SA): "SA provides a process for capturing system and software requirements in the form of explicit models, thereby helping reduce the gap between the requirements on software specified by systems engineers and the implementation of these requirements by software engineers." (Estefan, 2007, p. 33)

The previously mentioned methodologies help users to understand and apply MBSE approaches in an overarching way. Nevertheless, there are some issues that should be mentioned. According to Mhenni et al., - "In these methodologies, system functionality is mainly scenario-based or service request-based and serves as a support for the software coding. There is a lack of a hierarchical breakdown of system functions that facilitate the definition of alternative architectures by means of different functional groupings. There's also no traceability links clearly established between system functions and components in the presented methodologies." (Mhenni et al., 2014, p. 219) Especially the two most popular methodologies Harmony-SE and OOSEM show remarkable similarities to CONSENS and therefore to this modeling approach. In this context the process elements: requirement analysis, system functional analysis and the architectural aspect, have an important role. Therefore some definitions shall improve the understanding at this point. According to the IEEE Standards Association, a requirement is "a statement that identifies a product or process operational, functional, or design characteristic or constraint, which is unambiguous, testable or measurable, and necessary for product or process acceptability (by consumers or internal quality assurance guidelines)". (IEEE, 1998) The definition of requirements which sometimes can be derived from a development contract is essential for the satisfaction of the customer. Based on these requirements, first the system functions and then its architecture containing active structures shapes and so on, can be set up. The next chapters will treat those topics in more detail.

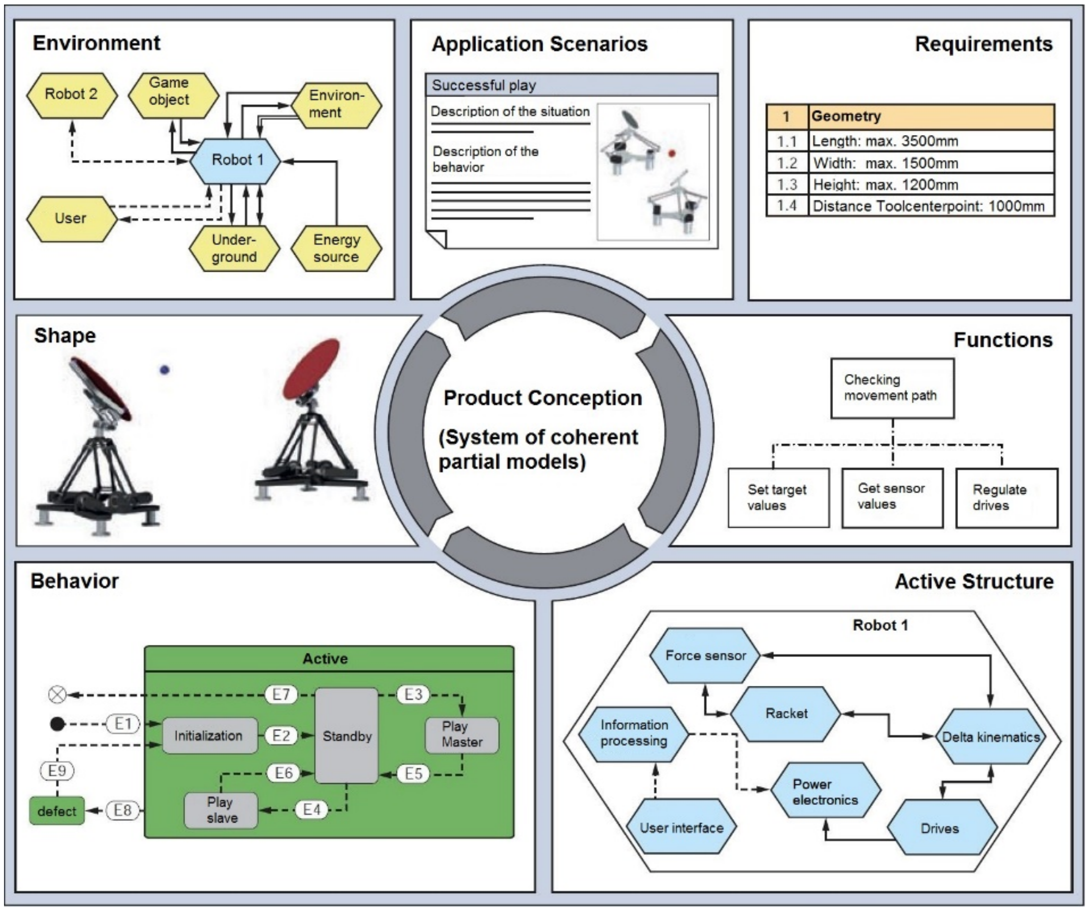
Not only in the area of MBSE methodologies have had some aspects, like the lack of understanding, decelerated the spreading process. Also in case of SysML, especially in consideration of tools like Cameo Systems Modeller the missing understanding and acceptance is a problem. Albers and Zingel voice the same concerns regarding MBSE methodologies and SysML, - "Over the last years, several modelling languages and approaches have been presented. They all promised to enable their users to model multidisciplinary and complex technical systems. […] Observations of the authors have shown that software engineers and electronics engineers cope well with the provided diagrams for modelling several system aspects like structures, sequences or states. Emerging graphical modelling languages for embedded systems like MARTE, AUTOSAR or EAST-ADL using similar principles and technologies like the meta-modelling standard MOF (Meta Object Facility) seem to underline this observation. On the contrary, mechanical engineers have much more trouble dealing with such kind of modelling languages. A possible reason is that technical terminologies between disciplines differ significantly. Furthermore, there is a huge leap of abstraction from concrete discipline-specific models to abstract multi-disciplinary system models. […] None of those methodologies has significantly established over the last years after this survey." (Albers and Zingel, 2013, p. 84) In his paper for the Proceedings of the Systems Engineering Test and Evaluation Conference in 2010, Kasser presented seven myths of Systems Engineering, due to persistent discussions about possible reasons for lacking acceptance and application in industrial product engineering. He ascertained, that especially the systems complexity is a huge problem (Kasser, 2010). There is neither a single board agreement upon systems engineering processes nor on the adequate application of the tools and methods to handle this complexity. This survey shall explain a methodology to implement SysML with the modelling tool on the example of developing a mechatronic system. It is attempted to enhance the approval of the modelling language and to improve dealing with systems complexity. Achieving consistency regarding the definition of requirements and functions as well as designing a coherent structure is fundamental and therefore considered as an overall objective.

3. Development of a mechatronic system by combining CONSENS and SysML

3.1. CONSENS

The specification technique CONSENS has been invented at the Heinz Nixdorf Institute, which is an interdisciplinary research institute at the University of Paderborn. CONSENS is very suitable for the intention of developing a SysML guideline for the Cameo Systems Modeller, because it introduces a step by step methodology for establishing a system model in the concept phase of mechatronic products (Maurer and Schulze, 2013). In contrast to SysML, CONSENS especially implements aspects which are mandatory for the design of mechatronic products from an engineering-scientific point of view (Gausemeier et al., 2011). As you can see in Figure 2, CONSENS structures the product conception into seven partial models (Gausemeier et al., 2014):

- Environment: regards every element in the surrounding of the system, such as user, underground, remote controls, objects or even other interacting systems.
- Application scenarios: determine the situations in which the system has to operate.
- Requirements: definition of customer and internal demands.
- Functions: derived from requirements the main function of the system has to be split up into partial functions to define the features of all subsystems and parts.
- Active structure: shows informative, energetic and material interactions between all parts of the system to get a better understanding of the functionality.
- Shape: derived from the active structure the 3D modelling of the system can be prepared.
- Behavior: describes the actions of the whole system in use, which is especially important for the interpretation of the software.



1	Geometry
1.1	Length: max. 3500mm
1.2	Width: max. 1500mm
1.3	Height: max. 1200mm
1.4	Distance Toolcenterpoint: 1000mm

Figure 2. CONSENS approach

Each partial model includes the elaboration of different development blocks like product requirements, specification of functions, construction drawings and so on. These different steps are guiding the engineer through the development process. The emerging partial solution determines the fundamental structure of the system, which is the basis for the further ascertainment.

3.2. Guideline resulting from the interfaces of CONSENS and SysML

The specification technique and the graphical modelling language show some interfaces in their structure which is why language concepts and approaches can be transferred from CONSENS to SysML (Gausemeier et al., 2014). These interfaces can be traced back to the basis of both methods which is systems engineering in general. MBSE consists of three blocks: requirements, architecture and system behavior. These blocks are also taken into account in those two regarded methods. As already mentioned SysML helps to describe and connect the structure, the behavior and the requirements of a system. The benefit of CONSENS is to get more information about a preferable order of the elaboration of specific development blocks. In this case the implementation order would be:

1. Defining requirements derived from potential customer wishes
2. Planning the functions on system and sub-system basis
3. Creating an active structure which includes components to realize all functions of the system
4. Displaying the behavior of the system in a typical use case
5. Designing and implementing the 3D model of the system into SysML

To minimize errors of the system and to realize a preferably perfect product these five steps have to be gone through repeatedly. It is also advisable to step back in specific stages of the process to implement adjustments. Later on, the paragraph "Implementation" describes the whole structure of the transcribed system in more detail. Nevertheless, the five mentioned aspects will be rediscovered.

3.3. Advantages of a basic SysML guideline

SysML doesn't have many consequent modeling guidelines yet, especially in view of applicable tools like the Cameo Systems Modeler. As a result inexperienced users have to think of their own approach. In most cases the system model includes more or less the same important parts, such as requirements, functions and so on. One more difficult aspect of the system modelling is the setup of a matching structure. Therefore it could be useful to work after a specific guiding principle. This would result in the following benefits:

- Time saving: Time savings can be achieved by indicating the user a specific guideline. The user may start with the modelling right away without designing a new model structure. The most important sections of the system modelling process are prescribed such as requirements, functions, structure and behavior.
- Comprehensibility: Guidelines also help users to understand the system model in less time. The user already knows the structure from previous projects which is why he/she can extract relevant information quicker.
- Error prevention: Certain standards can also avoid mistakes especially in the area of missing crucial information. The user knows which tasks he/she has to implement, this can also be considered as some kind of checklist.

3.4. Implementation

To accomplish the modelling of a system, SysML provides the user with different types of diagrams. For not getting into too much detail only the most important aspects will be mentioned at this point. There are two types of diagrams to work with:

- Diagrams for structural views: "The structural definitions of components and connectors are expressed using block definition diagrams (bdd). Configurations are defined with internal block diagrams (ibd). The block definition diagram is applied to specify the types of components and connectors as well as the types of data and ports. The internal block diagram is applied to specify the configuration in terms of components and connectors and how they are combined together.

Note that while architecture types are defined using bdd and architecture configurations are represented using ibd." (Oquendo et al., 2016, p. 54)

- Diagrams for behavioral views: "The behavioral definitions, i.e., definitions of activities and related actions, are expressed using block definition diagrams (bdd), activity diagrams (act), and parametric diagrams (par). The block definition diagram is applied to specify the interface of activities and actions. The activity diagram is applied to specify the internal behavior of components, connectors, and configurations as well as the protocols of ports in terms of actions, interactions, data, and control flows. The parametric diagram is applied to specify the behavior of actions in terms of equations that define the pre- and post-conditions." (Oquendo et al., 2016, p. 74)

Another diagram which can actually not really be assigned to one of these types is the requirement diagram which registers all relevant requirements of the system (Alt, 2012, p. 51).
The implementation process of this project can be divided into 6 parts in which only the first four are fundamental:

- Requirements
- Functions
- Structure
- Behavior
- Parameters
- Glossary

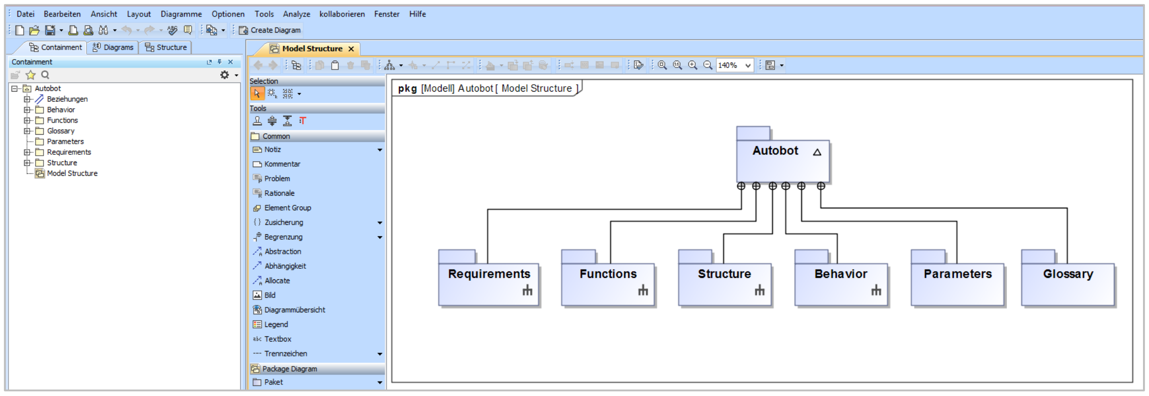
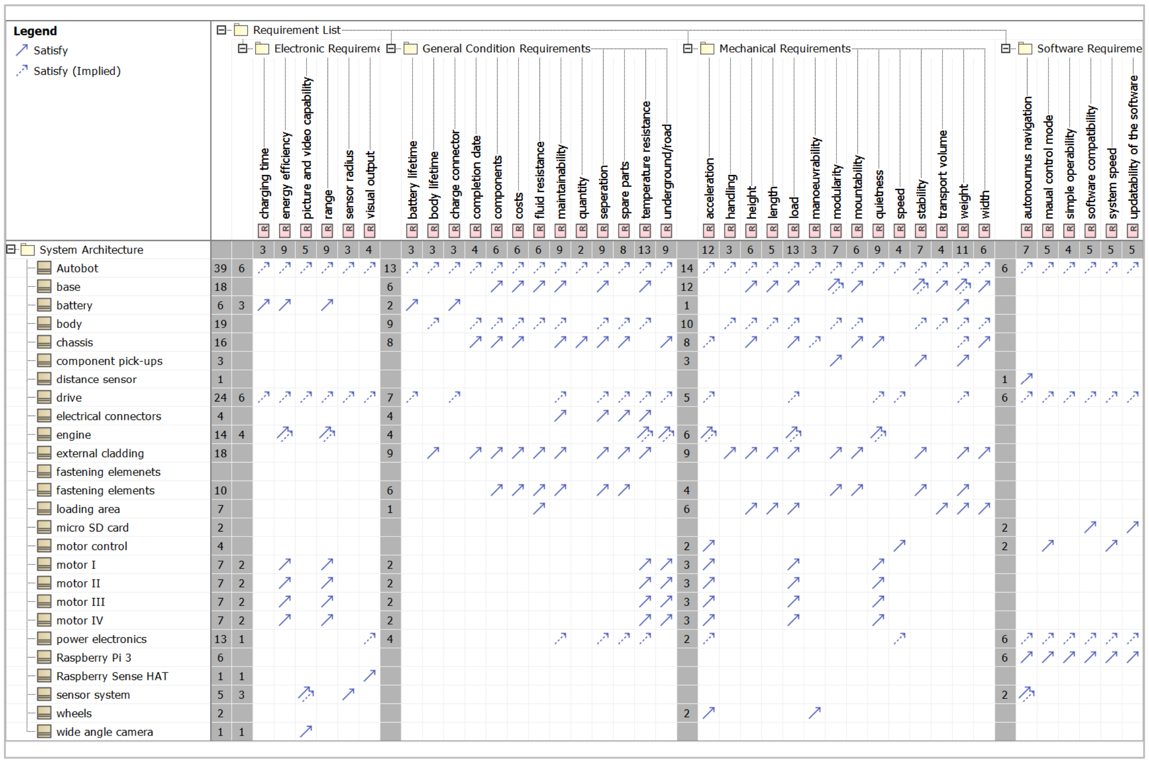


Figure 3. Program interface of Cameo Systems Modeler

The most important part of the Requirements section is the requirement list which includes all kinds of mechanical, electrical and software requirements. In case of the autonomous robot, the category general condition requirements, containing costs, quantity, maintainability etcetera, has been added. In addition to already mentioned definitions, according to the quality management standard DIN EN ISO 9001:2015 a requirement is a request or expectation, which is determined, usually presupposed or mandatory (DIN, 2015). The requirement phrasings must follow specific rules like describing the necessity of the requirement by using 'shall', 'should' and 'may'. The formulation also should contain, which part or sub-system is responsible for the fulfilment of the request (Alt, 2012, pp. 10–15). Before implementing the requirement structure and the requirement matrix all components have to be added to the system model which will be done at a later stage of the process. Nevertheless these two elements display which components are needed to satisfy a certain kind of requirement. Figure 4 shows the requirement matrix of the Autobot, presenting which system element satisfies what kind of requirement. The requirement list also represents a direct interface to the customer which sends requests and suggestions to the contractor in order to receive a satisfactory product at the end of the process. Other diagrams, lists and information is set up and collected by the respective departments and adapted to the requirements.



Legend
↗ Satisfy
⇢ Satisfy (Implied)

	Requirement List	Electronic Requireme	charging time	energy efficiency	picture and video capability	range	sensor radius	visual output
System Architecture			3	9	5	9	3	4
Autobot	39	6	⇢	⇢	⇢	⇢	⇢	⇢
base	18							
battery	6	3	↗	↗		↗		
body	19							
chassis	16							
component pick-ups	3							
distance sensor	1							
drive	24	6	⇢	⇢	⇢	⇢	⇢	⇢
electrical connectors	4							
engine	14	4		↗⇢		↗⇢		
external cladding	18							
fastening elemenets								
fastening elements	10							
loading area	7							
micro SD card	2							
motor control	4							
motor I	7	2		↗		↗		
motor II	7	2		↗		↗		
motor III	7	2		↗		↗		
motor IV	7	2		↗		↗		
power electronics	13	1						⇢
Raspberry Pi 3	6							
Raspberry Sense HAT	1	1						↗
sensor system	5	3			↗⇢		↗	
wheels	2							
wide angle camera	1	1			↗			

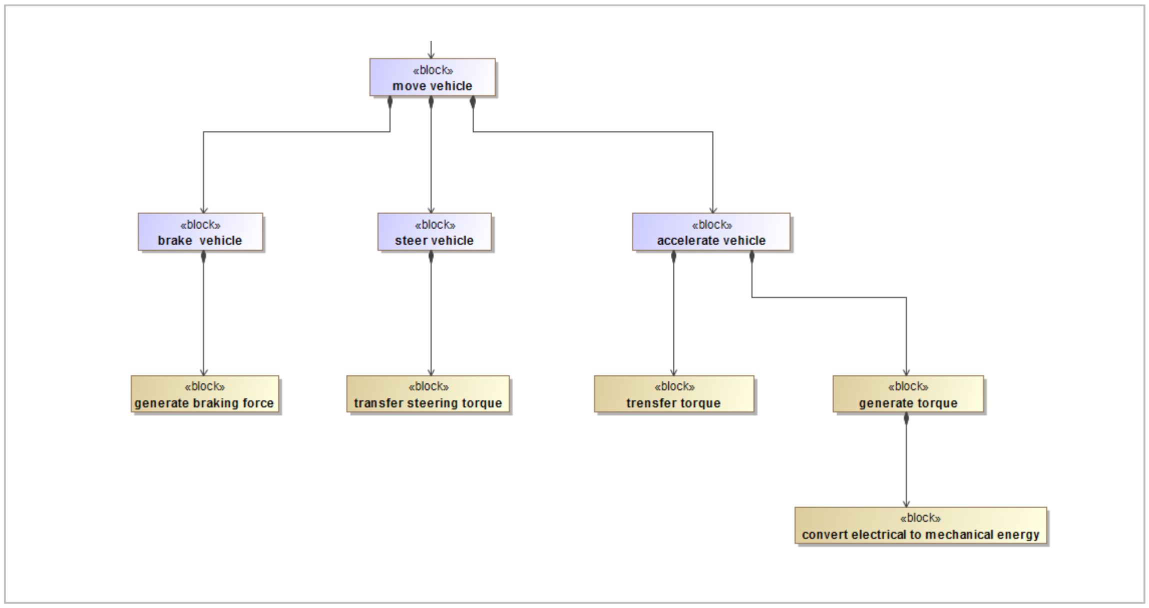
	General Condition Requirements	battery lifetime	body lifetime	charge connector	completion date	components	costs	fluid resistance	maintainability	quantity	seperation	spare parts	temperature resistance	underground/road
System Architecture		3	3	3	4	6	6	6	9	2	9	8	13	9
Autobot	13	⇢	⇢	⇢	⇢	⇢	⇢	⇢	⇢	⇢	⇢	⇢	⇢	⇢
base	6					↗	↗	↗	↗		↗		↗	
battery	2	↗		↗										
body	9		⇢		⇢	⇢	⇢	⇢	⇢		⇢	⇢	⇢	
chassis	8				↗	↗	↗		↗	↗	↗	↗		↗
component pick-ups														
distance sensor														
drive	7	⇢		⇢					⇢		⇢	⇢	⇢	⇢
electrical connectors	4								↗		↗	↗	↗	
engine	4												↗⇢	↗⇢
external cladding	9		↗		↗	↗	↗	↗	↗		↗	↗	↗	
fastening elemenets														
fastening elements	6					↗	↗	↗	↗		↗	↗		
loading area	1							↗						
micro SD card														
motor control														
motor I	2												↗	↗
motor II	2												↗	↗
motor III	2												↗	↗
motor IV	2												↗	↗
power electronics	4								⇢		⇢	⇢	⇢	
Raspberry Pi 3														
Raspberry Sense HAT														
sensor system														
wheels														
wide angle camera														

	Mechanical Requirements	acceleration	handling	height	length	load	manoeuvrability	modularity	mountability	quietness	speed	stability	transport volume	weight	width
System Architecture		12	3	6	5	13	3	7	6	9	4	7	4	11	6
Autobot	14	⇢	⇢	⇢	⇢	⇢	⇢	⇢	⇢	⇢	⇢	⇢	⇢	⇢	⇢
base	12			↗	↗	↗		↗⇢	↗			↗⇢	↗	↗⇢	↗
battery	1													↗	
body	10		⇢	⇢	⇢	⇢		⇢	⇢			⇢	⇢	⇢	⇢
chassis	8	⇢		↗		↗	⇢		↗	↗				⇢	↗
component pick-ups	3							↗				↗		↗	
distance sensor															
drive	5	⇢				⇢				⇢	⇢			⇢	
electrical connectors															
engine	6	↗⇢				↗⇢				↗⇢					
external cladding	9		↗	↗	↗	↗		↗	↗			↗		↗	↗
fastening elemenets															
fastening elements	4							↗	↗			↗		↗	
loading area	6			↗	↗	↗							↗	↗	↗
micro SD card															
motor control	2	↗									↗				
motor I	3	↗				↗				↗					
motor II	3	↗				↗				↗					
motor III	3	↗				↗				↗					
motor IV	3	↗				↗				↗					
power electronics	2	⇢									⇢				
Raspberry Pi 3															
Raspberry Sense HAT															
sensor system															
wheels	2	↗					↗								
wide angle camera															

	Software Requireme	autonoumus navigation	maual control mode	simple operability	software compatibility	system speed	updatability of the software
System Architecture		7	5	4	5	5	5
Autobot	6	⇢	⇢	⇢	⇢	⇢	⇢
base							
battery							
body							
chassis							
component pick-ups							
distance sensor	1	↗					
drive	6	⇢	⇢	⇢	⇢	⇢	⇢
electrical connectors							
engine							
external cladding							
fastening elemenets							
fastening elements							
loading area							
micro SD card	2				↗		↗
motor control	2		↗			↗	
motor I							
motor II							
motor III							
motor IV							
power electronics	6	⇢	⇢	⇢	⇢	⇢	⇢
Raspberry Pi 3	6	↗	↗	↗	↗	↗	↗
Raspberry Sense HAT							
sensor system	2	↗⇢					
wheels							
wide angle camera							

Figure 4. Requirement matrix

The next basic element of the system model is Functions, which describe the main tasks of the system and its sub-systems. All functions are shown in the function hierarchy. As shown in Figure 5 every main function has been broken down to very basic functions which can be allocated to specific components afterwards. The classification of the functions into functional (blue) and non-functional (yellow) has also been implemented at this point.



«block» move vehicle
«block» brake vehicle
«block» steer vehicle
«block» accelerate vehicle
«block» generate braking force
«block» transfer steering torque
«block» trensfer torque
«block» generate torque
«block» convert electrical to mechanical energy

Figure 5. Function hierarchy

The detail also shows the subdivision of the task 'move vehicle' into the functions braking, steering and accelerating. The subordinated, non-functional tasks, describe how the functional tasks are implemented on the mechanical point of view. For the sake of completeness, the other main functions of the robot are: autonomous control, energy supply, hardware protection and load carrying. After these main functions have been broken down, the system contains 45 functions in total.
For gaining a better overall view the Structure element is very helpful. It contains almost all important aspects to understand the implemented system, such as the architecture and the active structure. The architecture gives an overview of all components of the system (Figure 6) whereas the active structure displays all detailed interactions between these components (Figure 7). When designing the architecture, the system is split up from the top, down to more and more detail. The Autobot in this project contains three sub-systems: the drive, the chassis and the body. Sub-systems can either be broken down to sub-sub-systems or to the component layer. The chassis in this case for example contains no further components than 'fastening elements' and 'wheels' because there is a direct connection to the engine with no suspension or steering gear. Every sub-system should be broken down until it can be considered as a component. This simplifies the preparation of the list of parts and the active structure afterwards.

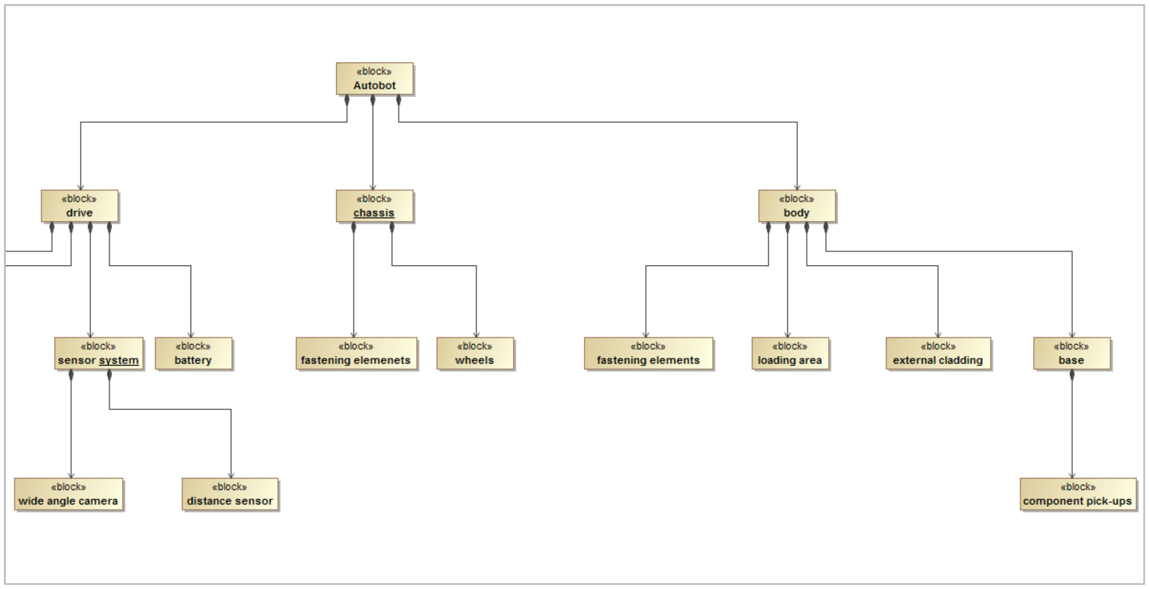


Figure 6. System architecture

The active structure displays and describes interactions (green) between these components (blue). It shows how every part is implemented in the whole system, how the system is connected to the environment (yellow) and where the interfaces are. Usually the component with the highest importance and adjustment effort in an active structure is the one with the most connections, because it has to be coordinated well. In the case of the Autobot, the Raspberry Pi, a single-board computer, represents the core of the system, by transmitting signals, data and electrical energy to many sub-components. Nevertheless every missing or damaged component within the system boundary can cause a malfunction.

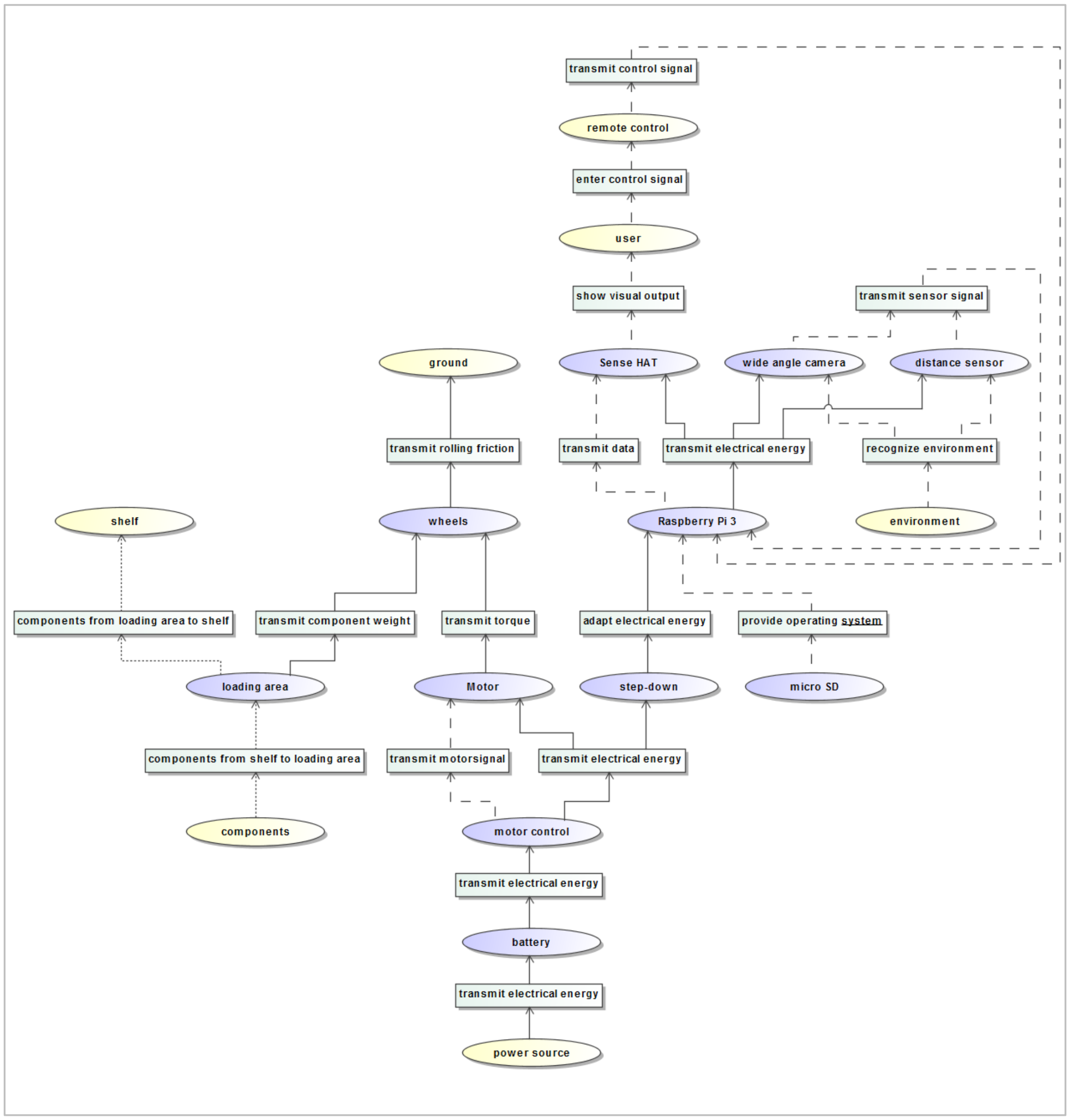


Figure 7. Active structure

The last central aspect of the system modeling presents the behavior of the product. It includes all steps of a use case from launching the system to having fulfilled its purpose (Figure 8).
As already mentioned the Autobot shall provide a production line with components autonomously by recognizing a lack of parts and calculating a route by itself. Therefore the consideration of the interfaces to environment (see above) is very important and should be implemented accurately.

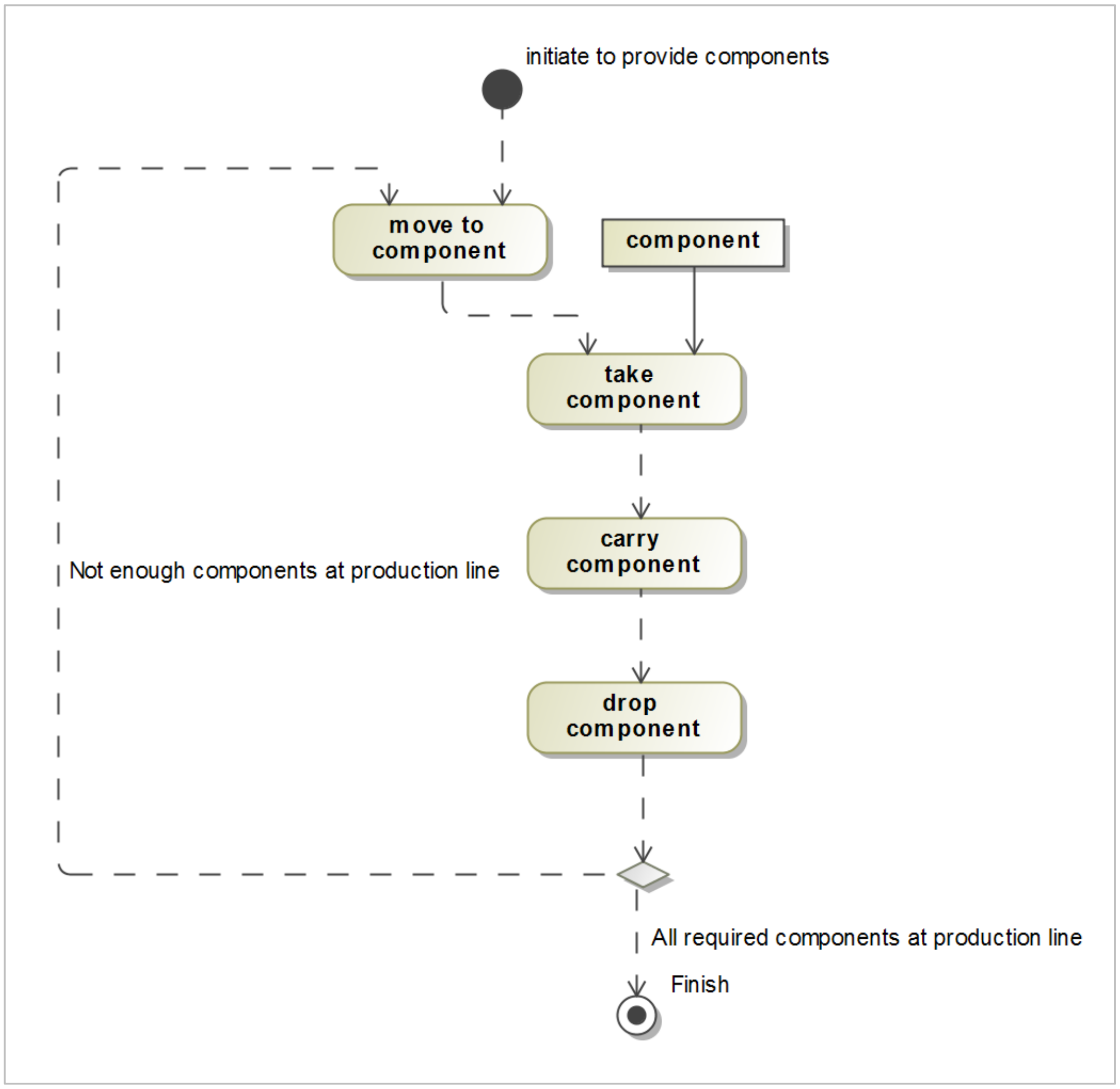

initiate to provide components
move to component
component
take component
carry component
Not enough components at production line
drop component
All required components at production line
Finish

Figure 8. Activity diagram

The elements Parameters and Glossary only have the purpose to deduce certain calculated values and to implement definitions. Nevertheless it can improve the understandability of the system model, which results in less communication effort due to fewer misunderstandings.

4. Benefits for the team using SysML

The advantages of SysML, by using the Cameo Systems Modeller, which have appeared during the project are obvious. Due to the exact structuring of the system with many different diagram types it is possible to understand complex systems in a short time. As already mentioned the system architecture and the active structure are very useful at this point. SysML is the perfect foundation for project discussions because it provides team members with key information. Finding solutions or implementing improvements is also more likely because the model displays all sub-systems and components of the product and shows them in interaction with the entire system. SysML is also a kind of central file location. Calculations, 3D-models and component information can be collected within the system model which can reduce administrative expenses in many ways.

5. Summary

As well as model-based systems engineering, SysML is a quite new procedure about which many potential users are skeptical. Lots of aspects will be optimized and overthought in the future to change this issue. The increasing dissemination of MBSE will lead to higher importance and acceptance of SysML. This approach of using some specific steps from the CONSENS method is one kind of optimization. It shall help to develop a guideline for the application of SysML with tools like the Cameo Systems Modeller.

At the launch of this project, the usage of the Systems Modeller had to be learned by the participants. Since the Cameo Systems Modeller leaves many structuring decisions to the user, which also has some advantages, the consistency is a huge problem. In the future the presented approach could help to facilitate the introduction of modelling languages and tools to inexperienced companies in that section as well as realizing higher consistency across different projects.

References

Albers, A. and Zingel, C. (2013), "Challenges of Model-Based Systems Engineering: A Study towards Unified Term Understanding and the State of Usage of SysML", *Smart product engineering: Proceedings of the 23rd CIRP Design Conference, Bochum, Germany, March 11-13, 2013*, Springer, Berlin, pp. 83–92. https://doi.org/10.1007/978-3-642-30817-8_9

Alt, O. (2012), *Modellbasierte Systementwicklung mit SysML*, Carl Hanser Fachbuchverlag, München. https://doi.org/10.3139/9783446431270.

DIN (2015), DIN EN ISO 9001:2015 Quality management systems – Requirements, Deutsches Institut für Normung e. V., Berlin.

Dori, D. (2016), *Model-Based Systems Engineering with OPM and SysML,* 1st ed., Springer New York, New York, NY. https://doi.org/10.1007/978-1-4939-3295-5

Estefan, J.A. (2007), "Survey of Model-Based Systems Engineering (MBSE) Methodologies", *Incose MBSE Focus Group*, Vol. 25 No. 8, pp. 1-12.

Gausemeier, J. and Behmann, B. (2012), *Produkte und Produktionssysteme integrativ konzipieren: Modellbildung und Analyse in der frühen Phase der Produktentstehung,* 1st ed., Carl Hanser Fachbuchverlag.

Gausemeier, J., Dumitrescu, R., Tschirner, C. and Stille, K.S. (2011), "Modellbasierte Konzipierung eines hybriden Energiespeichersystems für ein autonomes Schienenfahrzeug", *Tag des Systems Engineering,* Heinz Nixdorf Institut, Universität Paderborn.

Gausemeier, J., Trächtler, A., Schäfer, W. and Anacker, H. (2014), *Semantische Technologien im Entwurf mechatronischer Systeme: Effektiver Austausch von Lösungswissen in Branchenwertschöpfungsketten*, Hanser, München. https://doi.org/10.3139/9783446438453

Hampson, K. (2015), "Technical Evaluation of the Systems Modeling Language (SysML)", *Procedia Computer Science*, Vol. 44, pp. 403–412. https://doi.org/10.1016/j.procs.2015.03.054

Hoffmann, H.-P. (2006), "SysML-Based Systems Engineering Using a Model-Driven Development Approach", *INCOSE International Symposium*, Vol. 16 No. 1, pp. 804–814. https://doi.org/10.1002/j.2334-5837.2006.tb02782.x

IEEE (1998), IEEE Standard for Application and Management of the Systems Engineering Process, IEEE, Piscataway, NJ, USA. https://doi.org/10.1109/IEEESTD.1999.88825.

Kasser, J.E. (2010), "Seven systems engineering myths and the corresponding realities", *Proceedings of the Systems Engineering Test and Evaluation Conference; Adelaide, Australia*.

Long, J.E. (2000), *Systems Engineering (SE) 101: CORE: Product & Process Engineering Solutions*, Vitech training materials, Vienna.

Martin, J.N. (1997), *Systems engineering guidebook: A process for developing systems and products, Systems engineering series*, CRC Press, Boca Raton.

Maurer, M. and Schulze, S.-O. (2013), *Tag des Systems Engineering: Stuttgart 6. - 8. November 2013 ; Der Weg zu den technischen Systemen von morgen,* 1st ed., Carl Hanser Fachbuchverlag. https://doi.org/10.3139/9783446439467

Mhenni, F., Choley, J.-Y., Penas, O., Plateaux, R. and Hammadi, M. (2014), "A SysML-based methodology for mechatronic systems architectural design", *Advanced Engineering Informatics*, Vol. 28 No. 3, pp. 218–231. https://doi.org/10.1016/j.aei.2014.03.006

No Magic Inc. (2017), *Cameo Systems Modeler*. [online] Available at: https://www.nomagic.com/products/cameo-systems-modeler (accessed 09.05.2017).

Object Management Group (2017a), *About the Object Management Group*. [online] Available at: http://www.omg.org/gettingstarted/gettingstartedindex.htm (accessed 7 June 2017).

Object Management Group (2017b), *What is SysML? | OMG SysML*. [online] Available at: http://www.omgsysml.org/what-is-sysml.htm (accessed 24.05.2017).
Oquendo, F., Leite, J. and Batista, T. (2016), *Software Architecture in Action*, Springer International Publishing, Cham. https://doi.org/10.1007/978-3-319-44339-3
Weilkiens, T. and Soley, R.M. (2014), *Systems Engineering mit SysML/UML: Anforderungen, Analyse, Architektur,* 3rd ed.*,* Heidelberg.

Prof. Dr.-Ing. Vahid Salehi, Head of the Institute
Munich University of Applied Sciences, Engineering Design of Mechatronics
Loth Str. 34, 80335 Munich, Germany
Email: salehi-d@hm.edu