TECHNISCHE UNIVERSITÄT MÜNCHEN

Lehrstuhl für Automatisierung und Informationssysteme

Interdisciplinary structural modeling of mechatronic production systems using SysML4*Mechatronics*

Konstantin Emanuel Thassilo Kernschmidt

Vollständiger Abdruck der von der Fakultät für Maschinenwesen der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender: Prof. Dr.-Ing. Michael Zäh

Prüfer der Dissertation:

1. Prof. Dr.-Ing. Birgit Vogel-Heuser
2. Prof. Dr. Cesare Fantuzzi, Università degli studi di Modena e Reggio Emilia / Italien

Die Dissertation wurde am 10.10.2018 bei der Technischen Universität München eingereicht und durch die Fakultät für Maschinenwesen am 10.01.2019 angenommen.

Bibliografische Information der Deutschen Nationalbibliothek

Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie;
detaillierte bibliografische Daten sind im Internet über <http://dnb.ddb.de> abrufbar.

Interdisciplinary structural modeling of mechatronic production systems using SysML4Mechatronics

Autor:
Konstantin Kernschmidt

ISBN 13: 978-3-96548-025-4
1. Auflage 2019

Friedländer-Weg 54
37085 Göttingen
Tel.: +49 (0)551 5036647

Coverdesign: sierke MEDIA

“All models are wrong, but some are useful.”

George Box

Acknowledgments

Conducting research is like a journey of several years. Comparable to a travel guide, this dissertation is the compressed representation of all relevant information. However, the most important experiences arise from the people, who I met and with whom I worked together on this journey. I want to thank them for their support, ideas and discussions, which contributed extensively to writing this dissertation.

First, I want to thank Prof. Dr.-Ing. Birgit Vogel-Heuser from the Institute of Automation and Information Systems at the Technical University of Munich for giving me the opportunity to write this dissertation. Prof. Dr. Cesare Fantuzzi from the Università degli studi di Modena e Reggio Emilia I want to thank not only for acting as second supervisor of my dissertation, but also for giving me the chance to visit his institute and discuss my research. Prof. Dr.-Ing. Michael Zäh from the Institute for Machine Tools and Industrial Management I want to thank for chairing the examination committee.

The largest part of my research was carried out in the context of the Collaborative Research Centre SFB 768 “Managing cycles in innovation processes – Integrated development of product service systems based on technical products”. I want to thank all participants of the SFB 768 for their collaboration and the preparation of the proposal (until late at night). Together we applied successfully for the third funding period.

Prof. Christiaan J.J. Paredis, Univ.-Prof. Dipl.-Ing. Dr.sc.techn. Georg Schitter, Prof. Dr. Marga Marcos, Prof. Dr.-Ing. Elisabet Estévez, Assistant Prof. Mag.rer.soc.oec. Dr.rer.soc.oec. Manuel Wimmer, Dr. Giacomo Barbieri, Dr. Sebastian Herzig and Ahsan Qamar, PhD I want to thank for their collaborative projects during my research.

I want to thank all colleagues of the institute for their time and the coffees, which they shared with me, especially Gennadiy, Johann, Patrik, Sebastian and Stefan. Furthermore, I thank all students, whose theses I supervised, for their exchange of ideas on the development of mechatronic production systems. Representatively, I want to name Linda Hundegger and Leonardo von Lerchenfeld.

My special thanks go to all my family and my in-laws for their time and their support throughout my dissertation. Finally, lieve Debbie, I want to thank you for your patience and your understanding in all situations. Let me hold your hand forever.

Konstantin Emanuel Thassilo Kernschmidt
On a journey, March 2019

Abstract

Model-Based Engineering (MBE) in the machine and plant manufacturing industry poses an emerging and growing approach to enable an integrated development and information usage over the entire life cycle. Especially for modern mechatronic production systems, for the growing necessity to provide flexible, variant rich systems, and for upcoming trends such as Industry 4.0, traditional development and engineering methods lack sufficient possibilities to handle the system's complexity and consider the interdisciplinary dependencies. Furthermore, mechatronic systems usually evolve during their usage, e.g., due to new customer requirements or necessary reengineering, and require changes or updates of parts of the system. Such changes have various interdisciplinary influences on the system's structure and can lead to undesired downtimes or additional costs, if the new interfaces (called ports) are not compatible with the existing system.

The research presented in this thesis contributes to the goal of facing these challenges through the development of an interdisciplinary modeling approach for mechatronic production systems. On the one hand, it enables to analyze the dependencies of the different involved disciplines in the early development phase by integrating the components from mechanics, electrics/electronics, and software. On the other hand, the approach focuses on the integration of ports and properties in the model, which can be used as basis for analyzing the components' compatibility. Thus, the approach assists in the usage phase to estimate the impact of a change, before it is conducted in the real system.

Based on the standardized language for Model-Based Systems Engineering – SysML – the approach SysML4*Mechatronics* was developed and defined as a profile of the SysML metamodel in the MOF Model Driven Architecture Framework of the Object Management Group. Through according stereotypes, components from the different disciplines can be integrated in the interdisciplinary model. In order to facilitate composition and reuse, the approach enables, furthermore, a function-oriented modularization. The model purpose and thus, the included properties, constitute important prerequisites for the modeling and development process. In contrast, the associated required effort to create the model and add all relevant properties depicts a main reason for the resistance in industry to use model-based approaches. Therefore, this thesis investigates how properties can be included from standardized product classification systems, which are available, e.g., from design or (e-commerce) sourcing catalogs. Depending on the particular focus of the classification systems, they contain different relevant properties for analyzing compatibility, e.g., the wide spread classification system eCl@ss offers properties for industrial mechatronic components, while the NIST Functional Basis for Engineering Design offers a classification to define the components'

functionality. This thesis shows how such classified properties can be integrated as model library in SysML4*Mechatronics* in order to facilitate the modeling process. However, the applicability of such classification systems is strongly dependent on the utilized components in the system and thus, limitations, especially regarding module-properties, are discussed.

In order to evaluate the feasibility of the approach, a proof-of-concept implementation of a SysML4*Mechatronics* modeling editor was developed and applied to a production system use case. Based thereon, the required tasks in the modeling process and the potentials of the interdisciplinary modeling approach, as basis for analyzing the compatibility in case of a change, are discussed. The participants in two further academic case studies evaluated the practical usability of the approach, i.e., the perceived usefulness and perceived ease of use. Therefore, several modeling assignments were executed by the participants in the SysML4*Mechatronics* modeling editor and subsequently the approach was rated in a questionnaire based on the technology acceptance model (TAM).

Besides the described research contributions, this thesis also provides several implications for industrial practice by fostering a detailed comprehension of interdisciplinary dependencies in mechatronic production systems. This supports companies in the mechatronic production systems domain to increase their ability in handling the complexity and thus, accelerate the development process. Furthermore, companies have a more sophisticated possibility to analyze changes and foster the communication between the different involved disciplines.

Zusammenfassung

Modellbasiertes Engineering (MBE) im Maschinen- und Anlagenbau stellt einen neuen, aufstrebenden Ansatz dar, um eine integrierte Entwicklung und eine Durchgängigkeit über den gesamten Lebenszyklus zu ermöglichen. Herkömmliche Entwicklungs- und Engineeringmethoden bieten insbesondere für moderne mechatronische Systeme mit der steigenden Notwendigkeit, flexible und variantenreiche Produktionssysteme zur Verfügung zu stellen, sowie für neue Trends (z.B. Industrie 4.0) keine ausreichenden Möglichkeiten, um mit der Systemkomplexität und den interdisziplinären Zusammenhängen im System umzugehen. Zudem entwickeln sich mechatronische Systeme im Laufe ihres Lebenszyklus typischerweise weiter, z.B. aufgrund neuer Kundenanforderungen oder einem notwendigen Reengineering, weshalb Veränderungen oder Updates in Teilen des Systems vorgenommen werden müssen. Solche Änderungen beeinflussen verschiedene interdisziplinäre Systemzusammenhänge, was zu unerwünschten Ausfällen oder zusätzlichen Kosten führen kann, wenn die Eigenschaften und Schnittstellen (sogenannte Ports) der ausgetauschten Komponenten nicht mehr mit dem restlichen bestehenden System kompatibel sind.

Die in dieser Dissertation dargestellte Forschungsarbeit trägt zur Bewältigung der beschriebenen Herausforderungen bei. Hierzu wurde ein interdisziplinärer Modellierungsansatz für mechatronische Systeme entwickelt. Einerseits ermöglicht dieser Ansatz bereits in frühen Entwicklungsstadien eine Untersuchung der Abhängigkeiten zwischen den verschiedenen beteiligten Disziplinen, indem die Komponenten der Mechanik, Elektrik/Elektronik und Software in einem übergreifenden Modell integriert werden. Andererseits bilden die im Modell integrierten Komponentenmerkmale eine Basis für die modellbasierte Analyse von Änderungen. Somit unterstützt der Modellierungsansatz die Abschätzung der Änderungs-auswirkungen, bevor diese im realen System umgesetzt werden.

Basierend auf der im Modellbasierten Systems Engineering (MBSE) verbreiteten Sprache SysML wurde der Ansatz SysML4*Mechatronics* entwickelt und als Profil des SysML Metamodells im *MOF Model Driven Architecture Framework* der Object Management Group definiert. Durch entsprechende Stereotypen können Komponenten der verschiedenen Disziplinen in das interdisziplinäre Modell integriert werden. Um den Entwicklungsprozess und die Wiederverwendung zu erleichtern, ermöglicht der Ansatz zudem eine funktionsorientierte Modularisierung. Der Modellierungszweck und somit die enthaltenen Merkmale stellen wichtige Grundvoraussetzungen für den Modellierungs- und Entwicklungsprozess dar. Allerdings ist der Aufwand zur Erstellung eines Modells und zur Integration der relevanten Merkmale einer der Hauptgründe, weshalb modellbasierte Ansätze in der Industrie bisher noch keine umfassende Verbreitung gefunden haben. Daher wird in dieser Dissertation untersucht, in welchem

Umfang Merkmale aus standardisierten Produktklassifikationssystemen, welche beispielsweise bereits im Bereich E-Commerce oder Design existieren, genutzt werden können. Der weitverbreitete eCl@ss-Standard verfügt z.B. über Merkmale für industrielle (mechatronische) Komponenten, während die *NIST Functional Basis for Engineering Design* eine Klassifikation zur Funktionsbeschreibung von Komponenten bereitstellt. In dieser Dissertation wird untersucht, wie solche klassifizierten Merkmale als Bibliothek in SysML4*Mechatronics* integriert werden können, um den Modellierungsprozess zu vereinfachen, und wie diese Merkmale als Basis einer Kompatibilitätsprüfung genutzt werden können. Da die Anwendbarkeit solcher Klassifikationssysteme stark davon abhängt, welche Komponenten im System eingebunden werden sollen, werden resultierende Einschränkungen und notwendige Erweiterungen, insbesondere bezüglich Modulmerkmalen, diskutiert.

Zur Evaluation des Ansatzes wurde ein SysML4*Mechatronics*-Modellierungswerkzeug prototypisch implementiert. Auf dieser Basis wurden das Potential des interdisziplinären Modellierungsansatzes, die im Modellierungsprozess erforderlichen Aufgaben sowie die Integration der Merkmale als Basis einer Kompatibilitätsprüfung durch Anwendungsbeispiele überprüft. In Fallstudien wurden weiterhin die Akzeptanz und die Gebrauchstauglichkeit dieses Ansatzes evaluiert. Dazu wurden von den Studienteilnehmern verschiedene Modellierungsaufgaben im SysML4*Mechatronics*-Modellierungswerkzeug durchgeführt. Daraufhin wurde der Ansatz auf Basis des *Technology Acceptance Models* (TAM) evaluiert.

Neben den beschriebenen Beiträgen zur Forschung liefert der in dieser Dissertation entwickelte Ansatz auch Anknüpfungspunkte für die industrielle Praxis. Ein detailliertes Verständnis der interdisziplinären Zusammenhänge in mechatronischen System unterstützt Unternehmen des Maschinen- und Anlagenbaus in der Komplexitäts-beherrschung und beschleunigt folglich den Entwicklungsprozess. Unternehmen können mit dem entwickelten Ansatz die interdisziplinären Zusammenhänge in ihren Systemen und potentielle Änderungsauswirkungen detaillierter untersuchen und zudem die Kommunikation zwischen den unterschiedlichen beteiligten Disziplinen verbessern.

The following publications are part of the work presented in this thesis:

- **Kernschmidt, K.; Feldmann, S.; Vogel-Heuser, B.:** *A model-based framework for increasing the interdisciplinary design of mechatronic production systems.* In: Journal of Engineering Design, 2018, DOI: 10.1080/09544828.2018.1520205.
- **Feldmann, S.; Kernschmidt, K.; Vogel-Heuser, B.:** *Konzept eines wissens-basierten Frameworks zur Spezifikation und Diagnose von Inkonsistenzen in mechatronischen Modellen.* In: at - Automatisierungstechnik, vol. 64, no. 3, 2016, pp. 199–215, DOI: 10.1515/auto-2015-0081.
- **Feldmann, S.; Wimmer, M.; Kernschmidt, K.; Vogel-Heuser, B.:** *A Comprehensive Approach for Managing Inter-Model Inconsistencies in Automated Production Systems Engineering.* In: 12th annual IEEE International Conference on Automation Science and Engineering (CASE), Fort Worth, USA, 2016.
- **Kernschmidt, K.; Vogel-Heuser, B.; Preißner, S.; Christina, R.:** *From selling products to providing user oriented product-service systems – Exploring service orientation in the German machine and plant manufacturing industry.* In: 12th IFIP International Conference on Product Lifecycle Management (PLM), Doha, Qatar, 2015.
- **Feldmann, S., Herzig, S., Kernschmidt, K., Wolfenstetter, T., Kammerl, D., Qamar, A., Lindemann, U., Krcmar, H., Paredis, C. J. J., Vogel-Heuser, B.:** *A Comparison of Inconsistency Management Approaches Using a Mechatronic Manufacturing System Design Case Study.* In: 11th annual IEEE International Conference on Automation Science and Engineering (CASE), Gothenburg, Sweden, 2015.
- **Kernschmidt, K.; Barbieri, G.; Fantuzzi, C.; Vogel-Heuser, B.:** *Interdisziplinäre modellbasierte Entwicklung mechatronischer Systeme basierend auf SysML zur Steigerung der Wiederverwendung.* In: Kongress Automation, Baden-Baden, Germany, 2014.
- **Kernschmidt, K.; Behncke, F.; Chucholowski, N.; Wickel M.; Bayrak, G.; Lindemann, U.; Vogel-Heuser, B.:** *An integrated approach to analyze change-situations in the development of production systems.* In: 47th CIRP Conference on Manufacturing Systems (CMS), Ontario, Canada, 2014.
- **Kernschmidt, K.; Feldmann, S.; Vogel-Heuser, B.:** *Analyse von Änderungsauswirkungen in langlebigen Automatisierungssystemen.* In: 1st Collaborative Workshop on Evolution and Maintenance of Long-Living Systems (EMLS), Kiel, Germany, 2014.
- **Feldmann, S.; Legat, C.; Kernschmidt, K.; Vogel-Heuser, B.:** *Compatibility and Coalition Forming: Towards the Vision of an Automatic Synthesis of Manufacturing System Designs.* In: 23rd IEEE International Symposium on Industrial Electronics (ISIE), Istanbul, Turkey, 2014.
- **Feldmann, S.; Kernschmidt, K.; Vogel-Heuser, B.:** *Combining a SysML-based modeling approach and semantic technologies for analyzing change influences in manufacturing plant models.* In: 47th CIRP Conference on Manufacturing Systems (CMS), Ontario, Canada, 2014.

- **Barbieri, G.; Kernschmidt, K.; Fantuzzi, C.; Vogel-Heuser, B.:** *A SysML based design pattern for the high-level development of mechatronic systems to enhance re-usability.* In: 19th World Congress of the International Federation of Automatic Control, Cape Town, South Africa, 2014.
- **Maisenbacher, S.; Kernschmidt, Konstantin; Kasperek, Daniel; Vogel-Heuser, B.; Maurer, M.:** U*sing DSM and MDM Methodologies to Analyze Structural SysML Models.* In: International Conference on Industrial Engineering and Engineering Management (IEEM), Bangkok, Thailand, 2013.
- **Kernschmidt, K.; Wolfenstetter, T.; Münzberg, C.; Kammerl, D.; Goswami, S.; Lindemann, U.; Krcmar, H.; Vogel-Heuser, B.:** *Concept for an Integration-Framework to enable the crossdisciplinary Development of Product-Service Systems*. In: International Conference on Industrial Engineering and Engineering Management (IEEM), Bangkok, Thailand, 2013.
- **Kernschmidt, K.; Vogel-Heuser, B.:** *An interdisciplinary SysML based modeling approach for analyzing change influences in production plants to support the engineering.* In: 9th annual IEEE International Conference on Automation Science and Engineering (CASE), Madison, USA, 2013.
- **Kernschmidt, K.; Klein, P.; Jazdi, N.; Göhner, P.; Weyrich, M.; Vogel-Heuser, B.:** *Methodology for Identification of adaptive Reusable Modules in automated Production Systems.* In: Smart Product Engineering: Proceedings of the 23rd CIRP Design Conference, Bochum, Germany, 2013.
- **Kernschmidt, K.; Barbieri, G.; Fantuzzi, C.; Vogel-Heuser, B.:** *Possibilities and challenges of an integrated development using a combined SysML-model and corresponding domain specific models.* In: IFAC Conference on Manufacturing Modeling, Management, and Control (MIM), St. Petersburg, Russia, 2013.
- **Li, F.; Bayrak, G.; Kernschmidt, K.; Vogel-Heuser, B.:** *Specification of the Requirements to Support Information Technology-Cycles in the Machine and Plant Manufacturing Industry*. In: 14th IFAC Symposium on Information Control Problems in Manufacturing (INCOM), Bucharest, Romania, 2012.
- **Kernschmidt, K.; Hepperle, C.; Mörtl, M.; Vogel-Heuser, B.:** *Lifecycle oriented planning of mechatronic products and corresponding services.* In: IFIP WG 5.1 9th International Conference on Product Lifecycle Management (PLM), Montréal, Canada, 2012.
- **Kernschmidt, K.; Bayrak, G.; Vogel-Heuser, B.:** *A port-based approach for modelling the structure of mechatronic modules.* In: 14th International Dependency and Structure Modelling Conference (DSM), Kyoto, Japan, 2012.

Contents

1. Introduction 3
1.1. Motivation and problem statement 3
1.2. Research objectives and methodology 6
1.3. Structure of the thesis 10

2. Background and scope of research 12
2.1. Mechatronic systems in machine and plant manufacturing 12
2.2. Development processes of mechatronic systems 15
2.3. Model-Based Engineering 18
2.4. Summary 21

3. Requirements and boundary conditions 22
3.1. Identification of challenges in industry 22
3.1.1. Data collection method 23
3.1.2. Participants, results and analysis of the survey 23
3.2. Requirements for the modeling approach 29
3.3. Considered boundary conditions 36
3.4. Summary 37

4. State of the art 38
4.1. Modeling approaches 38
4.1.1. Unified Modeling Language (UML) 38
4.1.2. Systems Modeling Language (SysML) 40
4.1.3. MBE-approaches for mechatronic systems 44
4.2. Properties 51
4.2.1. Use of properties for engineering of mechatronic systems 51
4.2.2. Property classification systems 52
4.2.3. Property-based approaches in engineering 55
4.3. System modularization 57
4.3.1. Goals of modularization 58
4.3.2. Granularity of modules 59
4.3.3. Modularization-Based approaches 60
4.4. Summary and research gap 63

5. Concept of the interdisciplinary MBE-approach SysML4*Mechatronics* 66
5.1. SysML4*Mechatronics* metamodel 66
5.1.1. Components and modules 68
5.1.2. Ports and relations 69

5.1.3. Properties and functionalities 71
5.2. Concrete Syntax of SysML4*Mechatronics* 74
5.3. Integration of properties from standardized classification systems 76
5.3.1. Identification and classification of relevant properties 76
5.3.2. Integration in modeling approach 79
5.4. Application of SysML4*Mechatronics* 81
5.4.1. Development phase 82
5.4.2. Reengineering/Change phase 83
5.5. Summary 87

6. Evaluation of the concept 89
6.1. Evaluation concept 89
6.2. Implementation and case study 90
6.2.1. Proof-of-concept implementation 91
6.2.2. Case study 93
6.3. Usability inspection of the implementation 105
6.3.1. Cognitive walkthrough 106
6.3.2. Results of the cognitive walkthrough and refinement of the implementation 107
6.4. User evaluation 112
6.4.1. Technology Acceptance Model 112
6.4.2. Evaluation assignment 114
6.4.3. Evaluation results 116
6.5. Summary 122

7. Rating of specified requirements and realized results ... 124

8. Summary and Outlook 127
8.1. Summary 127
8.2. Outlook 131
8.3. Summary of publications and research cooperations 132

9. List of references 134

10. List of figures 152

11. List of tables 155

Appendix 156

1. Introduction

The interdisciplinary, model-based engineering of mechatronic production systems[1] is the main research subject of this thesis. The integrated consideration of mechanical, electrical/electronic and software engineering is used thereby for the development as well as for the analysis of changes in mechatronic production systems. After the initial situation and the motivation are described (chapter 1.1), the research objectives are derived and the research methodology is described in chapter 1.2. Completing this chapter, the structure of the thesis is described (chapter 1.3).

1.1. Motivation and problem statement

"Mechatronics is a natural stage in the evolutionary process of modern engineering design" [Bish06, pp.1–1] was proclaimed already over a decade ago. The realization of mechatronic systems and an according mechatronic development process, however, is up to now still no matter of course. While some sectors, e.g., the automotive industry, incorporated sophisticated mechatronic concepts, many companies retain traditional approaches. A study of the German Engineering Federation (VDMA) shows that at present in the machine- and plant manufacturing industry the mechanics still account for 69% of the overall system, as measured by the manufacturing costs [Vdma15]. At the same time, more than every third company (35%) expects a high or very high growth in the degree of digitalization for their systems until 2018 [Vdma15]. Existing sequential development approaches, however, are not sufficient, to manage the development of such interdisciplinary systems.

Mechatronic systems are characterized by the interaction of mechanical/physical system parts, sensors/actuators, and computers or logical systems [Bish06]. Thereby, these different components are not only brought together in one system, rather an important aspect of mechatronic systems is the tight interdisciplinary integration of the different disciplines [Bolt13]. This applies to the system itself, but also to an appropriate interdisciplinary development process. Mechatronic engineering therefore is often referred to as an *integration science* [Ciup15]. This mechatronic paradigm has been applied in the last years also to the development of automated production systems (aPS), due to the growing necessity to provide flexible, variant rich systems and at the same time shorten the engineering process and manage the systems' complexity [LFWZ10]. However, the large amount of sensors and actuators in aPS and their intensive interaction

[1] In order to improve readability, the terms *mechatronic production system* and *mechatronic system* are used synonymously for the description of the developed modeling approach in this thesis.

requires an adequate adaption of existing mechatronic approaches for this application domain [BSBF11].

In order to cope with the complexity of large systems and the multitude of involved stakeholders, Model-Based Engineering[2] (MBE) was established [DiMa13]. The core vision of MBE is the transition from a traditional, document-centric engineering process to a model-centric approach [Inte07]. Model-based approaches originate from software development, especially for industrial software projects. Many modeling approaches, e.g., for mechatronic production systems, have been developed and are commonly used in the software development process [VFST15, Vyat13]. Although software has become the dominant aspect in many mechatronic systems, modeling approaches, which just capture the software perspective, are not sufficient to consider the described interdisciplinary dependencies in mechatronic systems. Furthermore, an early domain specialization often leads to an inadequate consideration of discipline-spanning requirements, a suboptimal design of the overall system, and integration problems [Seli07]. Therefore, interdisciplinary Model-Based (Systems) Engineering approaches have been developed, which consider the overall system and enable the modeling of components/modules on different levels of abstraction. MBE describes the application "of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout the development and later lifecycle phases" [Inte07, p.15]. As MBE can significantly contribute in handling the systems' complexity and reduce the engineering time, various modeling approaches have been developed by different research groups, mainly focusing on the support of the development phase [BKFV14, BSBF11, HeSh12, Thra10] or on automatic code generation [EsMO07, FaSB11, HaVA16, VSFL14, Wits12].

Next to the necessary interdisciplinary development, mechatronic production systems possess the challenge of very long lifecycles of the overall system, which easily can reach up to 50 years [WoSU14]. During this long usage time the system itself as well as the requirements underlie frequent changes, e.g., due to maintenance or due to the implementation of a new functionality based on changed customer requirements [LBKV12, VoHe16]. Usually, a variety of solution possibilities exists. In some cases even a non-optimal solution is chosen to fulfill short term goals (referred to as "technical debt [VFST15]). The different possible solutions, i.e., the components or modules that fulfill the desired functionality, can affect different other parts of the existing system [KBCW14]. Thereby, the compatibility of the changed system part to the existing system has to be ensured or necessary adaptions have to be initiated. As MBE-approaches usually

[2] In literature several similar expressions, e.g., Model-Based Systems Engineering, Model-Driven Engineering/Design, are used. See chapter 2.3 for a discussion of the different terms and meanings.

lack a formal basis, formal models, e.g., using Semantic Web technologies, are needed to implement such compatibility checks [HQRP11]. Formal models on the other hand, are less intuitively and thus, not suitable as modeling interface to the stakeholders from the different involved disciplines [FeKV14]. This thesis focuses on the required information, which has to be captured in an MBE-approach as basis for a compatibility analysis. Through adequate model transformations the MBE-approach and the formal model can be integrated [JeRo12]. Further formal modeling frameworks focus e.g. on the formal specification and verification of models [CaBr18].

The described interaction between the different disciplines in mechatronic production systems as well as the necessity to ensure compatibility in case of a change, require the appropriate information to be modeled within the MBE-approach. Thus, the included properties constitute important prerequisites for the modeling process. Properties describe certain classified characteristics of a system component [Eppl11]. However, the required effort to model all relevant properties manually can be high and is error-prone. The expected high effort also depicts a main reason for the resistance in industry to use MBE [AlZi13]. A promising approach to face this challenge is the integration of properties from standardized classification systems. These are available, e.g., in product classification systems for e-commerce (e.g., eCl@ss [HeLS07]) or in design catalogues, e.g., the NIST Functional Basis for Engineering Design [HSMS02], and describe the semantics of the included properties. As these classification systems were developed for different purposes, the contained properties vary. The integration of properties from standardized classification systems and MBE-approaches was shown, e.g., by [ABDG11, KaEp13, KMWC12]. These approaches, however, do not focus on the properties, which are relevant to analyze the compatibility in case of a change.

The described challenges show that the development of mechatronic production systems requires an interdisciplinary Model-Based approach to manage the complexity, analyze the dependencies between the different disciplines, and compose the overall system. Changes during the lifecycle of mechatronic production systems require, furthermore, that the information, i.e., the properties, captured in the MBE-approach, are adequate to analyze the compatibility of the exchanged part to the existing system. Although existing approaches focus on specific aspects in detail, up to now, no MBE-approach considers all described aspects for the development and change analysis of mechatronic production systems.

1.2. Research objectives and methodology

Given the described current situation in the development and usage of mechatronic systems, this thesis aims at supporting the (re-)engineering process systematically through an interdisciplinary modeling approach.

Therefore, a main research objective lies in the investigation of the dependencies between the components from the different involved disciplines. The definition and modeling of these dependencies and resulting interdisciplinary influences depicts a necessary foundation for the mechatronic engineering. Based thereon, the interdisciplinary modeling approach and analysis of change situations in the usage phase of mechatronic systems can be developed. As described above, MBE offers a promising approach to manage the complexity in large systems. The focus in this thesis thus, lies on the integration of mechanical, electrical/electronic, and software components with their properties and ports in an interdisciplinary modeling approach for mechatronic productions systems. Thereby, an appropriate function-based modularization, which considers the boundaries of the different involved disciplines, is required to support reuse and thus, reduce the development time.

Models depict, according to their definition, an abstraction of reality and include only those aspects which are relevant for a specific stakeholder [Lude03]. Therefore, a main focus within this thesis is the investigation and definition of the information in the model, i.e., the aspects of the components from the different disciplines, that is required for the interdisciplinary development, the definition of new variants or versions of the system, and for the analysis of the system's compatibility in case of a change. Thereby, the modeling approach has to enable the stakeholders from the different disciplines, to add and update their relevant properties of the components and ports in the model comprehensively.

To facilitate the modeling process, this thesis, furthermore, aims at enabling the integration of properties from standardized classification systems into the model. As a variety of classification systems, with different goals, exists, it has to be analyzed, which properties can be integrated from the different classification systems and which properties have to be modeled individually by the developers. Such classification systems mostly focus only on components from specific disciplines or application areas and do not integrate interdisciplinary modules. Thus, this thesis investigates how module properties can be derived from the included components' properties and integrated in the model.

In order to investigate and answer these research objectives systematically, the research methodology shown in Figure 1-1, based on the Design Science Research Methodology [PTRC07], was followed. For each process step, the expected outcomes were defined.

Although the methodology is depicted as sequential process, iterations were carried out if necessary to refine the modeling approach. Furthermore, subsequently to the evaluation, the developed approach can be assessed regarding the fulfillment of the proposed requirements. As a sixth process step of the research methodology (not shown in Figure 1-1), Peffers et al. [PTRC07] suggest the communication of the results, which was conducted through various publications (see list of own publications) as well as through this thesis.

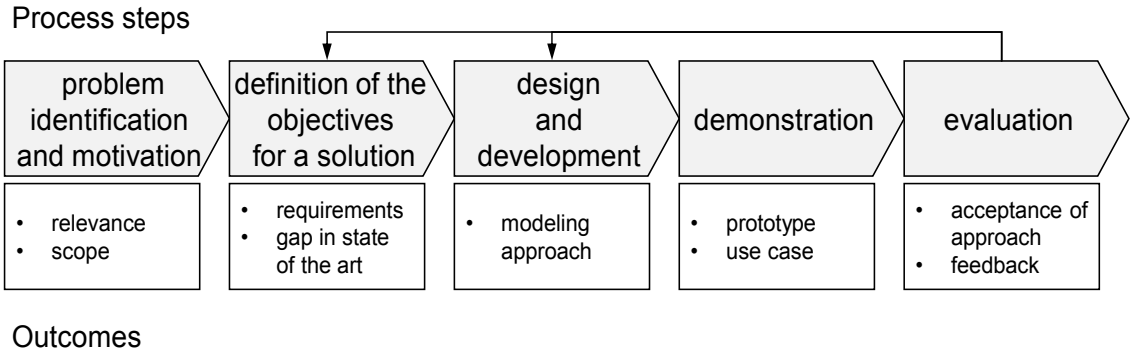


Figure 1-1: Applied research methodology, based on [PTRC07].

Previous works of the Model-Based Engineering group at the Institute of Automation and Information Systems (Technical University of Munich) provide a basis for the development of an interdisciplinary modeling approach for mechatronic production systems and are set in the context of this thesis in the following:

D. Witsch [Wits12] proposed a UML[3]-based development and IEC 61131-3 code generation for PLC software. Therefore, the UML profile plcML was developed. The structural representation of the system in plcML, however, focuses on the software perspective and does not investigate the other involved disciplines. Wannagat [Wann10] developed an agent-based approach for real-time control systems and, additionally, presented a first version of a description language for the application development. Based on the research of Wannagat [Wann10] and Witsch [Wits12], Schütz [Schü15] adapted the modeling language SysML[4] for the development and generation of software-agents for automated production systems. The developed approach, SysML-AT, was then integrated into an industrial engineering environment. SysML-AT enables, equally to plcML, the automatic generation of executable code and debugging directly inside the model. SysML-AT includes a *technical system layer* (including modules and boundary conditions) as well as an *automation system layer* (including sensors and actuators). The approach, however, focuses on the definition of appropriate agents on the described layers and not on an analysis of the dependencies between the components from different

[3] Unified Modeling Language, see chapter 4.1.1
[4] Systems Modeling Language, see chapter 4.1.2

disciplines. The modeling of distributed automation systems and the deployment of the respective automation software were the main focus in the work of Frank [Fran14]. Therefore, the modeling approach SysML-vAT was developed. It contains a *requirements view* (focusing especially on non-functional requirements), a *hardware view* (focusing on automation hardware, which is able to execute software functions, i.e., sensors, actuators and controllers)*, a software view* (focusing on automation functions), and a *deployment view* on the system. Thus, SysML-vAT supports the development process through a model-based approach, which enables engineers to define the required system's functions first and afterwards conduct the software deployment on the available hardware components. The consideration of further system components (e.g., mechanics) and the analysis of the compatibility in case of a change were not in the focus of the approach. As SysML-AT and SysML-vAT as well as the approach developed in this thesis are based on SysML, a combination of the approaches through the extensibility mechanisms of SysML (see chapters 4.1.1 and 5.1 for details) would be possible in a future step.

M. Witsch [Wits13] developed a modeling language, called MES-ML, for the interdisciplinary requirements engineering and functional specification of Manufacturing Execution Systems (MES). The modeling language is a domain-specific language, based on the Business Process Model and Notation (BPMN) [Omgo13], and aims at supporting a more efficient requirements engineering approach in the manufacturing and process industry. BPMN – and thus, MES-ML – can not be coupled with SysML-based approaches directly, due to their different metamodels and notations. However, the developed concepts, e.g., filtering the shown information for specific stakeholders, provide a basis for the development of the modeling editor. A further model-based approach, called modAT4rMS, was developed by Braun [Brau13]. The approach focuses on the development of modular automation software to enhance the reuse in manufacturing systems. It especially takes the aspects usability and software modularity into account. Based thereon, Obermeier [Ober15] examined in a series of usability tests the effects of programming approaches on usability and how they influence software modularity. Li [Li14] determined industrial needs concerning modularization and based thereon, investigated, how the device description language EDDL[5] has to be extended to meet the demands of a modular automation. EDDL-MA enables the combination of different components to *basic, application* and *plant modules* and thus, facilitates the operation and exchange of devices. A Model-Based Engineering process for mechatronic systems, however, was not focus of the research.

As described above, the compatibility analysis in case of a change requires on the one hand, an appropriate modeling approach as interface to the developer in order to provide

[5] Electronic Device Description Language, http://www.eddl.org, retrieved on 23 September 2018

the required properties and information (focus of this thesis). On the other hand, formal models and methods are needed to conduct the necessary compatibility checks and, if possible, resolve potential incompatibilities automatically. In his research, Feldmann investigates these aspects and develops a framework for the diagnosis and resolution of inconsistencies (including incompatibilities) in heterogeneous models for automated production systems. By providing an appropriate model combination (see chapter 5.4.2), an integrated compatibility analysis approach can be provided, offering a sophisticated possibility for engineers to estimate the impact of a change [FeKV14].

The focus of this thesis is based on the work of the author within the Collaborative Research Centre CRC/SFB 768 “Managing cycles in innovation processes – Integrated development of product service systems based on technical products”, funded by the German Research Foundation (DFG). Goal of the Collaborative Research Centre is the management of innovation processes under consideration of affecting cyclic influences in order to support companies in handling the complexity and maintaining their position in the market. The different affecting cycles are interlinked thereby in manifold ways and show mutual dependencies, e.g., within a mechatronic system the components of the different disciplines have varying lifecycles [LBKV12]. Companies face the challenge to act and react successfully along the innovation process under the influence of these dependent cycles, whereat not only the technical perspective is considered, rather the overall socio-technical system is included in the research of the SFB 768. The results of this thesis are based on the research in subproject A6 “Interdisciplinary module management of IT-cycles in innovation processes” of the SFB 768. Subproject A6 investigated during the second funding period (2012 – 2015) especially how the increasing utilization of IT in technical systems influences the dependencies within the system. The developed interdisciplinary modeling approach shall support companies in understanding IT driven innovation processes better and in improving the development and management of mechatronic systems.

Different parts of the research results have been published in several scientific publications. The results were discussed on international conferences with researchers from the same domain (mechatronics, automation) as well as with researchers from related research fields (e.g., manufacturing systems, product development, product lifecycle management), and, furthermore, the developed approach was applied and refined in several joint research cooperations. A list of the publications, which are part of the work presented in this dissertation, is given at the beginning of the thesis and a comprehensive overview of the different investigated aspects is presented in chapter 8.3. Next to the scientific discussion of the research results, the modeling approach was presented on several user-oriented symposia and discussed with experts from industry. Hence, industrial challenges as well as feedback on the presented modeling approach

were captured and used to refine the developed approach continuously. In addition, the development and evaluation of the modeling approach was supported through several student theses. The author of this dissertation guided those theses through definition of the assignment, current supervision and assistance, and discussion of the (intermediate) results of the work.

1.3. Structure of the thesis

The structure of this thesis (see Figure 1-2) is based on the described research methodology. After a description of the general motivation and problem statement, the research objectives and approach are described in chapter 1. In chapter 2 a literature-based overview of the field of research is given. Basic definitions, regarding mechatronic systems, their development processes and Model-Based Engineering, are provided and hence, the scope of research is defined.

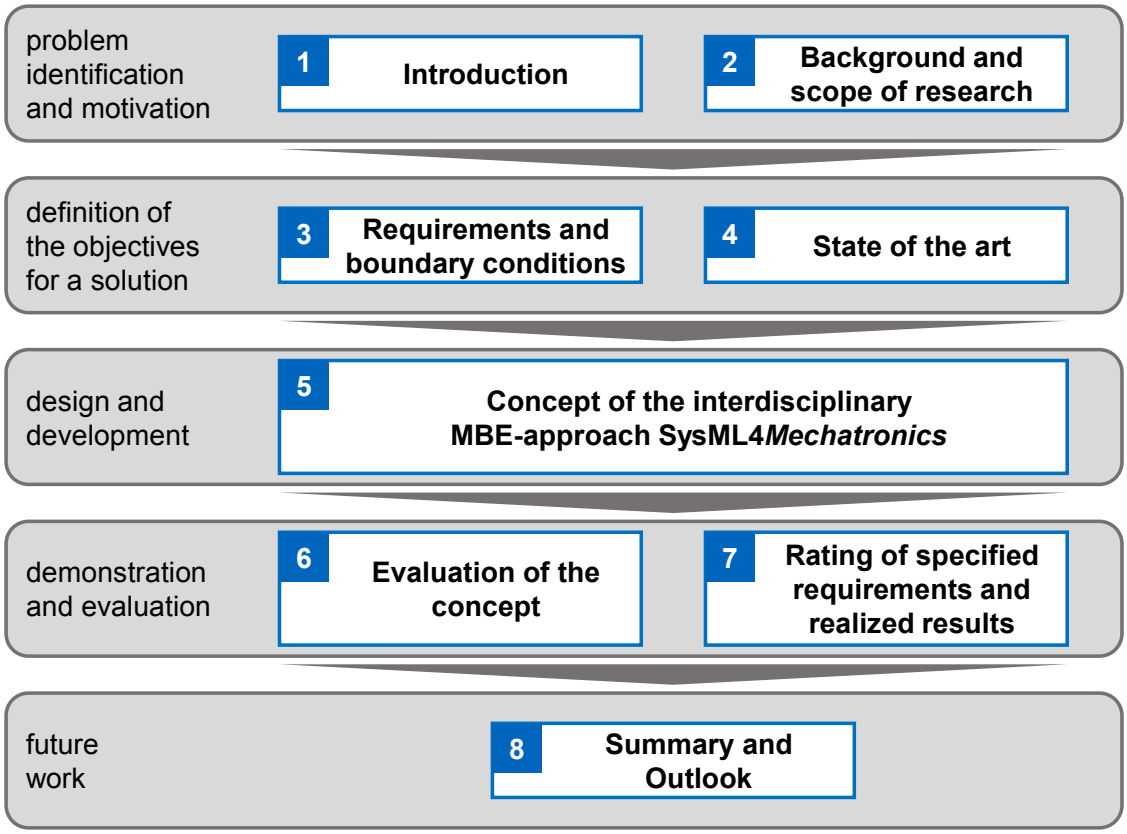


Figure 1-2: Structure of the thesis

In order to consider current industrial challenges for the development of the modeling approach, a survey with industry experts was conducted as basis for the requirements definition (chapter 3). Based on the survey and a literature review the requirements, concerning general modeling requirements, requirements for the interdisciplinary development process, requirements to support the analysis of changes during the

lifecycle, and requirements on usability of the modeling approach are derived (chapter 3). Chapter 4 examines in detail the state of the art regarding modeling approaches, the use of properties in engineering, and existing modularization approaches. The resulting research gap constitutes the starting point for the development of the modeling approach in this thesis. The approach, called SysML4*Mechatronics*, with its metamodel, its concrete syntax and the integration of properties from existing classification systems is described in chapter 5. A proof-of-concept prototype of the approach has been implemented, which serves as basis for the evaluation of the approach by means of a case study as well as under usability aspects (chapter 6). Based on the conducted evaluation, in chapter 7 the developed approach is rated in comparison to the proposed requirements (chapter 3) and their fulfillment is discussed. Concluding, in chapter 8 an outlook on future research and implications for industry are presented. Furthermore, a comprehensive overview of the author's publications and research cooperations, which accounted for the development of the modeling approach in this thesis, is given.

Note that chapters 4.1, 4.2, 5.1, 5.3, 5.4 and 6.2 are derived and extended from an article published in the Journal of Engineering Design [KeFV18] (date of publication: 07.09.2018, copyright Taylor & Francis, available online: https://doi.org/10.1080/09544828.2018.1520205).

2. Background and scope of research

The development of mechatronic systems requires the interaction and collaboration of experts from different disciplines, such as mechanics, electrics/electronics and software. The interdisciplinarity of mechatronic systems offers many advantages, e.g., enhanced functionalities or a reduced system size, which could not be obtained through monodisciplinary developments. However, such an interdisciplinary development and management along the systems' lifecycle leads to an increase of complexity, which has to be handled appropriately [MQAW12]. Model-Based Engineering approaches pose a promising possibility to manage the complexity of large systems and support the interdisciplinary development [BSBF11]. Mechatronic systems thereby exist in a wide range of applications, e.g., in technical (mass) products or technical plants. Depending on the application area, specific requirements arise, e.g., mass products usually have a limited number of actuators and sensors while automated plants possess a high number of sensors/actuators and have harder real time requirements [VDFJ14]. Accordingly, appropriate engineering methods have to be utilized. The focus of this thesis lies on mechatronic systems in the domain of automated production systems. Hence, this chapter introduces basic definitions and an overview of the application area as background for the scope of this thesis. After a definition of mechatronic systems in the machine and plant manufacturing industry in chapter 2.1, existing development approaches for mechatronics are presented in chapter 2.2. In chapter 2.3, Model-Based Engineering, especially in the context of mechatronics, is introduced. chapter 2.4 gives a summary of the background and scope of research.

2.1. Mechatronic systems in machine and plant manufacturing

After the term *Mechatronics* was defined for the first time in 1969 [More69], as combination of mechanics and electronics, it is nowadays understood as the synergetic combination of mechanical, electrical/electronic, and software engineering. According to the guideline 2206 of The Association of German Engineers (VDI) [Vdiv04] mechatronic systems are characterized by a functional or spatial integration of a basic system, actuators, sensors, and information processing. The goal of a mechatronic system thereby is to solve tasks though a combination of the mechanical and the digital-electronic part of the system [Iser96]. Through their sensors, actuators and the according software, mechatronic systems are able to react on changes of the environment, detect critical situations and optimize processes by using, e.g., methods from control theory. Mechatronic systems thus, are able to optimize the behavior of a technical system by

obtaining and utilizing information of the environment and of the system itself through its sensors and reacting accordingly via the actuators [Vdiv04].

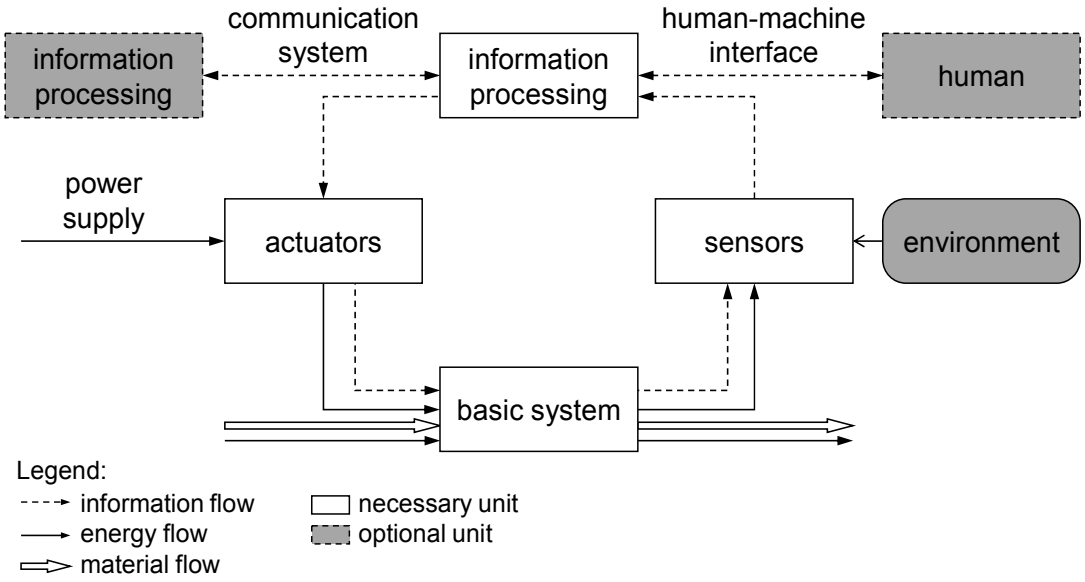


Figure 2-1: Basic structure of a mechatronic system according to VDI 2206 [Vdiv04]

The basic system (see Figure 2-1) thereby usually consists of mechanical, electro-mechanical, hydraulic or pneumatic components or a combination of them. The flows between the components in a mechatronic system can be divided generally in material (e.g., work pieces), energy (e.g., power supply), and information flows (e.g., software variables) [Vdiv04].

In the machine and plant manufacturing industry, aPS are particularly complex mechatronic systems, as they consist of a multitude of hardware, sensors and actuators, and an increasing software part [VDFJ14, VFST15]. Automation thereby refers to the capability of a system to execute a process, under defined boundary conditions, without required intervention of a human being [Dind14]. The degree of automation results from the number of automated functions in proportion to the total number of functions in the system [Dind14]. Next to the technical system, the automated system consists of the technical process, which is performed by the system [LaGö99]. The interrelation between the technical process and technical system, on which the technical process is performed, is shown in Figure 2-2. A process is defined as “complete set of interacting operations in a system” [Dind14], which are responsible for the transformation, transportation, or storage of material, energy or information [Dind14]. Processes can be divided into continuous (e.g., chemical processes), batch (e.g., batch production of medicines), and discrete processes (e.g., transport of work pieces). While continuous and batch processes are applied predominantly in the process technology, the manufacturing technology, which is focused in this thesis, uses mainly discrete processes [LaGö99]. Discrete

processes usually are executed in a sequence and aim at handling, transporting, and joining work pieces [Dind03].

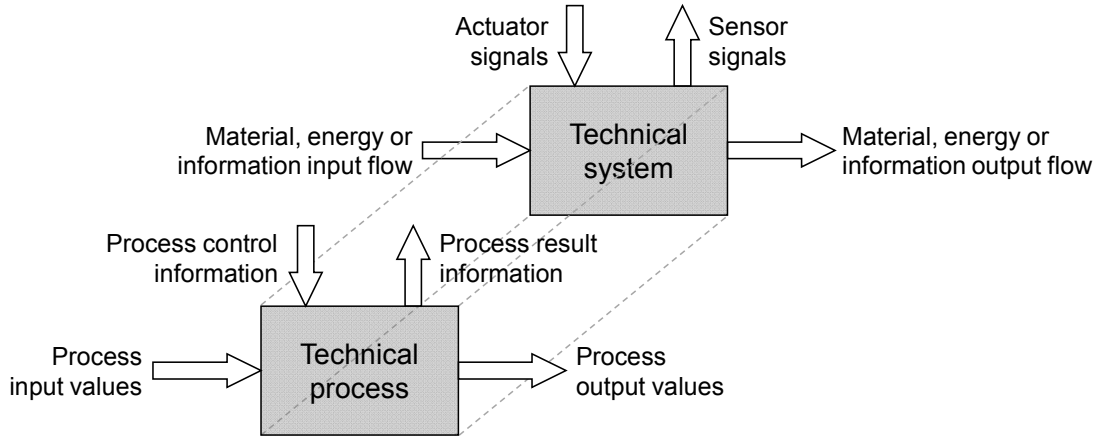


Figure 2-2: Interrelation between technical process and technical system according to [LaGö99]

While mechatronic systems in mass products often are controlled through microcontrollers (due to their cheap price and small dimensions), mechatronic production systems are controlled by automation-computer systems. The most commonly used systems thereby are Programmable Logic Controllers (PLC), industrial PCs (IPC), and Distributed Control Systems (DCS) [LaGö99, p.31]. Primary characteristics of these systems are an operation system which ensures real-time control and a cyclic program execution [Vyat13]. The software of these systems is programmed in most cases in the PLC programming languages according to the IEC 61131-3 standard [Ieci14a]. The world-wide accepted industry standard facilitates, furthermore, the migration of the software from one PLC to another PLC-type [Vyat13]. IEC 61131-3 consists of the textual languages Structured Text (ST) and Instruction List (IL, deprecated in the current version of the standard [Ieci14a, p.215]), and the graphical languages Function Block Diagram (FBD), Ladder Diagram (LD), and Sequential Function Chart (SFC). Next to IEC 61131-3 the standard IEC 61499 [Ieci12], especially for distributed systems, was developed in the last years [BaCG13, Vyat11]. IEC 61131-3 as well as IEC 61499 support modularity and reusability, which facilitates Model-Based Engineering [BoFS12]. Since the third edition of the IEC 61131-3, Object-Oriented Programming paradigms, e.g., by means of inheritance, were added to the standard [Wern09].

Next to the described common controllers, also more and more ‘intelligent’ actuators and sensors are equipped with a control unit (in general a microcontroller) and are able to execute certain tasks in the system, e.g., archive sensor values [FaBS09].

2.2. Development processes of mechatronic systems

The development of mechatronic systems and specifically mechatronic production systems requires appropriate processes, as traditional development approaches are not sufficient to deal with the complexity and interdisciplinarity of such systems [BSFB06]. An early, but frequently used, guideline for the development of mechatronic systems is provided by the guideline VDI 2422 [Vdiv94]. It was developed due to the rising degree of microelectronics in mechanical systems [Vdiv04]. As Figure 2-3 shows, the development process is divided explicitly into the disciplines mechanics, electrics and software.

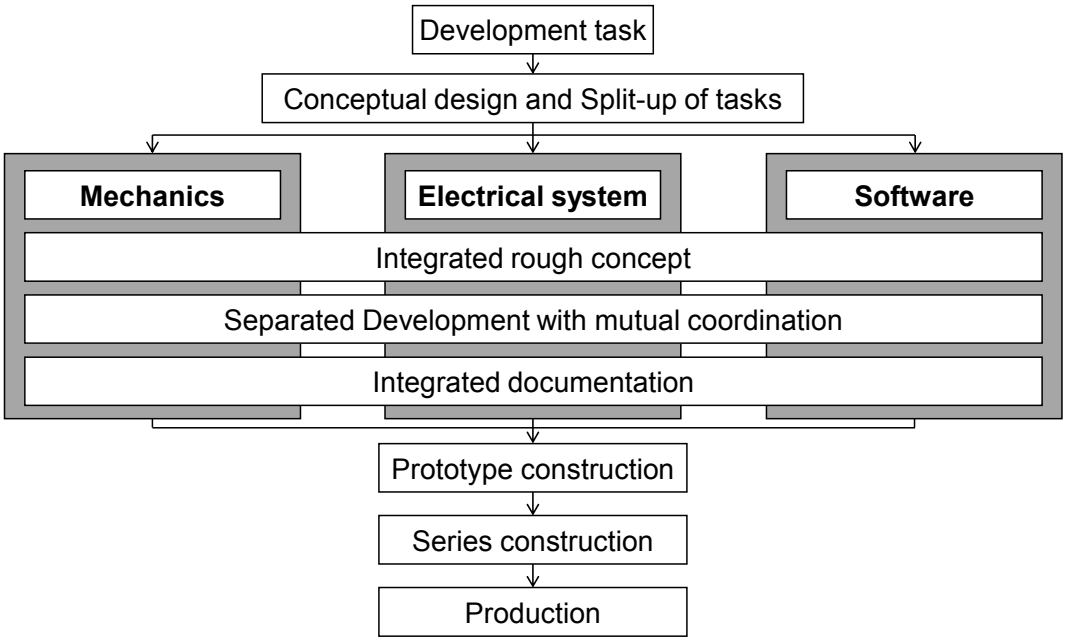


Figure 2-3: Systematic mechatronic development process based on VDI 2422 [Vdiv94]

Starting from a mechatronic development task, the concept is defined and then split up into discipline-specific development tasks. The integration is realized through mutual coordination and an integrated documentation of the project [Vdiv94]. The different steps of the process, however, are not defined in detail and the solutions are described informally in natural language in the guideline (e.g., define power supply and interfaces [Vdiv94, p.15]). The process according to VDI 2422, however, results in typical gaps between the different design phases (Figure 2-4) and between the separated discipline-specific tasks [AFTW10]. Despite the mutual coordination, the discipline-specific teams work separately and often are not located in the same place. Thus, the integration of the solutions is postponed to the prototyping phase [AFTW10].

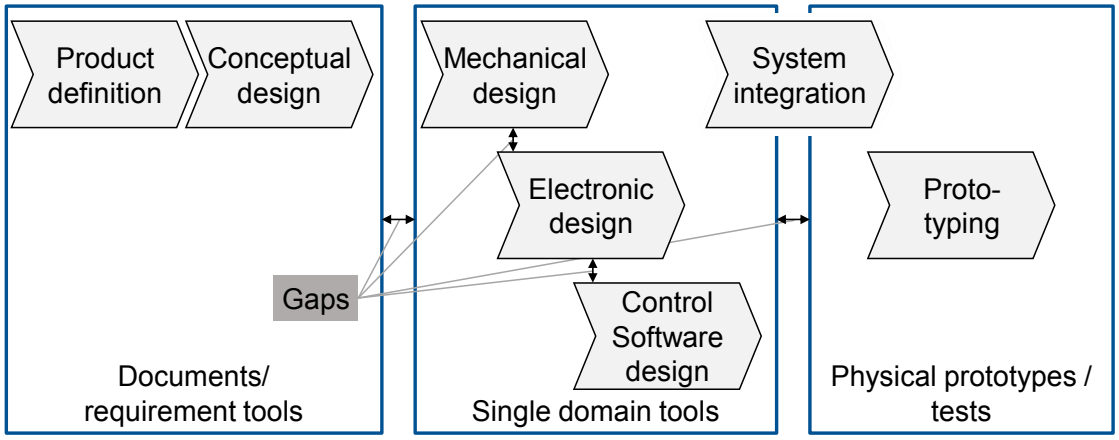


Figure 2-4: Gaps in the interdisciplinary development according to [AFTW10]

To overcome these disadvantages, the process was extended and adapted in the guideline VDI 2206 "Design methodology for mechatronic systems" [Vdiv04]. The guideline supports an interdisciplinary development of mechatronics in the early design phases. The macro-cycle of the development process is described in form of the V-model, which is depicted in Figure 2-5 [Vdiv04].

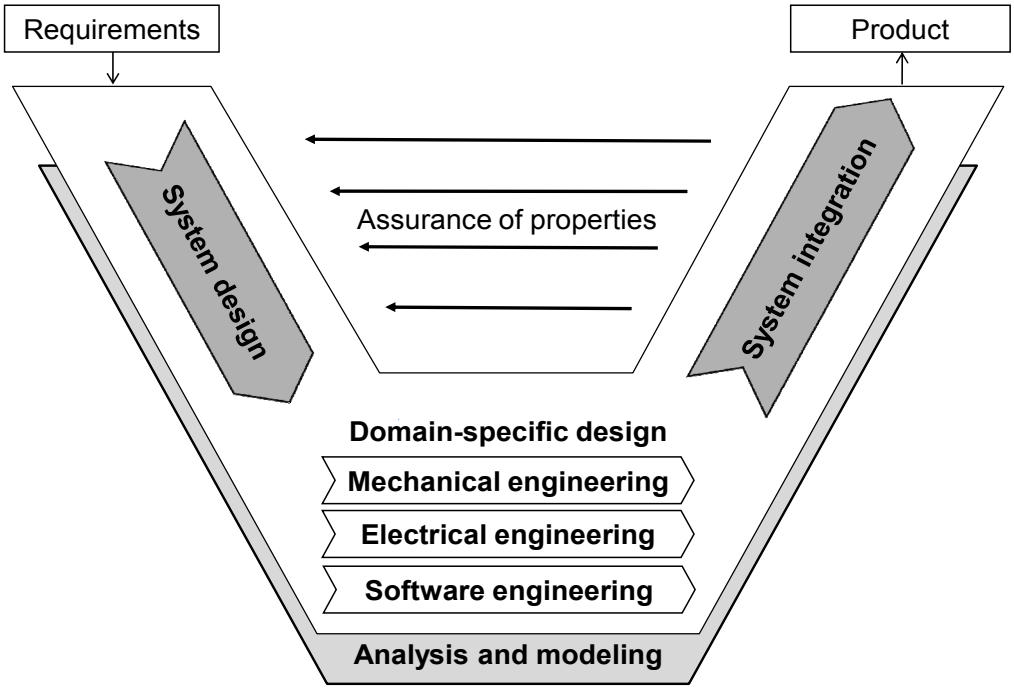


Figure 2-5: V-model according to VDI 2206 [Vdiv04]

Usually, a specific customer order constitutes the starting point for the development. Based on the defined requirements, a solution concept for the system is developed. The concrete design of the different components is then carried out in the single disciplines and integrated to the overall system. The design progress has to be checked continuously against the requirements, to ensure that the system properties fulfill the specification. If

necessary, simulations are carried out to analyze the system properties. At the end of the process, the V-model results in the intended product or system [Vdiv04].

The general development process steps of the VDI guideline 2206 have to be adapted for the domain of mechatronic production systems, as they are usually developed customer-specifically [VoHe16]. In order to handle the complexity and reduce the development time and costs, the development requires the engineers to reuse solutions of parts of the system [VFST15]. Therefore, the required tasks are divided in project-independent and in project-related tasks [Vdiv10] (see Figure 2-6, [VFST15]). The project-independent tasks (upper part in Figure 2-6) usually are developed according to mechatronic development processes (e.g., the described VDI 2206 [Vdiv04]) and are carried out, e.g., by system suppliers. These development tasks are decoupled from a specific development project of a mechatronic production system and the resulting solutions are stored in a solution repository for reuse. The project-related tasks for the development of a specific system are carried out in form of projects [Vdiv10] (lower part in Figure 2-6), which last between a few weeks and several months, depending on the complexity of the project [VFST15]. In order to reduce the development time, the project-independent solutions are reused within the project-related activities, especially during the phases of *system design* and *detail design*.

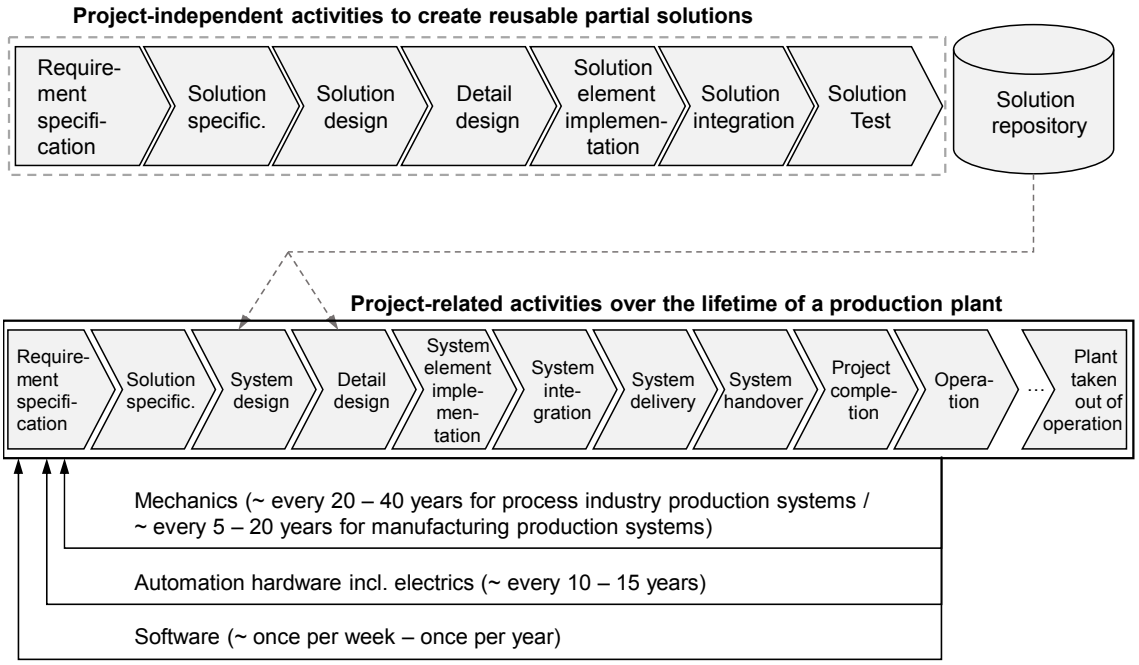


Figure 2-6: Project-independent tasks and project-related tasks in the lifecycle of aPS according to [VFST15]

During the operation phase of the system, which can reach up to 50 years [WoSU14], frequently changes of components or modules have to be conducted. Typical reasons therefore are changed requirements, e.g., resulting from the market, new laws or changed

customer requirements [LBKV12]. Such changes can often be solved through software-changes and may be even conducted during runtime of the system [VFST15]. Additionally, “forced changes” [LBKV12] have to be implemented, if a component has to be exchanged (e.g., due to wear and tear), which is not available anymore on the market or which shall be replaced by a newer one (e.g., a more energy-efficient component). This process is referred to as reengineering or modernization [VFST15]. According to the different lifecycles of the components from the different disciplines, the frequency of the described changes, within the overall mechatronic system, varies (see arrows for mechanics, automation hardware and software in Figure 2-6).

2.3. Model-Based Engineering

The usage of models can support the development process (cf. chapter 2.2) in terms of structuring complex systems, capturing their properties and dependencies in order to define and understand the structure and behavior, and conducting analyzes (e.g., simulations) [EGGM12]. Models in general are defined, according to [Stac73], as simplified representations of an original, which fulfill the *mapping criterion*, the *reduction criterion*, and the *pragmatic criterion* [Lude03]. The mapping criterion defines that the model represents an original, which can be a physical object, a phenomenon or an idea, e.g., for the development of a system. The reduction criterion indicates that not all properties of the original are represented in the model. However, at least some properties have to be included and on the other hand, also properties, which are not part of the original, can be included in a model, e.g., a coordinate system. According to the pragmatic criterion, the model has to be useful to for a specific (group of) stakeholder(s), i.e., it has to be developed for a specific purpose [Lude03].

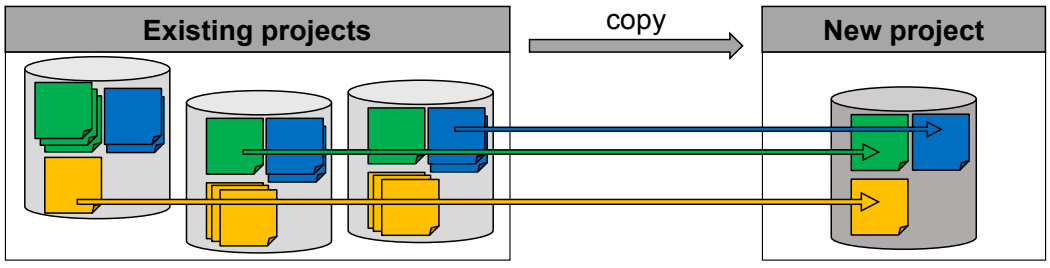


Figure 2-7: Reuse according to [Ange03]

For the engineering process in the machine and plant manufacturing industry, models have been used already for a long time [FrMS15], e.g., in terms of CAD-models or wiring diagrams [KHJM99]. However, this usage of models is limited to specific (discipline-specific) tasks. When these models shall be reused for a new project, they are usually copied individually from different existing projects (Figure 2-7, [Ange03]). This implies, however, disadvantages, e.g., in terms of quality, additional costs due to a lacking cross-

project coordination, or missing standardizations [Ange03]. Therefore, engineering approaches have been developed, which consider models not only as documentation artefacts [Rodr15], but focus on a model-centered process [Inte15]. Such approaches arose in different domains, e.g., product or software development, and as Systems Engineering approach. Accordingly, different terms can be found in literature:

- Model-Driven Design/Development – MDD [BuGT05, ESPO09, Seli03]
- Model-Driven (Software) Engineering – MDE [ObBV15, Rodr15, VSFL14]
- Model-Based (Systems) Engineering – MB(S)E [AlZi13, BSBF11, CaLP11]

Model-Driven approaches focus more on automating the development process through models [ScLV14], e.g., by means of code generation from models [VoWK05] or through model transformations. Model-Based approaches focus more on supporting the development through a transition from a document-centric to a model-centric approach [Inte07], e.g., by means of offering an interdisciplinary representation of the system under construction. However, a distinct separation of the terms is not always possible and depends, furthermore, on the discipline in which it is used. For the developed approach in this thesis the term Model-Based Engineering (MBE) is used. The MBE-approach thereby, is embedded in the overall development process. Hence, as the majority of information is captured in the model, MBE depicts the core of the development process (see Figure 2-8, [EGGM12]). Mechatronic production systems depict one domain of application for MBE and thus, MBE is applied for mechatronic engineering. Additionally, further systems engineering methods (e.g., management procedures) can support the overall development process for complex systems [EGGM12].

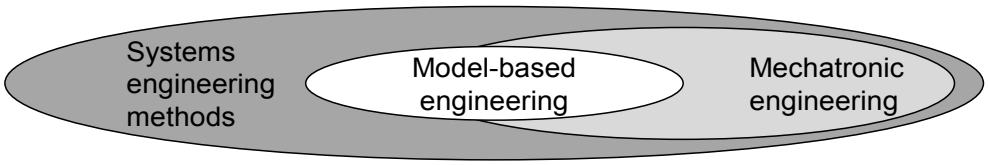


Figure 2-8: Classification of Model-Based engineering according to [EGGM12, p.42]

The application of a MBE-approach requires, as every modeling approach, a certain modeling process and an appropriate modeling language. The modeling process defines the necessary steps to develop the model and is represented, e.g., in a process model (a process model, which uses the modeling approach of this thesis was presented in [BKFV14]). The composition of a modeling language and its relations within the modeling domain are introduced in the following:

- Each model, as described above, represents a certain view of an original (e.g., a mechatronic system). The information, which shall be represented, can be modeled in many different ways, e.g., textually or graphically [Vdiv05]. Thus, a

model is described by using a specific modeling language and hence, conforms to the according metamodel. The metamodel represents (parts of[6]) the modeling language and is a model itself. Thus, the metamodel again is conform to a meta-metamodel (see also Figure 4-5 in chapter 4.1.2).

- The concrete syntax (also defined as notation) of a modeling language describes its visual or textual representation and thus, defines which symbols are used to represent the language, e.g., in a diagram [Mood09].
- The abstract syntax (also defined as schema) is independent from the particular representation of the modeling language and describes the elements that can be represented by the language as well as the rules, how the elements can be connected. The concrete syntax is mapped to the concepts of the abstract syntax [CeGR09, Scha13].
- The semantics define the meaning of the language, which represents a specific *semantic domain*. Through a *semantic mapping* the abstract syntax is related to the semantic domain [CeGR09]. For a formal language, the definition of formal semantics (e.g., in terms of model theory [DiMa13]) is required.
- In order to create a specific model, usually a modeling tool is used [Scha13]. Moody [Mood09] indicates that the modeling tool should be realized by a fitting representation medium, e.g., a computer-based modeling editor.

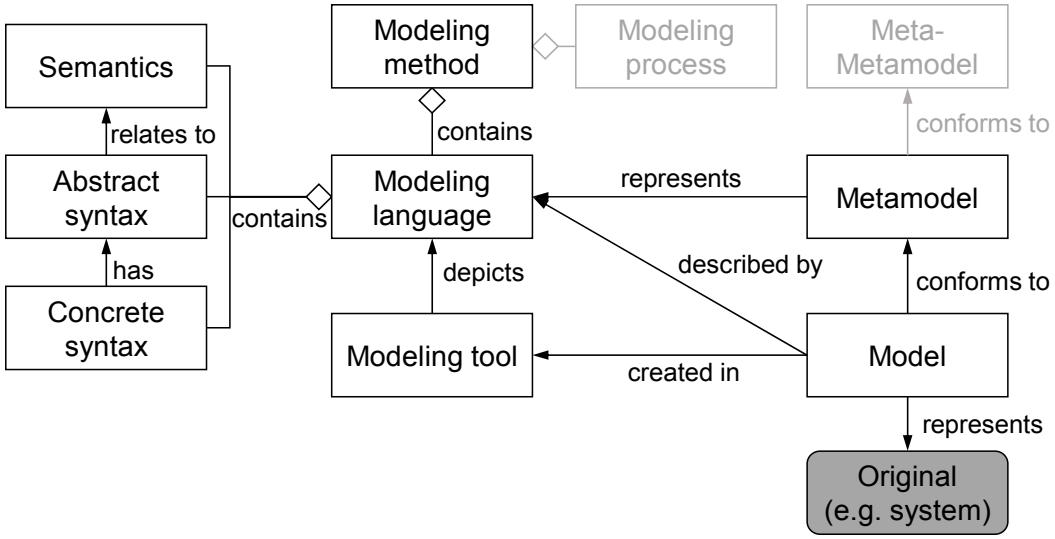


Figure 2-9: Modeling domain, based on [Fran14, Mood09, Omgo15a, Rodr15, Scha13] *(parts not focused in this thesis are shown grayed out)*

[6] Note that some definitions of a metamodel refer only to the abstract syntax, e.g., [Omgo15a]

2.4. Summary

In this chapter, an introduction to the scope of research and general definitions, regarding mechatronic systems, their development process and Model-Based Engineering, were presented. The focus of the developed approach in this thesis lies on mechatronic systems in the production systems domain, with a multitude of interrelated mechanical, electrical/electronic, and software components. Due to the usually customer-specific design, the development of such systems is specifically complex and has to be supported by an appropriate development process. A widely used approach in industry [FVFE15] is the V-model [Vdiv04]. Model-Based Engineering approaches help to represent the information in a structured, comprehensible way and thus, reduce the complexity. The development of an appropriate approach has to consider various aspects within the modeling domain (cf. Figure 2-9). This thesis focuses on the definition of an appropriate modeling language with a respective metamodel (chapter 5.1) and on a prototypical implementation of an associated modeling tool (chapter 6.2) for mechatronic production systems.

3. Requirements and boundary conditions

The overall goal of this thesis is to support engineers during (re-)engineering of mechatronic production systems through an interdisciplinary, model-based approach. As described in chapter 1, the model-based approach thereby shall be applicable in the development phase as well as in the reengineering phase (see description of phases in chapter 2.2, [Vdiv04, VFST15]). Therefore, on the one hand, the interdisciplinary development has to be supported and on the other hand, the adequate properties to analyze the compatibility in case of a change have to be captured in the model.

In this chapter, requirements and considered boundary conditions to develop such an interdisciplinary modeling approach are specified. Therefore, in a first step relevant challenges from an industry perspective are identified through a questionnaire (chapter 3.1). Based on these challenges from industry and a literature review, the requirements, which have to be fulfilled by the modeling approach, are derived in chapter 3.2. As an interdisciplinary development and innovation process comprises various aspects, which can not all be considered within the scope of this thesis, the assumed boundary conditions are described in chapter 3.3. A summary is given in chapter 3.4.

3.1. Identification of challenges in industry

As a first step, to identify the industrial needs regarding an interdisciplinary model-based development approach, a survey with industry experts was conducted. The goal of this survey was to get further insights, which modeling languages are already well known in industry, how interdisciplinary development and modularization is handled, and how change impacts are analyzed. Thus, the survey provides an important basis for the identification of the requirements for the model-based approach developed in this thesis.

Next to the survey in this thesis, the results of other relevant surveys in literature were analyzed and considered accordingly. An extensive survey, which provides a detailed insight into the development processes and modularization approaches of companies in the machine and plant manufacturing industry, was conducted by Vogel-Heuser et al. [VFFU17, VoOc18]. In this *modularity survey*, 16 German world-leading companies assessed their level of maturity concerning their development processes, software and reuse, quality assurance and the transfer to the customer. After answering the questionnaire, the companies can compare their individual result with the average values of the industry sector and thus, use it as a benchmark. The results of this questionnaire,

concerning the interdisciplinary development, were incorporated during the development of the approach in this thesis as well.

3.1.1. Data collection method

For the survey, data was collected through a standardized written questionnaire, which was conducted in February 2014 during the *Automation Symposium*. The *Automation Symposium* is an expert conference of the Institute of Automation and Information Systems at the Technical University of Munich, which brings together leading industry representatives to discuss current topics in the scope of automation in the machine and plant manufacturing industry.

The questionnaire comprised 16 questions covering the main topics of interest for the study. Some of the questions permitted multiple answers as well as the possibility to add comments. At the beginning of the survey, all participants were informed about the purpose of the questionnaire and were guaranteed anonymity to enhance trust and foster unbiased answers [Flic09]. The survey was conducted in German. The key questions and answers have been translated to English in the following for this thesis. The original questionnaire can be found in the appendix.

3.1.2. Participants, results and analysis of the survey

The goal of this survey is a practical assessment of the status quo in industry regarding the awareness level of modeling languages, the degree of interdisciplinary teamwork in the development and the current practices in the analysis of change impacts. Thus, the questions form an important basis for the deduction of relevant requirements for an adequate modeling approach for mechatronic systems.

Table 1: Structure of survey participants

Position	# of participants
Management position	9
Developer (mechatronic)	2
Developer (software)	8
Total	19

A total of n=19 questionnaires were filled in by the participants. Due to the setting of the questionnaire at the Automation Symposium, from some companies several employees answered the questionnaire. As the participants filled out the questionnaire anonymously, the specific company affiliation was not collected. The survey participants included software developers (especially for PLC software), mechatronic developers, product and

innovation managers, business segment leaders, as well as managing directors (Table 1). In the following, the main findings are presented and discussed with regard to their consequences for the development of an interdisciplinary model-based approach.

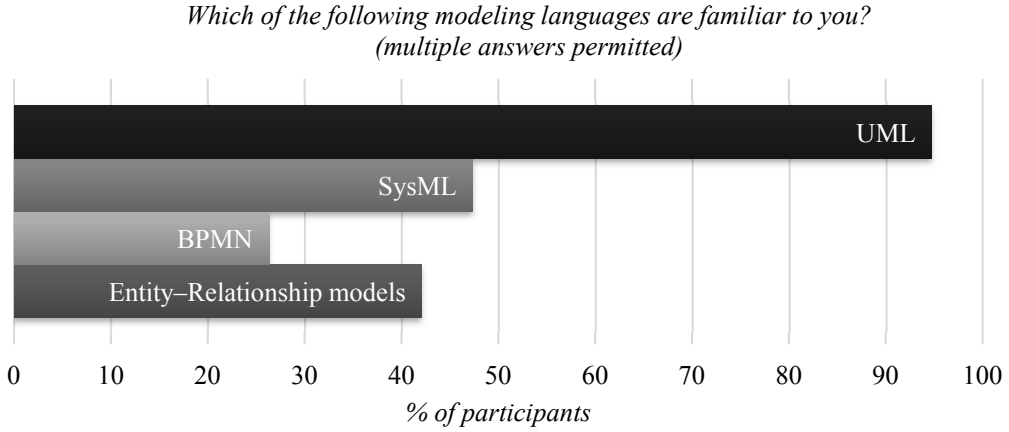


Figure 3-1: Survey results regarding modeling languages

Regarding the familiarity of existing standardized modeling languages, the questionnaire showed clearly that the Unified Modeling Language (UML, see chapter 4.1.1) is the most known modeling language among the participants (familiar to 94.74%, Figure 3-1). Thereby, all participating software developers (eight participants) stated that they are familiar with UML. The Systems Modeling Language (SysML, see chapter 4.1.2), which depicts an interdisciplinary language for the systems development and is based on UML, however, is only familiar to 47.37% of the participants. The more specific Business Process Model and Notation (BPMN) [Omgo13] and entity-relationship models (used in database design) [Chen76] were the least familiar languages in the survey and are only known to their specific stakeholders. As an open comment to this question, the Event-driven Process Chain (EPC) [Sche94] was mentioned, which also poses a specific approach for modeling service sequences and business processes [Sche94]. Thus, for the development of an interdisciplinary modeling approach, a general awareness of SysML can not be assumed. However, UML, as basis for SysML, has become nowadays the de-facto standard in software development [BrCe11] and is not only familiar to software developers, as the survey has shown.

The results of the survey show that interdisciplinary collaboration is a crucial point in industry (Figure 3-2). Although also specialized software developers were among the participants (cf. Table 1), no participant works solely in discipline-specific teams. In contrast, nearly 60% work *often* or *very often* in interdisciplinary teams. The development of mechatronic systems, which is focused in this dissertation, requires the tight collaboration of various disciplines in the development process, e.g., through mechanical, electrical and software engineers. In a modularity survey of Vogel-Heuser et al. [VoOc18], detailed information regarding the frequency of team coordination during the

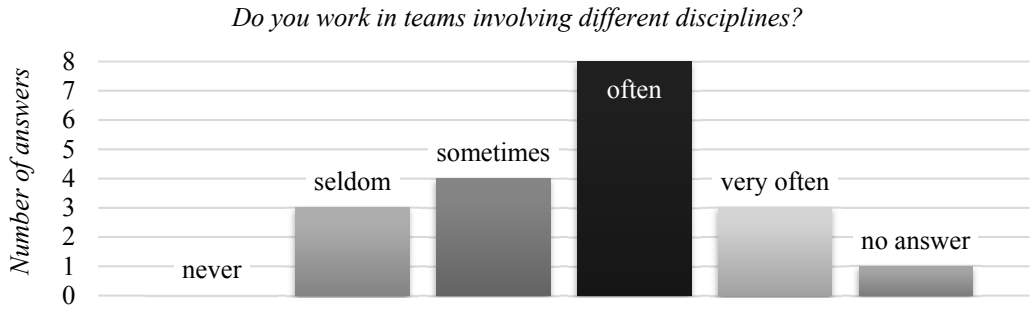


Figure 3-2: Answers regarding the interdisciplinarity of teams

development phase was assed. Thereby, 52% of the machine manufacturing companies stated that they exchange information on a weekly basis and 15% every two weeks. However, 21% answered that an information exchange only happens sporadically, which can be very critical for the achievement of the project goal. The most frequently mentioned disciplines for information exchange are electrics/electronics, software, mechanical engineering, and automation. The relevance of interdisciplinary development, especially regarding the variant and version management, was also shown in the research project BESTVOR [Best09]. In a survey in the context of the project, 52% of the 60 interviewed companies stated that they did not have an interdisciplinary software tool to manage their variants and versions [LBKV12].

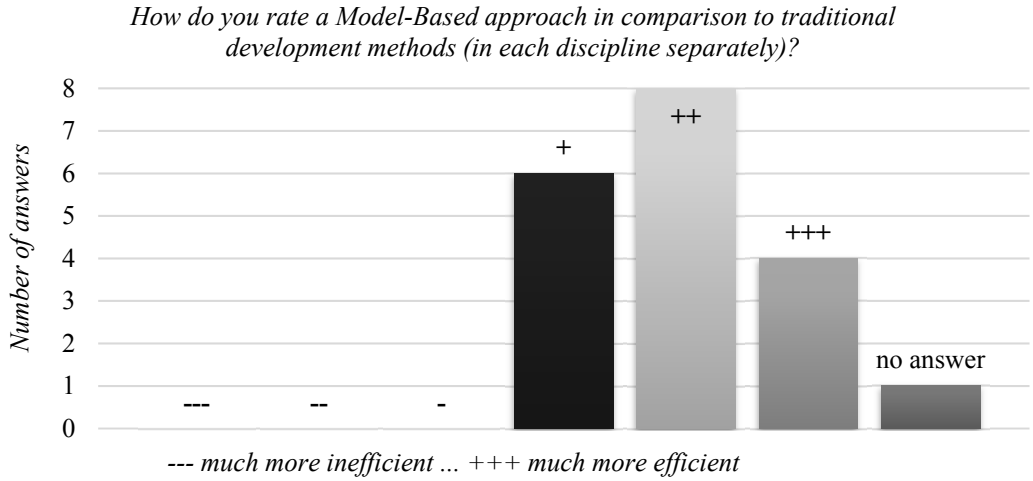


Figure 3-3: Survey results regarding the expected efficiency of Model-Based approaches

Based on their experience, the participants rated their expected benefit of a Model-Based approach, i.e., a model-centric development process, in comparison to traditional development methods, i.e., a development in each discipline separately (Figure 3-3). All 18 participants, who answered the question, expect a positive influence of a Model-Based approach regarding efficiency of the development process. On a scale from 1 (much more inefficient) to 6 (much more efficient) the mean value of the given answers lies at 4.89

(with a standard deviation of $\sigma = 0.74$.). Thus, although existing modeling languages for model-based engineering, like SysML, are not known to all participants (cf. Figure 3-1), Model-Based approaches pose a high potential from the industry experts' point of view. Motamedian [Mota13] conducted a survey regarding the actual usage of Model-Based Systems Engineering approaches, in which only 38% of 55 participants from various industries (e.g., aviation, automotive, IT) stated to currently use them. Thereby, 42.9% of the users of MBSE have less than 2 years of practical experience with MBSE.

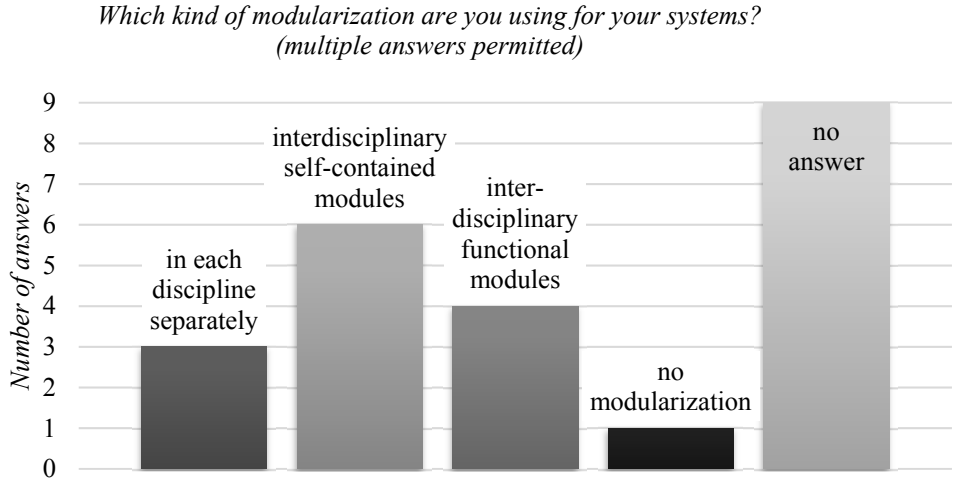


Figure 3-4: Survey results regarding modularity

Another important aspect in mechatronic system design is modularization [EbGö04, GePZ03]. According to the VDI guideline 2206 [Vdiv04] modularization is defined as "system elements or components which are combined to form a group and together perform a specific function." [Vdiv04, p.16]. The participants of the survey were asked, which kind of modularization they use for the systems they are developing (Figure 3-4)

- in each discipline separately, i.e., each discipline decides independently, if and how a modularization is conducted.
- interdisciplinary self-contained modules, i.e., the module has the same borders in all involved disciplines. The overall system then can be generated through composition of the desired modules
- interdisciplinary functional modules, i.e., 'virtual' modules are used during the development of the system. Thereby all components, relevant for a specific functionality, are grouped to a module. During implementation, the optimal structure for each discipline is determined.
- no modularization

Note that multiple answers were permitted as not all companies permanently use the same modularization in every project (three participants indicated that they use several different modularization approaches) and that from some companies several participants answered

the questionnaire. The wide variation of the answers shows that the kind of modularization is very company-/system-dependent. Thereby, in 21.43% of the cases no mechatronic modularization is used (answers: *in each discipline separately* and *no modularization*). The detailed modularity survey [VoOc18] arrives at similar results, i.e., 30% of the respondents do not incorporate a mechatronic modularization. If a modularization is conducted, the most commonly considered disciplines are, according to the modularity survey, software, electrics/electronics, and mechanical engineering. Hydraulics/pneumatics and safety engineering in contrast are considered seldom. In the survey of the BESTVOR-project, the participants from mechanical engineering assumed that they could manage the interdisciplinary, mechatronic engineering processes well, while a substantial need for improvement was claimed by the software engineers [Best09].

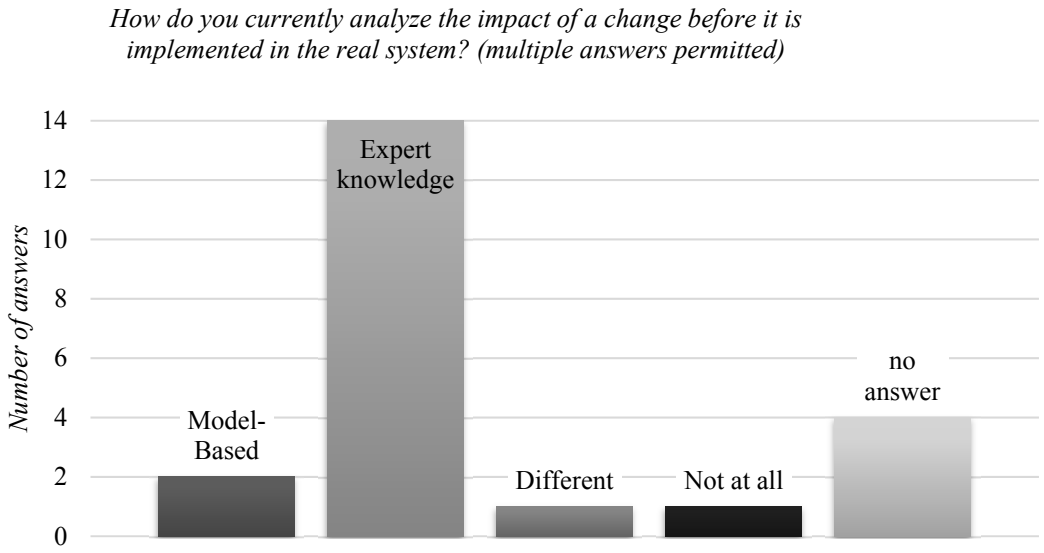


Figure 3-5: Answers regarding the analysis of change impacts

The last topic of interest deals with changes in the system. The necessity for changes can occur e.g., due to new customer requirements or required reengineering during the usage phase of the system. In a first question to this topic, 84.2% of the participants indicated that, during the system's usage phase, changes are conducted, which shows the high relevance of changes in industry. In a more detailed question (Figure 3-5), the participants should indicate how the impact of a change is analyzed, before it is implemented in the real system. The vast majority (73.7%) stated that their analysis to estimate the impact of changes is based on expert knowledge. A Model-Based analysis on the other hand is only used by a very small part of the participants (1.1%). One of the participants, who stated that *expert knowledge* is currently used, commented that this form of analysis, however, is not sufficient. Furthermore, the participant who answered that a *different* kind of analysis is used stated that they rely on their 'gut feelings'.

A further survey [KBHR15] shows that change management in production is an important interdisciplinary topic with very heterogeneous processes. The participants stated thereby, that an adequate IT support would be very beneficial for analysis and adaption tasks. In product development [Lang16, LWML12] change management plays a decisive role, as technical changes account for a quarter of the entire research and development efforts in a company [LWML12]. Although a third of these changes is rated as critical, the majority of companies only uses direct formal or informal communication to perform the required changes. A model-based or IT support to analyze and handle changes is used considerably less [LWML12].

Summing up, the following conclusions regarding the status quo in industry can be drawn from the conducted and analyzed surveys:

- The awareness level of currently existing modeling languages varies. While the standardized language UML is well known, the interdisciplinary SysML still needs to catch up, as less than half of the participants is currently familiar with the modeling language.
- Collaboration in interdisciplinary teams is currently required in nearly any industrial project. The frequency of information exchange, however, varies strongly among the companies. For an efficient development process, a ‘sporadic’ exchange of information can be very critical, as the existing (implicit) dependencies between the different involved disciplines can lead to undesired hold-ups, if they are not considered adequately.
- Modularization is handled very diverse by the different companies. Mechatronic modules (self-contained modules or functional modules) as well as a discipline-specific modularization is used in practice. Mainly, software, electrics/electronic and mechanics are considered for modularization. Other disciplines, e.g., hydraulics/pneumatics are included only rarely.
- Changes during the usage phase of the system have to be conducted by nearly every company. In order to avoid unexpected costs and prolonged downtimes through a change, an analysis of the resulting change impacts are crucially. Thereby, expert knowledge depicts the main source in most companies. Model-Based approaches to facilitate the estimation of change impacts are not widely spread among industry.

The results from this survey show how companies currently deal with model-based approaches, modularization and change analysis. Thus, this survey forms the basis for the development of a model-based approach in this thesis. Based thereon, in the following chapter concrete requirements are defined.

3.2. Requirements for the modeling approach

Based on the analyzed status-quo in industry and a literature review, the requirements for the interdisciplinary modeling approach SysML4*Mechatronics*, which shall be developed in this thesis, are defined in this chapter.

A model, independently if computer-based or any other kind of model, is the representation of an original object, phenomenon, or idea. It is made for a certain group of stakeholders with a specific intention at a determined point in time [Lude03, Stac73]. Thereby, each model is always a reduction of the original, i.e., not all properties of the original are included in the model [Lude03]. Depending on the intended use of models, different forms of modeling approaches can be used, i.e., a textual, graphical, or mathematical notation [Vdiv05]. The modeling approach, developed in this thesis, shall act as an interface to the engineers to model the different components, their properties, and their interrelations in mechatronic systems. Consequently, a human-readable modeling language is needed, which enables the adequate information input by the user. A graphical modeling language offers the possibility to capture the relevant information in a comprehensive manner through graphical elements representing the different modeling elements [HMSL07].

The potential benefits of a modeling approach always have to be seen alongside with the costs of introduction, not only financially but also regarding the required effort and time, in order to estimate if it delivers a benefit over the lifecycle. This is especially relevant for an industrial realization and can determine the potential success of the approach. Companies often state that a high learning effort to use Model-Based approaches is a significant entry barrier [AlZi13]. The reuse or adaption of well-known modeling languages offers a convenient possibility to reduce such barriers, as it saves potential users the time to learn a completely new modeling languages and its specific notation [FePi04]. Furthermore, using a standardized modeling language as basis has the advantage that it is interoperable with other existing models [CaLP11]. UML and SysML, which were defined as multipurpose languages, offer extension mechanisms to define own approaches for a specific intention [FrMS15]. The “light-weight” [CaLP11] method to extend SysML is the definition of a profile, in which specific stereotypes can be defined on the basis of the existing constructs of SysML. In this way, the integration in existing approaches is facilitated. Thus, the definition of the modeling approach for mechatronic systems shall be realized accordingly:

R 1:	***SysML4Mechatronics shall be defined as extension of SysML, using the provided profile mechanism***

The development of mechatronic systems requires profound knowledge from mechanical, electrical/electronic, and software engineering. Thereby, the tight integration of these different involved disciplines is especially important [BSBF11, Hehe14]. Through a traditional, separated development in each discipline, the optimal solution for the overall system, in order to meet the customer requirements, can hardly be achieved [GFDK09]. Gausemeier et al. [GaDK10] state that a profound, domain-spanning understanding of the system by all engineers from the beginning of the development process is essential. However, existing mechatronic design methodologies are often not suitable for the machine and plant manufacturing industry [BSBF11]. Missing adequate modeling approaches hinder the interdisciplinary communication and development of complex mechatronic systems [Hehe14]. Thus, one of the main goals of SysML4*Mechatronics* is the integration of the different disciplines in the development of complex, mechatronic production systems.

To represent the required aspects, in general two approaches can be pursued:

a) Coupling of different modeling languages
b) Interdisciplinary modeling language

The first alternative allows a separation of concerns [Rodr15], i.e., each user can work with his own, specific model. However, it has the disadvantage that transformation methods between the different coupled sub-models are required to receive a consistent overall model of the system [Schü15]. Interdisciplinary modeling languages integrate the aspects from different stakeholders and thus, foster their cooperation, as the modeling language can be understood easily by each discipline, reduce faults and increase the efficiency to exchange information [Wits13]. In order to make the different aspects, i.e., the components from the different disciplines, easily distinguishable in the modeling approach, visual variables [Mood09] (e.g., color, texture, brightness) should be used. Note that, in addition to an interdisciplinary modeling approach in engineering, still some discipline-specific models, e.g., for specific simulation purposes, can be required [PBBK10]. However, the interdisciplinary model should “drive the development activities” [BSBF11], while the specific models are used on a very detailed level of the process. Thus, in this thesis the second alternative, an interdisciplinary modeling language, is chosen for an integrated representation of the system structure, i.e., the components from the different involved disciplines. This enables the different stakeholders to add their relevant properties to their specific components within the joint model.

R 2: ***SysML4Mechatronics must be defined as an interdisciplinary modeling language, integrating the discipline-specific components from mechanical, electrical/electronic, and software engineering.***

The tight interaction of the different components is a fundamental characteristic of complex mechatronic systems [Hehe14, HPZA10]. Thus, not only the different discipline-specific components, but also especially their dependencies – discipline-specific as well as interdisciplinary – have to be considered in the development [EsMO07]. Such interdisciplinary dependencies arise due to joint fulfillment of functionalities in the mechatronic system, e.g., a sensor is installed mechanically in the system, is connected to a PLC and is represented in the software as a variable. Thus, the components and their dependencies also have to be considered by the different disciplines in order to achieve the best solution for the overall mechatronic system. The dependencies between the different components therefore pose a major factor for design decisions during engineering [MQAW12] and have to be represented by the modeling approach appropriately. SysML offers the concept of ports to depict the interfaces of a component [Omgo15b]. Interfaces can refer thereby, e.g., to mechanical, power or information connections [KHJM99] and thus, represent the interaction between two connected components [BDGS14]. As the port concept of SysML however, does not distinguish between ports of different disciplines, it shall be extended by SysML4*Mechatronics*. Thereby, not only standardized ports (e.g., an Ethernet port) can be considered as interaction points. However, non-standardized ports have to be abstracted as far as necessary (e.g., ‘force closure’ for describing the connection method) [LiPa03]. Like the components themselves, their ports can be specified through properties [AWFW07]. Thereby, the parameters often can be defined as range in which the connection to another component is valid, e.g., many sensors can work in a voltage range of 10 – 30 V DC. These port parameters are especially important to analyze their compatibility and enable an exchangeability of components. As described above, the components in mechatronic systems can have various interdisciplinary dependencies and thus, possess interfaces in the different disciplines. SysML4*Mechatronics* shall offer the possibility to model and parametrize the components’ ports of the different disciplines.

R 3: ***SysML4Mechatronics has to extend the SysML port concept to represent the structural interfaces of the different disciplines and show (interdisciplinary) dependencies between the components in mechatronic systems.***

The development of complex mechatronic systems is very costly and time-consuming, if it is performed entirely from scratch. In order to stay competitive, reduce the time-to-market and effectively develop new variants and versions, companies aim at reusing (parts of) prior projects in their development. Modularization facilitates this approach

[MaJG11a] and helps to handle the complexity of mechatronic systems [KKJG13]. Furthermore, modularization helps to structure the system at different levels of detail [BSBF11], which are required to answer different questions of interest during the development and show different viewpoints on the system (e.g., (sub)modules can be considered as black-boxes or shown in detail). Thus, modularization poses a critical nonfunctional requirement [VFST15]. Depending on the intended purpose and area of application modularization can be defined differently [EbGö04, GePZ03, RoLF07]. The status-quo in industry (see chapter 3.1.2) shows that companies use different kinds of modularization (discipline-specific as well as interdisciplinary). For an interdisciplinary modularization, in general, two viewpoints can be distinguished: On the one hand, the modularization can be carried out by grouping components to physical units, including their control intelligence [LEHM11] (often also referred to as mechatronic components [Thra10]). On the other hand, the modularization can be carries out function-oriented and contain all information of those components that are required to fulfill a specific functionality in the system [LEHM11]. Thus, these 'virtual' modules can be applied during the development process in the model [ScVo11] and implemented individually for each discipline in the real system. In comparison to standardized mass products with much larger lot sizes, physical units, which require same module borders in each discipline [KeFV15] (referred to as 1:1:1 modularization [BBOR12]), often are not profitable in the development of customer-specific mechatronic production systems. Thus, for the modeling approach in this thesis a function-oriented modularization in the model shall be chosen. Therefore, adequate stereotypes in SysML4*Mechatronics* have to be defined for the modules and their ports.

R 4:	***SysML4Mechatronics has to support an interdisciplinary, function-oriented modularization of the system in the model that does not require a 1:1:1 modularization during realization.***

During the development of a mechatronic system, a multitude of components has to be defined and integrated in the overall model. Thereby, each component possesses various properties, which describe certain classified characteristics (e.g., the component's mass, electrical connection, functionality etc.) [Eppl11]. Thereby, for each stakeholder different properties can be of interest (e.g., the mounting dimensions in mechanical engineering and the power connection in electrical engineering), representing their viewpoint of the system [Hehe14]. Next to the relevance of properties for design decisions during engineering, they depict crucial aspects for analyzing the compatibility of different components in the system [Eppl11]. This concerns on the one hand the properties that are directly related to the components' interfaces (port properties), but on the other hand can also relate to component properties that are relevant for the overall system (e.g., the mass

of each component is relevant for adhering to mass restrictions of the overall system). Hehenberger [Hehe14] distinguishes additionally between provided and required properties of a component. In order to use the properties effectively, they have to be described unambiguously in the model, which is especially challenging when the properties are added manually to the model. For purchase parts, especially through the importance of e-commerce, efforts have been made to standardize the data in product catalogs [VoCu91] and represent the components with their properties in product classification systems. By integrating the information from these classification systems directly into SysML4*Mechatronics*, the modeling effort can be reduced and mistakes or ambiguities can be prevented. Thereby, a combination of properties from several classification systems should be enabled, as each classification systems integrated different properties, depending on its application area. However, classification systems undergo continuous changes [HeLS07], e.g., due to innovations or new product versions, and thus, a definition as stereotypes in the metamodel of SysML4*Mechatronics* is not suitable, as each change would require an adaption of the metamodel. Therefore, the standardized properties shall be integrated to SysML4*Mechatronics* more flexibly in form of model libraries. Model libraries enable the reuse of model elements for a specific domain and can be adapted flexibly during usage without changing the metamodel of the modeling approach [Omgo15b]. Next to purchased components, usually also individually developed components are required for mechatronic systems, which are not considered in the product classification systems. However, if possible, the standardized properties from the library should also be used for the individually developed components, in order to prevent ambiguities. Furthermore, SysML4*Mechatronics* shall enable the manual integration of additionally required properties (e.g., a company specific labeling), for the individually developed components as well as for purchased components.

R 5: ***SysML4Mechatronics must enable the flexible integration of standardized properties and components from existing classification systems through model libraries.***

In order to enable an analysis of the components' compatibility in case of a change two aspects have to be combined: On the one hand, a comprehensible modeling approach is required, which allows the engineers from the different disciplines to add their relevant information in the model. On the other hand, formal methods are needed, which enable the interpretation of the modeled information and conduct compatibility checks [FeKV14]. The intended modeling approach focuses on the property-based information input and modeling (see requirements R 2 – R 5). However, in order to enable the described analysis, it has to be ensured that the models can be transformed in an according

formal model. The suitability of a modeling approach for model transformations is significantly dependent on the chosen modeling language (see requirement R 1) [FrLa03].

R 6: SysML4Mechatronics models shall be used as basis to enable a compatibility analysis with formal methods.

The success of modeling approaches depends, next to the definition of the modeling language itself, considerably on their implementation and tool support, as shown e.g., for IEC 61499 in [SCKP07]. In Model-Based Engineering the used modeling editor influences significantly, how complicated it is to create models and how the relevant information can be included in the model. Thus, an adequate tool implementation must be able to represent all relevant aspects and on the other hand be adaptable, to show each stakeholder only that information, which is relevant from his perspective. An unnecessary complexity and insufficient maturity of modeling tools are frequently mentioned inhibitors for companies to integrate them in their processes [BoCl10]. Witsch [Wits13] showed in the development of the MES-ML that a reduction of complexity in the diagrams, e.g., through filters, layout conventions or highlighting, is especially important for new users of a modeling language. Furthermore, a high number of different diagrams to represent a system can slow down the modeling process and lead to confusion of the user [Katz08, VBOJ12, Wits13]. If several diagrams of the same type are used thereby, Moody [Mood09] refers to it as homogeneous cognitive integration, otherwise (different diagram types) as heterogeneous cognitive integration. Thus, an adequate tool implementation for SysML4*Mechatronics* has to enable on the one hand, a comprehensible overall view of the mechatronic system structure and on the other hand, enable the different stakeholders to filter the included information. In this way, they can receive a view on the system, which only shows the relevant information for their discipline.

R 7: The implementation of a SysML4Mechatronics editor has to support the user by offering an interdisciplinary, structural view on the system in one diagram type with respective filters for the specific views.

SysML4*Mechatronics* and its respective implementation in a modeling editor shall be developed as interdisciplinary interface to engineers. Thus, it has to be ensured that the specific users, i.e., engineers from mechanical, electrical/electronic and software engineering, are able to apply the modeling approach in an effective, efficient and satisfying way [Dind16]. The requirements on usability can thereby, influence each other in a contrary manner, e.g., a modeling approach, which is easy to use, may not have the expressiveness, which is necessary to fulfill the task. Thus, the application- as well as the user-specific usability requirements have to be considered [FrLa03]. Obermeier et al.

[ObBV15, VBOJ12] investigated usability aspects for object-oriented modelling approaches in automation and showed that they lead to a reduction of engineering efforts and an increase of efficiency and comprehensibility. The users' acceptance of a new modeling approach, and thus its success, thereby is strongly depended on their perceived ease of use and their perceived usefulness of the approach [Davi89]. Perceived ease of use refers to the users' wish to apply the new modeling approach without major difficulties. The relevance of the perceived ease of use was demonstrated e.g., for conceptual models [MaPo07] and SysML requirements diagrams [SoNa14]. Thereby, the representation form of the modeling approach as well as the design of the respective modeling editor (see requirement R 7) play a decisive role for its comprehensibility and effortless usage. As SysML4*Mechatronics* is based on SysML, the models are represented in a graphical, semi-formal way. By implementing a prototypical SysML4*Mechatronics* modeling editor, potential users can interact with the system and evaluate how easy it is for them to use the modeling approach.

R 8: The SysML4Mechatronics modeling editor must achieve a positive perceived ease of use through potential users.

Next to the ease of use (see requirement R 8), the perceived usefulness of a modeling approach is equally essential [Davi89]. If users see a benefit in using a certain new modeling approach to reach their goal, they asses it as especially valuable and practicable, which leads to a higher acceptance of the approach [MaPo07]. Thus, the perceived usefulness of SysML4*Mechatronics* should consider the aspects of facilitating the mechatronic development, reusing components and modules, analyzing interdisciplinary dependencies and defining properties more effectively. While the perceived usefulness can be determined through questionnaires or interviews with the users after utilizing the new approach, a direct measurement of the 'real' usefulness would imply that its impact on subsequent modeling projects can be observed. However, as various factors have an influence on the outcome of a project, it is hardly possible to isolate the impact of the usefulness [Ritt10]. Therefore, after conducting an adequate modeling assignment that contains the required tasks, potential users shall assess their perceived usefulness of the modeling approach in a respective questionnaire.

R 9: SysML4Mechatronics must achieve a positive perceived usefulness for the interdisciplinary development of mechatronic systems through potential users after utilizing the approach.

3.3. Considered boundary conditions

The application of models and thus, also the development of according modeling approaches requires per definition that not all aspects of the original system can be represented [Lude03]. Instead, a simplification and limitation to the properties of interest for a specific purpose is necessary. As this thesis focuses on the interdisciplinary modeling of mechatronic production systems with their components, dependencies and properties, the following simplifying assumptions are made:

- Regarding the lifecycle of mechatronic systems, the approach emphasizes the support of the detailed mechatronic engineering phase as well as the reengineering phase. However, connections to Model-Based approaches that support other lifecycle phases, e.g., the high-level system's definition or the discipline-specific implementation, are shown throughout the thesis.
- The modeling approach focuses on the representation and analysis of the structural system view. Next to the structural view, further views on the system, e.g., requirements or behavior, are important for an overall development of a system. UML [Omgo15a] and SysML [Omgo15b] offer several diagram types, e.g. state machine or activity diagram (cf., Figure 4-3), to model the (functional) behavior of a system. As far as necessary for the approach in this thesis, the function is considered within the properties of the systems' components. However, the presented approach of this thesis can be combined with the behavioral diagrams of SysML in an overall system development.
- The information and properties in the model are considered as fixed after the model has been created. Changes of properties, e.g., during runtime, are not transferred back into the model. This approach is usually referred to as *Forward Engineering* [SeKü04].
- Next to the modeling approach, further (socio-)technical boundary conditions have to be considered for the innovation process of complex systems, e.g., different qualification levels of employees or the utilized tools and platforms of the company. Through cooperation within the SFB 768, SysML4*Mechatronics* was integrated in the overall management of innovation processes [KBCW14, KPRV16, KWMK13, MKKV13].

Through consideration of these described boundary conditions for the development of the modeling approach the research is narrowed to the relevant aspects within the scope of this thesis. Regarding the completeness of an overall Model-Based approach for mechatronic systems, further aspects during the development and use phase have to be investigated, e.g., through further diagram types and analysis methods. The resulting limitations and required steps for further research are discussed in chapter 7 and 8.

3.4. Summary

In chapter 3 the requirements for the development of the interdisciplinary modeling approach for mechatronic systems – SysML4*Mechatronics* – were defined.

In a first step, the status-quo and the challenges in industry were assessed through a survey with industry experts and the analysis of further relevant existing surveys. The results show that interdisciplinary collaboration is highly important in industry. Although a high potential of model-based approaches is expected, most modeling languages are still not very common in industry. Thus, for the estimation of change impacts, model-based approaches are only used scarcely. Rather, expert knowledge is used primarily. Regarding modularization, different approaches, discipline-specific as well as interdisciplinary, are followed by industry. Thereby, especially the disciplines software, electrics/electronics, and mechanical engineering are considered in the modularization process.

Based on the identified challenges in industry, the requirements for the development of SysML4*Mechatronics* were defined. Thereby, the definition of the modeling language itself, a respective implementation in a prototypical modeling editor and the perceived ease of use and perceived usefulness through potential users are considered.

Finally, necessary boundary conditions and simplifying assumptions for the development of the modeling approach in this thesis were pointed out.

4. State of the art

Mechatronics and thus, the mechatronic development play a decisive role in a wide range of modern products and systems. As such, the development requires the collaboration of stakeholders from various domains. An interdisciplinary comprehension, supported by appropriate modeling approaches, is essential. Furthermore, companies are forced by the global economic cost pressure, to develop their systems as efficient as possible, e.g., by using standardized components and through a suitable modularization strategy, which enables an effective variant and version management. In order to face these described challenges, various approaches have been developed in research and industry. This chapter describes the state of the art, regarding the three main aspects *modeling approaches* (chapter 4.1), *properties in engineering and the use of standardized classification systems* (chapter 4.2), and *modularization* (chapter 4.3) in the context of mechatronic systems.

4.1. Modeling approaches

Various model-based approaches in the field of software and (mechatronic) system development exist on different levels of detail, discipline-specific as well as interdisciplinary approaches. In this chapter the state of the art considering standardized as well as specifically defined modeling approaches are presented and discussed, regarding their applicability for the requirements in this thesis.

4.1.1. Unified Modeling Language (UML)

Since the emergence of object-oriented software development, various modeling languages and techniques have been developed (e.g., the Booch method [BrCe11] or object-oriented software engineering) to specify and construct software. As a standardization of these different languages, the Object Management Group (OMG) released in 1997 the Unified Modeling Language (UML) [BoRJ00].

UML is a graphical, implementation independent modeling language, which follows the object-oriented approach [SeBF07]. It is not only suitable for the specification and modeling of software systems, but can also be used for the development and documentation of software intensive systems. Therefore, UML nowadays is one of the most wide spread modeling languages in model-driven software engineering [SeBF07]. In its current version (Version 2.5, [Omgo15a]) UML comprises 14 diagram types, which are divided into the main categories *'Structure diagrams'* and *'Behavior diagrams'*, representing the two major aspects of the system under development, i.e., its structure and behavior (for a detailed description of the UML diagram types see e.g., [BoRJ00, Omgo15a]). The semantic foundations of these aspects, as they are represented in UML,

were defined by Selic [Seli04] and included into the UML 2.0 specification [Omgo05]. Thereby, as shown in Figure 4-1 [Seli04], three main semantic layers can be defined, in which each layer depends on the layers below. The *structural foundations* constitute the basic layer, which reflects the fundamental premise of UML, stating that every behavior results from actions of structural entities [BCDH06] and thus, there is no "disembodied behavior in UML" [Seli04]. On this lowest structural level, typical basic concepts are types, namespaces, relationships and dependencies in the system, which are complemented by specific constructs such as interfaces, components, instances or packages [Omgo15a].

Based on the structural foundation, the behavioral base is defined on the next layer and provides the basic framework for all higher level behaviors [BCDH06]. Thereby, *actions* pose the definition of the smallest behavioral unit, which can be used on the higher level for the detailed description of behavior. Furthermore, in this layer the behavior within a structural entity as well as between different structural entities is defined.

On the upper layer, the semantics of high-level behavioral constructs are described, which include activities, state machines and interactions. According diagram types are provided by UML to model these constructs [Omgo15a]. As described above, the higher layer can thereby make use of the lower level actions and the structural basis.

Additionally to the described layers, a layer for supplemental modeling constructs is defined in UML 2.5, which includes use cases, information flows and deployments, having structural as well as behavioral aspects [Omgo15a].

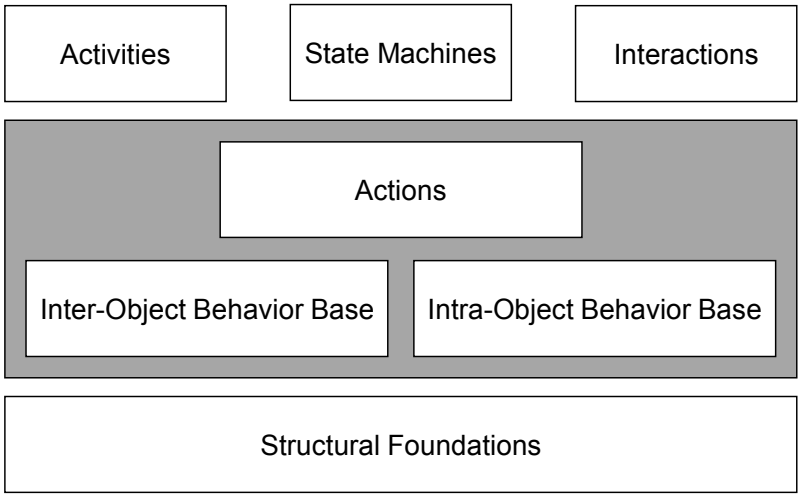


Figure 4-1: Semantic layers of UML according to [Seli04]

Especially since the change from UML 1.x to UML 2.0 its application areas expanded and UML also was adopted for the (software) development in the machine and plant manufacturing industry [SeBF07, Thra04]. In order to adapt the modeling language

thereby to the specific needs, UML offers the possibility to define *profiles*, as shown, e.g., for real-time, event-driven software architectures through specific communication-ports (UML-RT, [Seli98]) or for modeling and analyzing real-time systems (UML MARTE, [Omgo08]). In the field of automation software development further specific profiles have been developed, e.g., UML for process automation (UML-PA, [Katz08]). In order to accelerate the programming and reduce faults, many efforts have been made to offer the possibility to generate (parts of) the software code automatically from the models (e.g., [EsMO07, FaSB11]) or use directly executable models (plcML, [Wits12, WiVo11]). plcML thereby enables a direct Model-Based development and can be used during run-time to monitor the actual state of the system [Wits12].

Although there are some approaches that use UML to model also the physical part of software-intensive systems [BaDa03, Seli00], the modeling language does not have the expressiveness to cover all relevant aspects of the systems engineering process (e.g., requirements definition). Thus, a *UML for Systems Engineering Request for Proposal* [Omgo03] was initiated, which led to the development of the modeling language described in the following chapter.

4.1.2. Systems Modeling Language (SysML)

Based on UML, the Systems Modeling Language (SysML,[Omgo15b]) was specified as multipurpose modeling language, in order to fulfill the requirements to model multi-domain systems and thus, overcome the software-centered view. UML was adapted by reusing the necessary subset of UML 2 and extended in order to depict the missing systems engineering aspects (Figure 4-2, [Omgo15b]). SysML is object-oriented and supports hierarchical modeling.

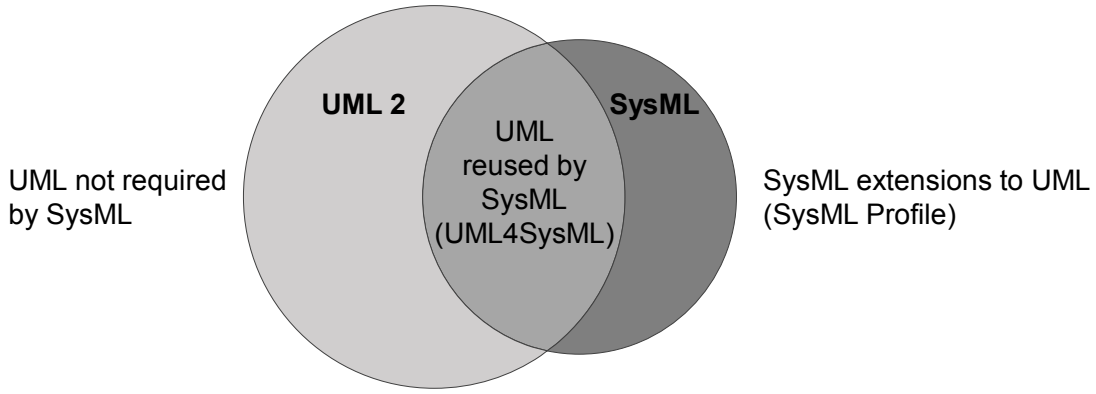


Figure 4-2: Relationship between UML 2 and SysML according to [Omgo15b]

Goal of the development of SysML was to provide a graphical modeling language, which supports the phases of specification, analysis, design and verification for complex systems. Such systems “often include hardware and equipment, software, data, people,

and facilities" [FrMS15, p.3]. SysML was developed to apply the MBSE approach, focusing on a model-centric development, and therefore, offers graphical representations with a semantic foundation for modeling [Omgo15b]. Nine diagram types (partly reused or adapted from UML, see Figure 4-3) are provided to model the structure, behavior, parametrics and requirements of the system under development. The structural representation comprises aspects of the system-composition and definition of system properties, connections as well as classifications, e.g., through inheritance. Properties can be limited through constraints. Behavior can be modeled in a state-based, flow-based, and message-based way. Furthermore, allocations between the structure, the behavior, the constraints, and the requirements can be defined. The requirements can be linked to other requirements (e.g., on different levels of detail), system's elements, and test cases [FrMS15].

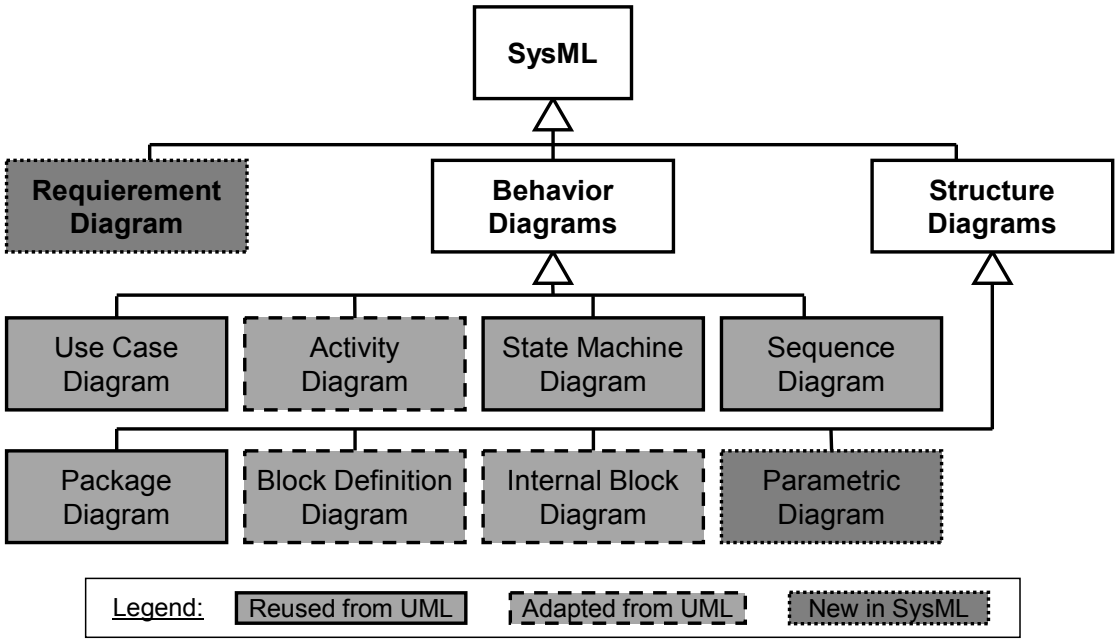


Figure 4-3: SysML diagram taxonomy, based on [Omgo15b]

The two most used diagram types to model the system's structure in SysML are the Block Definition Diagram (BDD) and the Internal Block Diagram (IBD) [BoCl10], which were adapted from the UML class diagram and composite structure diagram, respectively. While the basic element in UML is called *class*, it was adapted in SysML to the stereotype *block*, as not only software classes can be represented in SysML.

The BDD is used to define the hierarchy of elements in a system and their relationships. Thereby, elements can represent, next to blocks, also value types, constraint blocks, flow specifications, and interfaces. These elements can be detailed in the other eight diagrams of SysML. Regarding the relationships between the elements, inheritance (through generalization), associations (as part, shared or reference association) and general

dependencies can be depicted. Figure 4-4 shows on the left side an example BDD for a (simplified) water storage reservoir. Through part associations (♦), it is defined that the reservoir consists of the tank system and a HMI. The tank system itself consists of a tank and a valve. The flow rate is defined, as an example for a property, in the values section of the valve. As the BDD depicts the types and quantities of the components in a system as "black box" view, it can be compared to the first page of a furniture assembly instruction.

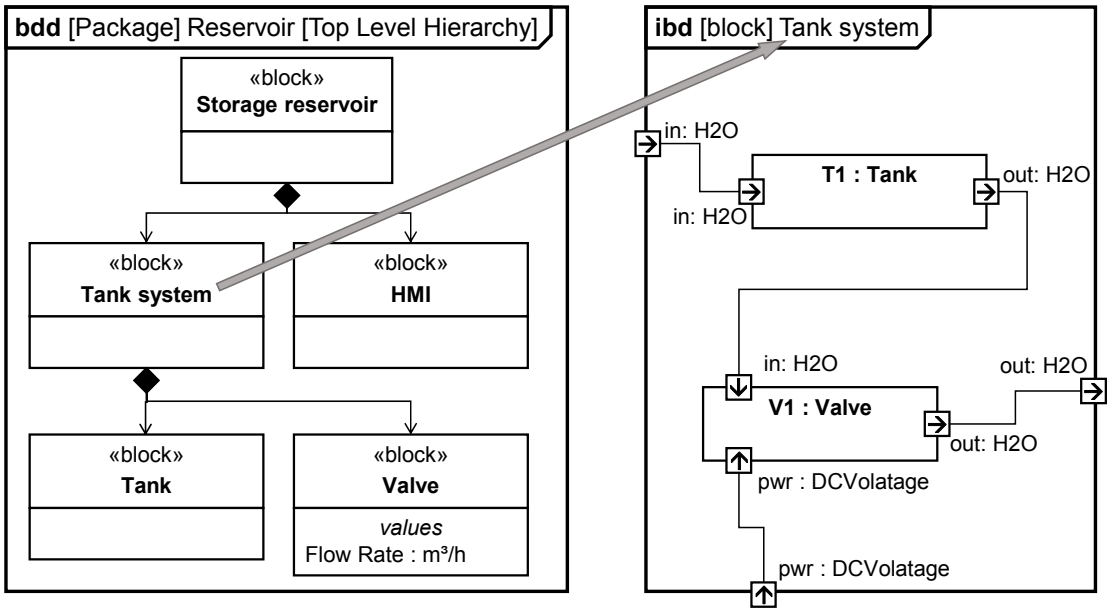


Figure 4-4: Example of a BDD (left) and IBD (right) in SysML

An internal, "white box" view on a system element can be depicted in SysML using the IBD. The blocks, represented in the BDD, are instantiated in the IBD as parts and their connections can be modeled through ports (proxy or full ports) and connectors. Thus, goal of the IBD is the depiction of the detailed structure or assembly of a block. On the right side of Figure 4-4 the IBD of the tank system is shown. The two parts T1 (tank) and V1 (valve) are connected to enable the flow of water from the tank through the valve. Ports on the border of the IBD depict that the tank system is connected externally to other elements, e.g., an in/outlet for the water and a power connection for the valve. Thus, the IBD can be compared to the last page of a furniture assembly instruction, which shows how the single components are connected to each other. For a detailed description of the other diagram types see, e.g., [FHPZ10, FrMS15]. UML and SysML, furthermore, enable the definition of profiles, which are based on the standardized language, but are adapted modeling approaches for a specific application area. Although the IBD offers the possibility to model in detail which parts are connected to each other, it should be noted that SysML only provides very limited means (e.g., as volume property) to model the geometry of the parts. Thus, to create an according geometric model, a Computer-Aided

Design (CAD) model is required additionally [Dell14]. Similarly, further specific models are required during the development, e.g., simulation models, which usually are carried out in well established, specialized tools [SKSP10]. This leads to a hierarchy of models in the development process, in which SysML depicts the high-level model view and the different discipline-specific models the detailed views [BSBF11, SKSP10]. Through appropriate approaches a (partial) transformation between the system model and the detailed, discipline-specific models can be established (e.g., for Modelica [PBBK10]). However, the existence of interrelated information among different models is a source for potential inconsistencies, especially if the consistency of the models has to be checked by the users manually [SKSP10].

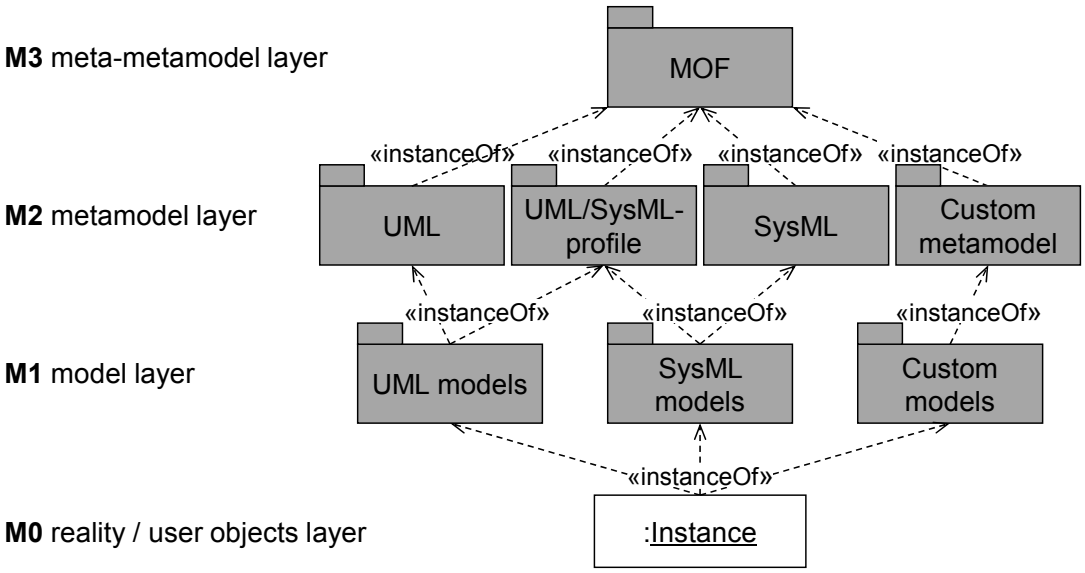


Figure 4-5: Four layer MOF-based MDA framework of the OMG [Omgo15c]

In order to facilitate the exchange of information between different models, the OMG specified the XML Metadata Interchange format (XMI, [Omgo15d]). In order to enable the XMI-based exchange of model data, the respective metamodels have to be conform to the *Meta Object Facility* (*MOF*, [Omgo15c]) in the four layer Model Driven Architecture framework (MDA framework, Figure 4-5). Core element of MOF is a meta-metamodel (M3 layer in Figure 4-5), which provides the foundations for defining and constructing conform modeling languages (M2 layer in Figure 4-5, see also Figure 2-9). Next to UML, SysML, and respective profiles of these languages, also customized MOF-based metamodels can be defined (see, e.g., Estevez and Marcos [EsMa12]). Instances of the metamodels are the models on the M1 layer, e.g., the presented SysML model of a storage reservoir (Figure 4-4). Finally, the M0 layer of the MDA framework represents the objects in the real system, e.g., the actual storage reservoir at a specific location.

The MOF core is extended by further specifications which focus on the handling and application of MOF-based (meta-)models, e.g., the Object Constraint Language (OCL) for the specification of constraints, MOF Query/View/Transformation (QVT) for defining transformation rules, or MOF Model to Text (MOFM2T) for the generation of text, e.g., software code [Omgo15c]. While the definition of bidirectional transformation rules with QVT requires MOF-conform models on both sides, MOFM2T only needs a MOF-conform model as starting point. The resulting text based outcome can be provided in any syntax.

Although SysML aims at providing the described benefits for the system development and the application of MB(S)E-approaches is growing [Mota13], there are various reasons and barriers, which still lead to a resistance of the industry to switch from traditional approaches to MB(S)E-approaches. Next to a lack of knowledge and skills for using Model-Based approaches, many companies do not see a high "perceived value of MBSE" [Mota13]. Reasons therefore are missing modeling methods and guidelines [AlZi13], missing adequate examples that give a description on generating SysML models [BoCl10, Mota13], and a lack of maturity and usability of many existing SysML tools [BoCl10]. Furthermore, the expected high learning efforts for using SysML [AlZi13] depicts a resistance to switch to MBE. Often users also have difficulties in abstracting real systems and thus, create large, inflexible and hard to maintain models. Such inflexible models pose a disadvantage, if (parts of) the model shall be reused, e.g., for new variants or versions of the system or in other projects [Kass10].

4.1.3. MBE-approaches for mechatronic systems

Based on the described advantages and challenges for using MBE, various approaches, which specifically focus on the development of mechatronic systems, have been developed. Vyatkin [Vyat13] points out that the development processes in different industries vary substantially and thus, also adequately adapted modeling approaches are required. In the following, existing modeling approaches from literature are reviewed and subsequently compared towards their fulfillment of the defined requirements for SysML4*Mechatronics* (chapter 3.2).

Fantuzzi et al. [FaBS09] introduced the use of UML for the development of automated machines. Based thereon, Bassi et al. [BSBF11, BSFB06] focused on supporting the design process of manufacturing systems. The authors state that the development of complex mechatronic systems requires a hierarchy of models to describe all necessary information on different levels of detail. Therefore, they propose a three level model hierarchy. The most detailed level includes the discipline-specific (executable) models, which are, e.g., necessary for carrying out specific simulations. The different detailed

models of the subsystems are linked on a higher level to the global executable model, where the dynamics of the overall system are analyzed. In order to offer additionally a system-wide perspective, a high-level model, created with SysML, is used. During the design process, based on the SysML models, different solution alternatives can be evaluated, refined, and, if they turn out to be inappropriate, eliminated. The level of detail within the diagrams of the high-level model are refined during the design process, starting from a depiction of the core concepts and general structure of the machine, till all information are included, which are necessary to create the global executable model.

Thramboulidis [Thra10, Thra13] proposes the use of a 3+1 SysML-view model. To represent a central view ('+1 view') of the mechatronic system, a SysML profile is used, while the other three views describe the discipline-specific models of the mechanical, electronic and software development. The approach is based on the concept of mechatronic components (MTC), which pose encapsulated components containing all mechanical, electronic and software parts and only have defined mechatronic interfaces to their surroundings. A mechatronic system can then be built through the combination of the MTCs. Thus, the approach requires a 1:1:1 modularization of the mechatronic system. Furthermore, an analysis of the interrelations between the different discipline-specific parts within the MTC is not conducted within the SysML model, rather Thramboulidis suggests to implement an interchange of the data between the different discipline-specific tools [Thra10].

A MBE-approach, which focuses on the concept development phase of mechatronic systems, is presented by Kruse et al. [KGSE14, KMWC12, KrSh16]. The approach utilizes SysML and includes thereby libraries with classified properties from the NIST functional basis [HSMS02] and from the Function-Behavior-Structure approach [Gero90] in order to generate new solution ideas during the early concept phase of product development. The approach thereby, focuses on a functional decomposition and assigns then possible (physical) components, which can be used to fulfill the desired functions. At this early stage in the development process, a detailed analysis of the interfaces between the different components in the mechatronic system is not yet possible, as not all properties are defined yet, and thus, not focused by the approach.

Another approach, which focuses on interdisciplinary conceptual modeling and an integration into Product Lifecycle Management (PLM) solutions, is presented by Eigner et al. [EiGZ12]. The SysML profile is used for modeling the requirements, functional elements and logical system elements in the "modeling and specification view". In the "modeling and first simulation" view, the authors propose to use multidisciplinary simulations such as Matlab or Modelica. In the detailed "discipline-specific modeling" view, the aspects are represented in the particular models (e.g., CAD models). To link the

different model elements, requirements, functions, and architecture elements to each other and ensure traceability a Product data management (PDM) solution is used. Thus, elements from SysML can be linked e.g., to the Bill of Material (BOM) and further to the CAD-element in the PDM system.

Gausemeier et al. [GaDK10, GFDK09] developed the approach CONSENS (Conceptual Design Specification Technique for the Engineering of Complex Systems), which supports the discipline independent development of mechatronic systems. Based on several specified metamodels, the approach enables to specify different system aspects, such as the active structure, describing the interdisciplinary structure on a high level of abstraction, application scenarios, showing the behavioral specification in state machines, or a function hierarchy, which shows the requirements. This high-level specification of the system focuses on defining principle solutions in the conceptual design phase, which then can be detailed within the different disciplines. In order to analyze the consistency and synchronize changes between the discipline-specific models and the high-level specification a Triple Graph Grammar (TGG) based approach is utilized [GSGK09]. In addition, the SysML profile SysML4CONSENS was developed [HBMS16].

In order to model the software-specific parts of a mechatronic system in CONSENS, the modeling approach MechatronicUML is used [GFDK09]. MechatronicUML [BDGS14, ScWe10] enables users to model and verify the software that is embedded in mechatronic systems. It is based on a subset of UML, and focuses specifically on the real-time behavior by combining UML state machines and timed automata [BDGS14]. On a structural level, MechatronicUML enables the development of the system architecture through a component-based approach using blocks, ports and connectors in UML component diagrams. Thereby, the components can communicate through messages, which they exchange with each other. Therefore, the port concept of UML is extended in order to specify the respective real-time communication protocols. Although non-software elements of the mechatronic system can be depicted in MechatronicUML to a certain extent, it is not the focus of the approach and thus, they are only shown on a high level of abstraction [BDGS14, p.162].

Hehenberger et al. [Hehe14, HPZA10] present a hierarchical design method for mechatronic systems, pointing out that a sequential development process is not satisfactorily for mechatronics. The model hierarchy enables the depiction of different viewpoints and granularity-levels of the system. The approach is based on mechatronic modules. They include the domain-specific components, which are linked to each other on the mechatronic coupling level. Also all property parameters, which influence other domains, are captured on this level. The authors point out that the requirements on the

parameters thereby can compete between the different involved domains and require a mutual communication of the different design teams to find the optimal solution.

Shah et al. [SKSP10] developed a multi-view modeling approach based on SysML. The approach focuses on the transformation of discipline-specific models to a joint model in SysML at a higher level of abstraction. Thereby, for each discipline a specific view is created as a subset of the joint model with a certain view of the system at a more detailed level. By using model transformations, the discipline-specific views in SysML are mapped to the respective domain specific models. Paredis et al. [PBBK10] show a specification of a SysML-Modelica transformation, to combine the analytic and descriptive modeling approaches.

Marcos and Estévez [EsMa12, EsMO07, MaEs08] define a description language for industrial control systems, which focuses on the automatic generation of PLC code. The XML-based approach enables the modeling of the industrial control system from different views, including control engineering, software engineering and electrical engineering. The structure of the software model is defined according to IEC 61131-3 [Ieci14a] in order to generate the code automatically. Through according XML style sheets, the exchange of information between the different views is enabled and thus, the integration of components from different vendors shall be facilitated.

A system-engineering environment for cyber-physical automation systems is presented by Harrison et al. [HaVA16]. The approach combines different models, such as CAD, control logic definition, simulation, visualization, deployment, and fault tracking in an integrated toolset. By using open standards, e.g., XML, the approach shall be independent from specific vendors and enable especially smaller companies to offer solutions for highly interconnected systems in the context of Industry 4.0.

SysML-AT (SysML for automation) [Schü15, VSFL14] is a modeling-approach that focuses specifically on the development and generation of software-agents for automated production systems. The SysML profile defines on a technical system layer and an automation system layer means to develop agents and automatically generate the executable code. Furthermore, requirements (functional and nonfunctional), software applications and hardware components can be modeled in SysML-AT. The development and deployment of automation software for distributed systems is the main focus of SysML-vAT (SysML for distributed automation systems) [FEHF13, Fran14]. The approach is based on the V-model, which was adapted to the requirements of distributed systems [FEHF13]. Through the use of characteristics, the (non-)functional requirements are mapped to the solution elements [FVFE15]. The model-based approach modAT4rMS (modular automation for reuse in manufacturing systems) [Brau13, ObBV15] aims at supporting the development of object oriented automation software with a special focus

on its usability and software modularity. Therefore, different features of UML and SysML were combined to model the structure and software behavior of automation systems. In a series of usability tests [Ober15] the benefits of the model-based approach were examined in comparison to classical IEC 61131-3 programming.

Next to UML/SysML-based approaches, the use of AutomationML (AML) [Ieci14b] in interdisciplinary engineering gained in importance within the last years. AutomationML was developed as an open, neutral and standardized data exchange format, based on XML. AML integrates different standards to realize the data exchange. The structure of the system and the top-level integration is realized in CAEX [Ieci08]. The kinematics and geometry information is represented in COLLADA [Isoi12]. Finally, for the PLC software, PLCopen [Plco09] is integrated in AML. Although AutomationML per se is not a modeling language, several approaches have been developed which use AML as basis for developing mechatronic and automated systems.

Biffl et al. present a MDE-approach, using AutomationML in combination with SysML [BBLM16, BMWL15]. A comparison between AML and SysML shows that the visualization strength of SysML, especially in the early development phase, can be combined with the model exchange capabilities of AML. In order to achieve interoperability between the two approaches, a mapping and model transformation between AML (CAEX) and SysML (BDD and IBD) is defined.

Glawe and Fay [GlFa16] combine AutomationML with knowledge-based models in OWL. During the engineering process the amount and preciseness of information is growing and different disciplines contribute with their specific views. AutomationML thereby is used to link the information, which is provided through various heterogeneous tools. The provided information is then enriched through additional knowledge (e.g., regarding security), which is captured in an OWL ontology. The enrichment is automated through a rule-based approach.

Abramovici et al. [AbGD16] developed a semantic data management approach to develop smart products. The approach focuses on information from components, architecture and the usage of smart products. The information is used as basis for the development of a new generation or reconfiguration. All information regarding the smart product, the specific instance, the process, the resource and the organization is stored in an ontology.

A framework regarding the integration of process and product models is presented by Eckert et al. [EABC15]. Next to the integration of different disciplines in the development process but also the integrated consideration of performance targets, which relate to the product- and process-domain, e.g., time, quality and cost are essential. As different company departments, e.g., engineering, finance and quality control, are involved,

various interrelations exist [EWMA17]. The authors show that an increased model integration would also support decision processes in case of an engineering change.

Ramsaier et al. [RHZS17, RSSR17] present a framework with a graph-based design language based on UML along the product lifecycle. An automated rule-based execution of the graph-based design language enables the deduction of digital models for specific disciplines or purposes (e.g., bill of materials, simulation, costs) from the design model. The approach reduces the required number of manual transformations between different tools and thus, potential inconsistencies between the models.

Halstenberg et al. [HaLS17] analyze in detail different Input-Output matching tools for the usage in the field of industrial symbiosis. Industrial symbiosis focuses on the usage of waste or by-products from one process as raw materials for another process. Therefore, information has to be exchanged along the material flows and process chains. The authors state that current tools usually focus on technical data and not on organizational data, e.g., from Product Data Management systems.

The described approaches cover a wide range of aspects in the engineering of mechatronic systems, reaching from conceptual (product) design in the early phases to detailed aspects such as automatic code generation. Thereby, depending on the focus of the different MBE-approaches, standardized languages like UML or SysML, profiles of these languages, or specifically developed languages are used. Table 2 analyzes the described approaches regarding their fulfillment of the requirements defined in chapter 3.2.

Table 2: MBE-approaches for mechatronic systems – comparison of existing modeling approaches

Reference	Language used for approach (req. R 1)	Scope (R 1, R 2)	Mech, E/E, SW (R 2, R 3)	Modulari-zation (R 4)	properties (R 5)	Transfor-mation (R 6)
[BSBF11, BSFB06]	SysML	development process (aPS)	◑	●	◑	◑
[Thra10] [Thra13]	SysML profile	development process (aPS)	●	◑	○	◑
[KGSE14, KMWC12, KrSh16]	SysML	conceptual design	◑	○	●	○
[EiGZ12]	SysML profile	conceptual design	◑	○	●	○
[GaDK10, GFDK09, HBMS16]	Own metamodel / SysML profile	conceptual design	●	●	◑	◑
[BDGS14, ScWe10]	UML extension	development process (SW)	○	○	●	●
[Hehe14, HPZA10]	Not specified / SysML	development process	◑	●	●	○
[PBBK10, SKSP10]	SysML and SysML profiles	development process	●	○	○	●
[EsMa12, EsMO07, MaEs08]	XML	development process (SW of aPS)	◑	◑	◑	●
[HaVA16]	XML	development process (CPS)	◑	●	○	◑
[Schü15, VSFL14]	SysML profile	development process (SW-agents)	◑	●	◑	●
[FEHF13, Fran14]	SysML profile	development process (distributed aPS)	◑	◑	●	○
[Brau13, ObBV15]	UML and SysML extension	development process (SW of aPS)	○	●	◑	●
[BBLM16, BMWL15]	AML / SysML	development process (aPS)	●	◑	◑	●
[GlFa16]	AML / OWL	development process (aPS)	◑	○	●	●
[AbGD16]	OWL ontology	(re-)development process	◑	◑	◑	◑
[EABC15, EWMA17]	Framework with different models	development process	◑	○	◑	○
[RHZS17, RSSR17]	UML extension/ graph-based	development process	◑	●	◑	◑

● addressed by the approach; ◑ partially addressed by the approach;
○ not addressed by the approach

4.2. Properties

Each model describes certain aspects of a system under development, which are relevant for the specific stakeholders. Within the model, the different components can be specified through properties[7]. In this chapter, relevant approaches for the use of properties in engineering are described and existing property classification systems are presented.

4.2.1. Use of properties for engineering of mechatronic systems

Properties, in general, describe a certain characterizing aspect of an object. Hadlich [Hadl15] defines a property as a distinguishable characteristic of an observed object, which can be used to classify the object. For the development of mechatronic systems, this means that components and modules can be classified through their properties and based thereon, e.g., the fitting component for satisfying a requirement can be chosen. For the approach in this thesis, properties are especially relevant for analyzing the compatibility of a component or module, i.e., if the (port) properties of a component are compatible with the (port) properties of the other components in the system.

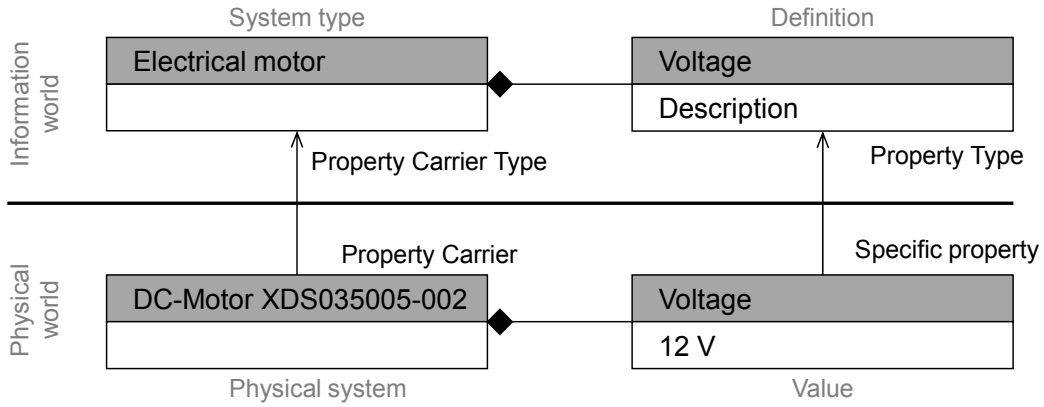


Figure 4-6: Example of a property and its property carrier, based on [Eppl11]

Epple [Eppl11] defines that a property can be specified through a single value, which does not vary in the observed period of time. Furthermore, he defines the system element that is specified through the property, as property carrier. Thus, on an abstract level, a property type (without specific values) is assigned to a property carrier type. By applying this classification to a specific system, the property type then is transformed to a specific property belonging to a specific property carrier. In this step the concrete values of the property are defined (Figure 4-6). Thereby, the allowed value-range and units should be defined beforehand, in order to avoid semantical problems at the interfaces [DöHe13]. The

[7] In literature several similar expressions, e.g., attributes, characteristics and features can be found. In the following, only the term property will be used.

properties of a system are independent from a specific modeling approach [Eppl11] However, the role of a property can vary depending on the property carrier to which the property is assigned [Hadl15]. A property can either be a statement of a requirement or specify a solution artefact [FVFE15]. If a manufacturer indicates, e.g., that his motor has a nominal power of 100 W, he guarantees this property as specification of the motor. On the other hand, if the manufacturer defines that the motor needs a 12 V power supply, this poses a requirement property in order to achieve the nominal power.

Typical relevant properties of purchased products can be found in the supplier product data, which usually are described in data sheets. However, the way of description and the naming thereby can vary widely between different suppliers, which makes it difficult to compare different components in a computable way. In order to overcome this problem, classification systems have been developed, as described in the following chapter.

4.2.2. Property classification systems

The goal of property classification systems is the standardization of property descriptions. Furthermore, in product classification systems the property carrier types can be specified according to their property types [Hadl15]. Since the growing importance of e-commerce in business-to-business transactions, companies seek at making their communication efficient and reduce costs and time for finding a certain product [FYOS01]. Thus, various classification systems that aim at providing a general, computable representation of properties and / or products have been developed. Their scope ranges from coarse-grained taxonomies to detailed descriptions for a specific segment of products. Among the variety of classification systems, the most wide spread approaches are UNSPSC, eOTD and eCl@ss, which aim at covering a wide range of industry sectors, as well as RNTD, which poses an industry-sector-specific classification system [HeLS05]. In the following, these approaches are presented briefly.

The United Nations Standard Products and Services Code (UNSPSC)[8] is a hierarchical taxonomy for a wide range of products from various industries. Its main goal is to provide a reference numbering to facilitate e-commerce. On the most detailed level, UNSPSC contains more than 50.000 commodities, i.e., products and services [Gsus15]. However, UNSPSC lacks some major aspects to serve as a basis for a property-based engineering. The classification system only classifies the products, i.e., property carriers, but does not define their properties. Furthermore, its usage is unintuitive and it lacks an appropriate level of detail [FYOS01].

[8] https://www.unspsc.org, retrieved on 23 September 2018

The ECCMA Open Technical Dictionary (eOTD)[9] is based on UNSPSC. However, properties were added to the classification system. eOTD is not a hierarchical classification system, it rather can be compared with a vocabulary or dictionary, in which the products can be identified unambiguously through their ConceptID. The ConceptIDs in the eOTD include the classification codes of other classification systems, such as UNSPSC or eCl@ss (see below). eOTD is defined as an XML schema and includes next to products and their characteristics also units and currencies.

The RosettaNet Standards are a collection of exchange formats that were developed specifically for the industry sectors information technology, electronics components and semiconductor manufacturing. One part of the collection is the RosettaNet Technical Dictionary (RNTD), which is used to describe semantically the characteristics, i.e., properties, of the included products. The dictionary is not hierarchical, but is composed of product classes, property definition sets, and the properties themselves. The classes are not related to each other. However, products with comparable properties can be grouped in one class. The classes can contain several property definition sets, which again contain several properties and describe a certain property area of the product in detail.

In the norm IEC 61360 [Ieci02] a Common Data Dictionary (CDD) is standardized, which is build up hierarchically. Each class contains a code, a definition, a relation to its superclass and a list of properties. The standard is the basis for several classification systems, e.g., eCl@ss. eCl@ss[10] is a wide spread classification system, which includes products and services in a hierarchical manner. It was developed by the eCl@ss e.V. organization with many members from the German industry, e.g., Festo AG & Co. KG, Phoenix Contact GmbH & Co. KG, and Siemens AG. Despite its German origin, eCl@ss is also used internationally and is available in English, French and Italian. Although eCl@ss aims at covering products from various industry sectors, it is especially detailed in the area of electrical and process control engineering, for which also other standards, such as ETIM or Prolist, were integrated into eCl@ss. The classification system is structured in a four level hierarchy (Figure 4-7). The first three levels classify the groups of products, while the fourth level contains the actual product. On this level, the properties are assigned to the products in a basic and a standard property list. The basic property list contains mainly those properties that are relevant for e-commerce (e.g., supplier name). The standard property list is used to specify the product by assigning properties from a technical perspective, e.g., supply voltage (see Figure 4-7). The properties are referenced by an identifier according to the standard ISO 29002-5 [Isoi00]. For an unambiguous identification of the products in eCl@ss, an eight-digit code, resulting from the hierarchy,

[9] http://www.eccma.org, retrieved on 23 September 2018
[10] http://www.eclass.eu, retrieved on 23 September 2018

is used. For example, the eCl@ss identification code for the inductive proximity switch in Figure 4-7 is 27-27-01-01.

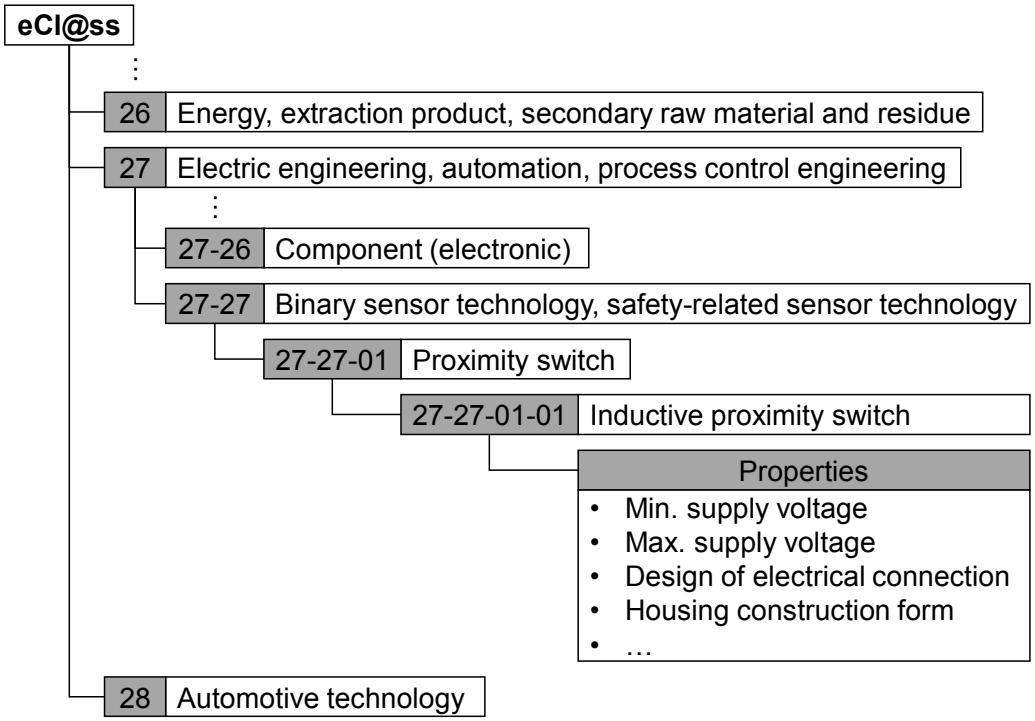


Figure 4-7: eCl@ss structure at the example of an inductive proximity switch

A classification system, which does not focus on the categorization of products for e-commerce but rather on the products' functionalities, is the NIST Functional Basis [HSMS02]. The authors state that "in engineering design, the end goal is the creation of an artifact, product, system, or process that performs a function or functions to fulfill customer need(s)" [HSMS02, p.3]. Therefore, the classification system provides a standardized functional terminology to serve as a basis for modeling. The vocabulary is based on a three-layer classification, which details the system functionalities and flows.

The exploration of classification systems shows that a wide range of standards exists and that through cooperation and links between the different standards (e.g., between eOTD and eCl@ss) many properties can be described in different classification systems. Among those, eCl@ss offers the advantage of a hierarchical and semantic system, which covers a wide range of properties and products, with a specific focus on electrical and electronic components. The level of detail of the properties and the logical structure of the product groups fits well to the required properties in engineering. Therefore, efforts are made to integrate eCl@ss, e.g., in AutomationML [GHJL15]. However, eCl@ss only contains a very limited description of the products' functionalities, which is provided much more detailed by the NIST Functional Basis. Thus, to use properties from classification systems a combination of product-related properties and a functional categorization is required.

4.2.3. Property-based approaches in engineering

In literature, several approaches aim at using classified properties in engineering. Vogwell and Culley [VoCu91] described already very early the potential of finding the 'optimal' component for a specific application from an electronic manufacturer catalog. The authors propose to list all available products and based thereon, make an automatic selection of a matching component. In order to select the best component for the particular use case, the catalog should not be limited to a specific manufacturer, but rather contain components from various manufacturers. However, product data often is provided by the manufacturers application-specifically and in a non-standardized way, e.g., in a textual way or as figures of graphs. The approach does not mention properties explicitly, but the described potentials of manufacturer-independent, categorized product data pose the basis for product classification systems as described in the chapter above. Furthermore, the approach seeks on achieving a compatibility between the requirements and the selected components based on the usage of product data.

Epple [Eppl11] developed an approach to use properties as a basis for system interoperability. Thereby, interoperability is defined as the ability of independent systems or tools to cooperate with each other. E.g., the different discipline-specific stakeholders who are involved in the development are coupled through the project but often work with independent tools to represent their views. This means that the planning, engineering, component sourcing etc. is conducted separately with different modeling languages / description forms and cannot be brought together in one semantic model. Rather a standardized property model is used to achieve a sufficient interoperability level. Thus, all models, which include properties according to the defined property model, are able to exchange this information between each other. Based thereon, the consistency of the system under development can be analyzed and it can be checked, if the components match to the defined requirements. This could also be used as a basis for conducting a compatibility analysis in case of a change. However, Epple does not define, how the property model would be integrated in a specific modeling approach, e.g., SysML, and which properties could be integrated from standardized classification systems.

Frank et al. [FEHF13] extend the V-model for the development of distributed automation systems and define thereby on the one hand a modeling approach (SysML-vAT, see chapter 4.1.3) and on the other hand define how properties, assigned to requirements and components, can be used to verify the chosen solution. The properties are also used to describe the functional and non-functional requirements of the system under development. The properties then can be used to analyze data, signal, and functional compatibility. Hadlich [Hadl15] extends the property model in order to define the properties from different modeling approaches during the development process

semantical unambiguously. Furthermore, the approach uses properties to describe the relations between different system components. Based thereon, it can be analyzed, if the properties of connected components are compatible to each other. The actual compatibility check, however, is not focused in the approach. Hadlich and Diedrich [HaDi13] define the relations between different properties more specifically and the use of property lists is proposed. These property lists can be assigned to the components in the development and the authors propose to use standardized property definitions, e.g., from the Common Data Dictionary of IEC 61360 (see chapter 4.2.2). The approach, furthermore, shows that the interaction of properties from different components can lead to 'new' properties. The 'surprising' emergence of such new properties should be avoided in engineering, as late changes in the development, due to such unexpected properties, can be very costly. However, not only 'surprising new' properties can emerge, but also existing properties can gain in importance at a certain time and thus, have to be taken into account, although they first were not relevant (e.g., energy efficiency gained very much in importance over the last years). Another reason for changing properties occurs, if different components are combined to a module. Thereby, 'new' properties with values that are a combination of the existing components' properties can emerge.

Döbrich and Heidel [DöHe13] focus on the development of cyber-physical systems (CPS) with the help of properties. In order to receive standardized property lists a combination of the IEC CDD and eCl@ss is used. Thereby, the CPS shall be modelled in a first step independently from a specific manufacturer. In a second step, the specific CPS is created through assignment of according property values. A production system is created through the combination of different components, whose properties match with each other. The properties are divided into the types Business, Construction, Function (Software), Location, and Performance according to standard IEC TR 62794. The sum of all properties represents the properties of the entire system. All property types and the relations between them are stored in a repository. Through the standardized data model of the properties, interoperability is achieved. As the components are combined to a CPS based on their properties, a compatibility check is implied by the approach. However, the concrete realization of the compatibility check is not focused by the approach. Furthermore, no modeling approach is used for the description of the properties.

The use of properties in engineering is especially important for complex systems, which require the interaction of different stakeholders. During the development and reengineering of mechatronic systems, engineers from mechanical, electrical/electronic, and software engineering have to define system components unambiguously to avoid inconsistencies and possible system failures. Thus, the integration of standardized properties into model-based approaches can reduce ambiguities and make the development process more efficient. Table 3 gives an overview of the described

approaches and their fulfillment of the requirements regarding the interdisciplinary development of mechatronic systems.

Table 3: Comparison of existing approaches focusing on properties in engineering

Reference	Integration into modeling language (req. R 1)	Mech, E/E, SW (R 2, R 3)	Consideration of module properties (R 4)	Use of standardized classification systems (R 5)	Use of properties for compatibility analysis (R 5, R 6)
[VoCu91]	○	○	○	◑	◑
[Eppll1]	○	◑	○	○	●
[Hadl15]	◑	○	○	◑	◑
[HaDi13]	◑	○	●	●	◑
[FEHF13]	◑	○	○	○	◑
[DöHe13]	○	◑	○	●	◑

● addressed by the approach; ◑ partially addressed by the approach;
○ not addressed by the approach

4.3. System modularization

For the terms module and modularity different definitions exist, depending on their application domain, e.g., software or engineering design [EbGö04, GePZ03, RoLF07]. Modularity can refer to the system itself, e.g., plug-and-play compatible modules, as well as to the organization or the design. Modularity describes the degree to which the architecture of a system is composed of modules, whose components within the module have strong dependencies and minimal interactions outside of the module [Schi00]. Components, in some approaches also called basic modules, are considered as the smallest possible system element, which is not divided further. Thus, modules are composed of several (different) components, which fulfill together a specific function, and are independent from components outside of the module. System architectures can then be composed through a combination of several different or identical modules [GePZ03]. In hierarchical module architectures, modules, furthermore, can consist of different submodules [BoFS12]. Huang and Kusiak [HuKu98] add the similarity of functional dependencies (e.g., geometrically) as a characteristic of modularity.

Katzke et al. [KaFV03] developed a module model in the area of automation for the machine and plant manufacturing industry (Figure 4-8). The model describes the different perspectives on the overall system through the module layers *basic module*, *application module*, and *plant/project module*, considering, e.g., their flexibility, the effort to manage variants, and their complexity [KaFV03]. Furthermore, the layers differ on the one hand in the possibility to reuse the contained modules and on the other hand on their degree of transparency (see left side in Figure 4-8). Basic modules provide fundamental functionalities and can be used for all industry sectors. They are encapsulated (black box) and parameterizable. Basic modules are defined as atomic basic modules, if they represent components on the most fine-grained layer and are not further decomposable. Application modules, in contrast to the basic modules, are characterized specifically for an industry sector through different requirements. By developing different variants of the modules, they can be applied to different projects. Application modules can be composed by several basic modules and are parameterizable as well. Plant/project modules are used to manage the complexity of the overall plant by dividing it in several sub-modules, representing parts of the plant. This is usually done through a Top-Down approach. The modularization in plant/project modules ends, if the respective modules can be composed through application modules in the further refinement. However, the reusability of plant/project modules is very limited, as plants usually are developed customer-specifically. The structure of the modules on the other hand is very transparent with application modules as smallest considered artifacts on this level. The modules on all levels can consist of automation hardware, PLC software, HMI and interfaces to higher-level systems. The modules on each level are recursive, e.g., application modules can contain further application modules [KaFV03].

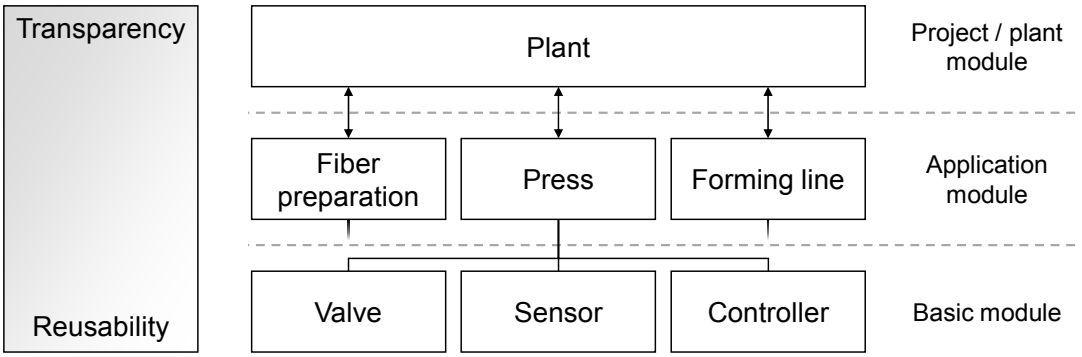


Figure 4-8: Three-level module model at the example of a fiberboard production system, according to [KaFV03]

4.3.1. Goals of modularization

Modularization can be considered from various points of views and thus, pursue different goals. Predominantly modularization supports the management of complexity in a

system, an improved extensibility and maintainability, a higher reusability and the development of system variants [GePZ03, MaJG11a, Ulri95]. Further advantages of modularization can lie in a cost reduction, a shortened development and set up and thus, a faster time-to-market [GHSR97]. Furthermore, lifecycle oriented aspects like recycling [GePA99] can be a goal of modularization efforts. In order to use modularization successfully and improve the development process, an adaption of the organizational structures is required [Oost01, PiEp94].

In mechanically dominated development approaches, modules are either defined during a new development according to a specific product function [Ehrl09, Ulri95] or, based on an analysis of the existing system, through regrouping of the components [PiEp94]. In both cases, at the beginning of the analysis, a subdivision of the considered system in subsystems and single components is conducted. Following this decomposition, the interrelations between the different system components are identified, and thus, the components are clustered to modules [BrEp02, PiEp94, StWC00]. According to Zhang and Gershenson [ZhGe03] the considered abstraction layer of the system during the modularization process, influences the modularity itself. Thus, various different views, such as customer requirements, functional structures, components and other (sub-) modules, have to be considered and harmonized in the modularization process [MuRo02, PiEp94].

4.3.2. Granularity of modules

Based on the modularization goals, the appropriate degree of modularization has to be defined. Thereby, the creation of reusable modules should take place independently of a specific project. According to Ulrich [Ulri95] the question, which degree of modularization is optimal, has to be answered individually, considering the resulting impact on the quality of the system and the development task. For the development of mechatronic systems, this question has to be answered, furthermore, for all involved disciplines. Jazdi et al. [MaJG11a] point out the necessity to define modules with an appropriate granularity in order to assure their reusability. Their analysis of an adequate granularity level for automation shows that different levels are required to enable on the one hand a high reusability in the different disciplines and on the other hand support domain and application engineering [MaJG11a]. As indicators for the level of granularity, the authors propose to analyze the module size in comparison to the entire system size and the degree of detail of the respective model. However, explicit metrics to achieve a module with an appropriate granularity level were not developed. Figure 4-9 shows the resulting conflicting aspects during the definition of the module granularity. Modules with a fine granularity lead to an increased reusability and a higher flexibility, coarse-grained modules on the other hand lead to an improved efficiency and robustness. Thus,

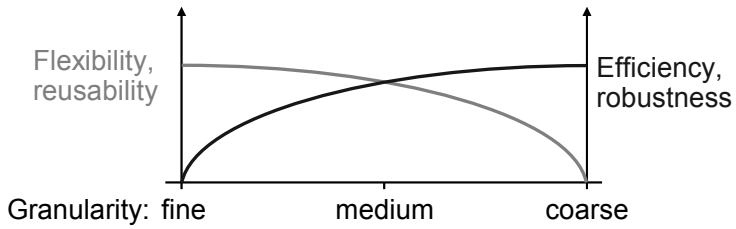


Figure 4-9: Conflicting aspects of module-granularity, according to [MaJG11a]

no universally valid level of granularity can be defined. Rather, the size and granularity of the modules is highly dependent on the plant structure, application domain, and company decisions.

4.3.3. Modularization-Based approaches

Lüder et al. [LEHM11] focus on an integrated, modular engineering approach using embedded mechatronical units to foster modularity and reusability. The mechatronical units contain "physical, electrical, and control components" [LEHM11]. The authors state, furthermore, that the mechatronic units can be seen not only as component but also as a container that consists of all required information from the different disciplines, e.g., functional, control, or technical data. Similar to Vogel-Heuser et al. [VFST15], the approach proposes a mechatronic engineering process containing customer-independent activities as well as customer-oriented tasks for a specific project. To transform and exchange the information between different modeling languages during the engineering process, the authors propose to use AutomationML and PLCopen XML as exchange format. Therefore, a meta-model, called intermediate modeling layer (IML), is defined for the required mappings. Therein, e.g., the SFC is mapped according to the IEC 61131-3 [Ieci14a]. The common, interdisciplinary model in the presented approach, however, focuses only on the transformation and exchange. An interdisciplinary, graphical modeling approach, which enables the visualization of modules, containing components from the different involved disciplines, is not focused by the approach.

Kuehnl and Grabbe [KuGr06] focus on the question, how a mechatronic modularization of the system and a mechatronic structure of the company in the machine manufacturing industry can be achieved. The authors state that a successful mechatronic modularization requires the collaboration of different disciplines and new development processes in order to receive a synchronized solution. The approach proposes self-contained mechatronic modules, which contain all required components within the module and the control panel. For such a mechatronic solution, the same module border in each discipline is mandatory.

As plant manufacturing usually is carried out very customer specific and thus, the quantities are not very large, it is advisable to leave parts of the module variable, in order to adjust it to different tasks [Poss10]. However, if a module is adaptable, e.g., by means

of electronic components, the software must be capable of being parameterized variant-specifically. Possel-Dölken [Poss10] raises two questions regarding the tool support of the engineering process:

- How can the developed modules with all their relevant information be stored and managed appropriately?
- How can the developed modules be presented to the developer at the right time and in an optimal way, in order to ensure implicitly that the developed standards are met?

The author shows in an analysis of leading product data management-systems (PDM-systems) in Germany that the mechatronic structure can be stored insufficiently with existing solutions. While the integration of the data structure from mechanical CAD-systems (MCAD) is already fairly advanced, deficits exist in the integration of the data from electrical CAD-systems (ECAD). Similarly, the integration of software development systems is not yet supported sufficiently. In order to use mechatronic modules efficiently, the PDM-systems would have to support an adequate search functionality fulfilling the following requirements [Poss10]:

- A coherent categorization and classification of the modules in a library
- An integrated search of product data, library elements and projects
- An integration of purchased and self-developed components
- A clear structuring of the found results, e.g., according to their discipline
- If an engineering system is coupled to a PDM-system, the search functionalities should be directly accessible from the engineering system

Although the described requirements for a tool support are not yet solved sufficiently, the authors in [KuGr06] and [Poss10] state that a mechatronic modularization poses a possibility for a structured development process and collaboration of the different involved disciplines. Thus, mechatronic modularization also requires new development processes and organizational structures. The application of mechatronic modules changes the development process from ‘engineering’ towards ‘tailoring’ the modules. Through the known and assured quality of the technical solution through the application of developed modules, the development and commissioning time can be reduced and the planning reliability, e.g., regarding the delivery time, can be increased. However, the authors state that the proposed approach with self-contained mechatronic modules can result in higher costs for the included components in comparison to an individual solution. Thus, for the application of the proposed approach, companies have to decide, if such a modularization can be implemented economically for their systems. Through the changed engineering process adequate approaches are necessary, which enable an interdisciplinary model-

based development of the mechatronic modules and of the overall system. However, these are not focus of the proposed approach in [KuGr06] and [Possl0].

Schmertosch [Schm11] proposes the combination of modular mechanics, compatible electrical/electronic hardware and object-oriented programming as basis for flexible, modular machines. The required additional effort in the beginning to develop the modular machine concept, is compensated rapidly, especially for complex systems. As criteria for the modularization, the author proposes the manufacturing sequence, customer requirements, modes of operation (e.g., customer requires possibility for own machine reconfiguration), services or the possible future development and modernization of the system. Regarding the project planning in a software tool, Schmertosch [Schm11] proposes to structure the system in groups according to their functionalities. Through different hardware configurations, varying modules can be created within one project. However, the approach does not consider the interdisciplinary dependencies, and furthermore, does not focus on a method to develop the structure of the modules before it can be included in the software tool.

An approach, which focuses on the identification of potential modules in existing systems to enable an efficient reuse is proposed by Weyrich et al. [WKLW11a, WKLW11b]. The modularization approach is based on a Design-Structure-Matrix (DSM) [PiEp94], in which the components (physical as well as software components) of a system and their relations are represented. Through clustering of the matrix, the system components, which have tight relations, are combined to modules. Thus, function fulfilling and connecting modules are generated. The modules, which are identified in the first step, refer to the analyzed system. In order to enable an adequate reuse various systems should be analyzed. Thereby, the size and structure of the modules have to be adapted, if necessary, to increase the possibility of reuse. The described approach is suitable to identify reusable (mechatronic) modules in existing machines/systems, in which the relations of the components are known explicitly. However, in order to use this approach for the engineering of new projects, the interfaces of the modules have to be specified more precisely in the different involved disciplines to ensure the interconnectivity. In a cooperation with the author of this thesis, a combination of the approach with a SysML model was presented in Kernschmidt et al. [KKJG13].

Next to the described challenges regarding the granularity of modules (cf., chapter 4.3.2), with the result that different levels of granularity are necessary [MaJG11a], Jazdi et al. [MaJG11b, MaJG11a] propose to divide the engineering in project dependent and project independent tasks in order to enable a systematic reuse. During the domain engineering (project independent tasks) an analysis, e.g., regarding module boundaries, and the design of reusable modules (independent from a specific customer order) is conducted. These

modules are stored in a central library (domain repository), which serves as interface between the domain engineering and the application engineering. During the application engineering, the developed modules are selected customer specifically and, if necessary, are adapted accordingly. The experience of the development and application in the customer specific projects is played back to the domain engineering in order to optimize the modules. The proposed approach shows that a systematization of the modules during domain engineering can comprise advantages, such as an improvement in efficiency, an increased technical maturity of the modules, a cost reduction and a minimization of the risk to handle a project. However, an according tool support for the interdisciplinary engineering of the modules was not focused by the approach.

Table 4 summarizes the described approaches, which focus on modularization of complex (mechatronic) systems and rates them according to the fulfillment of the requirements of this thesis (cf., chapter 3.2).

Table 4: Comparison of existing approaches focusing on modularization

Reference	Consideration of interdisciplinary modules (req. R 2, R4)	Functional modularization (R 4)	Integration of (standardized) properties (R 5)	Integration into modeling language (R 1, R 2, R 3)
[LEHM11]	●	◑	○	◑
[KuGr06]	◑	○	◑	○
[Poss10]	●	○	◑	○
[Schm11]	◑	○	○	◑
[WKLW11a] [WKLW11b]	●	◑	○	○
[MaJG11a] [MaJG11b]	◑	◑	○	●

● addressed by the approach; ◑ partially addressed by the approach;
○ not addressed by the approach

4.4. Summary and research gap

Various MBE-approaches for the development of mechatronic (production) systems have been developed. In many cases, standardized modeling languages (see also requirement

R 1) are used as basis of the approaches. Especially UML and SysML-based languages are used for approaches with a graphical visualization. If the exchange of information between different engineering tools is focused, AutomationML and XML-based formats are common.

The description of a system under development through respective properties has been focused already for a long time. Through the usage of standardized classification systems, e.g., eCl@ss, an unambiguous description of the properties is possible. However, not all industry sectors are covered by such classification systems. Furthermore, individually developed components as well as specifically defined modules are not included. The further usage of properties, as basis for analyzing compatibility, is focused so far only by a few approaches. In addition, the integration of standardized classification systems, especially in the domain of mechatronic production systems, in model-based approaches has not been exploited very much so far.

A third crucial aspect in the development of mechatronic systems is modularization. Through an appropriate modularization, the reuse of developed (and tested) solutions is fostered, which leads to a shorter time-to-market, retaining the desired quality level. For

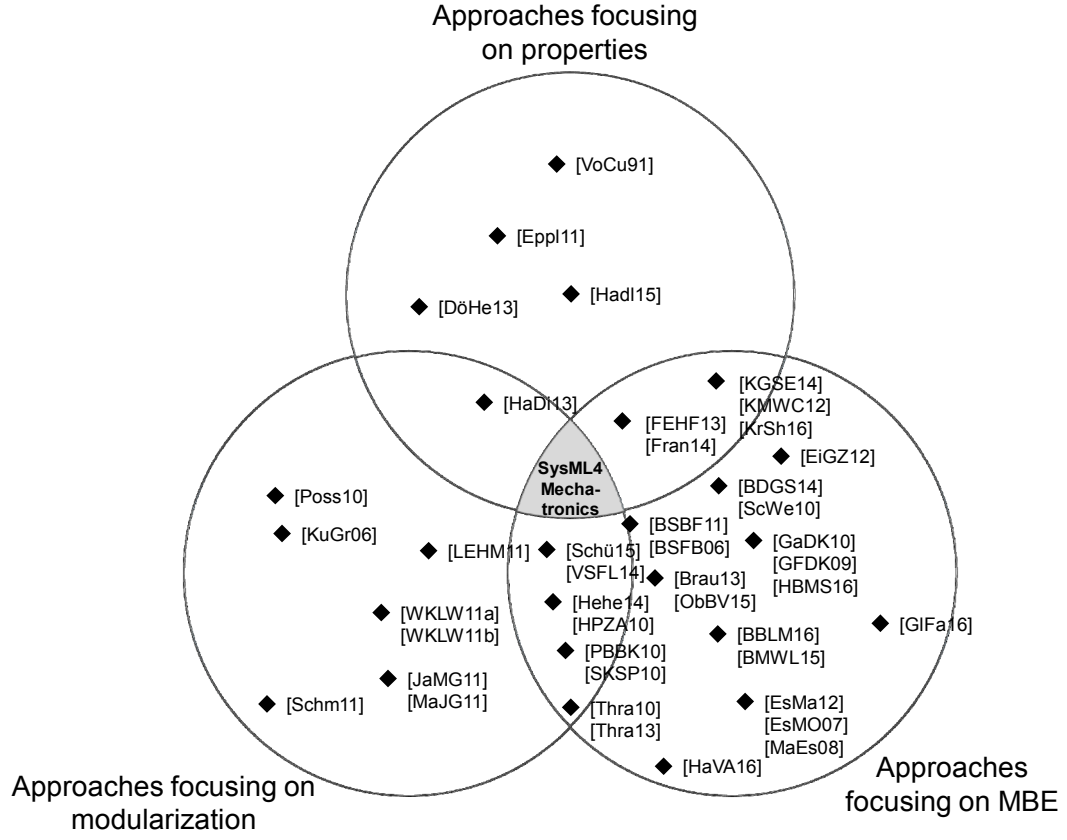


Figure 4-10: Comparison of existing approaches and resulting research gap in MDE of mechatronic systems

mechatronic systems, the usage of self-contained mechatronic modules, with the same module border in all disciplines, is proposed by many approaches. In contrast, up to now approaches that focus on a function-based modularization in the models and a flexible implementation in the real system (no 1:1:1 modularization) are rather rare.

Hence, the resulting research gap in the intersection of these fields of interest for the interdisciplinary (re-)engineering of mechatronic production systems is depicted in Figure 4-10. Based thereon, the MBE-approach SysML4*Mechatronics* is developed in the following chapter.

5. Concept of the interdisciplinary MBE-approach SysML4*Mechatronics*

In this chapter, the concept of SysML4*Mechatronics* is presented and the different parts of the modeling approach are described in detail. In chapter 4, current approaches were analyzed, based on the defined requirements presented in chapter 3, and the existing research gap was identified. The approach of this thesis – called SysML4*Mechatronics* – is based on the standardized modeling language SysML [Omgo15b]. Therefore, a respective SysML-profile for the SysML4*Mechatronics* metamodel was developed (requirement R 1), which is presented in chapter 5.1. It includes the components from the different involved disciplines (req. R 2), their properties (req. R 5), interfaces and relations (req. R 3), as well as the possibility to form function-oriented interdisciplinary modules (req. R 4). Chapter 5.2 shows how the SysML4*Mechatronics* profile is visualized. In order to facilitate the engineering process, the required properties of the components and modules shall be included, if possible, from existing classification systems (req. R 5). Thus, chapter 5.3 presents how according model libraries can be used in the modeling approach.

5.1. SysML4*Mechatronics* metamodel

In order to develop a customized modeling approach for a specific application area, i.e., mechatronic production systems in this thesis, UML and SysML offer the possibility to create *profiles*. The mechanisms of the profiles packages can be used to extend the existing metaclasses from the metamodels [Omgo15a, Omgo15b]. This adaption for the creation of a specific modeling approach is conform to the OMG MOF (cf., Figure 4-5) [Omgo15b].

Thereby, stereotypes, next to constraints, are the main mechanism for the creation of specific profiles. The defined stereotypes extend the SysML metamodel and can then be used in the model of the developed profile (e.g., respective stereotypes for the discipline-specific components are defined in chapter 5.1.1). Note that SysML itself contains stereotypes, which extend the UML metamodel. Figure 5-1 depicts how the developed SysML4*Mechatronics* profile is defined. The profile imports the SysML profile in order to use and extend the existing SysML stereotypes. SysML in turn is based on the UML metamodel and extends the *StandardProfile* of UML 2 [Omgo15b]. Next to the extension of SysML stereotypes, the SysML4*Mechatronics* profile can additionally extend UML metaclasses, if required.

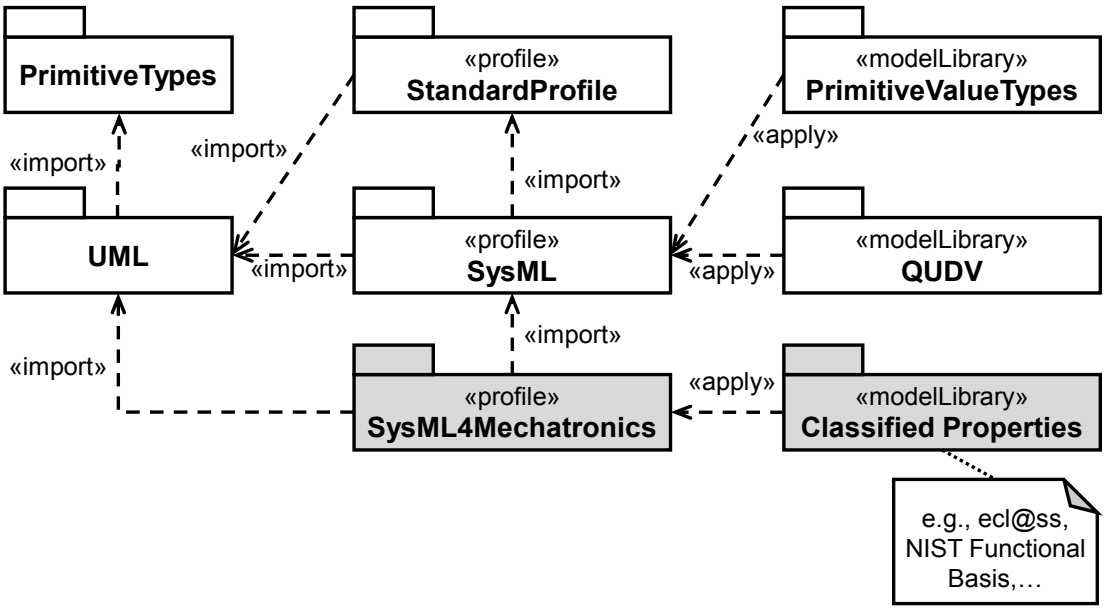


Figure 5-1: SysML4Mechatronics profile in the context of SysML and UML, based on [KeFV18]

Figure 5-1 shows, furthermore, on the right hand side, three model libraries called *PrimitiveValueTypes, QUDV,* and *Classified Properties*. Model libraries are collections of reusable modeling elements, such as blocks or type definitions, for a certain modeling domain. Such libraries are used, next to the specific profiles, to customize SysML for a certain domain, e.g., automotive or mechatronics [Omgo15b].

The library *PrimitiveValueTypes* is part of the SysML specification [Omgo15b] and defines the value types *Boolean, String* and the number types *Integer, Real* and *Complex*. Although UML defines already primitive types and thus, SysML could reuse them, they are provided in an own SysML library in order to enable engineers to extend them more easily through stereotypes, if required.

The model library for *Quantities, Units, Dimensions, and Values (QUDV)* is a non-normative collection provided by SysML to specify quantities, units, dimensions and values more comprehensible. For every development model, a solid foundation of clearly defined quantity and unit properties is crucial, especially considering possible inconsistencies with severe consequences that can arise from unit faults [FeKV16]. The QUDV library is based on the most wide spread quantity and unit systems, i.e., the International System of Quantities (ISQ) and the International System of Units (SI) [Omgo15b], which are standardized through the norm ISO/IEC 80000-1:2009 [Isoi09]. However, not for all applications and not in all countries these units are used, e.g., in North America the US customary system, based on the British Imperial units, is commonly used. Therefore, the QUDV library includes mechanisms to add such quantities and units that are not conform to ISO 80000 and define their relationships, i.e.,

conversion rules, explicitly. For example, the *LinearConversionUnit* can be used to convert foot to the SI reference unit meter through a linear conversion relationship (by multiplying foot with the factor 0.3048). The ValueTypes, defined in the SysML4*Mechatronics* profile (cf. chapter 5.1.3), can make use of the QUDV library for the specification of their units and quantities.

The third model library in Figure 5-1, called *Classified Properties*, is defined in addition to the SysML4*Mechatronics* profile in order to include properties from standardized classification in the developed modeling approach. For each classification system, e.g., eCl@ss or the NIST Functional Basis (cf. chapter 4.2.2) a sub-model library should be defined. Thus, the classified properties can be used during modeling by applying the respective model library. In chapter 5.3 the model library and its application to SysML4*Mechatronics* are described in detail.

5.1.1. Components and modules

The core concept of SysML4*Mechatronics* is the integration and visualization of the components from the different involved disciplines in an integrated model of the mechatronic system (req. R 2). In order to model the different discipline-specific components in the SysML4*Mechatronics* profile the SysML *block*, which represents the basic concept to model a certain artefact, was extended. Thus, *block* is stereotyped to the abstract *SysML4MechatronicsBlock*, containing the basic information of all development artifacts of the mechatronic system (Figure 5-2). Based thereon, the modeling elements for *MechanicalBlock*, *EEBlock*, and *SoftwareBlock* are defined, which inherit from *SysML4MechatronicsBlock*. Hence, the engineers from all disciplines can add their relevant elements within the joint model, representing the different aspects of interest on the system elements.

Next to the discipline-specific components, the stereotype *Module* is defined in the metamodel (req. R 4). As described in chapter 4.3, the use of modules is aspired by most companies in consideration of managing complexity, enabling reusability and reducing costs. Thereby, a 1:1:1 relation between mechanics, E/E and software often is not applicable in the field of mechatronic production systems [FeFV12] and a modularization defined by the mechanical structure or as mechatronic objects, including the components from all disciplines, often is not desirable and affordable. Rather, within the engineering process, modules should be considered as development artifacts that can easily be reused within the development of variants, versions or new systems. The SysML4*Mechatronics* stereotype *Module* incorporates this approach. As shown in Figure 5-2 (through the ♦ relations), *Modules* integrate different discipline-specific components, which are necessary for the functionality of the module, and can contain, furthermore, sub-modules.

In this way, a hierarchical system structure can be modeled. Equivalently to UML and SysML, the modules and discipline-specific blocks depict the pattern of an object, which they represent, and thus, can be saved in a library (cf., a construction design catalog). Specific instances thereof can then be used in the particular development project.

Regarding the metamodel definition, Figure 5-2 shows the two possible mechanisms for creating new stereotypes. On the one hand, stereotypes can be created through *extension* of an existing metaclass. The stereotype *SysML block* is created by extending (represented through an arrow with a line and a filled triangle) the metaclass *UML class* (UML4SysML represents the subset of UML, which is reused for SysML). On the other hand, an *inheritance* relation can be used for the creation of stereotypes. In Figure 5-2, the *SysML4MechatronicsBlock* stereotype was defined as sub-stereotype of the *SysML Block* stereotype. Thus, *SysML4MechatronicsBlock* inherits all properties from its super-stereotype *SysML block* and extends it [Omgo15b]. *SysML4MechatronicsBlock* is an abstract block (depicted through an italic font in Figure 5-2), which means it can not be instantiated directly during modeling [Omgo15a]. Rather, its concrete sub-stereotypes are used, i.e., the stereotypes for discipline-specific components *MechanicalBlock, EEBlock* and *SoftwareBlock*, as well as the *Module* stereotype.

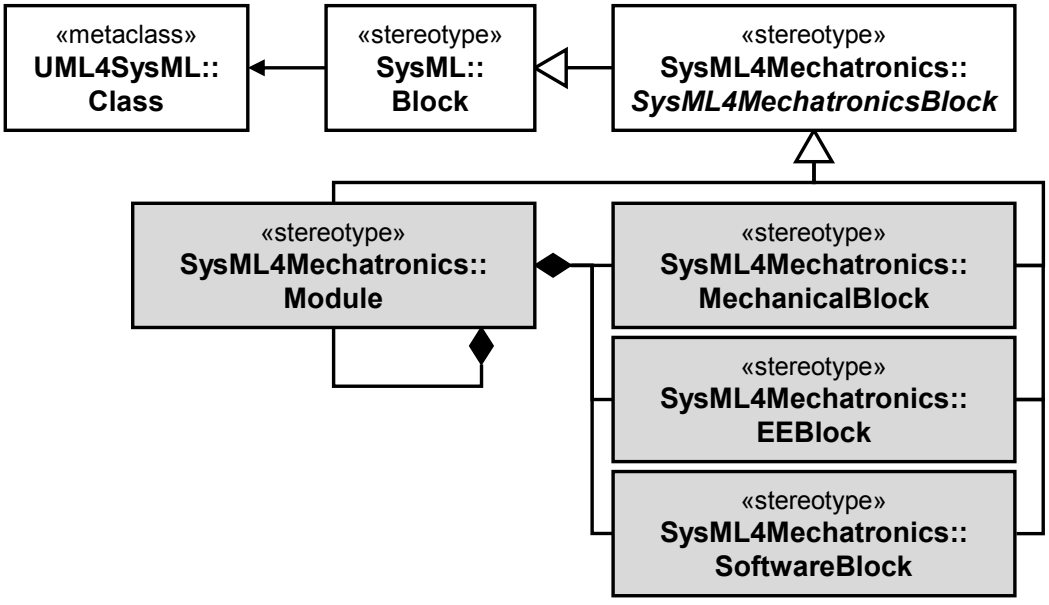


Figure 5-2: SysML4Mechatronics metamodel – components and modules [KeFV18]

5.1.2. Ports and relations

The blocks and modules of a system can not be considered isolated, as they are usually connected with other blocks or modules and interact with each other. Thus, their interfaces have to be taken into account and modeled appropriately. SysML offers therefore the concept of ports, more specifically in SysML 1.4 [Omgo15b] the stereotypes

ProxyPort and *FullPort* are defined. While the *FullPort* represents a separate element from the owning block [Omgo15b, p.85], the *ProxyPort* is defined as interface from the owning block, which makes certain features of this block available to external blocks. As the *ProxyPort* depicts an interface of the owning block and is not a separate element, it must be connected to internal parts to be completely specified. However, specified models can be used for this detailed 'internal' design [Omgo15b, p.87]. For example, if a port represents the interface for the power connection of a motor, the internal wiring and electrical allocation is not relevant for the mechatronic engineering and can be defined in a separate model (or is not necessary as it is, e.g., a purchased part). However, the specification of the interface, e.g., a required 12V DC connection, is relevant for the development of the overall system.

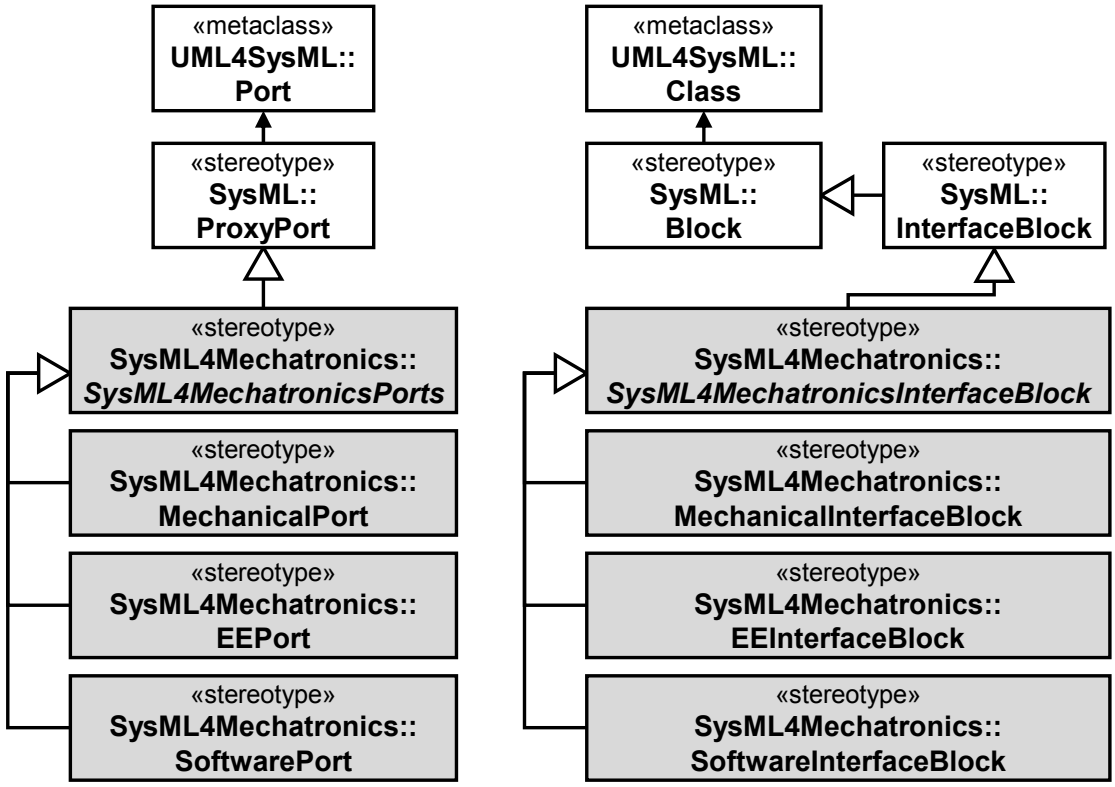


Figure 5-3: SysML4Mechatronics metamodel – ports and interfaces [KeFV18]

Thus, this basic concept of the *ProxyPort* was extended (via the abstract *SysML4MechatronicsPort*) to *MechanicalPort*, *EEPort*, and *SoftwarePort* (see Figure 5-3). By that, the described interfaces of each discipline can be considered specifically. A *MechanicalBlock* can be equipped thereby with one or more *MechanicalPort*s and a *SoftwareBlock* with one or more *SoftwarePort*s. *EEBlock*s – representing usually the sensors and actuators of a system – serve as 'linking pins' between the disciplines, as they are physical components, can communicate with the control and are represented in the software (e.g., as a variable representing a sensor value). Thus, the *EEBlock*s can contain *MechanicalPort*s, *EEPort*s, and *SoftwarePort*s. Similarly, *Module*s, which are a

combination of different discipline-specific blocks and sub-modules, can contain ports from all three disciplines.

In order to implement a specific functionality of a block or module, the required ports can be defined as mandatory. The mandatory ports define those ports that have to be connected to other ports in the final system, in order to enable the desired functionality. For example, the power connection of the motor described above has to be connected mandatorily in order to enable the functionality of the motor. In the context of modules, the definition of ports as mandatory is required to overcome a 1:1:1 modularization, as further components, next to module-specific ones, can be required within the entire system. This is the case for components, which ensure the functionality of different modules and thus, are not part of one specific module. For example, the valves of several modules can be connected mechanically in one valve cluster outside of a specific module. Therefore, module-ports can be defined as mandatory to specify the necessary connections over module borders. Note that SysML also defines a stereotype «optional» in the context of activity diagrams, which defines the multiplicity of parameters in activities. Although it is independent from the described concept of mandatory ports for SysML4Mechatronics blocks, it pursues a similar goal, i.e., if a parameter is stereotyped as «optional» it is not required to have a value in order to begin the execution of the activity [Omgo15b].

Each *ProxyPort* in SysML is typed through an *InterfaceBlock* (a specialized form of SysML *Block* [Omgo15b]). The *InterfaceBlock* defines the properties and operations of the corresponding ports. Thereby, several ports can refer to the same *InterfaceBlock*. Equivalent to the extension of the SysML ports in SysML4*Mechatronics*, the *InterfaceBlock* was extended (again via the abstract *SysML4MechatronicsInterfaceBlock*) to the specific interface blocks *MechanicalInterfaceBlock*, *EEInterfaceBlock*, and *SoftwareInterfaceBlock*. The SysML4*Mechatronics* interface blocks define the type of the port, i.e., name, direction, and unit, in each discipline. Thus, each port and its properties can be specified by the respective discipline.

5.1.3. Properties and functionalities

Each component or module of a system can be characterized through its properties [Hadl15]. As described in chapter 4.2.1, the properties can be used to classify the component and module and to analyze if a requirement can be met. Furthermore, the unambiguous description of the system properties is a prerequisite to conduct a compatibility analysis [FeKV14]. In order to describe the properties in the model, UML defines the *Property* which represents the attributes of a *class* [Omgo15a]. Thereby, the properties represent the state of a class related to a specific value. SysML reuses the

properties consistently to UML and recognizes in general four types of properties: references, parts, value properties, and constraint properties [Omgo15b]. SysML4*Mechatronics* extends (via the abstract *SysML4MechatronicsProperty*) this approach to differentiate a *ClassifiedProperty* and an *IndividualProperty* (Figure 5-4). Hence, properties from standardized product classification systems can be defined in the SysML4*Mechatronics* metamodel specifically. The usage of classified properties in the modeling approach is presented in chapter 5.3 in detail.

However, not all properties are classified in according classification systems. Thus, the *IndividualProperty* stereotype in the SysML4*Mechatronics* metamodel offers the possibility for the engineers to add further properties individually and include them in their model. In this way also a standardized component from a product classification system can be added as a block to the model with its classified properties and additional properties, e.g., a company specific identification code, can be added.

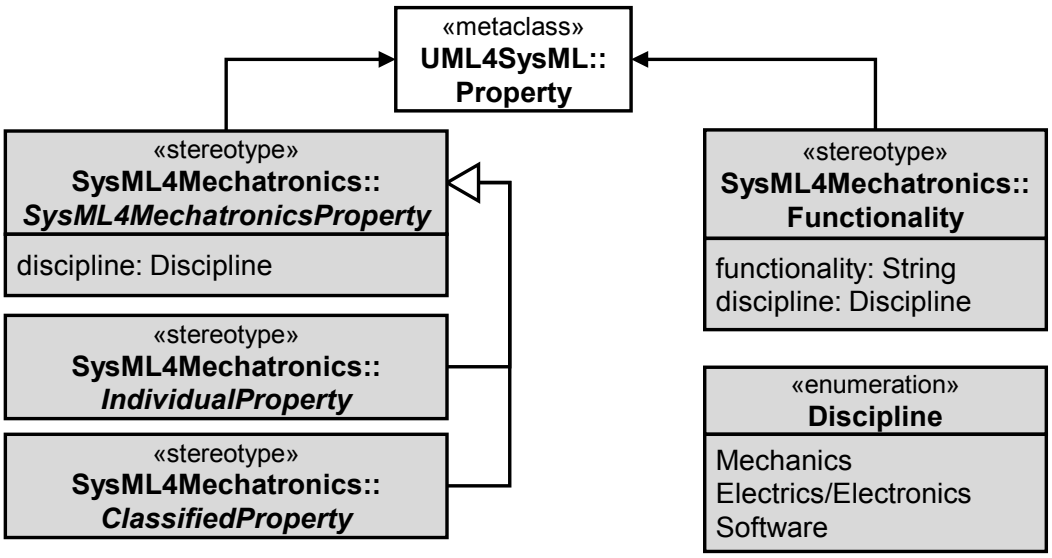


Figure 5-4: SysML4Mechatronics metamodel – properties and functionalities

Next to the SysML4*Mechatronics* properties, which define a certain characterizing aspect of the block or module, the stereotype *Functionality* is defined. The functionalities describe the interaction of the blocks or modules with the technical process, e.g., *Move workpiece*. Functionalities are independent from the specific block or module, which fulfills the functionality. Thus, different blocks or modules can fulfill the same functionality. For example, a cylinder and a conveyor fulfill the functionality *Move workpiece*. Through the *ownedFunctionality* relation, the functionalities are linked to the blocks or modules, which can fulfill the respective functionality.

Functionalities describe in a textual form, which behavior a block or module fulfills in the system. Thus, the functionalities can be used to categorize blocks and modules in a library according to the respective system functionalities that they fulfill. If a block or

module has to be exchanged, the engineers can search for other solution possibilities with the same functionality or comprise a compatibility check of an exchanged component regarding its functionality. As the functionalities are described in a textual form, it is necessary to ensure that the description is provided in an unambiguous way. The NIST Functional Basis [HSMS02] (cf. chapter 4.2.2) provides an extensive collection of classified functionalities, which can be used for the description. If detailed behavioral compatibility aspects shall be analyzed, behavioral models, such as SysML *State Machines* or *Activities,* and appropriate model checking methods have to be used, which, however, are not focused by SysML4*Mechatronics*.

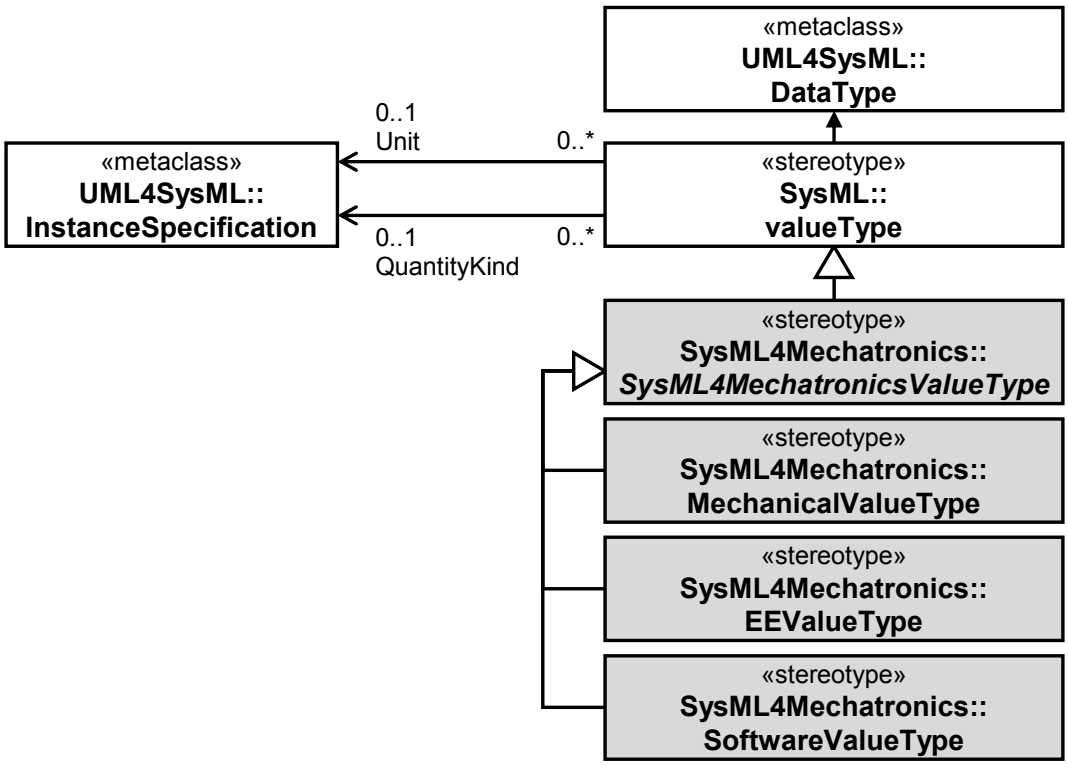


Figure 5-5: SysML4Mechatronics metamodel – value types

As described by Epple [Eppl11], properties are defined by a specific value (chapter 4.2.1). To describe the values that can be used in the model SysML provides the stereotype *valueType*. *valueType* is based on the UML *DataType*, but defines values in a more neutral form than UML, e.g., the SysML value "Real" does not necessarily have to express the concrete data representation form in software but rather relates to the mathematical concept [Omgo15b]. The *valueType* can type properties from, e.g., *block*s or *InterfaceBlock*s. The *valueType*s can be further specified through *units* and *quantity kind*s. A *quantity kind* defines thereby a certain physical quantity, e.g., length or power. Its value may be defined by a specific *unit*, e.g., meter or watt. The quantity kind and unit concepts can be used consistently across models and are usually captured in a model library [FrMS15], e.g., the QUDV model library (see Figure 5-1).

SysML4*Mechatronics* extends the valueType concept to allow the involved engineers to specify their values accordingly. As discussed above, a value definition can differ between different disciplines. Thus, they can be defined in SysML4*Mechatronics* through the stereotypes *MechanicalValueType*, *EEValueType*, *SoftwareValueType* (inheriting from the abstract stereotype *SysML4MechatronicsValueType*) accordingly. As units and quantities are defined generally and independent from a specific discipline, they were not changed from SysML for the SysML4*Mechatronics* approach.

5.2. Concrete Syntax of SysML4*Mechatronics*

As described in chapter 2.3, a modeling approach requires, next to the specification of its metamodel (cf. chapters 5.1.1 – 5.1.3), a concrete syntax to define in which way the different modeling elements are represented. However, in comparison to the metamodel definition there is no such standard format for the concrete syntax [Baar06]. For a visual modeling approach – e.g., for SysML4*Mechatronics* – the concrete syntax is realized through a graphical representation of the modeling elements. The definition of the visual language requires thereby in a first step the identification of all attributes for the visualization (e.g., shape, color) and then the formulation of well-formedness rules, i.e., when the graphical representation is syntactically correct. The application of the visualized modeling elements according to the well-formedness rules is conducted in *diagrams* [Baar06].

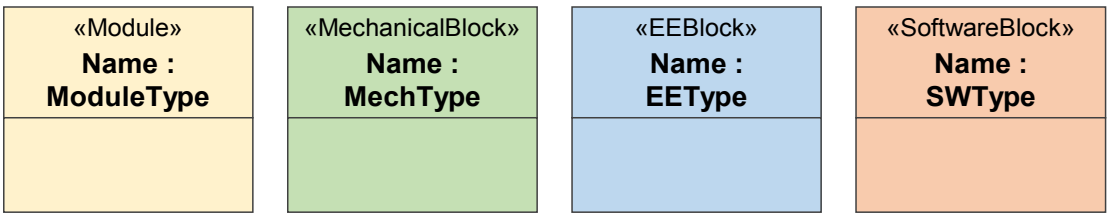


Figure 5-6: Concrete Syntax of SysML4Mechatronics blocks and modules

SysML4*Mechatronics* is based on the same principles as SysML for the visualization. The basic visualization form in SysML and thus, in SysML4*Mechatronics*, for blocks is a rectangle, divided in to several compartments [Omgo15b]. Figure 5-6 shows the SysML4*Mechatronics* visualization form for the discipline-specific blocks and the interdisciplinary modules. In the first compartment, equivalent to SysML, the applied stereotype, e.g., *Module*, is written in guillemets (« »). Thereunder the name and the type is specified, whereat the type defines the class (in terms of object orientation) and the name defines the instance of the block. Although the different block types and modules could be distinguished by the textual labels of their stereotype, a visual variation of the different blocks is beneficial for a faster differentiation. According to Moody [Mood09], the cognitive effectiveness, i.e., the speed, ease, and accuracy to interpret a representation,

should be optimized through an appropriate notation. Thus, the visual variable *color* was chosen to distinguish the different discipline-specific blocks and modules (as depicted in Figure 5-6) efficiently. Differing colors pose one of the most effective ways for distinction by humans (up to three times faster than shape differences) [Winn93].

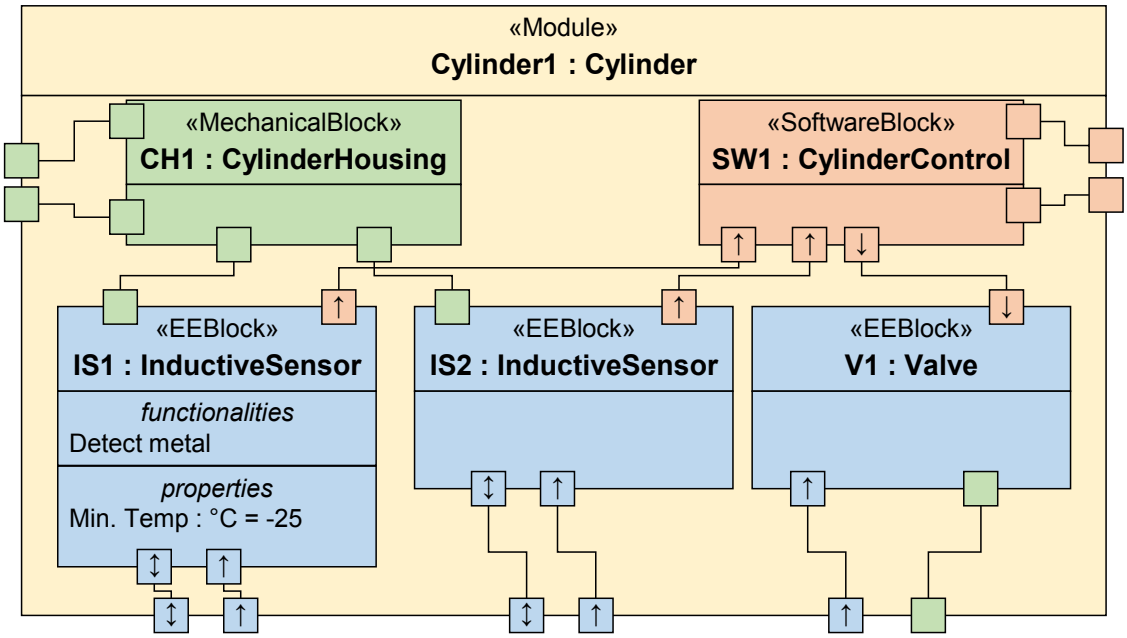


Figure 5-7: SysML4Mechatronics visualization at the example of a cylinder module

Figure 5-7 shows the concrete syntax of SysML4*Mechatronics* for a more complex model. The example represents a cylinder module with several discipline-specific blocks. To show the relationships between the blocks and the modules, i.e., that the blocks are part of the module, a spatial arrangement (blocks inside the module) was chosen, as this form of visualization is easy to interpret by humans [Mood09].

The ports of the blocks and module are visualized, similar to SysML [Omgo15b], through rectangles on the border of the owning block/module. Thereby, the color-coding for the ports of the different disciplines is defined correspondingly to the block colors of the respective discipline (mechanics: green, electrics/electronics: blue, software: orange). If a specific port direction (in, out, in-out) is defined in the interface block, an arrow, representing the respective direction, is added to the port visualization (→, ←, ↔).

For example, the valve of the cylinder in Figure 5-7 has two directed ports. Next to the power port (e/e), also the software port has a direction as it represents the variable of the valve, which is set to open it. In the compartments underneath the name of the block, further information can be presented. For example, the inductive sensor IS1 has compartments for *functionalities* and *properties*. Note furthermore, that two inductive sensors are integrated in the cylinder, i.e., IS1 and IS2, which represent two instances of

the same class *InductiveSensor*. For IS2 the compartments with further information were suppressed, in order to reduce the complexity of the model [Omgo15b].

5.3. Integration of properties from standardized classification systems

The integration of properties in the model is crucial for the description of the system and as a basis for a compatibility analysis in case of a change (req. R 5, R 6). However, if such properties are included manually, the process can be very time consuming and error-prone [Hadl15]. As more and more standardized property classification systems exist, which could be used to integrate the properties in the SysML4*Mechatronics* model directly, this chapter shows, which properties are relevant to integrate in the model (chapter 5.3.1), and then describes the integration into SysML4*Mechatronics* via model libraries (chapter 5.3.2).

5.3.1. Identification and classification of relevant properties

In order to decide, which properties should be integrated from product classification systems, it is necessary to analyze, which properties are relevant for the analysis of compatibility. Especially mechatronic systems are very complex systems with many components and thus, contain a multitude of different properties [Eppl11].

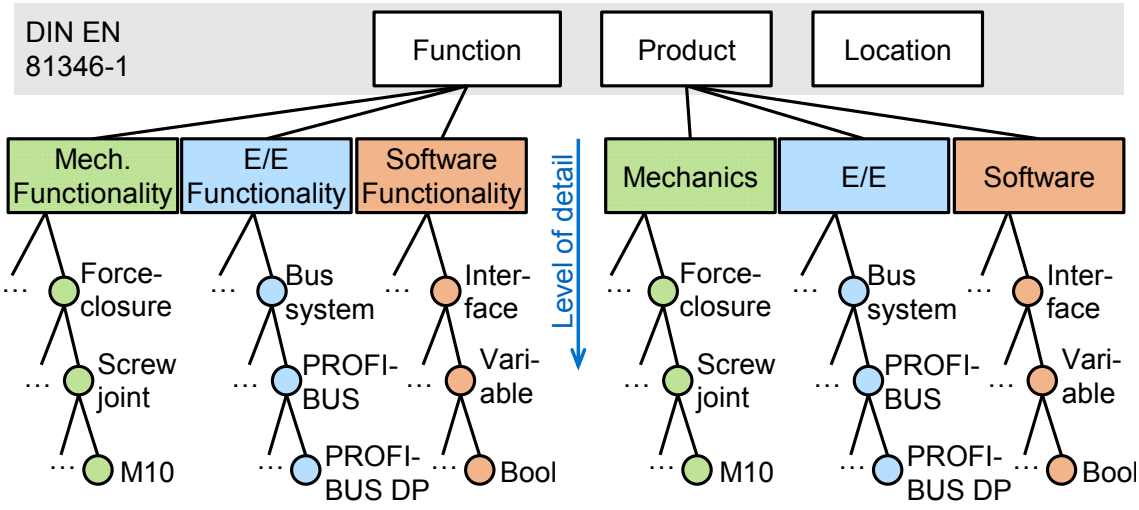


Figure 5-8: Example for levels of detail for properties from different aspects

Thereby, each component – or more generally speaking each object (which can be physically or virtually) – can be considered from different perspectives. The norm DIN EN 81346-1 [Dind10] defines these as *aspects* of an object and differentiates between *Product*, *Function*, and *Location* [Dind10]. The product aspect defines the

characteristics of the object. The function aspect defines what the object does or is ought to do. The location aspect, finally, describes the space or room in which the object is used. By using the aspect concept, the objects can be structured. Depending on the considered aspect, different properties are relevant. Thus, further properties can exist for an object, which, however, are not relevant for the particular analysis. The engineers have to decide during the development/modeling which properties are relevant for their specific perspective [Dind10].

The aspect concept is used as basis for the property concept of SysML4*Mechatronics*. However, as mechatronic systems are in the focus of the approach, the function and product aspects are divided further, regarding the relevant aspects of the different disciplines. Figure 5-8 shows the considered functional aspects *Mechanical Functionality*, *E/E Functionality*, and *Software Functionality* and the product aspects *Mechanics*, *Electrics/Electronics*, and *Software*. Each aspect represents the same object from a different view and thus, with different properties. Furthermore, properties can be considered on different levels of detail. Figure 5-8 shows some example properties on different levels of detail, e.g., the mechanical property can be considered on a high level as *force closure* down to the detailed property of an *M10* screw. The level of detail of an object can change during its lifecycle [Eppl11], e.g., if the Function-Behavior-Structure approach (cf. chapter 4.1.3) is used during the very early development stage, the properties are described on a very high level of abstraction. During the mechatronic development, which is focused by SysML4*Mechatronics*, the different components are considered and modeled in detail and thus, the properties and their respective values (cf. value types in chapter 5.1.3) are added.

In chapter 4.2.2 existing property classification systems have been presented, which focus mostly on the product aspect. Especially eCl@ss with its hierarchical and semantic systematization offers properties for a wide range of components, mostly from the electrics/electronics and mechanics field. However, an allocation of the properties to the different disciplines is not covered through eCl@ss. Furthermore, like in all product classification systems, only parts, which can be purchased from a supplier, are included. Own developed components and modules have to be considered separately. Nevertheless, the existing classified components and their properties can be integrated from eCl@ss into the modeling approach (chapter 5.3.2 describes how the integration as model library is conducted) and thus, can reduce the modeling time. The standardized properties from eCl@ss then have to be extended (regarding the discipline they are relevant for) and, if required, further individual properties have to be added to the component. In this way, a company specific library of components and properties can be realized and reused for each project.

Regarding the functionality of a component, eCl@ss or similar product classification systems offer an insufficient description. The NIST Functional Basis for Engineering Design [HSMS02], introduced in chapter 4.2.2, offers a classified representation of functions, which can serve as basis for the description of functionalities in SysML4*Mechatronics*. As the NIST Functional Basis was created especially for the design phase of physical systems, with a mechanical perspective on engineering design, it does not distinguish the functional description in different disciplines. As shown in Figure 5-8, this is, however, required for an integrated development of mechatronic systems, as different disciplines can use the same term to describe different functional aspects. For example, the NIST Functional Basis lists the function *Convert*, which can refer to different aspects. From a mechanical perspective, the workpiece material could be *converted* in a process step; from an electronic perspective *convert* could refer to the digitalization of a sensor value; and finally in software engineering the type of a variable could be *converted*. Therefore, a project or company specific library of functionalities has to be created based on the NIST Functional Basis and integrated in SysML4*Mechatronics*. In this library, the functionalities are assigned to the respective discipline as shown in Figure 5-8. The resulting *functionality library*, should contain, next to the different levels of detail, a textual description of the functionality, possible corresponding synonyms and an example [Lerc16]. In this way, engineers can determine for each component or module a matching functionality during the modeling process. In further projects, the resulting model library can be used to identify all components/modules, which are able to fulfill a specific desired functionality.

As described above, standardized classification systems, e.g., eCl@ss, include only detailed properties for single components and not for modules. However, modularization is a crucial factor (cf. chapter 4.3). As the modules are composed of several single components, the module properties can be derived in some cases from the properties of the included components. In order to create a relation between the module property and the component properties four different cases have to be considered (*Table 5*, based on [Hund16]). In the first case, the property of one or several component(s) can be adopted directly by the module. In this case, no changes to the value of the property have to be conducted and the property can be used directly by the module. In case No. 2 and 3, the component properties can be combined through a mathematical relation, e.g., sum, to the module property. Such a relation exists for example for the mass property. The mass of the module equals the mass of the included components. An example, in which the properties of the different components have to be compared and the minimum value has to be identified is a temperature range. The component with the smallest temperature range is decisive for the module. A special case of this relation is case No. 3. In this case, a module property can also be combined through the different component properties.

However, further information is required in this case. An example therefore are the overall dimensions of the module. Basically, the dimensions of the components can be added up, however, the spatial arrangement of the components has to be considered, e.g., through information from a CAD model. Case No. 4 applies, if the characteristics of the property do not allow a mathematical or logical combination. An example therefore can be the property *material*. If the components are manufactured with different materials, no uniform property for the module material can be defined. Thus, although module properties can not be included directly from a product classification system, it is possible for cases No. 1 to 3 to deduce them from the included components and define the module properties accordingly. This process, however, has to be done manually, when the module is defined for the first time. Thereupon, the module with its properties can be reused in different variants, versions or in new projects.

Table 5: Distinction of cases to generate module properties from component properties

No.	Description	Value of module property	Example
1.	Module property equals component property	$P_M = P_C$	Module voltage: $P_{M,Voltage} = P_{C,Voltage}$
2.	Module property results from mathematical relationship of component properties	P_M = (e.g., $\sum$, $\prod$, $\wedge$, $\vee$, min, max) $P_{C,1...i}$	Module mass: $P_{M,Mass} = \sum_{i=1}^{n} P_{Ci,Mass}$
3.	Module property results from mathematical relationship of component properties, but further structural information required to define value of module property	P_M = (e.g., $\sum$, $\prod$, $\wedge$, $\vee$, min, max) $P_{C,1...i}$ *if further information is available*	Module dimensions: $P_{M,Dim} = \sum_{i=1}^{n} P_{Ci,Dim}$ *Further information about position of components required*
4.	Module property can not be defined meaningfully / unambiguously from component properties	No statement possible	Module material: $P_{M,Material} \neq \sum_{i=1}^{n} P_{Ci,Material}$

P_M = Module property; P_C = Component property

5.3.2. Integration in modeling approach

In order to utilize the described classified properties during engineering they have to be integrated in the SysML4*Mechatronics* modeling approach. One possibility to add them would be the definition of respective stereotypes in the metamodel. This, however, would have the disadvantages that, if a change of a property is conducted or new properties shall be added, a change of the metamodel would be required. In chapter 5.1 the SysML concept of model libraries was introduced, which are special packages, containing

reusable model elements for a given domain [FrMS15]. The library – or several libraries – can then be applied to the modeling approach and used for an efficient modeling process.

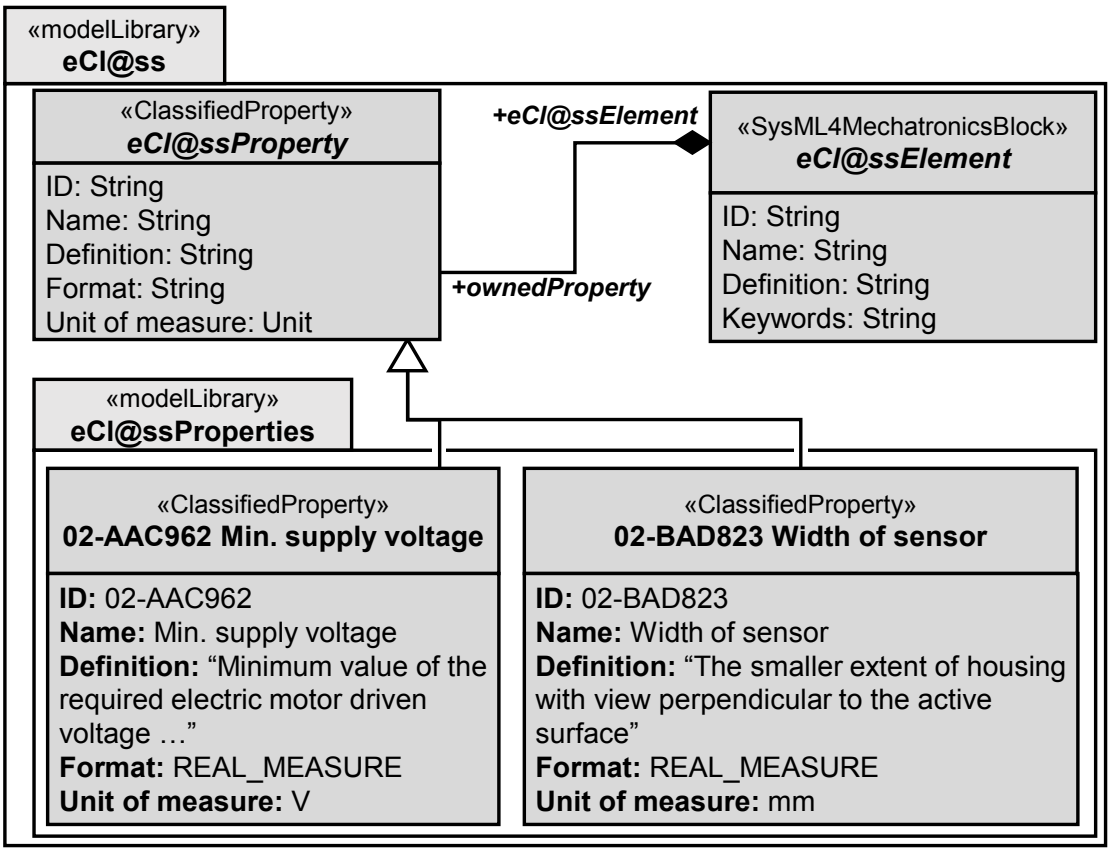


Figure 5-9: SysML4Mechatronics metamodel – properties in eCl@ss model library

The structure of the SysML4*Mechatronics* model library for eCl@ss is presented in Figure 5-9 and Figure 5-10. eCl@ss offers classified properties as well as a structured library of components, which are related to the properties (cf. Figure 4-7). Thus, the eCl@ss model library is based on the general *eCl@ssElement* and *eCl@ssProperty*. Thereby, the *eCl@ssProperty* is a SysML *ClassifiedProperty* and the *eCl@ssElement* a *SysML4MechatronicsBlock* respectively. Note that model libraries do not define new stereotypes, but apply existing stereotypes to create reusable modeling artefacts [Omgo15b]. The concrete eCl@ss properties and elements are stored in subfolders of the model library.

Each *eCl@ssProperty* in the SysML4*Mechatronics* model library represents the information from the respective property in eCl@ss, including *ID*, *Name*, *Definition*, *Format*, and *Unit of measure*. Figure 5-9 shows exemplarily the library elements for the minimum supply voltage and for the width of a sensor. The *Unit of measure* refers thereby to a unit from the standard QUDV model library (cf. chapter 5.1), while the other information is stored as String. Each property is independently and can be assigned to as many elements as necessary.

The subfolders for the eCl@ss elements are structured according to the eCl@ss four level hierarchy (cf. Figure 4-7). Each component is defined through *ID*, *Name*, *Definition* (optional), and *Keywords*. Figure 5-10 shows an inductive proximity switch *eCl@ssElement* and exemplarily an *ownedProperty* relation to the property *Min. supply voltage* in the *eCl@ssProperty* subfolder. According to the eCl@ss standard (eCl@ss Version 9.1), 65 properties are assigned to the inductive proximity switch, which can be represented in the model library accordingly.

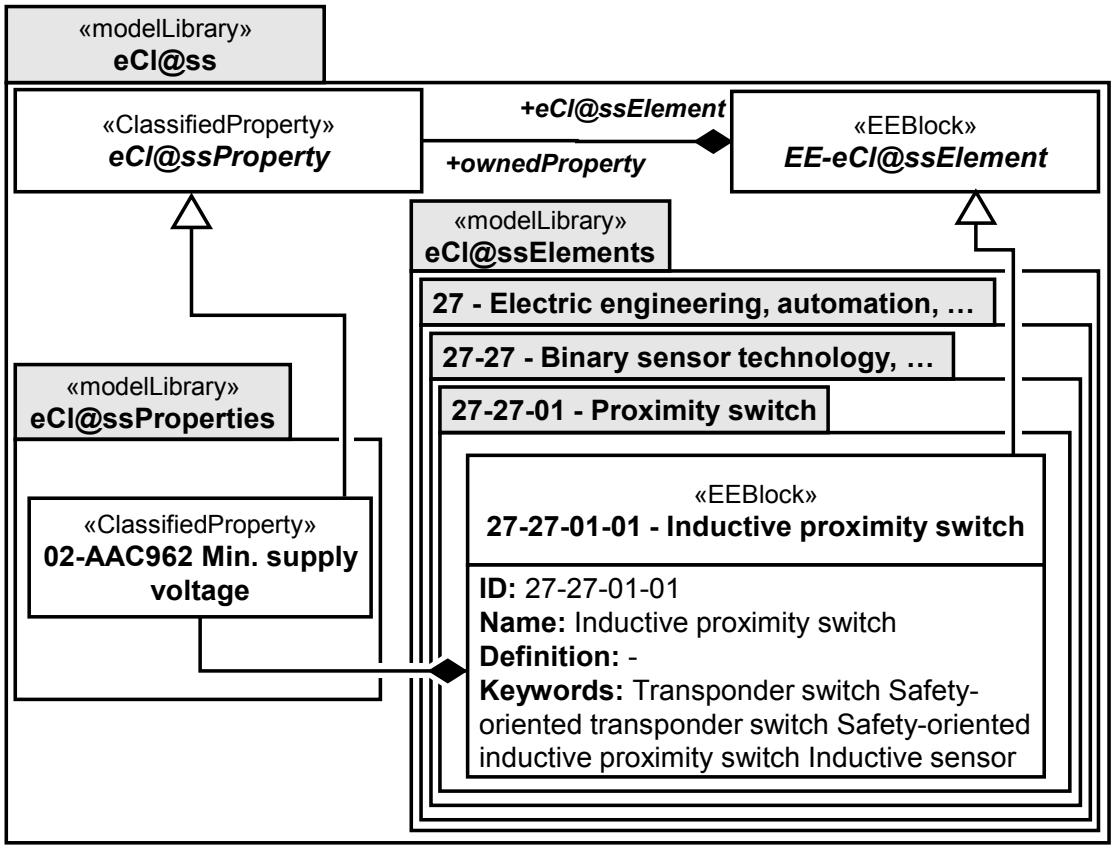


Figure 5-10: SysML4Mechatronics metamodel – eCl@ss component library [KeFV18]

In the same way as the eCl@ss properties are represented in a model library, the functionalities, based on the NIST Functional Basis, can be stored in an according SysML4*Mechatronics* model library (based on the SysML4*Mechatronics Functionality*) and assigned to the respective SysML4*Mechatronics* components and modules. As the eCl@ss library does not contain detailed information regarding the component's functionality, these elements have to be enriched through functionalities from the NIST Functional Basis model library during the modeling process.

5.4. Application of SysML4*Mechatronics*

After the concept of SysML4*Mechatronics*, including its metamodel, the concrete syntax, and the integration of standardized classification systems through model libraries, has

been presented, the two main scopes of application – i.e., the development and the reengineering/change of mechatronic systems – are described in the following.

5.4.1. Development phase

Following the MBE paradigm in engineering [BoFS12], SysML4*Mechatronics* supports the engineers in the detailed, mechatronic systems development through an integrated modeling approach. SysML4*Mechatronics* can be used for the development of completely new systems as well as for projects, which are (partly) based on prior projects [KeFV15]. The advantage of existing prior projects is the possibility to reuse models that stored in a repository. As they pose tested and reliable solutions the development time of the new project can be reduced, leading to a faster time-to-market.

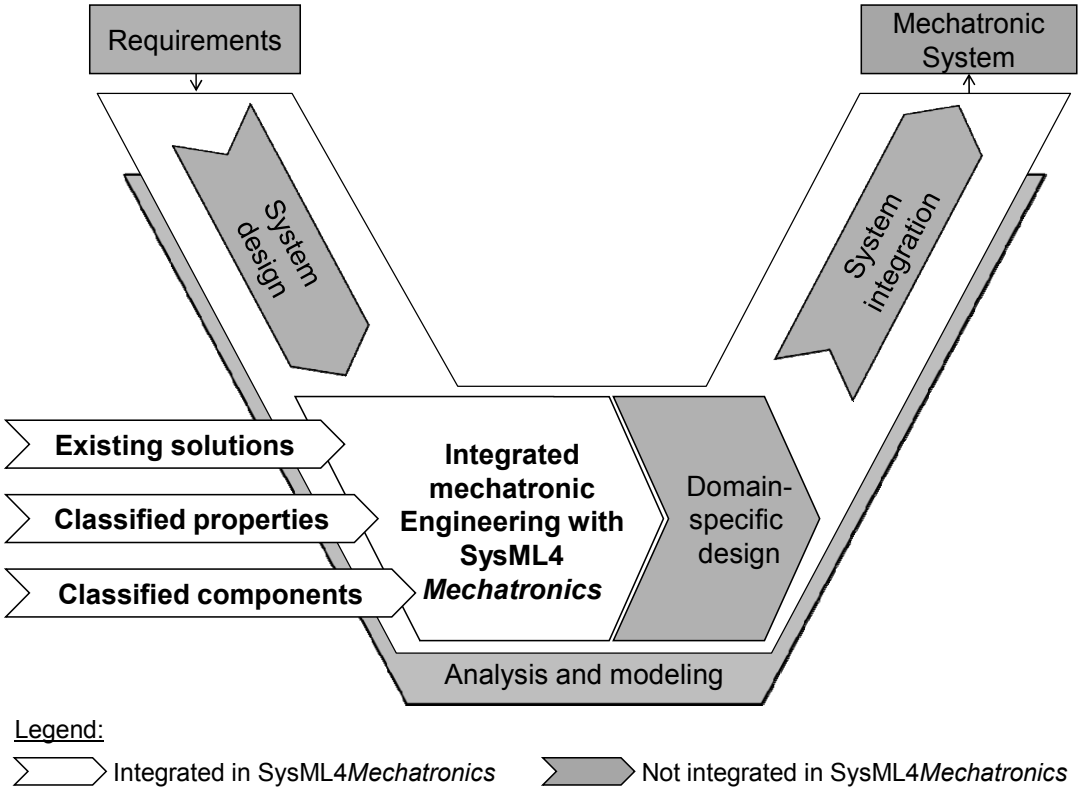


Figure 5-11: Usage of SysML4Mechatronics during engineering of mechatronic systems [KeFV18], extension of the V-model [Vdiv04]

Next to the repository of created solutions, i.e., models of components and modules that fulfill a specific system functionality, SysML4*Mechatronics* supports and accelerates the development process through the integration of classified components and properties as described in chapter 5.3. Figure 5-11 shows schematically, how SysML4*Mechatronics*, supported by these classified/pre-developed solutions, can extend the development process according to the V-model [Vdiv04]. The V-model is commonly used in industry, especially for the development of complex systems [MaCF15], however, offers in its original version (cf. Figure 2-5, [Vdiv04]) insufficient support for the detailed

mechatronic development. Thus, following the phases of *requirements definition* and *conceptual system design*, SysML4*Mechatronics* is applied. Following the integrated, mechatronic development, domain-specific models, e.g., CAD-models or analysis models, can be used, if required, and then be integrated to the overall system, resulting in the final mechatronic system, which is sold to the customer.

In [BKFV14] a design-pattern for the development of mechatronic systems, according to the V-model extended with SysML4*Mechatronics*, was presented. The approach is based on an iterative, hierarchical process on different levels of abstraction, covering the system, module, and component level. As a first step, based on the requirements, a high-level system model is developed. This information is inherited by the module level and a first modularization of the system is conducted on a high level of abstraction. The modules are then defined more detailed and required components are specified. Therefore, SysML4*Mechatronics* is used and already existing (commercial) components and modules are adopted in the model. After the break down phase (left side of the V-model), the information is integrated in the higher levels in order to ensure traceability of the overall system development and receive predefined solutions for further projects. Thus, this phase is called integration phase (right side of the V-model; the verification and validation phases were not focused specifically).

5.4.2. Reengineering/Change phase

The second major application area of SysML4*Mechatronics* is the reengineering or change phase in the system lifecycle. As described in the introduction of this thesis, mechatronic systems undergo frequent changes during the use phase. Reasons therefore can lie in new requirements, e.g., a customer demands to produce a new product, or in "forced changes", i.e., a component that becomes inoperable is not available anymore and thus, has to be exchanged through a different component [LBKV12].

In each change situation, the engineers have to choose an appropriate solution that shall be applied for the required change. Usually different alternative courses of action or solution alternatives are possible. In some cases, intentionally a non-optimal solution is chosen, as it is more important to fulfill short term goals, e.g., meet time restrictions of the customer. This is referred to as "technical debt", as the effects of the sub-optimal solution lead to "interest" that has to be paid on the long run [VFST15]. The term technical debt indicates its similarity to a financial debt.

Especially in mechatronic systems, with components from different disciplines, different solution alternatives can be chosen, e.g., a hardware or a software change (Figure 5-12). Depending on the chosen solution alternative, an exchange of different components or

modules is required to fulfill the new requirement. An exchange can include that different components have to be added, deleted or modified in the model [WaTT18].

On the one hand, this has a direct effect on the system (solution model in Figure 5-12), i.e., the exchanged component or module has to be compatible to the other elements in the system. On the other hand, also the subsequent lifecycle steps (lifecycle model in Figure 5-12) are influenced by the decision for a specific solution, e.g., the recycling phase [KBCW14]. Solution alternative 3 in Figure 5-12 "No Modification" could be chosen if the specification-change is not critical, which, however, could influence later lifecycle stages negatively. Thus, in this case the decision for a *technical debt* would be made.

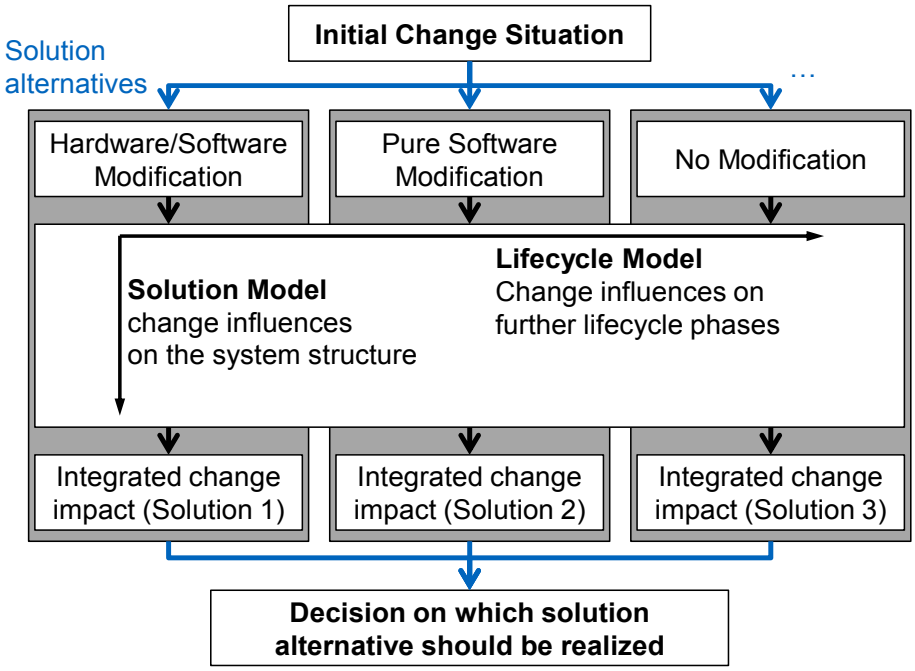


Figure 5-12: Integrated analysis of solution alternatives and their impacts, based on [KBCW14]

By considering both, the solution model as well as the lifecycle model, a more sophisticated analysis on the change impact of each solution alternative can be made and thus, the most appropriate course of action can be chosen. For analyzing the change impact in the lifecycle model expert knowledge is required, which can be captured, e.g., though check lists (for a detailed description of the analysis of change influences on the further lifecycle phases, which is not in the focus of SysML4*Mechatronics*, see [KBCW14]).

For the analysis of change influences within the system, i.e., the compatibility of the new component or module, SysML4*Mechatronics* is used. SysML4*Mechatronics*, however, as every semi-formal modeling language, lacks a formal basis, which is required for a compatibility check (requirement R 6). Formal models on the other hand are intended for

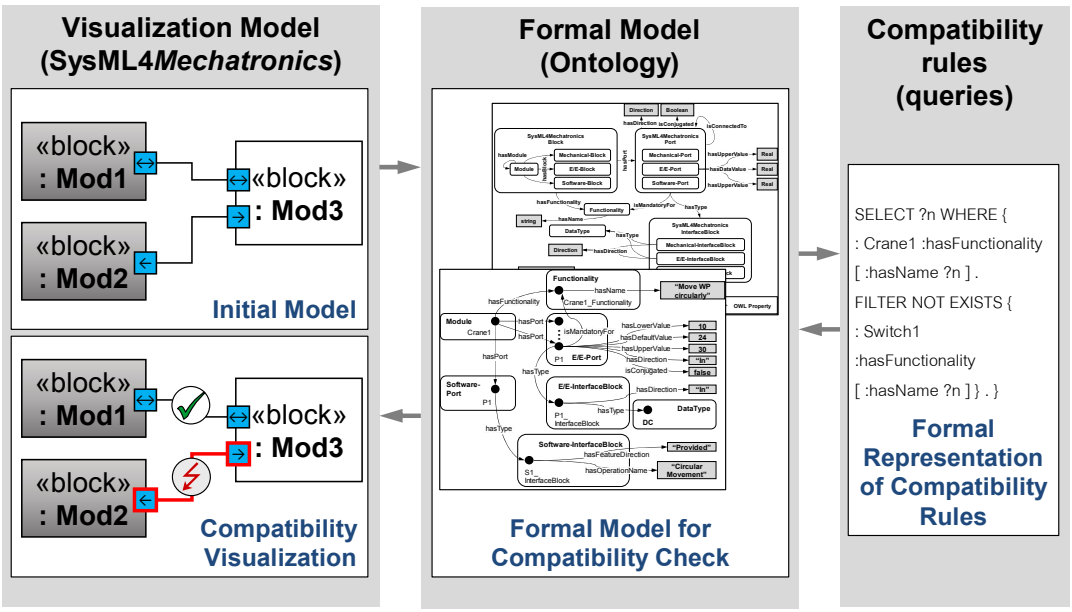


Figure 5-13: Overview of the compatibility analysis using SysML4Mechatronics and a formal model representation (based on [FeKV14]*)*

a computable knowledge representation and therefore, are less suitable as comprehensible model, which serves as interface to the engineers. Thus, an approach, which combines SysML4*Mechatronics* with an explicit knowledge representation and the application of inference mechanisms in the Web Ontology Language (OWL), was presented in [FeKV14] (Figure 5-13). Thereby, SysML4*Mechatronics* is used for designing the initial system, determine properties, and initiate a change. Through a Model-to-Model transformation [FeKV14] the semi-formal SysML4*Mechatronics* model is then transformed into an ontology, representing the captured information formally. In order to check the compatibility of an exchanged component or module, the compatibility rules are described by means of the SPARQL Protocol and RDF Query Language (SPARQL) and applied to the formal model. The detailed design of the formal model for compatibility and inconsistency check as well as possibilities for automatic resolution are under ongoing research.

Concerning the contents of the compatibility rules (independently from the implementation as SPARQL queries), two types can be defined: project-independent and project-specific compatibility rules. Inherent compatibility rules are defined project-independently, as they can be employed for each application. An example for a project-independent rule is the compatibility-check of two connected ports (cf., chapter 5.2): An *In-Port* can only be connected to an *Out-Port* or an *InOut-Port*. Project-specific compatibility rules pose additional compatibility-criteria, which are only relevant for the specific application, e.g., the definition of the maximum mass of the entire system, which may not be exceeded through an exchanged component.

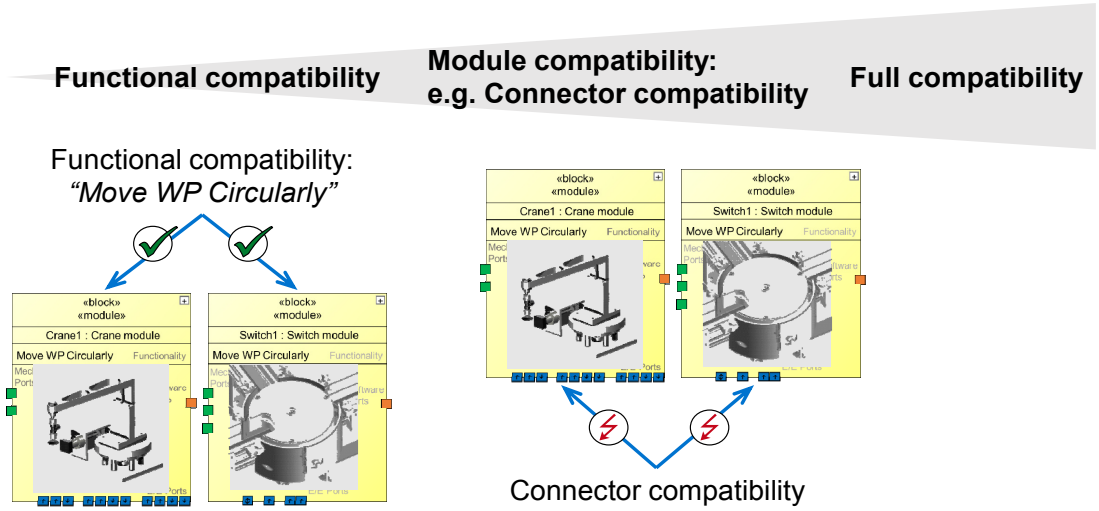


Figure 5-14: Comparison of different degrees of compatibility

As described above, the "right" solution for a system change is dependent on many aspects and can even include accepting a *technical debt* [VFST15]. This does not only include monetary effects, but also determines the "right" level of component or module compatibility. Thus, the required degree of compatibility for an exchanged component or module depends on the chosen solution alternative. Thereby, the following degrees of compatibility can be distinguished (based on [BFKK10]):

- Functional compatibility
- Component/module compatibility: Software compatibility, Signal compatibility, Data compatibility, Construction compatibility, Connector compatibility
- Full compatibility

Functional compatibility solely presupposes that the exchanged component or module can fulfill the same functionality as the former one. Component or module compatibility is related to several aspects, e.g., the (structural) compatibility of connectors, compatible data types or physical dimensions. Finally, full compatibility only is present, if the new component/module fulfills all compatibility aspects. Thus, depending on the required degree of compatibility, a certain component or module can either be suitable for a solution or not, as shown in Figure 5-14. While both, the crane and the switch, fulfill the same functionality, their ports differ (the crane module has analogue sensors and actuators, while the switch is connected via Profibus) and thus, the modules do not meet the criteria for module or full compatibility.

5.5. Summary

In this chapter, the developed approach – SysML4*Mechatronics* – for interdisciplinary structural modeling of mechatronic systems and the analysis of change influences was presented. The modeling approach forms the basis for an integrated system design.

SysML4*Mechatronics* is defined as profile of SysML (req. R 1) and extends its general modeling elements (e.g., block, property) through specific stereotypes for the disciplines mechanics, electrics/electronics, and software (req. R 2). By that, the engineers from the different disciplines can model their specific aspects of the system within the interdisciplinary SysML4*Mechatronics* model. Through the port concept, the different components can be connected with each other in a discipline-specific as well as an interdisciplinary way (req. R 3). Furthermore, interdisciplinary modules can be formed, which group several discipline-specific components and, if desired, submodules to functional, reusable units (req. R 4).

Although not all modeling elements of SysML are required for SysML4*Mechatronics* (e.g., behavioral elements), they still stay available in the metamodel due to the concept of SysML profiles, which constitute an extension of the existing metamodel. By using a standardized language as basis (req. R 1), the integration in existing engineering processes and the extensibility of SysML4*Mechatronics* can be assured. SysML4*Mechatronics* could be used together with further SysML-conform profiles.

In order to facilitate the modeling process further, the integration of properties from standardized classification systems was presented (req. R 5). By that, the engineers do not have to define each property manually, which can lead easily to mistakes or ambiguities, rather they can be chosen from a model library, which can be integrated in the modeling approach. Due to their suitability, an eCl@ss model library for structural properties as well as a model library based on the NIST Functional Basis for the functionalities of the components and modules have been defined. However, not all properties can be integrated from classification systems directly, e.g., module properties.

The two main areas of application of SysML4*Mechatronics* are the development phase and the analysis of change situations. Following a development process according to the V-model, which is commonly used in industry, SysML4*Mechatronics* is applied in the detailed mechatronic system development. Thus, SysML4*Mechatronics* can be used as central, integrated structural modeling approach for mechatronic systems from which further detailed models, e.g., for a specific discipline, can be derived through model transformations (req. R 6). In change situations the compatibility of an exchanged component or module can be analyzed, based on the information modeled in SysML4*Mechatronics*. Therefore, the semi-formal SysML4*Mechatronics* model is

transformed into a formal model (req. R 6). In the resulting ontology the compatibility analysis is conducted through the application of respective SPARQL queries.

6. Evaluation of the concept

In this chapter the developed modeling approach, SysML4*Mechatronics*, is evaluated regarding its applicability for interdisciplinary mechatronic modeling and a first assessment of its usability. Basis for the evaluation are the requirements defined in chapter 3.2. In chapter 6.1, the concept for the different parts of the evaluation is presented. In order to realize the evaluation, a prototypical implementation of a SysML4*Mechatronics* editor (req. R 7) is presented chapter 6.2 and used in a case study. Chapter 6.3 describes the refinement of the implementation as basis for the usability inspection (chapter 6.4). Finally, a summary of the SysML4*Mechatronics* evaluation is given in chapter 6.5.

6.1. Evaluation concept

The evaluation of modeling approaches is often considered under one specific aspect, e.g., evaluating the language quality [Krog03]. A more general framework for the evaluation of the quality of modeling approaches has been proposed by Lindland et al. [LiSS94] and was extended in several works, e.g., [KrLS95, Krog03, KrSJ06]. The framework has a theoretical basis, originating from semiotics and linguistics, and has been validated empirically, e.g., by Moody et al. [MSBS02, MSBS03]. Nowadays, it is usually referred to as SEQUAL framework (semiotic quality framework) [KrSJ06]. Figure 6-1 shows the main elements, i.e., quality statements and quality categories, of the framework. The four quality statements of the framework are the *model*, the *language*, the *domain*, and the *user interpretation*. The model is the set of statements actually made, the language includes all statements that are allowed according to the syntax, the domain consists of all possible statements that would be correct and relevant for the problem domain, and finally, the user interpretation represents what the users think that the model expresses [MaPo07, MSBS03].

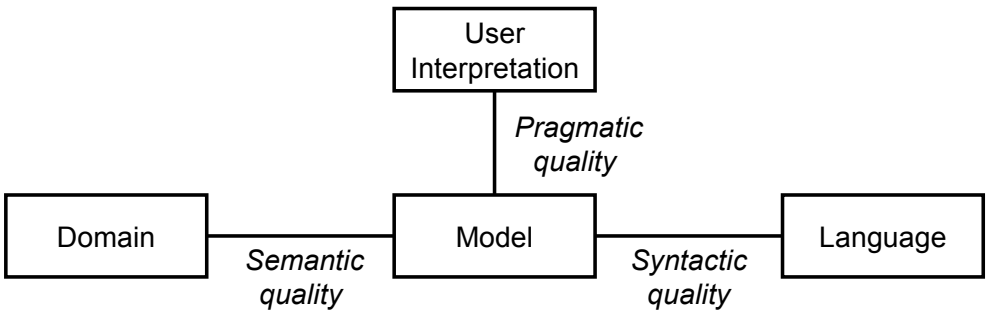


Figure 6-1: SEQUAL Framework for model quality, based on [LiSS94]

Regarding the three quality categories, the *semantic quality* describes the relation between the domain and the model, thus, the validity and the completeness of the model for the

specific problem domain. The *syntactic quality* is the relation of the model and the language and describes, if the model is conform to the grammar of the language. The *pragmatic quality* (relation of model and user interpretation) represents how good the model is understood by its users [MaPo07]. While the framework defines the different quality categories, it does not define how they have to be measured as this can vary for different modeling approaches or problem domains [Ritt10].

Thus, in order to evaluate the quality of the SysML4*Mechatronics* approach, two evaluation levels are defined:

The *domain – model – language* level contains the syntactic and semantic quality in the framework. In order to achieve models according to the SysML4*Mechatronics* metamodel (i.e., syntactically correct models), a prototypical implementation of a SysML4*Mechatronics* editor, with the respective modeling elements, was conducted (chapter 6.2.1). Semantic quality can not be measured directly, as the demand on completeness of the problem domain (see above) is always influenced by the perceptions of the user [Ritt10]. Thus, the knowledge about the domain is always subjective. Therefore, usually the proxy perceived semantic quality is used [KrSJ06, MaPo07, Ritt10], which considers the users perception if his knowledge of the domain can be represented through the model. Therefore, a use case (chapter 6.2.2) was conducted and analyzed, if an existing mechatronic system (problem domain) can be modeled sufficiently by using the SysML4*Mechatronics* approach.

In the *model – user interpretation* level the usability of the modeling approach is evaluated. Although the original definition of pragmatic quality only focuses on comprehension of the model by users [LiSS94], it has been extended in later works [KrSJ06, MaPo07] to include also the perceived ease of use and perceived value, i.e., usefulness, of the modeling approach. In order to analyze the usability of SysML4*Mechatronics*, the prototypical editor was improved after its first implementation (chapter 6.3) and then several user evaluations were conducted with students as test persons (chapter 6.4).

6.2. Implementation and case study

Although the definition and application of a modeling approach can be conducted independently from a tool implementation, an optimized tool support significantly contributes to the success of a modeling language in the development process [LFWZ10]. Thus, a prototypical editor has been developed for the SysML4*Mechatronics* modeling approach (chapter 6.2.1) and was applied for a use case (chapter 6.2.2) of a mechatronic system.

6.2.1. Proof-of-concept implementation

The SysML4*Mechatronics* editor was implemented in the context of this thesis to provide a comprehensible modeling environment for practically evaluating the developed modeling approach (req. R 7). This includes first of all offering the user the possibility to create models in accordance to the SysML4*Mechatronics* metamodel presented in chapter 5.1. In comparison to general-purpose modeling editors, which usually integrate various modeling languages and notations (e.g., MagicDraw® includes UML, SysML, BPMN, UPDM and others), the developed customized editor has the advantage that it focuses only on the relevant aspects of the SysML4Mechatronics approach. Thus, potential users can interact with the system easily, even if they are not familiar a specific (complex) industrial SysML editor. In order to represent the metamodel elements, e.g., with according symbols in a diagram [Mood09], a modeling language requires the definition of a concrete syntax (cf. chapter 2.3). Thus, the modeling editor offers the user specific modeling elements (req. R 2 – R 4), according to the concrete syntax of SysML4*Mechatronics* presented in chapter 5.2. Additionally, the tool offers supporting functionalities, which are not defined in the metamodel of a modeling language. For example, to support the development process and reduce complexity, the editor offers a filter concept (req. R 7) that allows users from a specific discipline to show only their blocks and, if existent, connections to parts of other disciplines. The elements, which are not relevant for that discipline, i.e., which have no connections to blocks from that discipline, are suppressed by the filter. An overview of the editor is given in Figure 6-2.

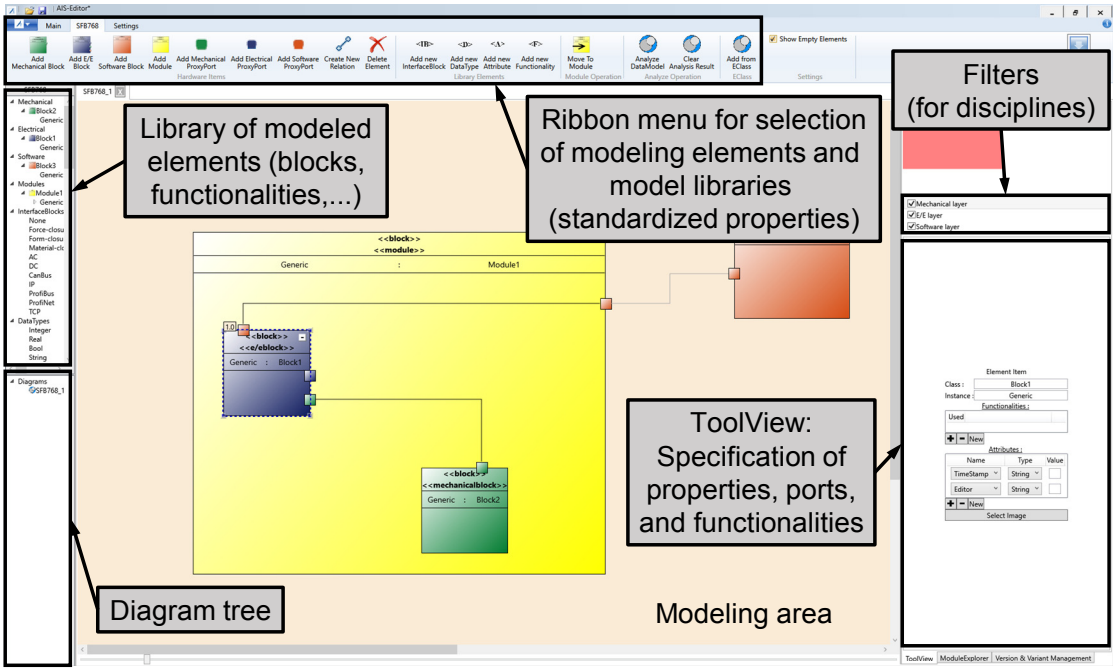


Figure 6-2: Screenshot of the SysML4Mechatronics modeling editor [KeFV18]

The *modeling area* is the principal component of the editor, in which the user develops the system model. Thereby, the different modeling elements can be chosen from the symbolic view at the top of the editor. In order to facilitate the modeling process the editor generates automatically an instance of a block, when it is placed in the modeling area. At the same time a *library* element of that block type is created, which can be reused as often as required in the project, for new variants or versions of the system, or for new projects.

In the *diagram tree* at the bottom left, the different created diagrams, e.g., for different variants of the system, are listed and can be selected accordingly. In the *ToolView* on the right-hand bottom side, the user can define the specific properties of the blocks and ports and define the functionalities.

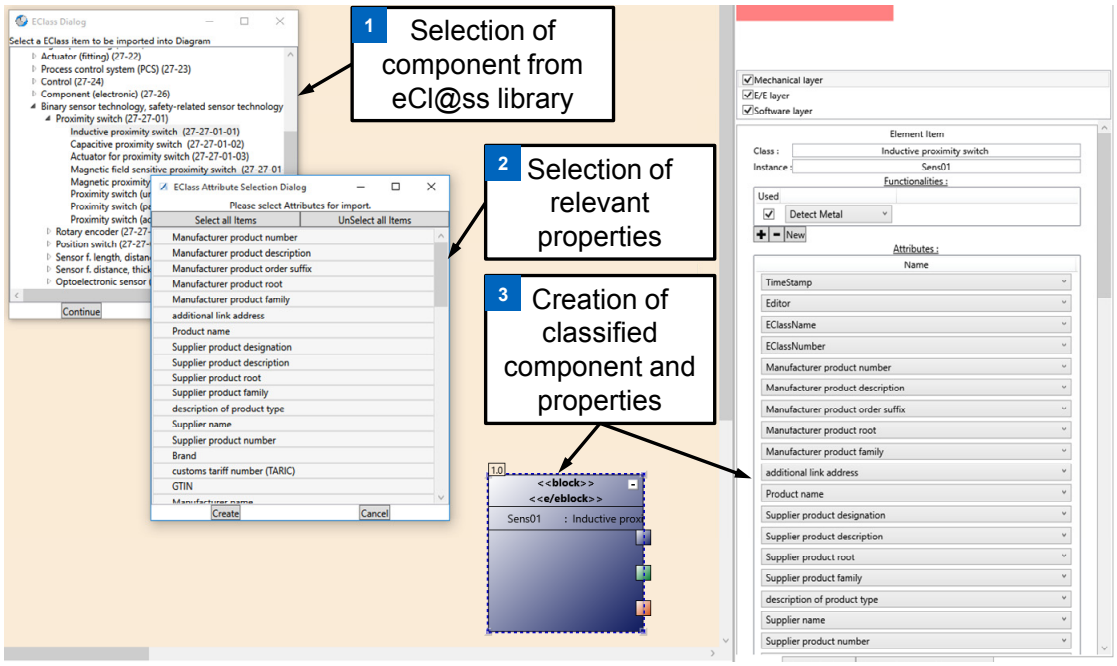


Figure 6-3: Selection of an eCl@ss component and properties in the editor [KeFV18]

As presented in chapter 5.3, components and properties from classification systems can be integrated as model library in SysML4*Mechatronics* to facilitate the modeling process and reduce ambiguities (req. R 5). Thus, the model library concept was included in the editor and is accessible via the *ribbon menu* at the top of the editor. Figure 6-3 shows the selection process of an eCl@ss component and standardized properties from the integrated library. In the first step, a component, according to the eCl@ss structure (cf. Figure 4-7 in chapter 4.2.2), has to be selected. After the user has selected a specific component, the eCl@ss properties of the component are presented in the second step. The user can choose from the list, which properties he wants to include in his model and omit the ones that are not relevant for the specific project, e.g., the customs number of the component. After the selection, the editor creates the respective element in the modeling

area and assigns the chosen properties to it automatically. The user, however, still has the possibility to adapt the eCl@ss properties or add properties manually, if required.

Regarding the technical perspective, the editor was developed according to the Model–View–Viewmodel (MVVM) concept, which allows separating the data (model) from the way it is presented to the user (view) and the interaction, e.g., through commands, filters etc., which are applied to the model or the view (view-model). The SysML4*Mechatronics* editor was implemented in C# in the .NET Framework 4[11] and the data model is realized as extension of the .NET UML metamodel.

6.2.2. Case study

In order to test the feasibility of the modeling approach and evaluate the applicability for the domain of mechatronic systems, i.e., evaluate the perceived semantic quality, a case study was conducted. Note that, although an existing system was used as basis, a design from scratch process is assumed in the first step of the case study. The usage of an existing system as basis has the advantage that the developed models can be compared with the real system.

The analyzed system for the case study is a "Pallet transportation and handling system" and is based on the "Testbed for Distributed Control" of the Automation and Control Institute (ACIN) at the Odo Struger Laboratory located at the TU Wien (Vienna, Austria) [LGMS13]. The modeling process was conducted together with three experts from ACIN, who have expertise in the development of mechatronic production systems as well as automation software development. The participants did not have any prior knowledge of SysML4Mechatronics and thus, could assess the developed approach in an unbiased way. After the modeling process, the results as well as the modeling approach itself were discussed with the experts in a qualitative manner (interview). Figure 6-5 shows schematically the general setup of the pallet transportation system. It is build up by *conveyors, handling stations,* intersections with *two-way* or *three-way switches* (Figure 6-4, right side) and *corner connections*. Furthermore, *RFID-reader*, *inductive proximity switches* and *stoppers* are located in various parts of the system (cf. Figure 6-5). Furthermore, each actuator is equipped with one or two reed switches to provide its current position to the PLC and control software. A set of PLCs (type CPX-CEC-C1 by Festo, not shown in Figure 6-5) is installed to control the conveyors, handling stations and intersections. Each RFID-reader is controlled by its own embedded controller (type Digi Connect ME). The handling station (Figure 6-4, left side) consists of two stoppers, two inductive proximity switches, a RFID-reader and a gripper (gripper not visible in Figure 6-4) and thus, can hold a pallet in a defined position to process its loading, e.g.,

[11] Microsoft® .NET Framework: http://www.microsoft.com/net retrieved on 23 September 2018

through a workstation (not part of the considered system). The conveyors, which are used to transport the pallets through the system, are operated by 24 V DC motors and can be moved in both directions.

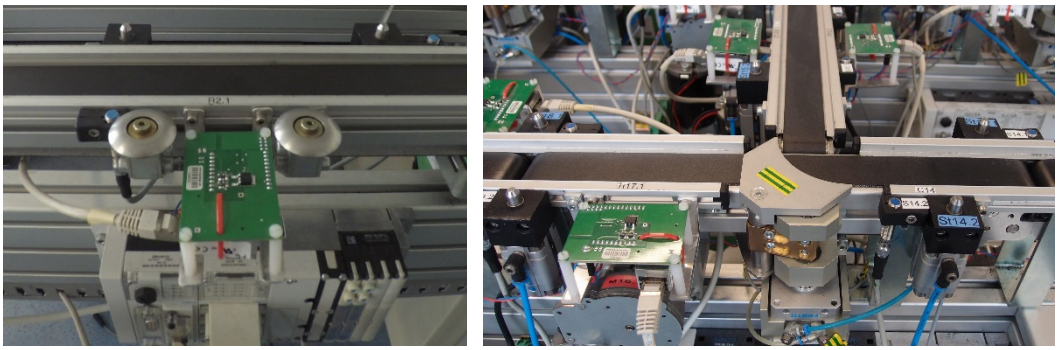


Figure 6-4: Handling station (left) and three-way switch (right)

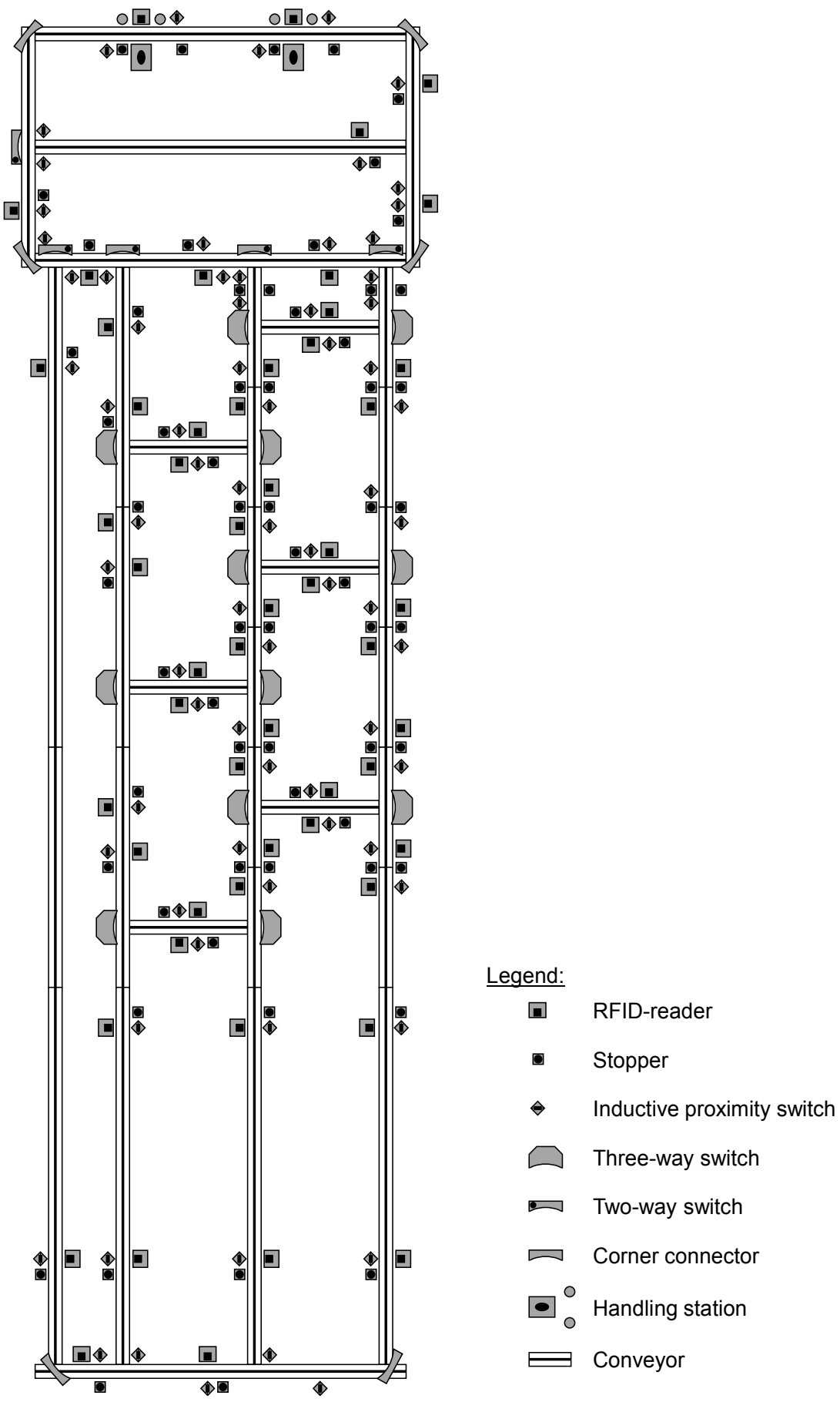


Figure 6-5: Schematic representation of the pallet transportation system [KeFV18]

As a first step of the feasibility evaluation, the development of the described mechatronic system is analyzed. In order to compare the development with and without SysML4*Mechatronics*, the required steps of the task for each case have to be determined. Therefore, a *Hierarchical Task Analysis* (HTA) [ADSG71] was conducted. The HTA approach was developed to decompose a task hierarchically into its required sub-tasks in order to receive an abstract model of the process. The result of an HTA is shown in a tree diagram, representing the different required (sub-)steps of the task. Depending on the application area and the goal of the conducted HTA, the abstraction level can vary from a high-level task description to a very detailed level, where each single step (e.g., mouse click) a user has to conduct is listed. Due to its hierarchical structure, the HTA can be applied very well to technical tasks such as the development [Brau13]. In previous works at the Institute of Automation and Information Systems, the HTA has been applied for usability evaluations of model-driven object oriented programming approaches [OBSV11, VBOJ12].

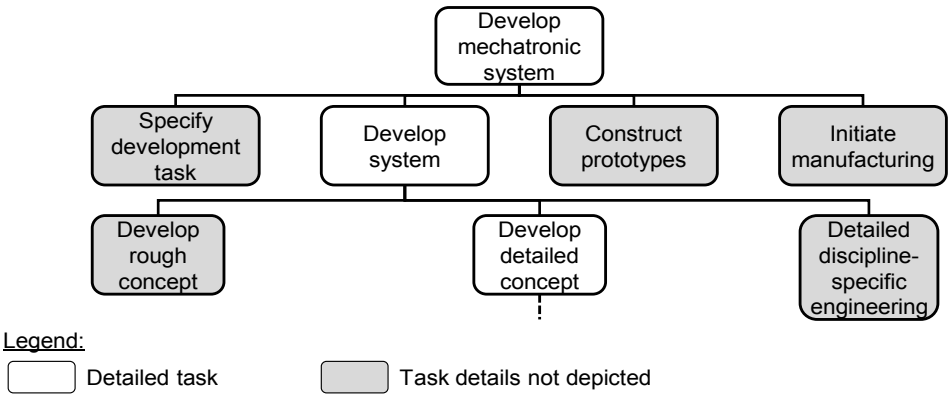


Figure 6-6: High-level HTA of "Design mechatronic system" task

The development of mechatronic systems is a complex task, which persists of many sub-tasks. Hence, in a first step the abstract task "develop mechatronic system" is divided into its main sub-tasks (see Figure 6-6). These high-level sub-tasks ("specify development task", "develop system", "construct prototypes", and "initiate manufacturing") form the basis for most standardized development processes, e.g., according to the V-model [Vdiv04, Vdiv94], and can be applied to model-based as well as to traditional, non model-based, approaches. The steps before and after the actual development (grayed out tasks in Figure 6-6) are not detailed further, as they are not focused by this thesis. The task "develop system" is divided in the next hierarchical level into the development of the rough concept (e.g., use cases based on high-level requirements), the detailed (interdisciplinary) development, and the detailed discipline-specific engineering (e.g., code, which still can be required even if an interdisciplinary model-based approach is used, see Figure 5-11 in chapter 5.4.1). This level still can be seen independently from the chosen development approach. However, in a model-based development process, the

information between these tasks can be exchanged automatically, if appropriate model transformations are defined [FWKV16, KWMK13] or respective languages for the exchange, e.g., AutomationML [DLPH08], are used.

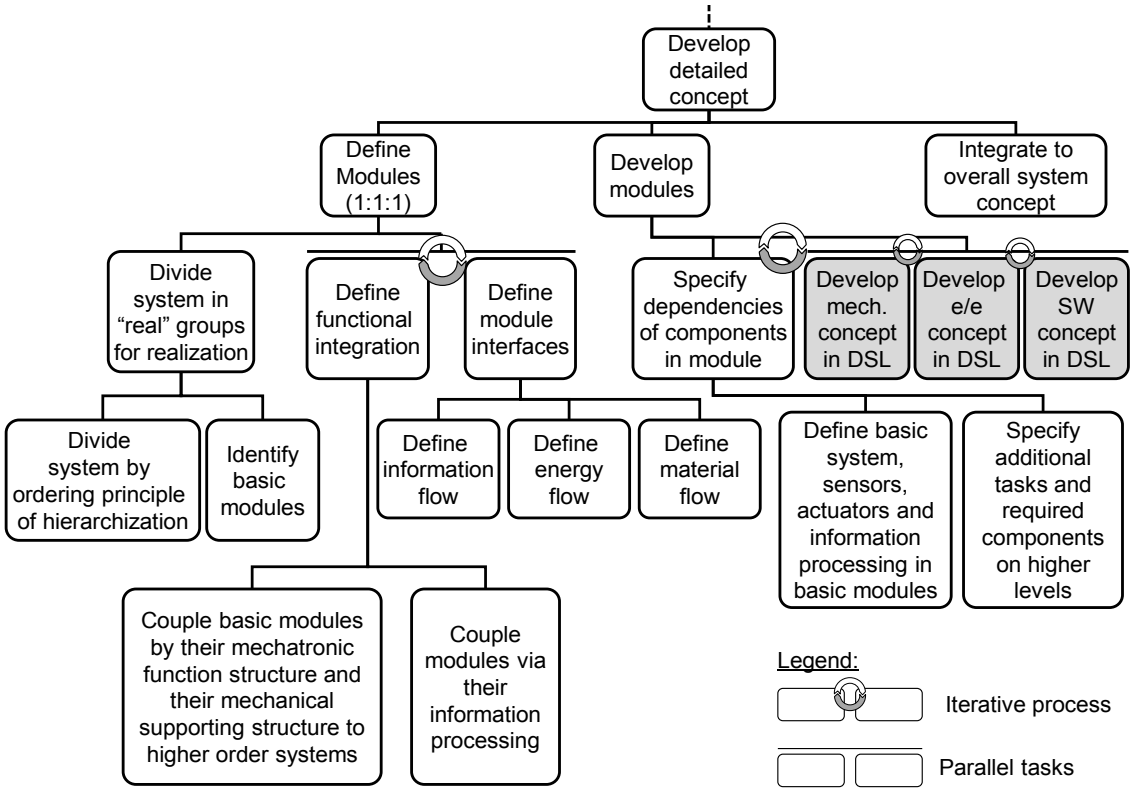


Figure 6-7: HTA of "Develop detailed concept" without integrated modeling approach

In Figure 6-7 the hierarchization of the task "develop detailed concept" in its detailed sub-tasks is shown for a traditional development process according to the VDI guidelines [Vdiv04, Vdiv93, Vdiv94], as described in chapter 2.2. According to the guideline VDI 2206 [Vdiv04], mechatronic modules should be defined on different levels of hierarchy, beginning with *basic modules*. These basic modules represent a functional modularization, but also require an integration of mechanics, electrics/electronics and software. Thus, a 1:1:1 modularization of all disciplines (cf. chapter 3.2) is conducted and according interfaces have to be defined. After the module structure has been defined, the modules can be used during the development. Through a coupling of modules by their mechatronic function structure [Vdiv04] a complex mechatronic system can be defined. Additional tasks and further components on higher hierarchy levels can be added as well. Based on the defined components of the mechatronic module and the module border, the further concretization takes place separately in each domain, e.g., with domain specific languages (DSL). Finally, the different developed modules are integrated to the overall mechatronic system. An example for a basic module in the pallet transportation system is the three three-way switch (cf. Figure 6-4). The module consists of the mechanical construction, two reed switches at the end positions and two valves for actuating the

switch. As a mechatronic module according to the described approach (guideline VDI 2206 [Vdiv04]), additionally the controller with the information processing would have to be integrated in the module. However, the described modularization is disadvantageous, as each three-way switch would be a self-contained module with, e.g., separate controllers and valves. On a higher hierarchy-level, the basic modules are connected with other modules. Figure 6-8 shows a module for the connecting conveyors for which the valves of the two included switches could be integrated into one valve cluster. However, with a 1:1:1 modularization of the basic modules as described above, this can not be conducted, as each module would contain its own valve cluster.

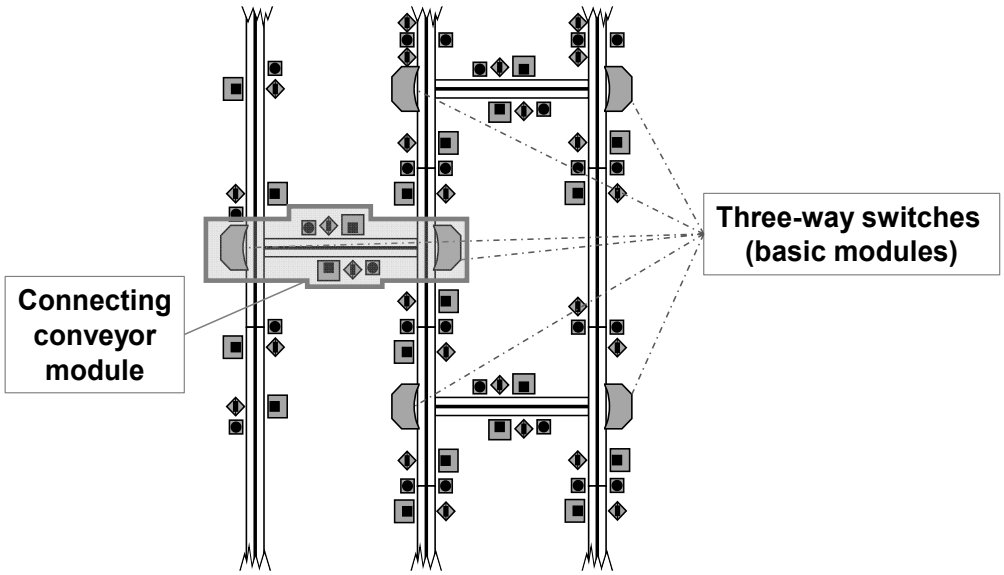


Figure 6-8: Excerpt of the system with six three-way switches [KeFV18]

In comparison to the tasks following the traditional approach, Figure 6-9 shows the HTA for the detailed development using SysML4*Mechatronics*. The sub-task "*Create new component in SysML4Mechatronics*" shows in detail the required steps to define a component, including its discipline, properties (from a classification system or manually), and its ports. These steps only have to be conducted if a respective component does not yet exist in the library (the prototypical SysML4*Mechatronics* editor adds the component automatically to the library, when it is modeled for the first time, cf. chapter 6.2.1). If the developer chooses to use an existing component from the library for his system, he can integrate it directly to the modeling area and adapt the parameters accordingly. The different components in the system can be connected discipline-specifically as well as interdisciplinary, as described in chapter 5.1.2. SysML4*Mechatronics* allows the user to define functional modules after the single components have been added to the modeling area or, if desired, the module can be defined a priori as well. For the definition of a module, the relevant components (which are required to fulfill the specific module functionality) in each discipline are identified and integrated in the module. However, in

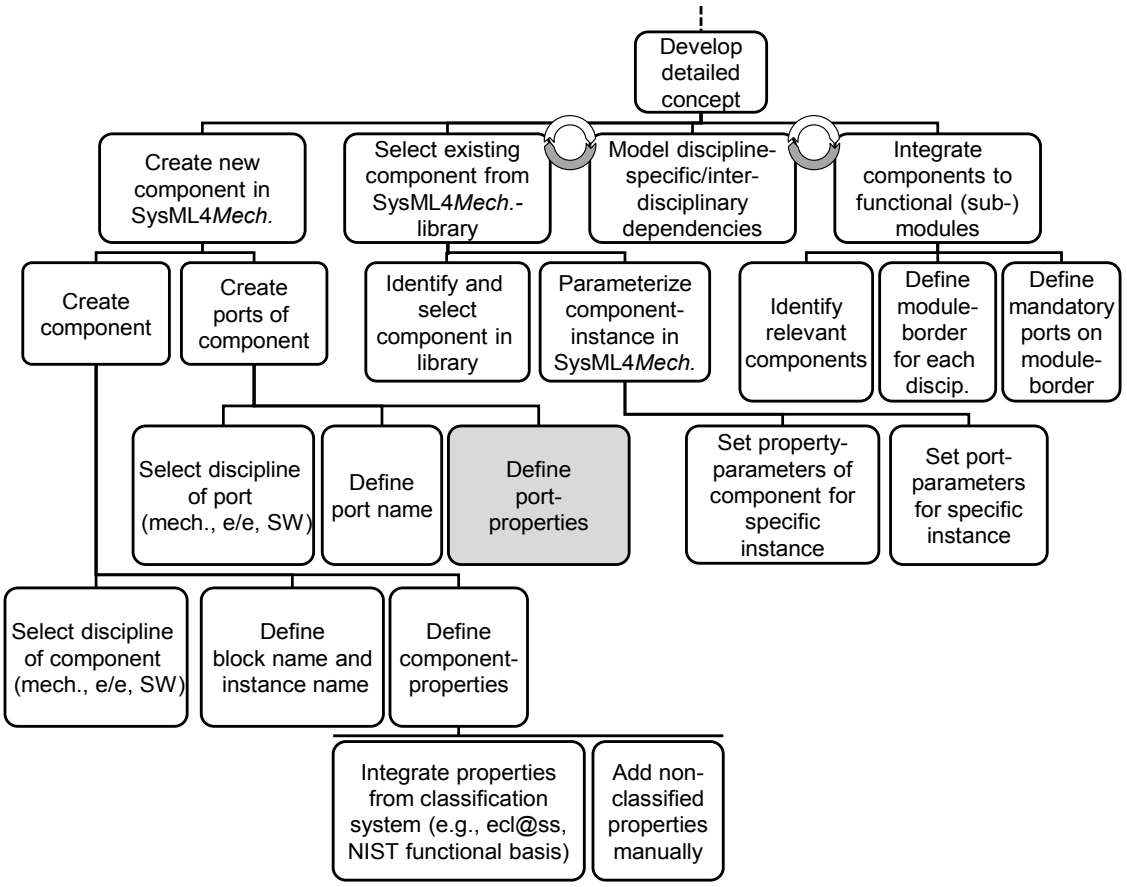


Figure 6-9: HTA of "Develop detailed concept" with SysML4Mechatronics

comparison to the 1:1:1 modularization, SysML4*Mechatronics* does not require that the module is self-contained. Rather, mandatory ports on the module border can be defined (cf. respective sub-task in Figure 6-9), which have to be fulfilled by components outside of the module. Thus, the three way-switch of the pallet transportation system is modeled as SysML4*Mechatronics* module as shown in Figure 6-10. The module contains all components to fulfill the functionality. However, during the realization of the overall mechatronic system, the components of the module do not necessarily have to be located together. For example, the two required valves of the three-way switch could be mounted mechanically together with other valves in one valve cluster, which is not part of the module. Therefore, the mechanical connection ports of the valve are defined as mandatory and have to be connected to an adequate port outside of the module. Figure 6-10 shows, furthermore, exemplarily the integration of two reed switches from the eCl@ss model library (cf. Figure 5-10). Thus, for the reed switch (classified under the eCl@ss-code 27-27-01-04 as "Magnetic field sensitive proximity switch") the sub-tasks for creating the component manually are not required.

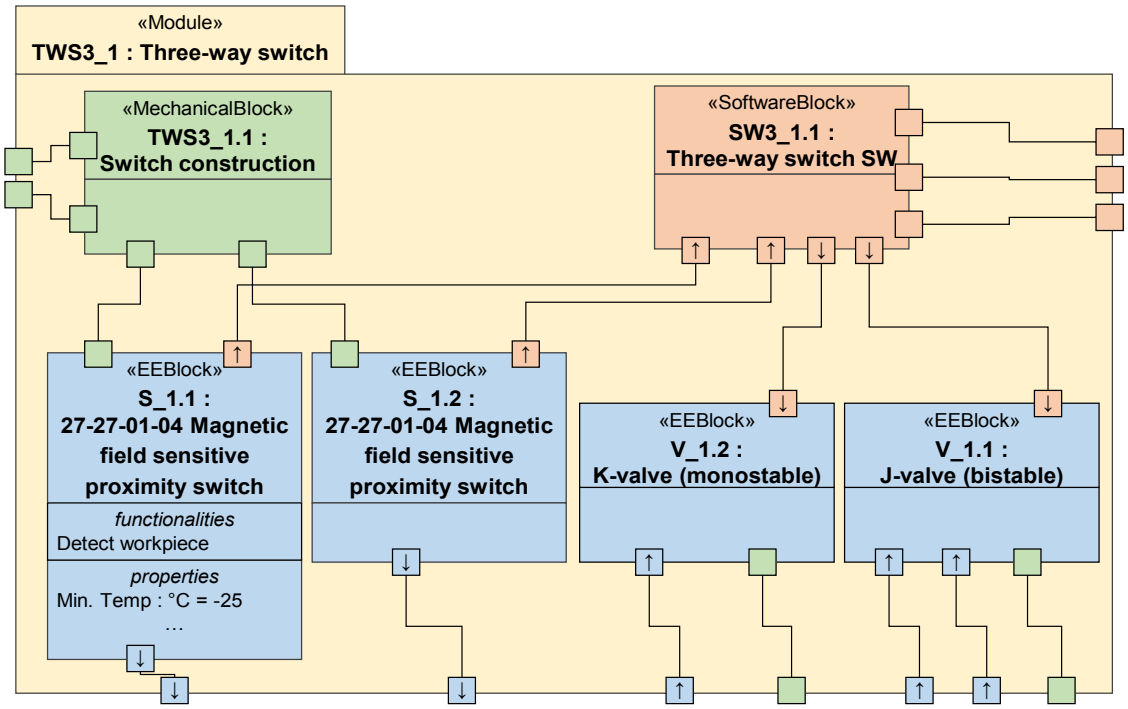


Figure 6-10: SysML4Mechatronics model of the three-way switch [KeFV18]

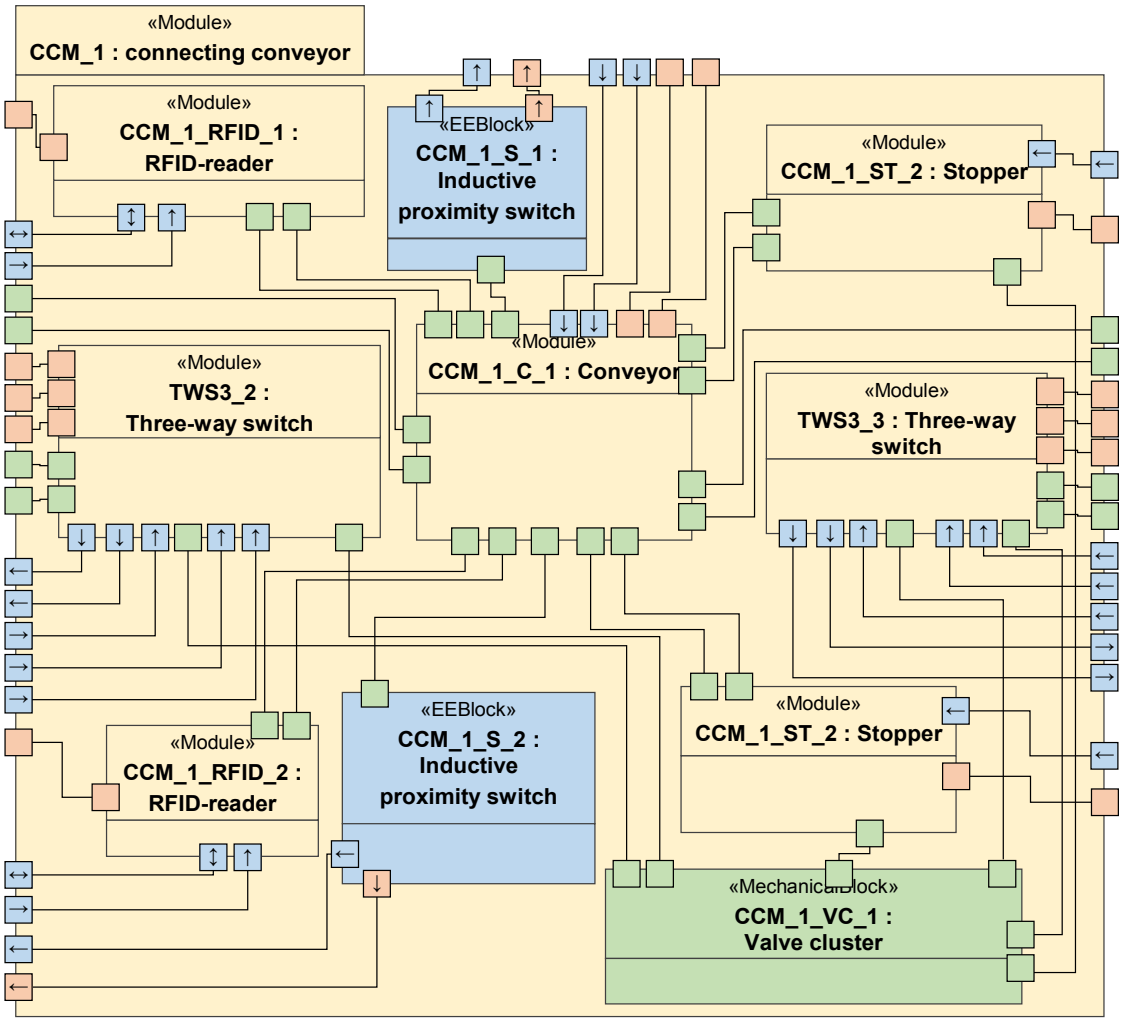


Figure 6-11: SysML4Mechatronics model of the connecting conveyor module [KeFV18]

Figure 6-11 shows, how the switch module is integrated with other components and modules into the *connecting conveyor* module. Thereby, the four valves of the two switches are integrated together with the four valves of the stoppers into one valve cluster. Note that the valve cluster in this case is part of the connecting conveyor module and thus, contains only the valves of the parts within the module. The choice of this module border, however, is not mandatory and is dependent, e.g., on the possible reuse of modules in further projects or on the company strategy. While the valve cluster was appointed to the module in the considered case, no software block for the connecting conveyor module was defined. Thus, the sub-modules and components can be controlled directly from software blocks outside of the module and make the module more flexible.

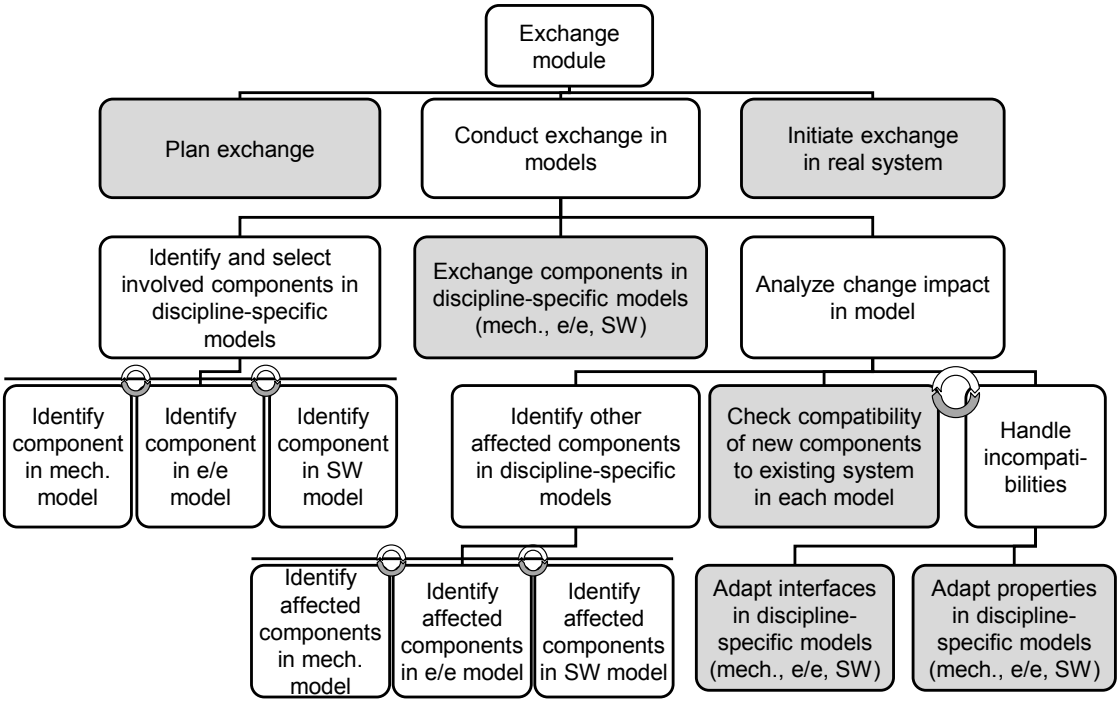


Figure 6-12: HTA of "Exchange module" without integrated modeling approach (discipline-specific models)

As a second part of the case study, an exchange of a module during the usage phase of the system is analyzed. Figure 6-12 shows the respective HTA, if the system was developed without an integrated modeling approach. As several discipline-specific models were used for the development (cf. Figure 6-7), the respective components have to be identified in the different models. This task can be especially error-prone and time-consuming, if not all models have been synchronized and kept updated, as it can be hard to identify the components in all specific models. After all involved models have been identified, the respective components have to be exchanged in the different discipline-specific models. Thereafter, the impact of the change can be analyzed in each discipline by checking the compatibility of the exchanged components to the existing system and, if required, handle the identified incompatibilities, e.g., by adapting the interfaces or

properties. In many cases however, companies up to now rely on expert knowledge or their gut feelings for analyzing change impacts (cf. results of the questionnaire with experts from industry, Figure 3-5).

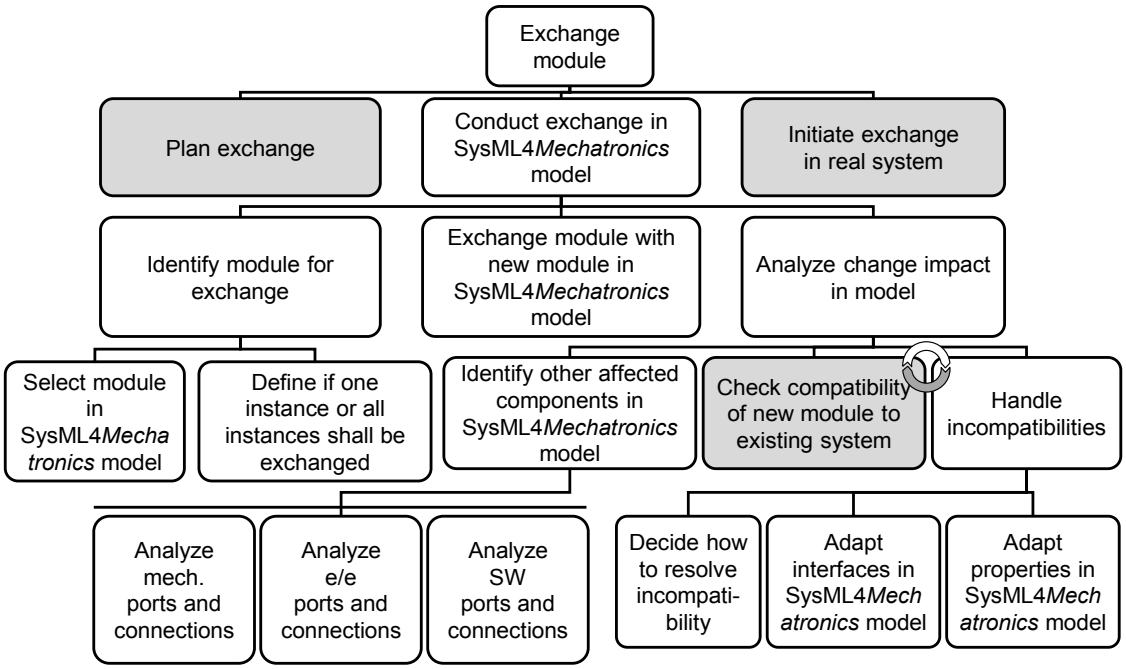


Figure 6-13: HTA of "Exchange module" with SysML4Mechatronics

If an exchange of a module is conducted with help of SysML4*Mechatronics* (Figure 6-13), the module for exchange can be identified within the interdisciplinary model. As each modeling element in SysML4*Mechatronics* represents an instance of the respective class (cf. chapter 5.1.1), it can be chosen to exchange only one specific instance or all instances of that class, e.g., through an updated version of the module. By exchanging the module in the SysML4*Mechatronics* model, the interdisciplinary dependencies and thus, the change impact, can be analyzed. Therefore, the ports in the different disciplines have to be analyzed regarding their compatibility to the existing system. As described in chapter 5.4.2, a (semi-)automatic compatibility analysis (sub-task Check compatibility of new module to existing system") and management (sub-task "Handle incompatibilities") requires, next to the SysML4*Mechatronics* model, formal methods and a respective framework. If incompatibilities arise, the developers have to decide, if they can be eliminated through a respective adaption of the interfaces and/or properties or if a different module would be more convenient for the exchange. Thus, based on the compatibility analysis, the developers can estimate in a more sophisticated way, which impact the planned change has and initiate the change in the real system. Through the interdisciplinary perspective in the model, faults through unconsidered interdisciplinary dependencies can be reduced and thus, time-consuming and costly hold-ups can be prevented.

As an example in the case study of the pallet transportation system, the exchange of a barcode-scanner in the handling station to a RFID-reader (as shown in Figure 6-4) shall be analyzed. Such changes occur frequently during the usage phase of production systems, as, e.g., new technologies are available (from barcode to RFID) or the customer requires integrating new functionalities (more specific product information that can be stored in an RFID-tag in comparison to a barcode). Figure 6-14 shows the primary version of the handling station with the barcode-scanner included. The barcode-scanner module has two mechanical connection ports (ports B1, B2), an Ethernet communication port (B3), a port for the power supply (B4, marked as conjugated as the default direction of power supply ports was defined as *Out*) and a port representing the software interface (B5). The electrical ports are connected to components outside of the module, however, as they are required for the functionality of the module, they have been defined as mandatory (*IsMandatoryFor: Scan barcode*). The software port provides the result of the scanned barcode (13-digit number), which then can be used by the software block of the handling station module.

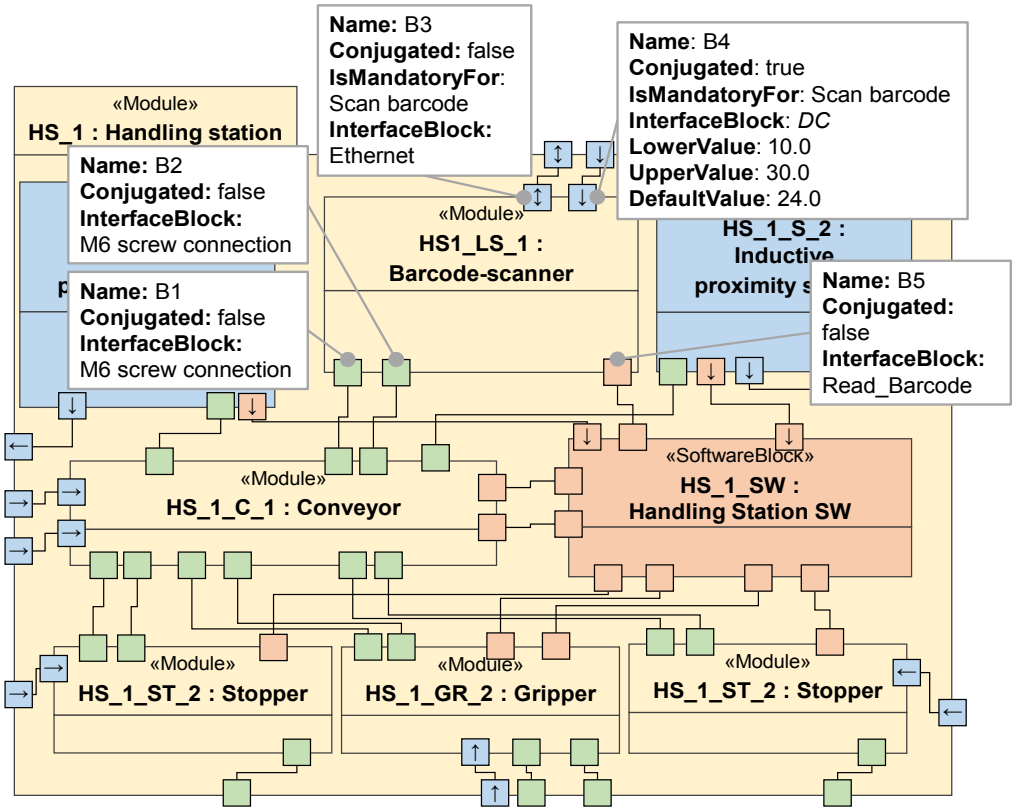


Figure 6-14: Handling station with barcode-scanner in SysML4Mechatronics [KeFV18]

In order to analyze the change from a barcode-scanner to an RFID-reader, the respective module is exchanged in the SysML4*Mechatronics* model (Figure 6-15). As the interface analysis shows, only the communication port of the new RFID-scanner can be used directly with the existing system. As the RFID-scanner is microcontroller-based (note that

the used RFID-scanner in the pallet transportation and handling system of the case study is a prototypical module for research purposes, cf. Figure 6-4), it requires a power supply of 5 V DC. Thus, a respective power supply or converter has to be integrated in the system. The mechanical ports of the RFID-reader are designed smaller, as the entire module is lighter than the barcode-scanner. A respective modification of the conveyor fixture, e.g., new mounting points, have to be planned for the exchange. The software port requires the most extensive adaption. While the barcode-scanner only delivered a 13-digit number, RFID-tags can contain multiple information, e.g., regarding the required processing steps for the specific product. Thus, the information from the RFID-scanner can be used in the software, e.g., for triggering the respective tool for handling the product. This, however, requires not only the adaption of the software interface to the RFID-reader, but also possible new interfaces to subsequent modules for processing the products on the pallet. The extent of the required adaption, however, is dependent on the goal of the developers. If the RFID-scanner shall only replace the barcode-scanner to identify the current pallet, the adaption has a lesser extent.

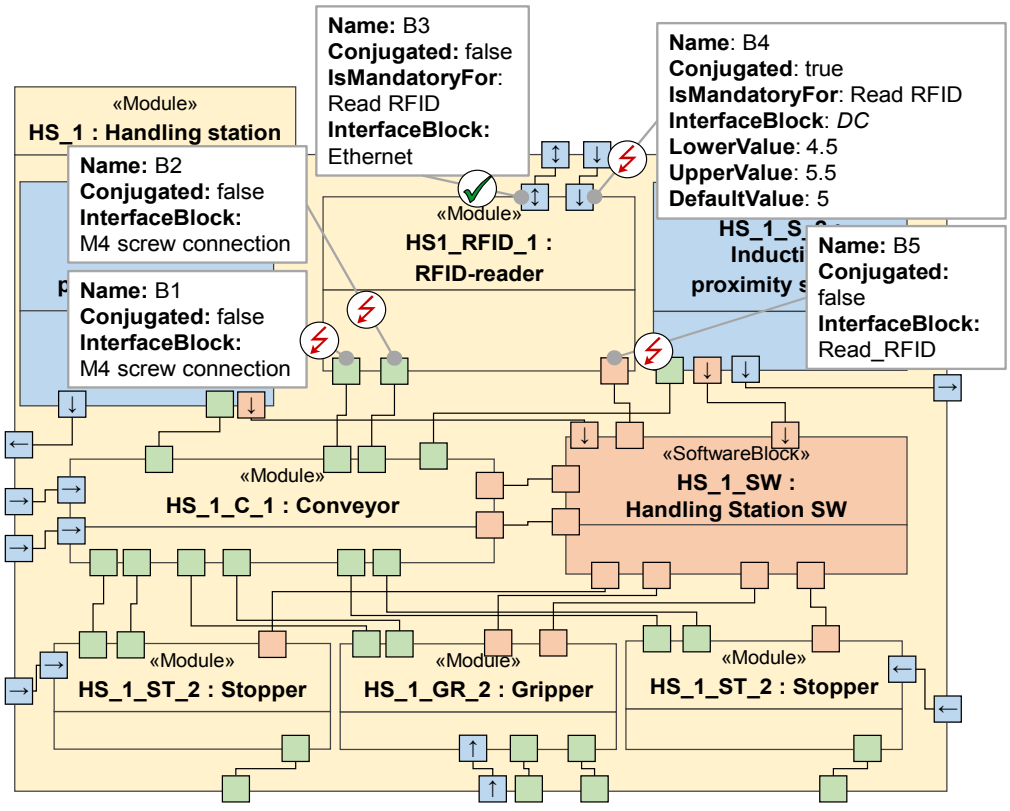


Figure 6-15: Handling station with RFID-reader in SysML4Mechatronics [KeFV18]

Thus, based on the analysis in the model, an estimation of the required effort to fulfill the change can be conducted in the different disciplines and the exchange can be performed in the real system. If larger modules, e.g., the entire handling station, shall be exchanged, a multitude of ports has to be analyzed. Therefore, the SysML4*Mechatronics* model is

transformed to a formal ontology and therein a (semi-)automatic compatibility analysis, as described above, can be conducted. In addition, further compatibility rules (cf. inherent and project-specific compatibility rules in chapter 5.4.2) as for example the maximum mass of the entire module can be checked (semi-)automatically.

The results of the modeling process were discussed with domain experts in an interview, in order to gain qualitative insights regarding the usability of the approach and as basis for further improvements of the implementation. The experts had knowledge of UML and were also familiar with the basics of SysML. Thus, they were able to rate the specific characteristics of SysML4*Mechatronics* and not only the advantages/ disadvantages of model-based approaches in general. Although the modeling of the system was conducted by the author of this thesis, the experts stated that it was easy for them to follow the process and that the modeling editor gave a clear depiction of the components from the different disciplines in the overall mechatronic system. As positive aspects, the experts stated that the usage of *SysML4Mechatronics* as a common description language is well suited to improve the information exchange among the involved disciplines. Especially in the early phases of multidisciplinary developments, an approach to share the information of all engineers among each other is especially important, as an insufficient communication leads to a significant increase of the number of required changes and of the resulting change impacts [FGNI00]. As a further benefit of the approach, the consistent data model was mentioned, which enhances fault detection, consistency checks and change management among the components from the different involved disciplines in the overall mechatronic system design.

On the negative side, crashes and some missing functions (e.g., copy-paste) of the software were stated. Furthermore, the experts stated that it would be beneficial, if an overlay of certain other diagrams, e.g., piping and instrumentation diagrams, in the mechatronic model would be possible.

6.3. Usability inspection of the implementation

Besides the feasibility of the approach, the usability is a crucial factor for the success of the modeling approach. This chapter describes the first part of the usability evaluation, in which the SysML4*Mechatronics* editor is analyzed regarding potential drawbacks for its usage. In the following chapter (chapter 6.4), the editor then is utilized for a user evaluation.

6.3.1. Cognitive walkthrough

The cognitive walkthrough technique [PLRW92] is used to analytically evaluate the design of a user interface in an early phase before an empirical test with real users is possible. Thus, the method enables, without the involvement of real users, to analyze the perspective of potential users and identify problems that could arise through the interaction with the system. The cognitive walkthrough should be carried out by the developer of the software and a small group of experts, who are familiar with the domain and can adopt the perspective of a potential user. As a preparation of the walkthrough the following aspects have to be defined [PLRW92, WRLP94]:

- Definition of the characteristics of the potential user group, e.g., which knowledge they possess, which experience they have, or which affinity to similar systems they have
- Description of typical, representative tasks that shall be fulfilled through the usage of the system
- Definition of the ideal sequence of actions that is required to execute the desired task

In the next step the actual walkthrough is performed. Thereby, each task is analyzed regarding the following questions [PLRW92, WRLP94]:

- *Will the user try to execute the right steps of action to fulfill the task?* The user has to realize, which steps he has to undertake in order to achieve the goal. Therefore, the mental model of the user has to match to the system of usage.
- *Will the user recognize that the right steps of action are available in the system?* The user has to notice that he can conduct the steps in the system, i.e., the option for action has to be visible in the system.
- *Will the user recognize that the right steps of action will lead to the desired effect?* The user has to be able to link his aim with the steps of action in the system.
- *Will the user see the progress after executing the correct action step?* The user has to recognize that his correct steps of action lead to the desired effect, e.g., through an appropriate feedback of the system.

The cognitive walkthrough team answers each of the described questions and notes them in an according questionnaire. If all questions can be answered positively, no changes to the user interface are required. If a task, however, is not clear, e.g., a user would not recognize how to execute the step, potential difficulties and possible solutions are discussed by the cognitive walkthrough team. The walkthrough is then executed further, as if the task was fulfilled. After the cognitive walkthrough is completed, the user

interface has to be adapted according to the identified problems for the execution of the specific tasks [WRLP94].

6.3.2. Results of the cognitive walkthrough and refinement of the implementation

The cognitive walkthrough for the SysML4*Mechatronics* editor (cf. chapter 6.2.1) was conducted in November 2014 by a team of five experts from the Institute of Automation and Information Systems at the Technical University of Munich. The team included engineers, a psychologist with in-depth knowledge of the cognitive walkthrough methodology and user interfaces in the machine and plant manufacturing industry, as well as the author of this thesis. At the beginning of the cognitive walkthrough, the technique was explained to all participants and questionnaires were handed out, in which the participants could note their answers for each step of the walkthrough. In the following, the different aspects of the SysML4*Mechatronics* cognitive walkthrough and its results are described.

Definition of the user group

As preparation for the evaluation process, the potential user group of the SysML4*Mechatronics* editor was defined and discussed with the participants of the cognitive walkthrough. This step is very important, as the cognitive walkthrough team shall put itself in the position of a user and rate the software from this perspective. According to the goals of SysML4*Mechatronics*, potential users of the system are developers of mechatronic systems, i.e., mechanical, electrical, and software engineers. Their ages and work experience may differ. However, it can be assumed that this user group has good knowledge of using office PCs (Microsoft Windows) and typical computer programs (e.g., Microsoft Office). Thus, the potential users are familiar with standard Windows commands and symbols (e.g., an “x” symbol for closing a program). Regarding the development process, it can be assumed that the engineers have an in-depth knowledge of their discipline and a general understanding of the relations in mechatronic systems. However, the engineers do not necessarily have a profound knowledge of model-based development processes or modeling languages such as SysML.

In order to gain a sufficient number of participants, the SysML4*Mechatronics* user evaluation (cf. chapter 6.4) is conducted with master students from mechanical engineering. This group of participants for the evaluation is comparable to the described group of potential users. However, the students have only limited industrial work experience in the development of mechatronic systems, which, however, is also true for newly hired engineers in industry. Due to the mandatory courses, all participating students

have good knowledge of IT and (at least basic) knowledge of automation technology, mechatronics, and model-based development.

Furthermore, all participating students of the user evaluation and also the experts of the cognitive walkthrough receive a training of the SysML4*Mechatronics* approach in order to understand the approach (independently of the tool that is used).

Task selection

The model-based development of mechatronic systems is a complex task, requiring the specification of many components, their properties and interrelations. The SysML4*Mechatronics* editor is a prototypical research tool that supports the modeling process. The evaluated version included all requisite functionalities, except for the automatic integration of components/properties from classification systems through model libraries (e.g., an eCl@ss library). At the time of the cognitive walkthrough, the SysML4*Mechatronics* editor has only been used by the author of this thesis but not by 'real' users.

Based on these given factors, the following four tasks were chosen for the cognitive walkthrough, which are exemplary for the development process with SysML4*Mechatronics* and comprise several steps of action to fulfill them correctly:

1. Creating a new SysML4*Mechatronics* project, creating a discipline-specific (E/E) component, and saving the project (7 steps of action required)
2. Specifying (manually) the properties and functionalities of a component in the SysML4*Mechatronics* editor (35 steps of action required for four properties and one functionality)
3. Selecting three components from the library and connecting the components' ports (18 steps of action required)
4. Merging existing components in the modeling area to a functional module (3 steps of action required)

According to [WRLP94, p.6] "[…] the analyses should be limited to a reasonable but representative collection of benchmark tasks". Thereby, some "benchmark tasks should be sampled from the core functionality of the application, that is, the basic operations the system is intended to support. In addition, some tasks should be included that require combinations of these basic functions" [WRLP94, p.7]. Accordingly, two simple and two complex task scenarios were selected for the interaction with the system. By using a simple task scenario in the beginning, the cognitive walkthrough team can gain experience with the method, which is necessary for evaluating the more complex tasks [WBJF92]. Even though the tasks were chosen as realistically as possible, the cognitive walkthrough can only cover an excerpt of the complexity for modeling a mechatronic

system. Hence, every cognitive walkthrough depicts a simplification, considering mostly the direct path to fulfill a task. This can consequently also lead to not identifying all potential problems of the user interface [WBJF92].

Evaluation results and refinement of the implementation

While performing the cognitive walkthrough the participants took notes in the provided questionnaires for each action step regarding the questions described in chapter 6.3.1. As the evaluators of a cognitive walkthrough often notice additional problems, which are not directly relevant for the current action step [WBJF92], the used questionnaire provided a space for additional comments. Thus, these issues were captured but did not disturb the walkthrough of the respective task scenario. The identified additional problems were then discussed afterwards.

During the walkthrough, many aspects of the SysML4*Mechatronics* editor have been discussed and served as input for the further improvement of the editor before the user evaluation was conducted. As many steps did not reveal any potential problems and on the other hand many detailed issues were detected, in the following only some of the key findings of the cognitive walkthrough can be described.

In task scenario 1, the steps for creating a new SysML4*Mechatronics* project and saving a project are comparable with other standard PC programs, e.g., creating in Microsoft Word a new document and saving it. Thus, only minor issues, e.g., an unclear differentiation between the labels *project* and *diagram* in the menu were stated by the evaluation team (A project in SysML4*Mechatronics* can contain several diagrams to handle the complexity of large systems or depict different variants of the system). The creation of a discipline-specific component (in case of the walkthrough a E/E component was created) is conducted in the SysML4*Mechatronics* editor by selecting the desired component type from the ribbon menu (Figure 6-16) and placing it with a left mouse click in the modeling area (cf. Figure 6-2).

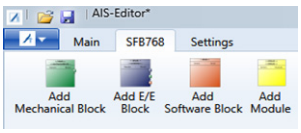


Figure 6-16: Excerpt of the SysML4Mechatronics ribbon for creating components

This procedure is also common for many computer programs (e.g., Microsoft PowerPoint). Thus, it can be easily understood by computer-experienced users. Task scenario 1 shows that it is important to define the potential user group beforehand of the cognitive walkthrough, as the described steps would probably cause major difficulties for users who have no experience with computers.

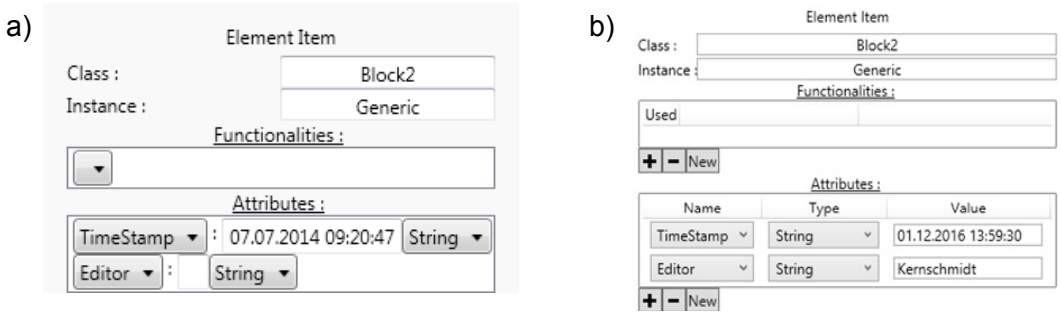


Figure 6-17: Specification of component properties in the SysML4Mechatronics editor a) original interface, b) revised interface after cognitive walkthrough

In task scenario 2, the specification of component and port properties was analyzed in the SysML4*Mechatronics* editor for an inductive sensor. The task showed that even a relative simple component – one functionality, one component and three port properties were considered for the cognitive walkthrough – required 35 steps of action. In general, properties in the SysML4*Mechatronics* editor can be created and specified by clicking on the respective component and then making the desired adaptions in the ToolView (cf. Figure 6-2). In the version of the editor, which was used for the walkthrough, the specification of properties in the ToolView was not intuitively enough and required many additional steps. Figure 6-17 shows on the left side the interface, as it was presented during the cognitive walkthrough. The main drawback thereby is that only existing properties and functionalities can be chosen from the presented dropdown menus. In case that a new property shall be defined, e.g., a property for the maximum operating ambient temperature, this has to be done by defining the property independently from the specific component (new properties can be created in the ribbon menu at the top of the SysML4*Mechatronics* editor, cf. Figure 6-2). The property is then saved to the library and can then be assigned in another step to the specific component. This procedure is not directly clear for a potential user of the system, requires several steps and thus, is time consuming. Based on the cognitive walkthrough, this part of the SysML4*Mechatronics* editor was revised thoroughly. In the new version (right side of Figure 6-17), the user has the possibility to add, delete, and create new functionalities and properties directly in the ToolView and thus, does not have to search for the respective menu items. Additionally, if a user creates a new property or functionality, it is added on the one hand directly to the respective component and on the other hand is added to the library for potential future reuse. In the creation dialog, the SysML4*Mechatronics* editor, furthermore, makes suggestions on existing properties, based on the input of the user. In this way, the multiple creation of the same property shall be avoided. Although these improvements of the SysML4*Mechatronics* editor facilitate the process, the manual specification of requirements remains a time-consuming and potentially error-prone task. Thus, the automatic inclusion of properties from existing classification systems (not analyzed in the cognitive walkthrough) can significantly contribute to a shortening of the modeling

process with SysML4*Mechatronics*. In addition, the usage of classified components and properties prevents a faulty specification, which can lead to further delays in the development process or even to failures of the real system.

Task scenario 3 deals with the integration of existing components from the SysML4*Mechatronics* library in the modeling area and the port-connection of these components. The cognitive walkthrough task considered the selection and integration of three components (two E/E and one SW component) from the library and the connection of four ports. Although the task scenario has 18 required steps of action and thus, is quite complex, the walkthrough-team rated most steps as intuitively. However, a necessary clearer labeling, e.g., regarding the different disciplines of the components in the library, was identified. A major suggestion of improvement concerned the connection of ports. According to the SysML4*Mechatronics* approach, only ports from the same discipline can be connected (e.g., an E/E port can only be connected to another E/E port). An unexperienced user would recognize this in the SysML4*Mechatronics* editor first, when he tries to set the connection, i.e., clicks on the second port (from a different discipline), and not before pulling the connection line. In this case, after clicking on the second port, the SysML4*Mechatronics* editor would not set the connection. Based on the cognitive walkthrough, this issue was facilitated by disabling all ports from other disciplines as soon as the user starts to pull a connection line (Figure 6-18). Thus, the user receives directly a visual feedback to which port he can connect the currently selected one.

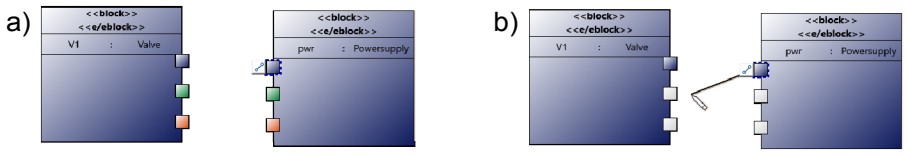


Figure 6-18: a) E/E components with ports from different disciplines, b) disabled ports of the other disciplines while connecting an E/E port

Task scenario 4 shows, in comparison to task scenario 2, that the integration of components in the modeling area to a functional module can be conducted already very efficiently in SysML4*Meachtronics*. The desired components only have to be selected by holding the left mouse button clicked (equivalent to selecting several objects in, e.g., Microsoft PowerPoint), then the right mouse button has to be clicked and the item “Move to Module” has to be selected from the context menu by clicking on it with the left mouse button. After these three steps of action, the SysML4*Meachtronics* editor creates automatically a module with the selected components and stores the module in the library for future reuse. Furthermore, the user has the possibility to name the module accordingly, attach ports on the module border, and use it accordingly in the overall project.

The cognitive walkthrough revealed several major and minor issues for the usability, as described above, and thus, the SysML4*Meachtronics* editor was refined accordingly before the actual user evaluation was conducted. For the larger issues, e.g., as described for task scenario 2, it is often more effective to consider a new solution than trying to fix the problem locally [WRLP94].

6.4. User evaluation

After the usability of the SysML4*Mechatronics* editor has been evaluated analytically through a cognitive walkthrough and the implementation has been revised according to the findings of the walkthrough (cf. chapter 6.3.2), the usability was evaluated with potential users. Therefore, two studies with voluntarily participating students were conducted in November 2015 and in November 2016 in the context of the Master's course *Industrial Automation II* at the Technical University of Munich. In total, a number of 52 participants (26 participants per year) joined the studies. All participants were Master's students from mechanical engineering (including different specializations such as Mechatronics and Information Technology, Automotive Technology, and Production and Logistics). All participants were familiar with the basics of IT (through their study curriculum) and received within the scope of the course *Industrial Automation II* an introduction to mechatronics and model-based development.

For the evaluation of SysML4*Mechatronics*, the participants received in the first step a theoretical training of the approach through a presentation and exercises (in total two hours). Furthermore, additional material was handed to the participants as preparation for the utilization of the SysML4*Mechatronics* editor. The practical evaluation with the editor was carried out one week after the theoretical training. For the practical evaluation, first a questionnaire regarding the participants' prior knowledge in the area of programming/modeling was conducted. Then three assignments (chapter 6.4.2) were carried out with the SysML4*Mechatronics* editor and subsequently, the participants filled out a questionnaire according to the widespread Technology Acceptance Model and the usability norm ISO 9241-10 [Isoi95] (chapter 6.4.3) to evaluate the approach. The entire user evaluation took two and a half hours and according to the required protection of privacy, all answers of the questionnaire were handled anonymously for the evaluation.

6.4.1. Technology Acceptance Model

The success of a new technology or approach in practice is strongly dependent on its users' acceptance. According to Davis [Davi89], this acceptance is significantly dependent on how the users perceive the *usefulness* and the *ease of use* of the technology or approach. Thereby, perceived usefulness is defined as "the degree to which a person

believes that using a particular system would enhance his or her job performance" [Davi89, p.320]. Accordingly, perceived ease of use refers to "the degree to which a person believes that using a particular system would be free of effort" [Davi89, p.320].

Based thereon, the Technology Acceptance Model (TAM) [Davi89] has been developed to predict the user acceptance. Originating from the evaluation of information systems, the TAM has been applied for various application areas, also for modeling approaches (e.g., by Moody et al. [MSBS02]) and is nowadays one of the most widespread methods for measuring user acceptance [GhLB12]. In an extension of the TAM, called TAM2 ([VeDa00], Figure 6-19), it was shown that social influence, e.g., subjective norms, as well as cognitive instrumental processes, e.g., job relevance, also have a significant influence on the user acceptance [VeDa00]. In order to evaluate the user acceptance with the TAM a questionnaire was developed by Davis [Davi89]. Based on this original TAM-questionnaire, the contained statements were adapted for SysML4*Mechatronics*. For both areas, *perceived usefulness* and *perceived ease of use*, six questions were defined (each with a seven level scale ranging from *'I totally disagree'* to *'I totally agree'*).

Regarding the *perceived usefulness*, the following measures were included:

- Using SysML4*Mechatronics* makes interdisciplinary models more expressive
- By using SysML4*Mechatronics*, the components' interdisciplinary dependencies can be estimated more effectively
- By using SysML4*Mechatronics*, properties of components and modules can be specified more effectively
- SysML4*Mechatronics* facilitates the reuse of components and modules

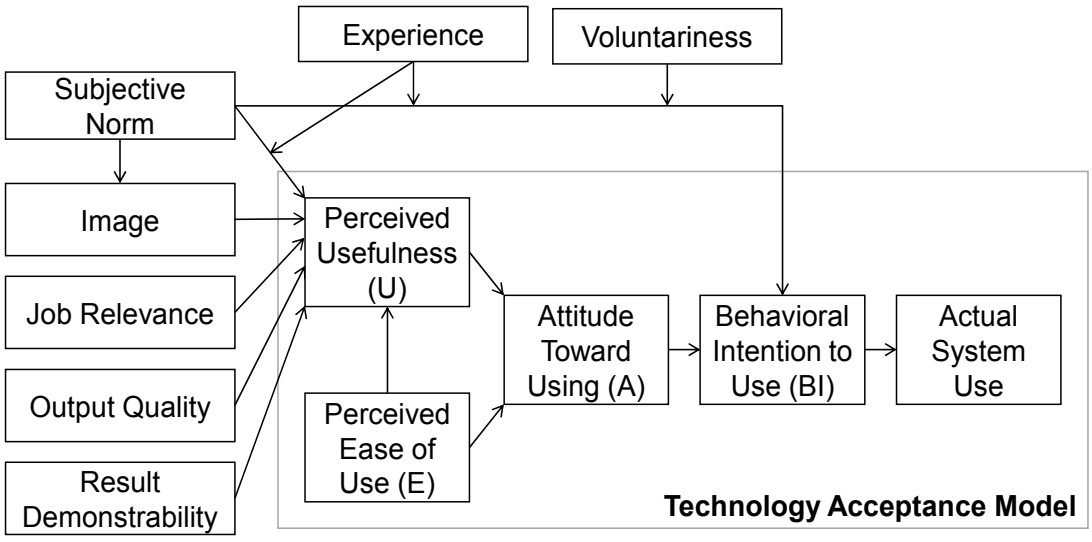


Figure 6-19: Extended Technology Acceptance Model according to [VeDa00]

- Using SysML4*Mechatronics* would quicken the development of mechatronic systems
- SysML4*Mechatronics* models would be useful in later lifecycle phases (e.g., for analyzing change influences)

For the *perceived ease of use*, the questionnaire included:

- It is easy for me to learn SysML4*Mechatronics*
- It is easy for me to identify the different disciplines and interfaces in SysML4*Mechatronics* diagrams
- It is easy for me to create interdisciplinary modules in SysML4*Mechatronics*
- It is easy for me to use SysML4*Mechatronics* for the engineering of mechatronic systems
- Using SysML4*Mechatronics* diagrams, it would be easy for me to talk about the system to other disciplines (e.g., Electrical Engineering, Computer Science)
- It would be easy for me to extend my knowledge in SysML4*Mechatronics*

In addition to these questions regarding perceived usefulness and perceived ease of use, questions regarding the usability according to the norm ISO 9241-10 [Isoi95] were included into the questionnaire. The scope of the questions includes the *notation, suitability for the task, self-descriptiveness, controllability, conformity with user expectations, error tolerance, suitability for individualization,* and *suitability for learning*. In total, the evaluation questionnaire consisted of 59 questions, of which one question was designed as open question where participants could specify their positive and negative aspects of using SysML4*Mechatronics*. The entire questionnaire (in German language) can be found in the appendix.

6.4.2. Evaluation assignment

The user evaluation contained three assignments. The first assignment (*assignment 0, modeling of an inductive sensor*) served as recapitulation of the theoretical training and was conducted conjointly.

The second assignment (*assignment 1*) comprised of modeling a cylinder, consisting of a mechanical housing, two inductive sensors (which could be reused from assignment 0) indicating the cylinder position, one valve for extending/retracting the cylinder and an according software block for controlling the cylinder. This assignment served as a training for the participants, in order to get familiar with the different functionalities of the SysML4*Mechatronics* editor. During this assignment, questions of the participants regarding the editor were answered, in order to avoid possible ambiguities.

In the following assignment (*assignment 2*) a system with two silos had to be modeled by the participants (a schematic representation of the system is shown in Figure 6-20). The system is composed of various components from the different considered disciplines and according modules that facilitate the reuse during modeling should be created. Therefore, the participants received the following description of the system as starting point:

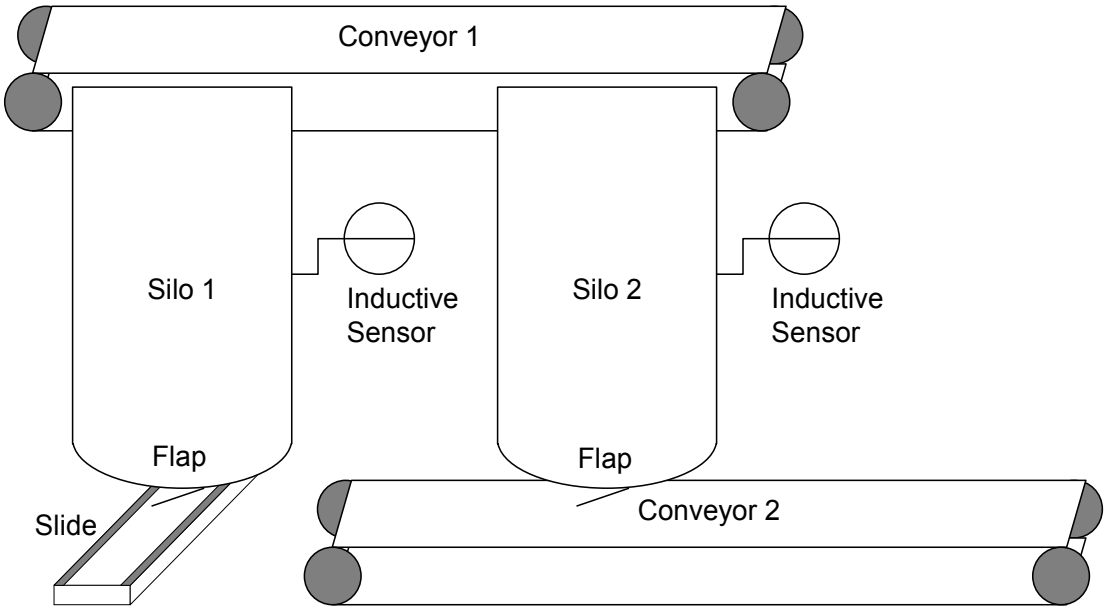


Figure 6-20: Silo system of the evaluation assignment

"The system contains two silos (height: 3m), which can be filled through conveyer 1. Each silo possesses an inductive sensor (modeled in assignment 0) in order to identify metal pieces, which are accidently located in the bulk material. At the bottom of each silo a controllable flap is located (boolean variable: OpenFlap), which can guide the bulk material to conveyer 2 (from silo 2) or to a mechanical slide (from silo 1). Furthermore, the silos possess a silo housing to which the inductive sensor, the flap as well as the conveyer or the slide are attached to. Via a software component, which receives the value of the inductive sensor and sets the variable of the flap, the silo is controlled. Towards external parts, the software component provides the function (operation) silo control.

The two identical conveyers (length: 7m, mass: 250kg) consist of a motor (boolean variable: MotorOn) and the mechanical parts of the conveyor which are attached to it. All actuators and sensors are powered with 24 V DC. Assignment:

- *In a first step, define (on paper) the type of each component (Mech, E/E, SW) and the design of the modules.*
- *Model the described system. If possible, try to realize a high degree of reusability. Specify the given attributes (properties) in the model.*

Note: Further components (e.g., power supply) of the system are not considered in this exercise. In this case guide the ports to their respective module border, so that they can be connected in later steps."[12]

After the participants finished the task, they evaluated the SysML4*Mechatronics* approach in a questionnaire according to chapter 6.4.1. In the following chapter, the main results are presented.

6.4.3. Evaluation results

The evaluation was carried out with two groups of participants in November 2015 and November 2016 with 26 participants per year. In 2015, the average age of the participants was 23.7 years and the ratio of female participants was 26.9%. The average participants' age in 2016 was also 23.7 years and the ratio of female participants 23.1%, respectively. Figure 6-21 shows a summary on the participants' self-assessment, which was surveyed at the beginning of the evaluation, regarding programming (C/C++) and modeling (UML/SysML). Further languages (e.g., Java, IEC-61131-3, Matlab/Simulink) were mentioned only by a few individual participants. While the two groups have similar prior knowledge regarding the programming languages C/C++, the number of participants who have no knowledge of the modeling languages UML and especially SysML is much higher in 2016.

[12] The original evaluation assignment and Figure 6-20 were provided to the participants in German language and translated to English in the context of this thesis.

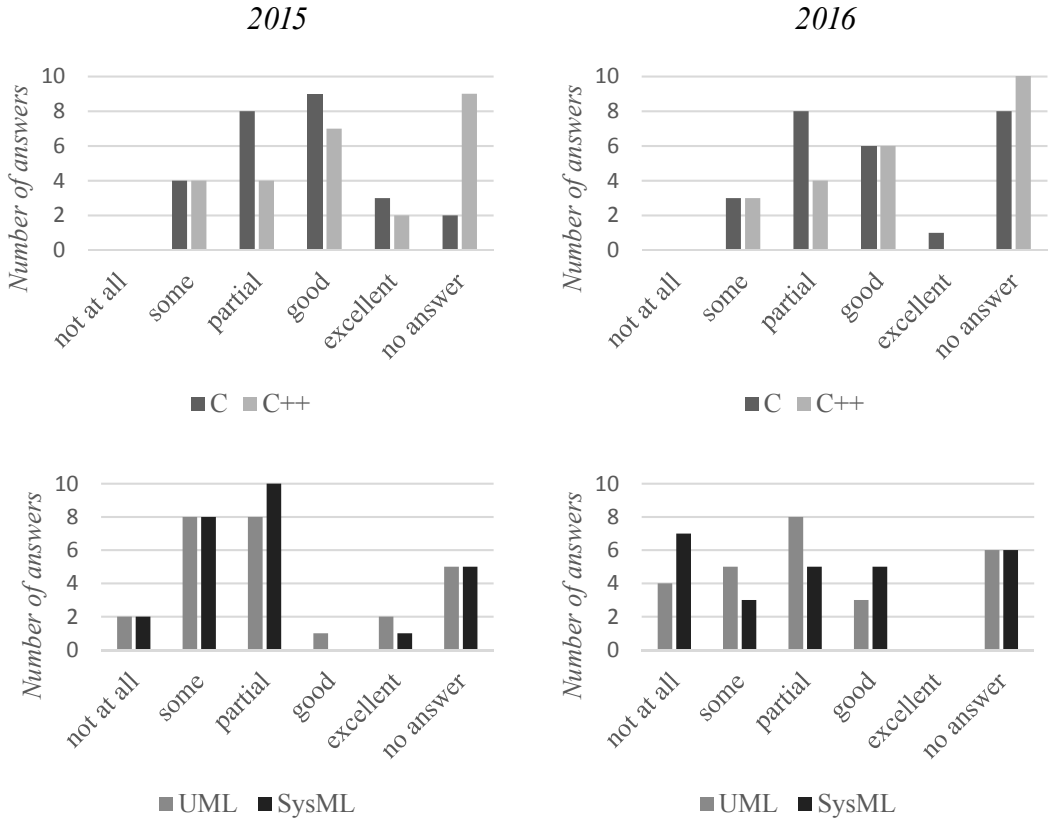


Figure 6-21: Participants' knowledge in programming/modeling languages, multiple answers permitted (left side 2015, right side 2016)

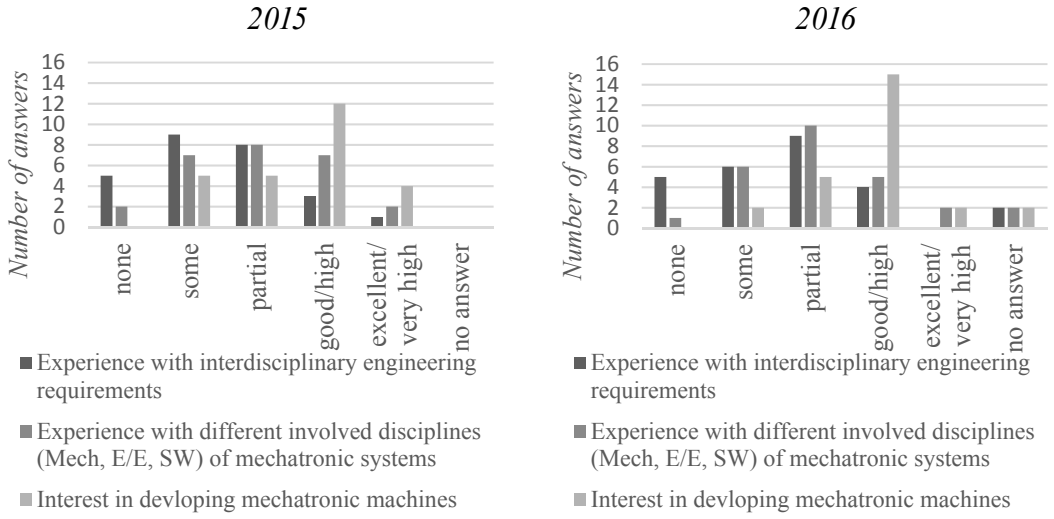


Figure 6-22: Participants' experience with the interdisciplinarity of mechatronic systems and their interest in developing machines (left side 2015, right side 2016)

On the other hand, in 2016 more participants than 2015 state to have good knowledge of SysML. Thus, the group of participants in 2016 was more inhomogeneous regarding their prior modeling knowledge than in 2015. Regarding the participants' experience with

interdisciplinary engineering (requirements, involved disciplines, see Figure 6-22) both groups had similar prior knowledge. 92.3% of the participants in 2015 and 88.5% in 2016 had at least some knowledge of the involved disciplines in mechatronic systems. Finally, the participants' interest in developing mechatronic machines was inquired before the evaluation assignment was executed. In both groups, more than 60% (61.5% in 2015 and 65.4% in 2016) stated to have a high or very high motivation.

Previous works (e.g., [Fran14]) have shown that in evaluations, which are carried out with students that have to join mandatorily, e.g., due to their curriculum, the participants have a lower motivation. Likewise, the TAM2 (cf. chapter 6.4.1, Figure 6-19) has shown that social influence, e.g., voluntariness, has a significant influence on the users' acceptance of an approach. The SysML4*Mechatronics* evaluation was carried out within the scope of the course *Industrial Automation II*. However, it was not mandatory for the students and thus, all participants joined voluntarily.

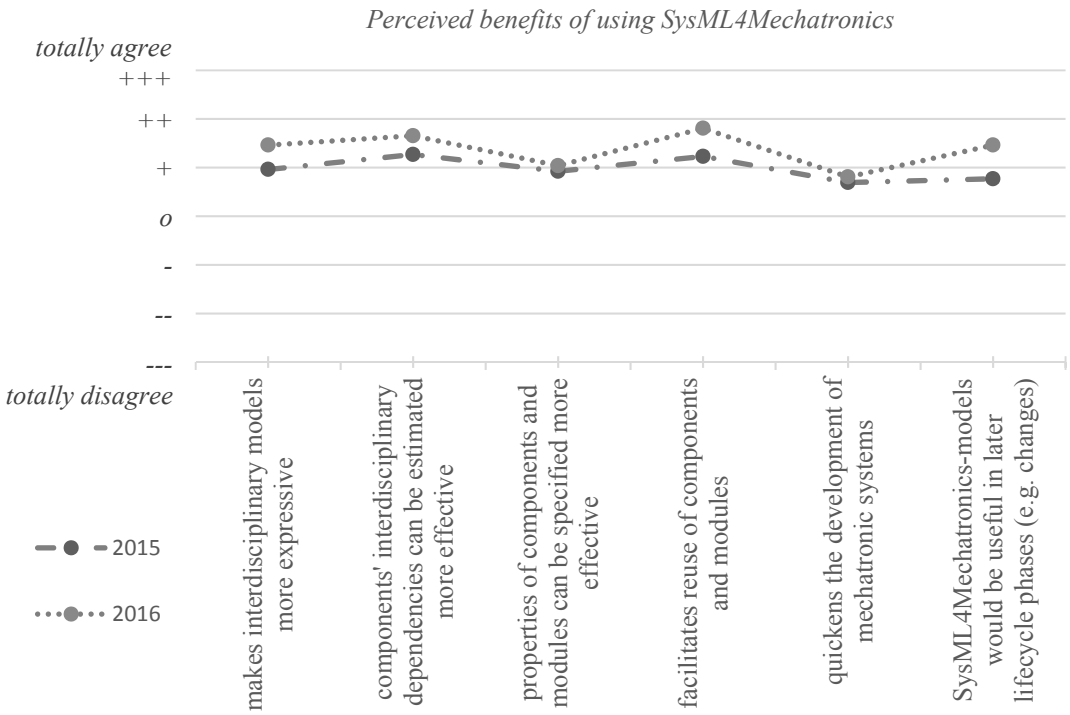


Figure 6-23: Results of the TAM-questionnaire regarding the perceived benefits of using SysML4Mechatronics

After the evaluation assignment, the participants evaluated the SysML4*Mechatronics* approach according to the TAM and ISO 9241-10, as described in chapter 6.4.1. Figure 6-23 shows the mean values of the participants' answers on the perceived usefulness of SysML4*Mechatronics*. In both groups, all questions were evaluated on average positively, i.e., the participants agreed to the postulated statements. 69.2% of the 26 respondents in 2015 and 96.2% of the 26 respondents in 2016 agreed that the components' interdisciplinary dependencies can be estimated more effectively by using

SysML4*Mechatronics* (i.e., answered either *agree*, *strongly agree* or *totally agree*). In addition, the majority (76.9% in 2015 and 96.2% in 2016) believes that SysML4*Mechatronics* facilitates the reuse of components and modules. These two questions in the questionnaire received the highest average agreement of the participants in both groups. The comparison of the two groups shows that the perceived usefulness of SysML4*Mechatronics* was higher for the group in 2016. Between the evaluation in 2015 and the evaluation in 2016 the SysML4*Mechatronics* modeling editor has been improved, based on the results of the first user evaluation as well as through discussions with experts in industry, which could reflect the better evaluation. However, as the students who participated in the evaluation were different in 2015 and 2016, a direct comparison can not be conducted.

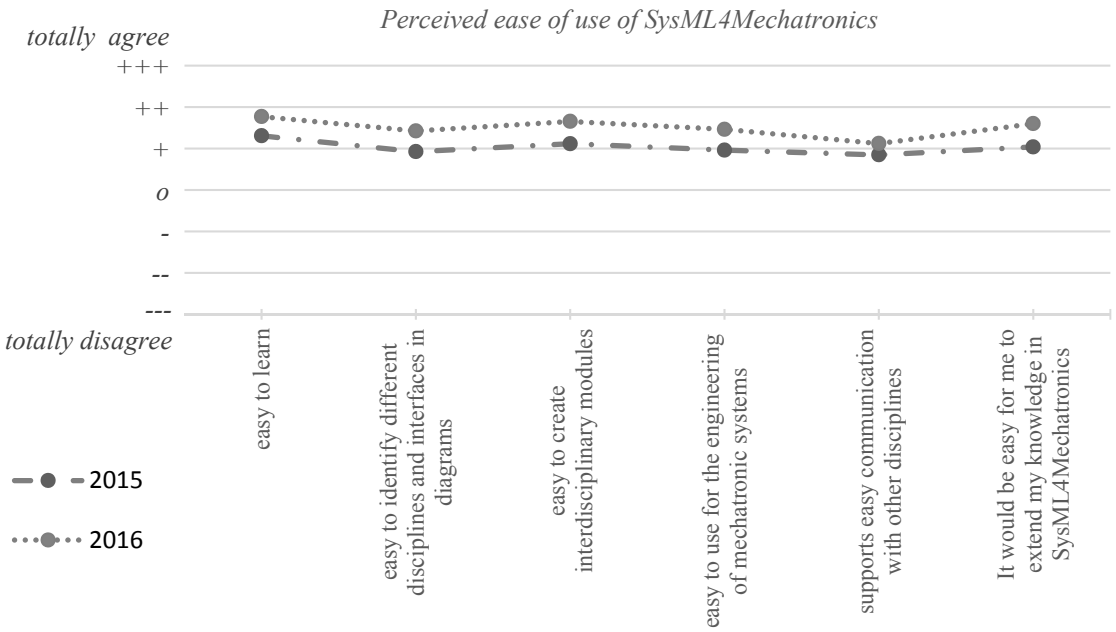


Figure 6-24: Results of the TAM-questionnaire regarding the perceived ease of use of SysML4Mechatronics

The aspects regarding the perceived ease of use of SysML4*Mechatronics* (Figure 6-24) were evaluated by both groups positively, again with a slightly better rating in 2016. The majority of the participants (73.1% of the 26 respondents in 2015 and 96.2% of the 26 respondents in 2016) stated that it is easy for them to learn SysML4*Mechatronics* (i.e., answered to the respective question either with *agree*, *strongly agree* or *totally agree*). For 69.2% of the participants in 2015 and 88.5% in 2016 SysML4*Mechatronics* makes it easy to identify the different involved disciplines and interdisciplinary interfaces. Furthermore, 73.1% of the participants in 2015 and 88.5% in 2016 state that it easy for them to create interdisciplinary modules with SysML4*Mechatronics*.

The evaluation of the perceived usefulness and the perceived ease of use shows that potential users, although they worked for the first time[13] with the approach and the respective modeling editor, assume that SysML4*Mechatronics* would enhance the development process and thereby, require manageable learning effort. As shown in the extended TAM [VeDa00], the perceived usefulness and the perceived ease of use have a direct influence on the users attitude towards using a specific approach, which again has a significant influence on his intention and finally his actual usage of the approach. Thus, through the conducted evaluation of the perceived usefulness and the perceived ease of use two major influencing factors could be verified. However, as the students used the modeling editor only within a well-defined assignment, the conducted evaluation can only be seen as a first step of the evaluation and the long-term perceptions of users towards SysML4*Mechatronics* would require further evaluations after defined periods of time, in which the approach has to be used on a regular basis.

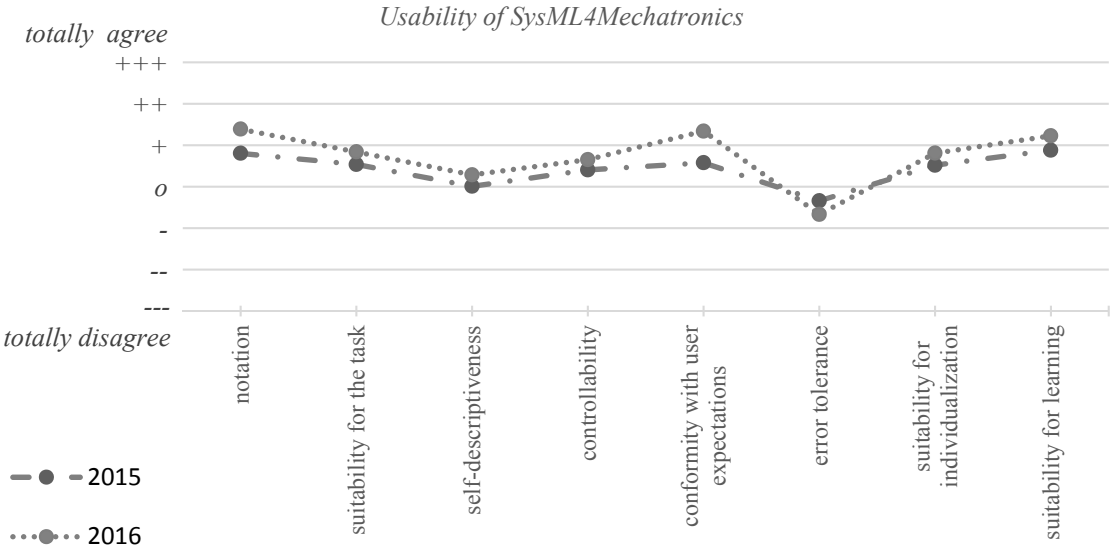


Figure 6-25: Results of the questionnaire regarding the usability of SysML4Mechatronics (aggregated mean values from the respective question blocks)

Next to the questions regarding the TAM, the participants were asked 59 questions regarding usability according to ISO 9241-10 [Isoi95]. Each question block focused on a specific usability aspect of SysML4*Mechatronics* (*notation, suitability for the task, self-descriptiveness, controllability, conformity with user expectations, error tolerance, suitability for individualization,* and *suitability for learning*) and included five to seven questions. Due to space limitations, Figure 6-25 shows the aggregated mean values from

[13] The participants were asked within the questionnaire, if they have worked with SysML4*Mechatronics* before. All participants, except for one student, stated that it was their first time to use SysML4*Mechatronics*.

the respective question blocks. While the majority of the questions were evaluated positively by the participants, the self-descriptiveness was rated indifferent and the error tolerance slightly negatively. An analysis of the single questions concerning self-descriptiveness shows that three of the five questions were evaluated positively (*the software gives a good overview of its functionality*, *the software uses easy comprehensible terms, labeling and symbols*, and *the software offers enough information, which input is allowed or necessary*). However, the two questions *the software offers on demand context specific explanations, which help in the concrete situation* and *the software offers context specific explanations by itself, which help in the concrete situation* were evaluated negatively. As the SysML4*Mechatronics* modeling editor is a prototypical implementation, these features were not in the focus of the implementation. The evaluation shows, however, that it is important to improve these assistance features in future (industrial) SysML4*Mechatronics* tools.

The question block on error tolerance contained five questions regarding the *consequences of a mistake*, *feedback on incorrect input*, *comprehensibility of error messages*, *required effort for correcting mistakes*, and *information on correcting mistakes*. Thereby, the participants indicated that an improvement of each of the aspects is required. Similar to the two negatively evaluated questions on self-descriptiveness the prototypical implementation did not focus on the aspects of indicating and resolving errors during modeling. Therefore, the respective mechanisms and notifications in the tool have to be improved in future.

In contrast, the question blocks on the SysML4*Mechatronics* notation and the conformity with user expectations received together with the suitability for learning the best evaluation results. These evaluated aspects, including complexity, efficiency, and operability of the modeling approach, were the main focus of the prototypical implementation. For example, 75.5% of all participants (2015 and 2016) stated that it is easy to model dependencies (the respective question was asked in the question block *notation*) and 74.5% agree that the usage principle (i.e., how to model the different blocks and modules) is realized consistently (question block *conformity with user expectations*) with SysML4*Mechatronics* in the modeling editor.

Next to the described questions above, in an additional free-text question the participants could give voluntarily further comments on the evaluated approach. As positive aspects, the possible reuse of components and modules was stated (3 participants), the interdisciplinary connections through the ports (2 participants), as well as the possibility to visualize modules as black-box in more complex systems (1 participant). On the negative side, too many crashes of the software were stated (3 participants), some missing/not-working functions, e.g., copy-paste (3 participants). As further improvement,

4 participants stated that the connections between the ports should be colored more clearly and the routing algorithm of the port connections should be improved (e.g., no overlaps), as in complex systems a high amount of connectors exists. Thus, the comments of the participants show on the one hand that the prototypical implementation requires further improvement or that an industrial tool would be required. On the other hand, the positive comments regard the approach directly independently from the used tool.

6.5. Summary

Based on the theoretical definition of SysML4*Mechatronics* in chapter 5, an evaluation concept based on [KrSJ06], which considers the model, language, domain, and user interpretation of the approach, was applied. In order to conduct the evaluations practically, in a first step a prototypical SysML4*Mechatronics* modeling editor was implemented.

The feasibility of the approach was analyzed in a case study of a pallet transportation and handling system, based on the "Testbed for Distributed Control" of the Automation and Control Institute (ACIN) at the Odo Struger Laboratory located at the TU Wien (Vienna, Austria) [LGMS13]. Through Hierarchical Task Analyses (HTA) [ADSG71] the required steps for developing systems with SysML4*Mechatronics* in comparison to traditional approaches were examined. The analysis of system modules (three-way switch and a conveyor module in the case study) shows that a 1:1:1 modularization often is disadvantageous and thus, rather a functional modularization in the model, as implemented in SysML4*Mechatronics*, should be conducted. After the interdisciplinary model has been set up during the development of the system, it can be used to analyze changes, which occur frequently during the usage phase of the system. In order to ensure the compatibility a new module with the existing system, SysML4*Mechatronics* can be used to analyze the ports in the different disciplines. For a (semi-)automatic compatibility analysis, SysML4*Mechatronics* has to be coupled with formal methods. The analysis of a change in the model enables developers to estimate the change impact more sophisticated before it is implemented in the real system and thus, decide if the exchange is economically and can be realized in the given time.

The second part of the evaluation focused on the usability of SysML4*Mechatronics*. Therefore, the prototypical implementation was evaluated at first analytically in a cognitive walkthrough. Five experts went systematically through predefined tasks and discussed, from the perspective of potential users, if the users will find and recognize the correct steps of action for the task and get an appropriate feedback, if their action was successful and led to the desired result. Conducting such an analytical evaluation before the modeling editor is tested with 'real' users has the advantage that possible issues, e.g.,

the design of the menu for specifying component properties in the SysML4*Mechatronics* modeling editor, can be recognized in an early stage and improved before the practical evaluation. However, cognitive walkthroughs are a very time-consuming method, as every step has to be discussed by the experts. In the SysML4*Mechatronics* modeling editor for example, one task considering the selection of components from the library and connecting the components' ports required 18 steps of action, which had to be evaluated.

After the cognitive walkthrough, the SysML4*Mechatronics* modeling editor was used in two practical evaluations (in 2015 and 2016) with students. In total 52 voluntary participants conducted an evaluation task and subsequently evaluated SysML4*Mechatronics* according to the Technology Acceptance Model (TAM) [Davi89] and the usability norm ISO 9241-10 [Isoi95]. TAM takes the *perceived ease of use* (req. R 8) and *perceived usefulness* (req. R 9) of an approach into account. Both aspects have a significant influence on the actual intention of participants to use a certain system. For the SysML4*Mechatronics* evaluation, a questionnaire, based on the original TAM questionnaire [Davi89], was developed that contained in each of the two described aspects six questions. Both evaluation groups (2015 and 2016) evaluated the perceived usefulness as well as the perceived ease of use positively.

In addition to the questions based on the TAM, 59 questions on usability were answered by the participants. The results of these questions show that the aspects, which were in the focus of the prototypical implementation (e.g., consistency and simplicity of usage), were evaluated positively by the participants. In contrast, aspects, which are not directly related to the actual modeling approach (e.g., error messages), and thus, not implemented in the prototype, still require improvement in future versions of the modeling editor. In an additional free-text question, the participants suggested further improvements, e.g., to minimize software crashes during the usage.

7. Rating of specified requirements and realized results

In the previous chapters, the interdisciplinary modeling approach SysML4*Mechatronics* was defined and evaluated concerning the specified requirements. After the theoretical definition of the metamodel (chapter 5.1), SysML4*Mechatronics* was implemented in a prototypical modeling editor (chapter 6.2.1) and used in an academic case study to show its feasibility for mechatronic systems (chapter 6.2.2). Based on the developed prototype, a cognitive walkthrough was conducted in order to analyze possible usability-problems (chapter 6.3). The optimized SysML4*Mechatronics* editor was then used in two practical, academic evaluation studies to assess its usefulness and ease of use (chapter 6.4). A final rating of the requirements' fulfillment through the different parts of the evaluation is summarized in Table 6.

In order to ensure extensibility and integration with other approaches, the SysML4*Mechatronics* metamodel was developed in accordance to the provided extension mechanisms of SysML and thus, represents a SysML profile as demanded through requirement R 1. The profile enables through specific stereotypes to model components from mechanics, electrics/electronics, and software in one integrated model (req. R 2) as well as their ports and connections (req. R 3). Through the implementation of a prototypical modeling editor, the practical usage of SysML4*Mechatronics* according to its metamodel was proven. The editor offers additional features, e.g., filters, to reduce the complexity. The cognitive walkthrough revealed that the port concept was implemented correctly according to the metamodel, however, from a usability perspective a further filter to show the user only the permitted connection possibilities would be favorable. The implementation was enhanced accordingly and thus, requirement R 7 can be rated as fulfilled.

In the case study of a mechatronic production system, the modularization on different levels of detail was analyzed in depth. According to requirement R 4, a functional modularization of the system could be achieved with SysML4*Mechatronics*, which contains on the one hand all required components in the model and on the other hand can be implemented in the real system with flexible module borders in each discipline. Therefore, the concept of mandatory ports was introduced in SysML4*Mechatronics*. The use case showed, furthermore, the exemplary usage of classification systems to integrate standardized components and properties in the model (req. R 5). In comparison to stereotypes, model libraries can be extended and adapted flexibly without a required change of the metamodel. As shown in chapter 5.3.2, the combination of several classification systems is required, to represent all required aspects in the model. Through

Table 6: Summary of the requirements fulfillment of SysML4Mechatronics

Req.	Description	Proof-of-concept implementation and case study	Usability evaluation	Overall evaluation
R 1	SysML4*Mechatronics* shall be defined as extension of SysML, using the provided profile mechanism	●	●	●
R 2	SysML4*Mechatronics* must be defined as an interdisciplinary modeling language, integrating the discipline-specific components from mechanical, electrical/electronic, and software engineering.	●	●	●
R 3	SysML4*Mechatronics* has to extend the SysML port concept to represent the structural interfaces of the different disciplines and show (interdisciplinary) dependencies between the components in mechatronic systems.	●	●	●
R 4	SysML4*Mechatronics* has to support an interdisciplinary, function-oriented modularization of the system in the model that does not require a 1:1:1 modularization during realization.	●	●	●
R 5	SysML4*Mechatronics* must enable the integration of standardized properties and components from existing classification systems through model libraries.	●	n.a.	●
R 6	SysML4*Mechatronics* models shall be used as basis to enable a compatibility analysis with formal methods.	◑	n.a.	◑
R 7	The implementation of a SysML4*Mechatronics* editor has to support the user by offering an interdisciplinary, structural view on the system in one diagram type with respective filters for the specific views.	●	●	●
R 8	The SysML4*Mechatronics* modeling editor must achieve a positive perceived ease of use through potential users.	n.a.	●	●
R 9	SysML4*Mechatronics* must achieve a positive perceived usefulness for the interdisciplinary development of mechatronic systems through potential users after utilizing the approach.	n.a.	●	●

● Fully satisfied by the approach; ◑ Partially satisfied by the approach;
○ Not satisfied by the approach; n.a.: not applicable

the usage of model libraries it is possible to include several classification systems, e.g., as presented for eCl@ss and the NIST Functional Basis.

Requirement R 6 postulates the usage of SysML4*Mechatronics* as basis for a (semi-) automatic compatibility analysis with according formal methods. Through the conducted case study, the general applicability of SysML4*Mechatronics* to analyze changes was shown. With the integrated information in the model, a manual compatibility analysis of the ports in case of a change was presented at the example of a barcode-scanner exchange to a RFID-scanner. In the current state of the prototypical modeling editor, an automatic transformation into a formal ontology and a reverse transformation of the results into the modeling editor, however, is not yet fully possible. Thus, requirement R 6 can only be rated as partially satisfied. However, as SysML4*Mechatronics* was defined as SysML profile, according Model-to-Model transformation mechanisms can also be used for SysML4*Mechatronics*, if an according (industrial) toolchain is implemented.

According to the Technology Acceptance Model, the perceived ease of use (req. R 8) and the perceived usefulness (req. R 9) of an approach are two significantly relevant aspects for the adoption and actual usage of a modeling language. In two academic usability evaluations, potential users evaluated these aspects after conducting several tasks with the SysML4*Mechatronics* modeling editor. Both groups evaluated their perceived ease of use as well as the perceived usefulness of SysML4*Mechatronics* positively and thus, the approach fulfills requirements R 8 and R 9. Through the Technology Acceptance Model, the intention of (potential) users can be evaluated. However, in order to assess the benefits of SysML4*Mechatronics* during the real usage further investigations after defined periods of time would be necessary.

Summarizing it can be concluded that SysML4*Mechatronics* fulfills the specified requirements for an interdisciplinary model-based development of mechatronic production systems.

8. Summary and Outlook

Within this dissertation, the interdisciplinary model-based development of mechatronic systems has been investigated. Based on an analysis of the research gap in the state of the art, a model-based engineering approach – called SysML4*Mecharonics* – has been developed, implemented prototypically and evaluated. In this concluding chapter a summarizing overview of the achieved results as well as implications for research and industry are given. An outlook shows starting points for further research activities to extend the developed approach and establish its use in industry.

8.1. Summary

The development of modern mechatronic systems, specifically in the production systems' domain, requires the knowledge and tight integration of various disciplines, which results in a high complexity of the engineering process [VFST15]. Companies have to cope with the challenge to develop customer specific, flexible, and variant rich systems with a high quality at reasonable costs in order to survive in the growing dynamic global competition. Traditional development approaches lack sufficient methods to cope with this complexity and consider the interdisciplinary dependencies in the mechatronic systems adequately. Furthermore, innovations, e.g., solutions for Industry 4.0, changing boundary conditions or new customer requirements, have to be considered during the entire system lifecycle and thus, integrated into existing systems. If adaptions or changes have to be conducted, the overall system functionality has to be assured by analyzing the compatibility of the changed components and, if required, adapting the system accordingly. As such changes have various interdisciplinary influences on the system, an insufficient analysis of the change influences can lead to undesired downtimes or additional costs.

Based on this initial situation, the main aim of this thesis is to support the development and (re-)engineering process of mechatronic systems through an interdisciplinary model-based approach. The focus of the approach lies on the integration of mechanics, electrics/electronics, and software and the modeling of their components' interfaces (ports) and dependencies. Such an integrated model reduces the complexity during the development phase and facilitates the creation of function-oriented modules, which consider the different boundaries in the involved disciplines. The approach focuses, furthermore, on the integration and usage of the components' properties in the model. For many purchased components, especially for electrics/electronics, standardized product classification systems with relevant properties exist in industry. The aim of this thesis is to integrate the properties from such classifications systems through model libraries in order to reduce the modeling effort and reduce faults. The components' properties can be used as basis for analyzing the compatibility in case of a change. This supports the

engineers to estimate the impact of a change in the model, before it is conducted in the real system.

In a first step to achieve the described aims a survey with 19 experts from the machine and plant-manufacturing industry was conducted to identify current challenges in industry. The survey as well as current approaches in literature show that Model-Based Engineering (MBE) approaches pose an emerging and growing possibility to enable an integrated, more efficient development and (re-)engineering process and thus, cope with the rising complexity. Based on the identified needs, the MBE-approach SysML4*Mechatronics* was developed. In order to ensure a high compatibility to existing standards and tools and to enable an easy extensibility, the modeling approach was defined as profile of the standardized SysML metamodel [Omgo15b] in the MOF Model Driven Architecture framework of the Object Management Group [Omgo15c]. As SysML is a general-purpose modeling language [BSBF11], it has to be adapted for its specific case of use, e.g., mechatronic systems, and offers therefore extension possibilities. SysML4*Mechatronics* makes use of stereotypes, which pose a specialization of the general *block* in SysML. The interdisciplinary SysML4*Mechatronics* model integrates the different disciplines through according stereotypes and offers further stereotypes, e.g., for the function-oriented modularization. Through stereotyped ports the components can be connected to each other in the system, discipline-specifically as well as interdisciplinary, and thus, show which dependencies exist and have to be considered in case of a change.

The integration of standardized properties from existing classification systems, e.g., eCl@ss for industrial mechatronic components or the NIST Functional Basis for Engineering Design [HSMS02] for the components' functionalities, was realized in SysML4*Mechatronics* through model libraries. Libraries contain reusable model elements for a given domain [FrMS15], e.g., the components classified in eCl@ss, and can be applied during the modeling process. A further benefit of model libraries is that no adaption of the metamodel is required, if they shall be changed or extended. Although eCl@ss offers a classification for a wide range of components, it is limited in the description of the components' functionality and does not contain an assignment of the relevant discipline. Thus, the SysML4*Mechatronics* approach enables on the one hand the integration of properties from different classification systems through according model libraries and on the other hand enables the developers to add further properties manually, if required.

After the interdisciplinary model with according properties has been developed during the engineering phase, it can be used in later lifecycle phases, i.e., the usage phase of the system, to analyze the components' discipline-specific and interdisciplinary dependencies to other system components. By using the modeling approach, engineers

thus can decide in case of a required change, e.g., due to new customer requirements, if a new component is compatible to the existing system or if adaptions have to be conducted. In this way, a more sophisticated estimation of the interdisciplinary change influences can be achieved.

Based on the theoretical definition of the approach, a proof-of-concept implementation of a SysML4*Mechatronics* modeling editor was implemented in order to evaluate its feasibility and conduct a practical evaluation with participants in an academic usability test. The modeling editor is intentionally restricted to the developed aspects and elements of SysML4*Mechatronics* and does not consider, e.g., further diagrams of SysML. In this way, the usage of the modeling editor can be understood more easily by the participants of the evaluation, which do not necessarily have a detailed knowledge of SysML. In order to ensure the compatibility to existing SysML models as well as the extensibility of the approach, especially for the application of SysML4*Mechatronics* in industrial projects, an implementation in a standard SysML modeling tool would have to be conducted.

In a first step, the prototypical modeling editor was applied to a production system use case. The case study shows how the system can be represented in SysML4*Mechatronics* and focuses, furthermore, on a comparison of the traditional development process and a development with SysML4*Mechatronics*. Therefore, a hierarchical task analysis was conducted, which analyzes the different required steps. SysML4*Mechatronics* shows its advantages thereby especially in the integrated modeling, the definition of functional modules and the analysis of (interdisciplinary) dependencies.

Before the SysML4*Mechatronics* modeling editor was used for the practical evaluation with potential users, an analytical usability evaluation, i.e., a cognitive walkthrough, was performed. In the walkthrough five experts, who were familiar with the modeling domain, took over the perspective of potential users in order identify possible problems that users could have during the interaction with the system. In this way, the design of the modeling editor, e.g., the positioning and labeling of menus and buttons, could be improved before the practical evaluation was conducted.

In an academic case study, the perceived usefulness and ease of use of the approach was evaluated through voluntary student participants. Therefore, several modeling tasks were conducted by the students in the SysML4*Mechatronics* modeling editor. Subsequently, the participants rated the approach in a questionnaire based on the Technology Acceptance Model [Davi89]. Both, the perceived usefulness and ease of use of the SysML4*Mechatronics* approach, were evaluated positively. As the used modeling editor is a prototypical implementation, the stability of the editor should be improved further.

In this dissertation, three main scientific contributions are made: The extension of the standardized modeling language SysML with definition of a metamodel (chapter 5.1) and concrete syntax (chapter 5.2) to represent mechatronic systems, the integration of standardized properties from existing classification systems in the modeling approach through model libraries (chapter 5.3), and the application of the developed modeling approach to support the development and reengineering/change of mechatronic systems (chapter 5.4).

SysML is a comprehensive, standardized modeling language for the systems' development. However, it was developed as general-purpose language to support a wide range of systems and requires an adaption for the use in a specific context, e.g., mechatronic systems. For this purpose, SysML4*Mechatronics* was developed as SysML-profile with a focus on structural modeling aspects. Next to the definition of the respective metamodel, a concrete syntax was developed to increase the cognitive effectiveness [Mood09], i.e., the ease for a user to interpret the represented model. A key concept of the developed modeling approach is the exploitation of stereotypes and respective ports for representing mechanics, electrics/electronics, and software as well as their dependencies within the mechatronic model.

The unambiguous representation of component's properties is a prerequisite to use them as a basis for analyzing the compatibility in case of a change. In order to overcome a manual definition of all properties, which can lead easily to mistakes or ambiguities, the integration of properties from standardized classification systems was presented. As such classification systems are extended frequently, an integration in the metamodel of the modeling approach would be unfavorable, as each change would require an adaption of the metamodel. Thus, rather the concept of respective model libraries, which can be applied to the metamodel of SysML4*Mechatronics*, was presented. This enables furthermore, that several model libraries (e.g., eCl@ss and the NIST Functional Basis) are integrated, representing different specific properties.

As third major contribution, the application of the developed modeling approach in the development phase and for the analysis of change situations was discussed. SysML4*Mechatronics* supports typical development processes such as the V-model and extends thereby the detailed mechatronic system development, in which the components of the different disciplines and their dependencies are defined. In addition, further detailed models, e.g., for a specific discipline, can be deduced from the SysML4*Mechatronics* model. In change situation or for reengineering the compatibility of an exchanged component or module can be analyzed based on the properties modeled in SysML4*Mechatronics*. The analysis of required adaptions, due to incompatibilities of

properties, supports engineers to decide more sophisticated, if an exchange makes economic sense or if rather a different component should be used for the change.

Besides the described scientific contributions, this thesis also provides several implications for industrial practice. The detailed comprehension of interdisciplinary dependencies in mechatronic production systems supports companies in the mechatronic production systems domain to increase their ability in handling the complexity and thus, accelerate the development process. By using the developed model-based approach, companies can analyze the interdisciplinary dependencies within their systems more detailed, foster the communication between the different involved disciplines, and have a more sophisticated possibility to analyze changes.

8.2. Outlook

Through the developed approach for an integrated model-based engineering of mechatronic systems, this thesis shows new and promising ways to support the interdisciplinary development and (re-)engineering process. The detailed comprehension of interdisciplinary dependencies in mechatronic systems enables companies to increase their ability in handling the complexity and accelerate the development. While SysML4*Mechatronics* focuses on the interdisciplinary modeling of mechatronic systems, it offers several linking points to further methods and approaches that support the innovation process. For example, the impact of decisions, which are taken at early stages in the innovation process, can be estimated in most cases only much later and thus, may lead to potentially costly changes. Therefore, SysML4*Mechatronics* could be linked with other models and documents and thus, enable a more sophisticated analysis in this phase.

In a first step, the developed SysML4*Mechatronics* approach has to be integrated into an industrial toolchain, in order to foster its usage in industry and link the created models with other models and documents in the innovation process, e.g., CAD models or PLM data. Furthermore, the models can be linked with documents and models for process- and resource-planning in digital factory systems [Zafi14, Zafi17]. The goal of digital factory systems is the integrated data management throughout the factories' lifecycle. A combination of the models would allow, e.g., virtual commissioning, material flow simulations or a more sophisticated production planning. Within the transfer project T3 of the Collaborative Research Centre SFB 768 "Managing Cycles in Innovation Processes" the SysML4*Mechatronics* approach shall be integrated into the industrial tool PTC Integrity Modeler[14] and refined for the specific needs of the product and manufacturing automation industry [GSLM15]. By coupling the models and documents,

[14] http://www.ptc.com/model-based-systems-engineering/integrity-modeler, retrieved on 23 September 2018

the interdisciplinary decision-making in early phases of the development process can be improved.

While the models in SysML4*Mechatronics* are created during the development and (re-)engineering phase, information from change cycles during the use phase of the system could be used to improve the models. In this way, engineers could use optimized model elements for the development of future systems and thus, enable a faster development and less necessary changes. Subproject A6 of the SFB 768 will follow this goal of a 'self-maintance' of the models. Therefore, an assistance system, which synchronizes the models with the change cycles in the use phase and, if necessary, adapts them (semi-)automatically will be developed.

Furthermore, SysML4*Mechatronics* can be used as a basis for a model-based training system for maintenance engineers [LKKP18]. Within a broader scope, SysML4*Mechatronics* models also could be coupled with non-technical models, e.g., sociological or psychological models, to achieve an integrated, interdisciplinary management of innovation processes. A major focus for research thereby is the heterogeneity and differing information quality of the different involved models.

8.3. Summary of publications and research cooperations

This additional, final chapter shall give a brief overview of the research contributions that have accounted to the development of the approach presented in this thesis.

The foundation for this work was set through the research at the Institute of Automation and Information Systems at the Technical University of Munich. In chapter 1.2, prior theses of the institute are discussed in the context of this work. At the beginning of the research work, the requirements of IT related cycles in mechatronic development processes were investigated [LBKV12], which served as basis for the development of the approach in this thesis [KeBV12, KeVo13].

A MBE-approach, such as SysML4*Mechatronics*, is always embedded in a company's entire innovation process. Thus, within the collaborative research centre "CRC 768: Managing Cycles in Innovation Processes" SysML4*Mechatronics* was combined with other (socio-)technical models to investigate the mutual dependencies and influences in the innovation process. Different project collaborations involved the link with lifecycle models [KHMV12], planning models [KBCW14], design structure matrices and multiple domain matrices [MKKV13], requirements and service models [KWMK13], as well as user interaction models [KPRV16].

In a close research cooperation with Prof. Dr. Cesare Fantuzzi and Dr. Giacomo Barbieri (Automation, Robotics and System Control group, Università degli studi di Modena e Reggio Emilia) the challenges of an integrated development process were investigated [KBFV13] and a design pattern for the development of mechatronic systems, which utilizes SysML4*Mechatronics*, was presented [BKFV14].

As presented in chapter 5.4.2, SysML4*Mechatronics* can be used as basis for a (semi-) automatic change analysis with formal methods [FeKV14, FeKV16]. In a cooperation with Prof. Christiaan J.J. Paredis, Dr. Sebastian Herzig and Dr. Ahsan Qamar (Georgia Institute of Technology) inter-model inconsistencies between SysML4*Mechatronics* and other models in the development process were investigated [FHKW15a, FHKW15b]. A framework, dealing with the management of such inconsistencies, was focused in a further research cooperation with Assistant Prof. Mag.rer.soc.oec. Dr.rer.soc.oec. Manuel Wimmer (Business Informatics Group, Vienna University of Technology) [FWKV16].

This brief summary illustrates that not only the development of mechatronic systems requires an intensive collaboration, but that also science lives from an active dialogue and a constant exchange with other researchers.

9. List of references

[ABDG11] ARLITT, R.M.; BALINSKI, K.D.; DAGLI, C.H. and GRANTHAM, K.: Functional Analysis of Systems Using a Functional Basis. In: *IEEE International Systems Conference*. Montreal, 2011, pp. 563–568

[AbGD16] ABRAMOVICI, M.; GÖBEL, J.C. and DANG, H.B.: Semantic data management for the development and continuous reconfiguration of smart products and systems. In: *CIRP Annals* Vol. 65 (2016), No. 1, pp. 185–188

[ADSG71] ANNETT, J.; DUNCAN, K.D.; STAMMERS, R.B. and GRAY, M.J.: *Task analysis*. London: Her Majesty´s Stationery Office, 1971

[AFTW10] ALVAREZ CABRERA, A.A.; FOEKEN, M.J.; TEKIN, O.A.; WOESTENENK, K.; ERDEN, M.S.; DE SCHUTTER, B.; VAN TOOREN, M.J.L.; BABUŠKA, R.; VAN HOUTEN, F.J.A.M. and TOMIYAMA, T.: Towards automation of control software: A review of challenges in mechatronic design. In: *Mechatronics* Vol. 20 (2010), No. 8, pp. 876–886

[AlZi13] ALBERS, A. and ZINGEL, C.: Challenges of Model-Based Systems Engineering: A Study towards Unified Term Understanding and the State of Usage of SysML. In: ABRAMOVICI, M.; STARK, R. (eds.): *CIRP Design Conference*, *Lecture Notes in Production Engineering*. Berlin, Heidelberg: Springer, 2013, pp. 83–92

[Ange03] ANGERBAUER, R.: Baukastensystematik in der Steuerungstechnik auf Basis einer Föderalen Informationsarchitektur. In: THOBEN, K.-D.; FRITZ, S.; LÜNEMANN, M. (eds.): *Fit für die Maßgeschneiderte Massenfertigung durch agile, rekonfigurierbare Fertigungssysteme – Bremer Schriften zur integrierten Produkt- und Prozessentwicklung / Band 46*. Aachen: mainz Verlag, 2003, pp. 41–55

[AWFW07] ABELE, E.; WÖRN, A.; FLEISCHER, J.; WIESER, J.; MARTIN, P. and KLÖPPER, R.: Mechanical module interfaces for reconfigurable machine tools. In: *Production Engineering* Vol. 1 (2007), No. 4, pp. 421–428

[Baar06] BAAR, T.: Correctly Defined Concrete Syntax for Visual Modeling Languages. In: NIERSTRASZ, O.; WHITTLE, J.; HAREL, D.; REGGIO, G. (eds.): *Model Driven Engineering Languages and Systems. MODELS 2006. Lecture Notes in Computer Science*. Vol. 4199. Berlin, Heidelberg: Springer, 2006, pp. 111–125

[BaCG13] BASILE, F.; CHIACCHIO, P. and GERBASIO, D.: On the Implementation of Industrial Automation Systems Based on PLC. In: *IEEE Transactions on Automation Science and Engineering* Vol. 10 (2013), No. 4, pp. 990–1003

[BaDa03] BAHILL, T. and DANIELS, J.: Using Objected-Oriented and UML Tools for Hardware Design: A Case Study. In: *Systems Engineering* Vol. 6 (2003), No. 1, pp. 28–48

[BBLM16] BERARDINELLI, L.; BIFFL, S.; LÜDER, A.; MÄTZLER, E.; MAYERHOFER, T.; WIMMER, M. and WOLNY, S.: Cross-disciplinary engineering with AutomationML and SysML. In: *at - Automatisierungstechnik* Vol. 64 (2016), No. 4, pp. 253–269

[BBOR12] BRAUN, S.; BARTELT, C.; OBERMEIER, M.; RAUSCH, A. and VOGEL-HEUSER, B.: Requirements on Evolution Management of Product Lines in

Automation Engineering. In: *IFAC Proceedings Volumes* Vol. 45 (2012), No. 2, pp. 340–345

[BCDH06] BROY, M.; CRANE, M.L.; DINGEL, J.; HARTMAN, A.; RUMPE, B. and SELIC, B.: 2nd UML 2 Semantics Symposium: Formal Semantics for UML. In: KÜHNE, T. (ed.): *Proceedings of the 2006 International Conference on Models in Software Engineering*. Berlin, Heidelberg: Springer, 2006, pp. 318–323

[BDGS14] BECKER, S.; DZIWOK, S.; GERKING, C.; SCHÄFER, W.; HEINZEMANN, C.; THIELE, S.; MEYER, M.; PRIESTERJAHN, C.; POHLMANN, U. and TICHY, M.: *The MechatronicUML Design Method - Process and Language for Platform-Independent Modeling: Technical Report tr-ri-14-337*. Paderborn, 2014

[Best09] BESTVOR: *Projektabschlussbericht des Forschungsprojekts BESTVOR – Betriebliche Einführungsstrategie für ein anwendungsorientiertes Vorgehensmodell für die Entwicklung zuverlässigerer mechatronischer Systeme im Maschinen- und Anlagenbau*, 2009

[BFKK10] BIRKHOFER, R.; FELDMEIER, G.; KALHOFF, J.; KLEEDÖRFER, C.; LEIDNER, M.; MILDENBERGER, R.; MÜHLHAUSE, M.; NIEMANN, J.; SCHRIEBER, R.; WICKINGER, J.; WINZENICK, M. and WOLLSCHLÄGER, M.: *Life-Cycle-Management für Produkte und Systeme der Automation – Ein Leitfaden des Arbeitskreises Systemaspekte im ZVEI Fachverband Automation*. Frankfurt: Berthold Druck, 2010

[Bish06] BISHOP, R.H.: *Mechatronics: An Introduction*. Boca Raton: CRC Press, 2006

[BKFV14] BARBIERI, G.; KERNSCHMIDT, K.; FANTUZZI, C. and VOGEL-HEUSER, B.: A SysML based design pattern for the high-level development of mechatronic systems to enhance re-usability. In: *IFAC Proceedings Volumes* Vol. 47 (2014), No. 3, pp. 3431–3437

[BMWL15] BIFFL, S.; MAETZLER, E.; WIMMER, M.; LUEDER, A. and SCHMIDT, N.: Linking and versioning support for AutomationML: A model-driven engineering perspective. In: *International Conference on Industrial Informatics (INDIN)*, 2015, pp. 499–506

[BoCl10] BONE, M. and CLOUTIER, R.: The Current State of Model Based Systems Engineering : Results from the OMG ™ SysML Request for Information 2009. In: *Conference on Systems Engineering Research (CSER)*, 2010, pp. 225–232

[BoFS12] BONFÈ, M.; FANTUZZI, C. and SECCHI, C.: Design patterns for model-based automation software design and implementation. In: *Control Engineering Practice* Vol. 21 (2012), No. 11, pp. 1608–1619

[Bolt13] BOLTON, W.: *Mechatronics: Electronic Control Systems in Mechanical and Electrical Engineering*. 5th ed. Upper Saddle River: Pearson/Prentice Hall, 2013

[BoRJ00] BOOCH, G.; RUMBAUGH, J. and JACOBSON, I.: *The Unified Modeling Language User Guide*. 6th ed. Upper Saddle River: Addison-Wesley, 2000

[Brau13] BRAUN, S.: *Anpassung und praxisnahe Evaluation gebrauchstauglicher Notationen zur modularen, objektorientierten Programmierung von*

Steuerungstechnik im Maschinen- und Anlagenbau, Dissertation, Technical University of Munich, 2013

[BrCe11] BROY, M. and CENGARLE, M.V.: UML formal semantics: lessons learned. In: *Software & Systems Modeling* Vol. 10 (2011), No. 4, pp. 441–446

[BrEp02] BROWNING, T.R. and EPPINGER, S.D.: Modeling impacts of process architecture on cost and schedule risk in product development. In: *IEEE Transactions on Engineering Management* Vol. 49 (2002), No. 4, pp. 428–442

[BSBF11] BASSI, L.; SECCHI, C.; BONFÈ, M. and FANTUZZI, C.: A SysML-Based Methodology for Manufacturing Machinery Modeling and Design. In: *IEEE/ASME Transactions on Mechatronics* Vol. 16 (2011), No. 6, pp. 1049–1062

[BSFB06] BASSI, L.; SECCHI, C.; FANTUZZI, C. and BONFÈ, M.: An object-oriented approach to manufacturing systems modeling. In: *IEEE International Conference on Automation Science and Engineering (CASE)*, 2006, pp. 442–447

[BuGT05] BURMESTER, S.; GIESE, H. and TICHY, M.: Model-Driven Development of Reconfigurable Mechatronic Systems with Mechatronic UML. In: AßMANN, U.; AKSIT, M.; RENSINK, A. (eds.): *Model Driven Architecture. Lecture Notes in Computer Science*. Berlin, Heidelberg: Springer, 2005

[CaBr18] CAMPETELLI, A. and BROY, M.: Modelling Logical Architecture of Mechatronic Systems and Its Quality Control. In: WINNER, H.; PROKOP, G.; MAURER, M. (eds.): *Automotive Systems Engineering II*. Cham: Springer, 2018, pp. 73–91

[CaLP11] CAO, Y.; LIU, Y. and PAREDIS, C.J.J.: System-level model integration of design and simulation for mechatronic systems based on SysML. In: *Mechatronics* Vol. 21 (2011), No. 6, pp. 1063–1075

[CeGR09] CENGARLE, M.V.; GRÖNNIGER, H. and RUMPE, B.: Variability within Modeling Language Definitions. In: SCHÜRR, A.; SELIC, B. (eds.): *Model Driven Engineering Languages and Systems. MODELS 2009. Lecture Notes in Computer Science*. Vol. 5795. Berlin, Heidelberg: Springer, 2009, pp. 670–684

[Chen76] CHEN, P.P.-S.: The Entity-Relationship Model - Toward a Unified View of Data. In: *ACM Transactions on Database Systems* Vol. 1 (1976), No. 1, pp. 9–36

[Ciup15] CIUPEK, M.: "Der Maschinenbau hat sich zu einer Integrationswissenschaft entwickelt" - Interview mit Professor Jürgen Gausemeier. In: *VDI nachrichten* (2015), No. 4, pp. 12–13

[Davi89] DAVIS, F.D.: Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. In: *MIS Quarterly* Vol. 13 (1989), No. 3, pp. 319–340

[Dell14] DELLIGATTI, L.: *SysML distilled: A brief guide to the systems modeling language*. Upper Saddle River: Addison-Wesley, 2014

[DiMa13] DICKERSON, C.E. and MAVRIS, D.: A Brief History of Models and Model Based Systems Engineering and the Case for Relational Orientation. In: *IEEE Systems Journal* Vol. 7 (2013), No. 4, pp. 581–592

[Dind03] DIN DEUTSCHES INSTITUT FÜR NORMUNG: DIN 8580:2003-09, Manufacturing processes — Terms and definitions, division (2003)

[Dind10] DIN DEUTSCHES INSTITUT FÜR NORMUNG: DIN EN 81346-1:2010-05, Industrial systems, installations and equipment and industrial products – Structuring principles and reference designations – Part 1: Basic rules (IEC 81346-1:2009); German version EN 81346-1:2009 (2010)

[Dind14] DIN DEUTSCHES INSTITUT FÜR NORMUNG: DIN IEC 60050-351:2014-09, International electrotechnical vocabulary – Part 351: Control technology (IEC 60050-351:2013) (2014)

[Dind16] DIN DEUTSCHES INSTITUT FÜR NORMUNG: DIN EN ISO 9241-11:2016-01, Ergonomics of human-system interaction – Part 11: Usability: Definitions and concepts (ISO/DIS 9241-11:2015); German and English version prEN ISO 9241-11:2015. Berlin (2016)

[DLPH08] DRATH, R.; LÜDER, A.; PESCHKE, J. and HUNDT, L.: AutomationML – the glue for seamless Automation Engineering. In: *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2008, pp. 616–623

[DöHe13] DÖBRICH, U. and HEIDEL, R.: Modell zur Beschreibung cyber-physischer Systeme. In: *atp edition - Automatisierungstechnische Praxis* Vol. 55 (2013), No. 12, pp. 38–45

[EABC15] ECKERT, C.; ALBERS, A.; BURSAC, N.; CHEN, H.X.; CLARKSON, P.J.; GERICKE, K.; GLADYSZ, B.; MAIER, J.F.; RACHENKOVA, G.; SHAPIRO, D. and WYNN, D.: Integrated product and process models: Towards an integrated framework and review. In: *International Conference on Engineering Design (ICED)*, 2015

[EbGö04] EBERLE, S. and GÖHNER, P.: Softwareentwicklung für eingebettete Systeme mit strukturierten Komponenten - Teil 1: Komponentenorientierte Zerlegung. In: *atp - Automatisierungstechnische Praxis* Vol. 46 (2004), No. 3, pp. 41–52

[EGGM12] EIGNER, M.; GERHARDT, F.; GILZ, T. and MOGO NEM, F.: *Informationstechnologie für Ingenieure*. Berlin, Heidelberg: Springer, 2012

[Ehrl09] EHRLENSPIEL, K.: *Integrierte Produktentwicklung*. München: Hanser, 2009

[EiGZ12] EIGNER, M.; GILZ, T. and ZAFIROV, R.: Proposal for functional product decription as part of a PLM solution in interdisciplinary product development. In: *International Design Conference (DESIGN)*, 2012, pp. 1667–1676

[Eppl11] EPPLE, U.: Merkmale als Grundlage der Interoperabilität technischer Systeme. In: *at - Automatisierungstechnik* Vol. 59 (2011), No. 7, pp. 440–450

[EsMa12] ESTÉVEZ, E. and MARCOS, M.: Model-Based Validation of Industrial Control Systems. In: *IEEE Transactions on Industrial Informatics* Vol. 8 (2012), No. 2, pp. 302–310

[EsMO07] ESTÉVEZ, E.; MARCOS, M. and ORIVE, D.: Automatic generation of PLC automation projects from component-based models. In: *The International Journal of Advanced Manufacturing Technology* Vol. 35 (2007), No. 5, pp. 527–540

[ESPO09] ESTÉVEZ, E.; SARACHAGA, I.; PEREZ, F.; ORIVE, D. and MARCOS, M.: Model driven design in industrial automation. In: *IEEE Conference on Decision and Control (CDC)*, 2009, pp. 6262–6267

[EWMA17] ECKERT, C.; WYNN, D.C.; MAIER, J.F.; ALBERS, A.; BURSAC, N.; XIN CHEN, H.L.; CLARKSON, P.J.; GERICKE, K.; GLADYSZ, B. and SHAPIRO, D.: On the integration of product and process models in engineering design. In: *Design Science* Vol. 3 (2017), No. May, pp. 1–41

[FaBS09] FANTUZZI, C.; BONFÈ, M. and SECCHI, C.: A Design Pattern for Model Based Software Development for Automatic Machinery. In: *IFAC Proceedings Volumes* Vol. 42 (2009), No. 4, pp. 1429–1434

[FaSB11] FANTUZZI, C.; SECCHI, C. and BONFÈ, M.: A Design Pattern for translating UML software models into IEC 61131-3 Programming Languages. In: *IFAC Proceedings Volumes* Vol. 44 (2011), No. 1, pp. 9158–9163

[FeFV12] FELDMANN, S.; FUCHS, J. and VOGEL-HEUSER, B.: Modularity, variant and version management in plant automation – Future challenges and state of the art. In: *International Design Conference (DESIGN)*, 2012, pp. 1689–1698

[FEHF13] FRANK, T.; ECKERT, K.; HADLICH, T.; FAY, A.; DIEDRICH, C. and VOGEL-HEUSER, B.: Erweiterung des V-Modells® für den Entwurf von verteilten Automatisierungssystemen. In: *at - Automatisierungstechnik* Vol. 61 (2013), No. 2, pp. 79–91

[FeKV14] FELDMANN, S.; KERNSCHMIDT, K. and VOGEL-HEUSER, B.: Combining a SysML-based modeling approach and semantic technologies for analyzing change influences in manufacturing plant models. In: *CIRP Conference on Manufacturing Systems (CMS)*, 2014, pp. 451–456

[FeKV16] FELDMANN, S.; KERNSCHMIDT, K. and VOGEL-HEUSER, B.: Konzept eines wissensbasierten Frameworks zur Spezifikation und Diagnose von Inkonsistenzen in mechatronischen Modellen. In: *at - Automatisierungstechnik* Vol. 64 (2016), No. 3, pp. 199–215

[FePi04] FERNÁNDEZ-MEDINA, E. and PIATTINI, M.: Extending OCL for Secure Database Development. In: BAAR, T.; STROHMEIER, A.; MOREIRA, A.; MELLOR, S. J. (eds.): *«UML» 2004 — The Unified Modeling Language. Modeling Languages and Applications*, *Lecture Notes in Computer Science*. Vol. 3273. Berlin, Heidelberg: Springer, 2004, pp. 380–394

[FGNI00] FRICKE, E.; GEBHARD, B.; NEGELE, H. and IGENBERGS, E.: Coping with changes: causes, findings, and strategies. In: *Systems Engineering* Vol. 3 (2000), No. 4, pp. 169–179

[FHKW15a] FELDMANN, S.; HERZIG, S.J.I.; KERNSCHMIDT, K.; WOLFENSTETTER, T.; KAMMERL, D.; QAMAR, A.; LINDEMANN, U.; KRCMAR, H.; PAREDIS, C.J.J. and VOGEL-HEUSER, B.: A comparison of inconsistency management approaches using a mechatronic manufacturing system design case study. In: *IEEE International Conference on Automation Science and Engineering (CASE)*, 2015, pp. 158–165

[FHKW15b] FELDMANN, S.; HERZIG, S.J.I.; KERNSCHMIDT, K.; WOLFENSTETTER, T.; KAMMERL, D.; QAMAR, A.; LINDEMANN, U.; KRCMAR, H.; PAREDIS, C.J.J. and VOGEL-HEUSER, B.: Towards Effective Management of Inconsistencies

in Model-Based Engineering of Automated Production Systems. In: *IFAC-PapersOnLine* Vol. 48 (2015), No. 3, pp. 916–923

[FHPZ10] FOLLMER, M.; HEHENBERGER, P.; PUNZ, S. and ZEMAN, K.: Using SysML in the Product Development Process of Mechatronic Systems. In: *International Design Conference (DESIGN)*, 2010, pp. 1513–1522

[Flic09] FLICK, U.: *An introduction to qualitative research*. 5th ed. London: Sage, 2009

[Fran14] FRANK, T.: *Entwicklung und Evaluation einer Modellierungssprache für den Architekturentwurf von verteilten Automatisierungsanlagen auf Basis der Systems Modeling Language (SysML)*, Dissertation, Technical University of Munich, 2014

[FrLa03] FRANK, U. and VAN LAAK, B.L.: Anforderungen an Sprachen zur Modellierung von Geschäftsprozessen. In: *Arbeitsberichte des Instituts für Wirtschaftsinformatik* Vol. 34 (2003), pp. 1–109

[FrMS15] FRIEDENTHAL, S.; MOORE, A. and STEINER, R.: *A Practical Guide to SysML: The systems modeling language*. 3rd ed. Waltham: Morgan Kaufmann, 2015

[FVFE15] FAY, A.; VOGEL-HEUSER, B.; FRANK, T.; ECKERT, K.; HADLICH, T. and DIEDRICH, C.: Enhancing a model-based engineering approach for distributed manufacturing automation systems with characteristics and design patterns. In: *Journal of Systems and Software* Vol. 101 (2015), pp. 221–235

[FWKV16] FELDMANN, S.; WIMMER, M.; KERNSCHMIDT, K. and VOGEL-HEUSER, B.: A comprehensive approach for managing inter-model inconsistencies in automated production systems engineering. In: *IEEE International Conference on Automation Science and Engineering (CASE)*, 2016, pp. 1120–1127

[FYOS01] FENSEL, D.; YING DING; OMELAYENKO, B.; SCHULTEN, E.; BOTQUIN, G.; BROWN, M. and FLETT, A.: Product data integration in B2B e-commerce. In: *IEEE Intelligent Systems* Vol. 16 (2001), No. 4, pp. 54–59

[GaDK10] GAUSEMEIER, J.; DOROCIAK, R. and KAISER, L.: Computer-Aided Modeling of the Principle Solution of Mechatronic Systems: A Domain-Spanning Methodology for the Conceptual Design of Mechatronic Systems. In: *ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, 2010, pp. 1109–1118

[GePA99] GERSHENSON, J.K.; PRASAD, G.J. and ALLAMNENI, S.: Modular Product Design : A Life-Cycle View. In: *Journal of Integrated Design and Process Science* Vol. 3 (1999), No. 4, pp. 13–26

[GePZ03] GERSHENSON, J.K.; PRASAD, G.J. and ZHANG, Y.: Product modularity: Definitions and benefits. In: *Journal of Engineering Design* Vol. 14 (2003), No. 3, pp. 295–313

[Gero90] GERO, J.S.: Design Prototypes: A Knowledge Representation Schema for Design. In: *AI Magazine* Vol. 11 (1990), No. 4, pp. 26–36

[GFDK09] GAUSEMEIER, J.; FRANK, U.; DONOTH, J. and KAHL, S.: Specification technique for the description of self-optimizing mechatronic systems. In: *Research in Engineering Design* Vol. 20 (2009), No. 4, pp. 201–223

[GHJL15] GRÄSER, O.; HUNDT, L.; JOHN, M.; LOBERMEIER, G.; LÜDER, A.; MÜLHENS, S.; ONDRACEK, N.; THRON, M. and SCHMELTER, J.: White paper AutomationML and eCl@ss integration, Common Working Group of AutomationML e.V and eCl@ss e.V (2015)

[GhLB12] GHAZIZADEH, M.; LEE, J.D. and BOYLE, L.N.: Extending the Technology Acceptance Model to assess automation. In: *Cognition, Technology & Work* Vol. 14 (2012), No. 1, pp. 39–49

[GHSR97] GU, P.; HASHEMIAN, M.; SOSALE, S. and RIVIN, E.: An Integrated Modular Design Methodology for Life-Cycle Engineering. In: *CIRP Annals - Manufacturing Technology* Vol. 46 (1997), No. 1, pp. 71–74

[GlFa16] GLAWE, M. and FAY, A.: Wissensbasiertes Engineering automatisierter Anlagen unter Verwendung von AutomationML und OWL. In: *at - Automatisierungstechnik* Vol. 64 (2016), No. 3, pp. 186–198

[GSGK09] GAUSEMEIER, J.; SCHÄFER, W.; GREENYER, J.; KAHL, S.; POOK, S. and RIEKE, J.: Management of cross-domain model consistency during the development of advanced manufacturing systems. In: *International Conference on Engineering Design (ICED)*, 2009, pp. 1–12

[GSLM15] GALLASCH, A.; SCHMIDT, R.; LEIPRECHT, J. and MÜLLER, T.: Model Based Systems Engineering und PLM, warum und wie dies bei der Entwicklung von Maschinen zusammengehört. In: *Tag des Systems Engineering*. München: Carl Hanser Verlag GmbH & Co. KG, 2015, pp. 267–278

[Gsus15] GS1 US (UNITED NATIONS DEVELOPMENT PROGRAMME): United Nations Standard Products and Services Code (UNSPSC) – Codeset (2015)

[HaDi13] HADLICH, T. and DIEDRICH, C.: Using properties in systems engineering. In: *IEEE Conference on Emerging Technologies & Factory Automation (ETFA)*, 2013, pp. 1–7

[Hadl15] HADLICH, T.: *Verwendung von Merkmalen im Engineering von Systemen*, Dissertation, Otto-von-Guericke-University Magdeburg, 2015

[HaLS17] HALSTENBERG, F.A.; LINDOW, K. and STARK, R.: Utilization of Product Lifecycle Data from PLM Systems in Platforms for Industrial Symbiosis. In: *Procedia Manufacturing* Vol. 8, Elsevier B.V. (2017), No. October 2016, pp. 369–376

[HaVA16] HARRISON, R.; VERA, D. and AHMAD, B.: Engineering Methods and Tools for Cyber–Physical Automation Systems. In: *Proceedings of the IEEE* Vol. 104 (2016), No. 5, pp. 973–985

[HBMS16] HOLTMANN, J.; BERNIJAZOV, R.; MEYER, M.; SCHMELTER, D. and TSCHIRNER, C.: Integrated and iterative systems engineering and software requirements engineering for technical systems. In: *Journal of Software: Evolution and Process* Vol. 28 (2016), No. 9, pp. 722–743

[Hehe14] HEHENBERGER, P.: Perspectives on hierarchical modeling in mechatronic design. In: *Advanced Engineering Informatics* Vol. 28 (2014), No. 3, pp. 188–197

[HeLS05] HEPP, M.; LEUKEL, J. and SCHMITZ, V.: A quantitative analysis of eCl@ss, UNSPSC, eOTD, and RNTD content, coverage, and maintenance. In: *IEEE International Conference on e-Business Engineering (ICEBE)*, 2005, pp. 572–581

[HeLS07] HEPP, M.; LEUKEL, J. and SCHMITZ, V.: A quantitative analysis of product categorization standards: content, coverage, and maintenance of eCl@ss, UNSPSC, eOTD, and the RosettaNet Technical Dictionary. In: *Knowledge and Information Systems* Vol. 13 (2007), No. 1, pp. 77–114

[HeSh12] HELMS, B. and SHEA, K.: Computational Synthesis of Product Architectures Based on Object-Oriented Graph Grammars. In: *Journal of Mechanical Design* Vol. 134 (2012), No. 2, pp. 021008-1-021008-14

[HMSL07] HE, X.; MA, Z.; SHAO, W. and LI, G.: A metamodel for the notation of graphical modeling languages. In: *IEEE International Computer Software and Applications Conference (COMPSAC)*, 2007, pp. 219–224

[HPZA10] HEHENBERGER, P.; POLTSCHAK, F.; ZEMAN, K. and AMRHEIN, W.: Hierarchical design models in the mechatronic product development process of synchronous machines. In: *Mechatronics* Vol. 20 (2010), No. 8, pp. 864–875

[HQRP11] HERZIG, S.J.I.; QAMAR, A.; REICHWEIN, A. and PAREDIS, C.J.J.: A Conceptual Framework for Consistency Management in Model-Based Systems Engineering. In: *ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference (IDETC/CIE)*, 2011, pp. 1329–1339

[HSMS02] HIRTZ, J.; STONE, R.B.; MCADAMS, D.A.; SZYKMAN, S. and WOOD, K.L.: A functional basis for engineering design: Reconciling and evolving previous efforts. In: *Research in Engineering Design* Vol. 13 (2002), No. 2, pp. 65–82

[HuKu98] HUANG, C.-C. and KUSIAK, A.: Modularity in design of products and systems. In: *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* Vol. 28 (1998), No. 1, pp. 66–77

[Hund16] HUNDEGGER, L.: Nutzung von Merkmalen aus Produktklassifikationssystemen in der modellbasierten Entwicklung, Bachelor Thesis, Technical University of Munich, Supervisor: Dipl.-Ing. Konstantin Kernschmidt (2016)

[Ieci02] IEC INTERNATIONAL ELECTROTECHNICAL COMMISSION: IEC 61360-1:2002, Standard data element types with associated classification scheme for electric components – Part 1: Definitions – Principles and methods (2002)

[Ieci08] IEC INTERNATIONAL ELECTROTECHNICAL COMMISSION: IEC 62424:2008, Representation of process control engineering – Requests in P&I diagrams and data exchange between P&ID tools and PCE-CAE tools (2008)

[Ieci12] IEC INTERNATIONAL ELECTROTECHNICAL COMMISSION: IEC 61499-1:2012, Function blocks – Part 1: Architecture (2012), No. September

[Ieci14a] IEC INTERNATIONAL ELECTROTECHNICAL COMMISSION: IEC 61131-3:2014, Programmable Controllers - Part 3: Programming languages (2014)

[Ieci14b] IEC INTERNATIONAL ELECTROTECHNICAL COMMISSION: IEC 62714-1:2014, Engineering data exchange format for use in industrial automation systems engineering – Automation markup language – Part 1: Architecture and general requirements (2014)

[Inte07] INTERNATIONAL COUNCIL ON SYSTEMS ENGINEERING (INCOSE): *Systems Engineering Vision 2020, Technical Report, Document No.: INCOSE-TP-2004-004-02*, 2007

[Inte15] INTERNATIONAL COUNCIL ON SYSTEMS ENGINEERING (INCOSE): *Systems Engineering Handbook - A Guide for System Life Cycle Processes and Activities*. 4th ed. Hoboken: Wiley, 2015

[Iser96] ISERMANN, R.: Modeling and design methodology for mechatronic systems. In: *IEEE/ASME Transactions on Mechatronics* Vol. 1 (1996), No. 1, pp. 16–28

[Isoi00] ISO INTERNATIONAL ORGANIZATION FOR STANDARDIZATION: ISO/TS 29002-5:2009, Industrial automation systems and integration - Exchange of characteristic data - Part 5: Identification scheme

[Isoi09] ISO INTERNATIONAL ORGANIZATION FOR STANDARDIZATION: ISO 80000-1:2009, Quantities and units - Part 1: General (2009)

[Isoi12] ISO INTERNATIONAL ORGANIZATION FOR STANDARDIZATION: ISO/PAS 17506:2012, Industrial automation systems and integration - COLLADA digital asset schema specification for 3D visualization of industrial data (2012)

[Isoi95] ISO INTERNATIONAL ORGANIZATION FOR STANDARDIZATION: ISO 9241-10:1995, Ergonomic requirements for office work with visual display terminals (VDTs) - Part 10: Dialogue principles (1995)

[JeRo12] JENKINS, J.S. and ROUQUETTE, N.F.: Semantically-Rigorous Systems Engineering Modeling Using SysML and OWL. In: *International Workshop on Systems & Concurrent Engineering for Space Applications (SECESA)*, 2012, pp. 1–8

[KaEp13] KAMPERT, D. and EPPLE, U.: A service interface for exchange of property information. In: *Annual Conference of the IEEE Industrial Electronics Society (IECON)*, 2013, pp. 6922–6927

[KaFV03] KATZKE, U.; FISCHER, K. and VOGEL-HEUSER, B.: Entwicklung und Evaluation eines Modells für modulare Automatisierung im Anlagenbau. In: *Verteilte Echtzeitsysteme*. Boppard, 2003, pp. 69–77

[Kass10] KASSER, J.E.: Seven systems engineering myths and the corresponding realities. In: *Systems Engineering Test and Evaluation Conference*, 2010, pp. 1–13

[Katz08] KATZKE, U.: *Spezifikation und Anwendung einer Modellierungssprache für die Automatisierungstechnik auf Basis der Unified Modeling Language (UML)*, Dissertation, Universität Kassel, 2008

[KBCW14] KERNSCHMIDT, K.; BEHNCKE, F.G.H.; CHUCHOLOWSKI, N.; WICKEL, M.C.; BAYRAK, G.; LINDEMANN, U. and VOGEL-HEUSER, B.: An integrated approach to analyze change-situations in the development of production systems. In: *CIRP Conference on Manufacturing Systems (CMS)*, 2014, pp. 148–153

[KBFV13] KERNSCHMIDT, K.; BARBIERI, G.; FANTUZZI, C. and VOGEL-HEUSER, B.: Possibilities and challenges of an integrated development using a combined SysML-model and corresponding domain specific models. In: *IFAC Proceedings Volumes* Vol. 46 (2013), No. 9, pp. 1465–1470

[KBHR15] KOCH, J.; BRANDL, F.; HOFER, A. and REINHART, G.: *Studie: Änderungsmanagement in der Produktion*: Institut für Werkzeugmaschinen und Betriebswissenschaften (iwb), Technical University of Munich, 2015

[KeBV12] KERNSCHMIDT, K.; BAYRAK, G. and VOGEL-HEUSER, B.: A port-based approach for modelling the structure of mechatronic modules. In: ONISHI, M.; MAURER, M.; EBEN, K.; LINDEMANN, U. (eds.): *Gain competitive advantage by managing complexity - Proceedings of the 14th International DSM Conference*. München: Hanser, 2012, pp. 111–123

[KeFV15] KERNSCHMIDT, K.; FELDMANN, S. and VOGEL-HEUSER, B.: Lebenszyklusorientierte modellbasierte Entwicklung. In: *atp edition - Automatisierungstechnische Praxis* Vol. 57 (2015), No. 05, pp. 32–39

[KeFV18] KERNSCHMIDT, K.; FELDMANN, S. and VOGEL-HEUSER, B.: A model-based framework for increasing the interdisciplinary design of mechatronic production systems. In: *Journal of Engineering Design* (2018), pp. 1–27

[KeVo13] KERNSCHMIDT, K. and VOGEL-HEUSER, B.: An interdisciplinary SysML based modeling approach for analyzing change influences in production plants to support the engineering. In: *IEEE International Conference on Automation Science and Engineering (CASE)*, 2013, pp. 1113–1118

[KGSE14] KRUSE, B.; GILZ, T.; SHEA, K. and EIGNER, M.: Systematic Comparison of Functional Models in SysML for Design Library Evaluation. In: *Procedia CIRP* Vol. 21 (2014), pp. 34–39

[KHJM99] KOREN, Y.; HEISEL, U.; JOVANE, F.; MORIWAKI, T.; PRITSCHOW, G.; ULSOY, G. and VAN BRUSSEL, H.: Reconfigurable Manufacturing Systems. In: *CIRP Annals - Manufacturing Technology* Vol. 48 (1999), No. 2, pp. 527–540

[KHMV12] KERNSCHMIDT, K.; HEPPERLE, C.; MÖRTL, M. and VOGEL-HEUSER, B.: Lifecycle Oriented Planning of Mechatronic Products and Corresponding Services. In: RIVEST, L.; BOURAS, A.; LOUHICHI, B. (eds.): *Product Lifecycle Management. Towards Knowledge-Rich Enterprises. PLM 2012. IFIP Advances in Information and Communication Technology*. Vol. 388. Berlin, Heidelberg: Springer, 2012, pp. 349–358

[KKJG13] KERNSCHMIDT, K.; KLEIN, P.; JAZDI, N.; GÖHNER, P.; WEYRICH, M. and VOGEL-HEUSER, B.: Methodology for Identification of Adaptive Reusable Modules in Automated Production Systems. In: ABRAMOVICI, M.; STARK, R. (eds.): *Smart Product Engineering - Lecture Notes in Production Engineering*. Berlin, Heidelberg: Springer, 2013, pp. 125–135

[KMWC12] KRUSE, B.; MÜNZER, C.; WÖLKL, S.; CANEDO, A. and SHEA, K.: A Model-Based Functional Modeling and Library Approach for Mechatronic Systems in SysML. In: *ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference (IDETC/CIE)*, 2012, pp. 1217–1227

[KPRV16] KERNSCHMIDT, K.; PREIßNER, S.; RAASCH, C. and VOGEL-HEUSER, B.: From Selling Products to Providing User Oriented Product-Service Systems – Exploring Service Orientation in the German Machine and Plant Manufacturing Industry. In: BOURAS, A.; EYNARD, B.; FOUFOU, S.; THOBEN, K. (eds.): *Product Lifecycle Management in the Era of Internet of Things. PLM 2015. IFIP Advances in Information and Communication Technology*. Vol. 467. Cham: Springer, 2016, pp. 280–290

[KrLS95] KROGSTIE, J.; LINDLAND, O.I. and SINDRE, G.: Defining quality aspects for conceptual models. In: FALKENBERG, E. D.; HESSE, W.; OLIVÉ, A. (eds.): *Information System Concepts - Towards a consolidation of views*. Cham: Springer, 1995, pp. 216–231

[Krog03] KROGSTIE, J.: Evaluating UML using a generic quality framework. In: ARLOW, J.; NEUSTADT, I. (eds.): *UML and the Unified Process*. Hershey: IGI Publishing, 2003, pp. 1–22

[KrSh16] KRUSE, B. and SHEA, K.: Design Library Solution Patterns in SysML for Concept Design and Simulation. In: *Procedia CIRP* Vol. 50 (2016), pp. 695–700

[KrSJ06] KROGSTIE, J.; SINDRE, G. and JØRGENSEN, H.: Process models representing knowledge for action: a revised quality framework. In: *European Journal of Information Systems* Vol. 15 (2006), No. 1, pp. 91–102

[KuGr06] KUEHNL, C.P. and GRABBE, W.: Praktische Erfahrungen beim Einsatz von Mechatronik im Maschinenbau. In: *Tagungsband SPS/IPC/DRIVES 2006 / Elektrische Automatisierung, Systeme und Komponenten Fachmesse & Kongress*, 2006, pp. 1–9

[KWMK13] KERNSCHMIDT, K.; WOLFENSTETTER, T.; MÜNZBERG, C.; KAMMERL, D.; GOSWAMI, S.; LINDEMANN, U.; KRCMAR, H. and VOGEL-HEUSER, B.: Concept for an Integration-Framework to enable the crossdisciplinary Development of Product-Service Systems. In: *IEEE International Conference on Industrial Engineering and Engineering Managemen (IEEM)*, 2013, pp. 340–345

[LaGö99] LAUBER, R. and GÖHNER, P.: *Prozessautomatisierung 1*. 3rd ed. Berlin, Heidelberg: Springer, 1999

[Lang16] LANGER, S.: *Kritische Änderungen in der Produktentwicklung – Analyse und Maßnahmenableitung*, Dissertation, Technical University of Munich, 2016

[LBKV12] LI, F.; BAYRAK, G.; KERNSCHMIDT, K. and VOGEL-HEUSER, B.: Specification of the Requirements to Support Information Technology-Cycles in the Machine and Plant Manufacturing Industry. In: *IFAC Proceedings Volumes* Vol. 45 (2012), No. 6, pp. 1077–1082

[LEHM11] LÜDER, A.; ESTÉVEZ, E.; HUNDT, L. and MARCOS, M.: Automatic transformation of logic models within engineering of embedded mechatronical units. In: *The International Journal of Advanced Manufacturing Technology* Vol. 54 (2011), No. 9–12, pp. 1077–1089

[Lerc16] LERCHENFELD, L.M.F. VON: Interdisciplinary Modeling of Behavioral Properties for Mechatronic Systems, Bachelor Thesis, Technical University of Munich, Supervisor: Dipl.-Ing. Konstantin Kernschmidt (2016)

[LFWZ10] LÜDER, A.; FOEHR, L.H.M.; WAGNER, T.; ZADDACH, J.-J. and HOLM, T.: Manufacturing system engineering with mechatronical units. In: *IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, 2010, pp. 1–8

[LGMS13] LEPUSCHITZ, W.; GROESSING, B.; MERDAN, M. and SCHITTER, G.: Evaluation of a multi-agent approach for a real transportation system. In: *IEEE International Conference on Industrial Technology (ICIT)*, 2013, pp. 1273–1278

[Li14] LI, F.: *Anforderungsermittlung an ein Beschreibungsmittel und Erweiterung einer Gerätebeschreibungssprache für die modulare Automatisierung im Maschinen- und Anlagenbau*, Dissertation, Technical University of Munich, 2014

[LiPa03] LIANG, V.-C. and PAREDIS, C.J.J.: A port ontology for automated model composition. In: *Proceedings of the 2003 International Conference on Machine Learning and Cybernetics*, 2003, pp. 613–622

[LiSS94] LINDLAND, O.I.; SINDRE, G. and SOLVBERG, A.: Understanding quality in conceptual modeling. In: *IEEE Software* Vol. 11 (1994), No. 2, pp. 42–49

[LKKP18] LOCH, F.; KOLTUN, G.; KARASEVA, V.; PANTFÖRDER, D. and VOGEL-HEUSER, B.: Model-based training of manual procedures in automated production systems. In: *Mechatronics* (2018)

[Lude03] LUDEWIG, J.: Models in software engineering - an introduction. In: *Software and Systems Modeling* Vol. 2 (2003), No. 1, pp. 5–14

[LWML12] LANGER, S.; WILBERG, J.; MAIER, A. and LINDEMANN, U.: *Änderungsmanagement-Report 2012 – Studienergebnisse zu Ursachen und Auswirkungen, aktuellen Praktiken, Herausforderungen und Strategien in Deutschland*: Technical University of Munich, 2012

[MaCF15] MARGINI, A.; CUTRONA, G. and FANTUZZI, C.: Structured Product Development Process Implementation for a Packaging Company. In: *IFAC-PapersOnLine* Vol. 48 (2015), No. 10, pp. 190–196

[MaEs08] MARCOS, M. and ESTÉVEZ, E.: Model-driven design of Industrial Control Systems. In: *IEEE International Conference on Computer-Aided Control Systems (CACSD)*, 2008, pp. 1253–1258

[MaJG11a] MAGA, C.R.; JAZDI, N. and GÖHNER, P.: Reusable Models in Industrial Automation: Experiences in Defining Appropriate Levels of Granularity. In: *IFAC Proceedings Volumes* Vol. 44 (2011), No. 1, pp. 9145–9150

[MaJG11b] MAGA, C.R.; JAZDI, N. and GÖHNER, P.: Requirements on engineering tools for increasing reuse in industrial automation. In: *IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, 2011, pp. 1–7

[MaPo07] MAES, A. and POELS, G.: Evaluating quality of conceptual modelling scripts based on user perceptions. In: *Data & Knowledge Engineering* Vol. 63 (2007), No. 3, pp. 701–724

[MKKV13] MAISENBACHER, S.; KERNSCHMIDT, K.; KASPEREK, D.; VOGEL-HEUSER, B. and MAURER, M.: Using DSM and MDM methodologies to analyze structural SysML models. In: *IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, 2013, pp. 351–355

[Mood09] MOODY, D.: The "Physics" of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering. In: *IEEE Transactions on Software Engineering* Vol. 35 (2009), No. 6, pp. 756–779

[More69] MORE, T.: Mecha-tronics. In: *Yaskawa Internal Trademark Application Memo 21.131.01*, 1969

[Mota13] MOTAMEDIAN, B.: MBSE Applicability Analysis. In: *International Journal of Scientific & Engineering Research* Vol. 4 (2013), No. 2, pp. 1–7

[MQAW12] MØRKEBERG TORRY-SMITH, J.; QAMAR, A.; ACHICHE, S.; WIKANDER, J.; HENRIK MORTENSEN, N. and DURING, C.: Challenges in Designing

Mechatronic Systems. In: *Journal of Mechanical Design* Vol. 135 (2012), No. 1, pp. 011005-1-011005-11

[MSBS02] MOODY, D.L.; SINDRE, G.; BRASETHVIK, T. and SØLVBERG, A.: Evaluating the Quality of Process Models: Empirical Testing of a Quality Framework. In: SPACCAPIETRA, S.; MARCH, S. T.; KAMBAYASHI, Y. (eds.): *Conceptual Modeling — ER 2002. Lecture Notes in Computer Science*. Vol. 2503. Berlin, Heidelberg: Springer, 2002, pp. 380–396

[MSBS03] MOODY, D.L.; SINDRE, G.; BRASETHVIK, T. and SOLVBERG, A.: Evaluating the quality of information models: empirical testing of a conceptual model quality framework. In: *IEEE International Conference on Software Engineering (ICSE)*. Vol. 6, 2003, pp. 295–305

[MuRo02] MUFFATTO, M. and ROVEDA, M.: Product architecture and platforms: a conceptual framework. In: *International Journal of Technology Management* Vol. 24 (2002), No. 1, pp. 1–16

[ObBV15] OBERMEIER, M.; BRAUN, S. and VOGEL-HEUSER, B.: A Model-Driven Approach on Object-Oriented PLC Programming for Manufacturing Systems with Regard to Usability. In: *IEEE Transactions on Industrial Informatics* Vol. 11 (2015), No. 3, pp. 790–800

[Ober15] OBERMEIER, M.: *Untersuchung von Programmieransätzen im Maschinen- und Anlagenbau bezüglich ihrer Gebrauchstauglichkeit sowie ihrer Einflüsse auf die Softwaremodularisierung*, Dissertation, Technical University of Munich, 2015

[OBSV11] OBERMEIER, M.; BRAUN, S.; SOMMER, K. and VOGEL-HEUSER, B.: Fundamental Aspects Concerning the Usability Evaluation of Model-Driven Object Oriented Programming Approaches in Machine and Plant Automation. In: MARCUS, A. (ed.): *Design, User Experience, and Usability. Theory, Methods, Tools and Practice. DUXU 2011. Lecture Notes in Computer Science*. Vol. 6770. Berlin, Heidelberg: Springer, 2011, pp. 497–506

[Omgo03] OMG OBJECT MANAGEMENT GROUP: UML for Systems Engineering Request For Proposal, ad/03-03-41 (2003)

[Omgo05] OMG OBJECT MANAGEMENT GROUP: Unified Modeling Language: Superstructure, Version 2.0, formal/05-07-04 (2005)

[Omgo08] OMG OBJECT MANAGEMENT GROUP: UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded systems, ptc/2008-06-09 (2008)

[Omgo13] OMG OBJECT MANAGEMENT GROUP: Business Process Model and Notation (BPMN), Version 2.0.2, formal/2013-12-09 (2013)

[Omgo15a] OMG OBJECT MANAGEMENT GROUP: OMG Unified Modeling Language (OMG UML), Version 2.5, formal/2015-03-01 (2015)

[Omgo15b] OMG OBJECT MANAGEMENT GROUP: OMG Systems Modeling Language (OMG SysML), Version 1.4, formal/2015-06-03 (2015)

[Omgo15c] OMG OBJECT MANAGEMENT GROUP: OMG Meta Object Facility (MOF) Core Specification, Version 2.5, formal/2015-06-05 (2015)

[Omgo15d] OMG OBJECT MANAGEMENT GROUP: XML Metadata Interchange Specification (XMI), Version 2.5.1, formal/2015-06-07 (2015)

[Oost01] OOSTERMAN, B.: *Improving product development projects by matching product architecture and organization*, Dissertation, University of Groningen, 2001

[PBBK10] PAREDIS, C.J.J.; BERNARD, Y.; BURKHART, R.; DE KONING, H.-P.; FRIEDENTHAL, S.; FRITZSON, P.; ROUQUETTE, N. and SCHAMAI, W.: An overview of the SysML-Modelica transformation specification. In: *INCOSE International Symposium*, 2010, pp. 1–14

[PiEp94] PIMMLER, T.U. and EPPINGER, S.D.: Integration analysis of product decompositions. In: *ASME Conference on Design Theory and Methodology*, 1994

[Plco09] PLCOPEN TECHNICAL COMMITTEE 6: XML Formats for IEC 61131-3 - Version 2.01 – Official Release (2009)

[PLRW92] POLSON, P.G.; LEWIS, C.; RIEMAN, J. and WHARTON, C.: Cognitive walkthroughs: a method for theory-based evaluation of user interfaces. In: *International Journal of Man-Machine Studies* Vol. 36 (1992), No. 5, pp. 741–773

[Poss10] POSSEL-DÖLKEN, F.: Mechatronische Modularisierung im Sondermaschinenbau. In: *HNI-Verlagsschriftenreihe* Vol. 272 (2010), pp. 357–370

[PTRC07] PEFFERS, K.; TUUNANEN, T.; ROTHENBERGER, M.A. and CHATTERJEE, S.: A Design Science Research Methodology for Information Systems Research. In: *Journal of Management Information Systems* Vol. 24 (2007), No. 3, pp. 45–78

[RHZS17] RAMSAIER, M.; HOLDER, K.; ZECH, A.; STETTER, R.; RUDOLPH, S. and TILL, M.: Digital representation of product functions in multicopter design. In: *Proceedings of the 21st International Conference on Engineering Design (ICED 17) Vol. 1: Resource Sensitive Design, Design Research Applications and Case Studies*, 2017

[Ritt10] RITTGEN, P.: Quality and perceived usefulness of process models. In: *ACM Symposium on Applied Computing (SAC)*, 2010, p. 65

[Rodr15] RODRIGUES DA SILVA, A.: Model-driven engineering: A survey supported by the unified conceptual model. In: *Computer Languages, Systems & Structures* Vol. 43 (2015), pp. 139–155

[RoLF07] RO, Y.K.; LIKER, J.K. and FIXSON, S.K.: Modularity as a Strategy for Supply Chain Coordination: The Case of U.S. Auto. In: *IEEE Transactions on Engineering Management* Vol. 54 (2007), No. 1, pp. 172–189

[RSSR17] RAMSAIER, M.; SPINDLER, C.; STETTER, R.; RUDOLPH, S. and TILL, M.: Digital Representation in Multicopter Design Along the Product Life-cycle. In: *Procedia CIRP* Vol. 62, The Author(s) (2017), pp. 559–564

[Scha13] SCHALLES, C.: *Usability Evaluation of Modeling Languages*. Wiesbaden: Springer Fachmedien, 2013

[Sche94] SCHEER, A.-W.: *Business Process Engineering*. 2nd ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1994

[Schi00] SCHILLING, M.A.: Toward a General Modular Systems Theory and Its Application to Interfirm Product Modularity. In: *Academy of Management Review* Vol. 25 (2000), No. 2, pp. 312–334

[Schm11] SCHMERTOSCH, T.: Maschinen modulorientiert automatisieren. In: *5. Ostsächsische Maschinenbautage*, 2011

[Schü15] SCHÜTZ, D.: *Automatische Generierung von Softwareagenten für die industrielle Automatisierungstechnik der Steuerungsebene des Maschinen- und Anlagenbaus auf Basis der Systems Modeling Language*, Dissertation, Technical University of Munich, 2015

[SCKP07] SIERLA, S.; CHRISTENSEN, J.; KOSKINEN, K. and PELTOLA, J.: Educational approaches for the industrial acceptance of IEC 61499. In: *IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, 2007, pp. 482–489

[ScLV14] SCHÜTZ, D.; LEGAT, C. and VOGEL-HEUSER, B.: MDE of manufacturing automation software — Integrating SysML and standard development tools. In: *IEEE International Conference on Industrial Informatics (INDIN)*, 2014, pp. 267–273

[ScVo11] SCHÜTZ, D. and VOGEL-HEUSER, B.: Modellintegration von Verhaltens- und energetischen Aspekten für mechatronische Module. In: *at - Automatisierungstechnik* Vol. 59 (2011), No. 1, pp. 33–41

[ScWe10] SCHÄFER, W. and WEHRHEIM, H.: Model-Driven Development with Mechatronic UML. In: ENGELS, G.; LEWERENTZ, C.; SCHÄFER, W.; SCHÜRR, A.; WESTFECHTEL, B. (eds.): *Graph Transformations and Model-Driven Engineering*. Berlin, Heidelberg: Springer, 2010, pp. 533–554

[SeBF07] SECCHI, C.; BONFE, M. and FANTUZZI, C.: On the Use of UML for Modeling Mechatronic Systems. In: *IEEE Transactions on Automation Science and Engineering* Vol. 4 (2007), No. 1, pp. 105–113

[SeKü04] SENDALL, S. and KÜSTER, J.: Taming model round-trip engineering. In: *Proceedings of Workshop on Best Practices for Model-Driven Software Development*, 2004, p. 13

[Seli00] SELIC, B.: A generic framework for modeling resources with UML. In: *Computer* Vol. 33 (2000), No. 6, pp. 64–69

[Seli03] SELIC, B.: The pragmatics of model-driven development. In: *IEEE Software* Vol. 20 (2003), No. 5, pp. 19–25

[Seli04] SELIC, B.: On the Semantic Foundations of Standard UML 2.0. In: BERNARDO, M.; CORRADINI, F. (eds.): *Formal Methods for the Design of Real-Time Systems*. Vol. 3185. Berlin, Heidelberg: Springer, 2004, pp. 181–199

[Seli07] SELIC, B.: From Model-Driven Development to Model-Driven Engineering. In: *IEEE Euromicro Conference on Real-Time Systems (ECRTS)*, 2007, pp. 3–3

[Seli98] SELIC, B.: Using UML for modeling complex real-time systems. In: MUELLER, F.; BESTAVROS, A. (eds.): *Languages, Compilers, and Tools for Embedded Systems. Lecture Notes in Computer Science*. Vol. 1474. Berlin, Heidelberg: Springer, 1998, pp. 250–260

[SKSP10] SHAH, A.A.; KERZHNER, A.A.; SCHAEFER, D. and PAREDIS, C.J.J.: Multi-view Modeling to Support Embedded Systems Engineering in SysML. In: ENGELS, G.; LEWERENTZ, C.; SCHÄFER, W.; SCHÜRR, A.; WESTFECHTEL, B. (eds.): *Graph Transformations and Model-Driven Engineering*, *Lecture*

Notes in Computer Science (LNCS). Vol. 5765. Berlin, Heidelberg: Springer, 2010, pp. 580–601

[SoNa14] SOARES, M.S. and DO NASCIMENTO, R.P.C.: Evaluation of SysML diagrams to document requirements using TAM. In: *ACM Euro American Conference on Telematics and Information Systems (EATIS)*, 2014, pp. 1–6

[Stac73] STACHOWIAK, H.: *Allgemeine Modelltheorie*. Wien, New York: Springer, 1973

[StWC00] STONE, R.B.; WOOD, K.L. and CRAWFORD, R.H.: A heuristic method for identifying modules for product architectures. In: *Design Studies* Vol. 21 (2000), No. 1, pp. 5–31

[Thra04] THRAMBOULIDIS, K.: Using UML in control and automation: a model driven approach. In: *IEEE International Conference on Industrial Informatics (INDIN)*, 2004, pp. 587–593

[Thra10] THRAMBOULIDIS, K.: The 3+1 SysML View-Model in Model Integrated Mechatronics. In: *Journal of Software Engineering and Applications* Vol. 03 (2010), No. 02, pp. 109–118

[Thra13] THRAMBOULIDIS, K.: Overcoming Mechatronic Design Challenges: the 3+1 SysML-view Model. In: *The Computing Science and Technology International Journal* Vol. 1 (2013), No. 1, pp. 6–14

[Ulri95] ULRICH, K.: The role of product architecture in the manufacturing firm. In: *Research Policy* Vol. 24 (1995), No. 3, pp. 419–440

[VBOJ12] VOGEL-HEUSER, B.; BRAUN, S.; OBERMEIER, M.; JOBST, F. and SCHWEIZER, K.: Usability evaluation on teaching and applying model-driven object oriented approaches for PLC software. In: *IEEE American Control Conference (ACC)*, 2012, pp. 4463–4469

[VDFJ14] VOGEL-HEUSER, B.; DIEDRICH, C.; FAY, A.; JESCHKE, S.; KOWALEWSKI, S.; WOLLSCHLAEGER, M. and GÖHNER, P.: Challenges for Software Engineering in Automation. In: *Journal of Software Engineering and Applications* Vol. 07 (2014), No. 05, pp. 440–451

[Vdiv04] VDI VEREIN DEUTSCHER INGENIEURE: VDI 2206, Design methodology for mechatronic systems (2004)

[Vdiv05] VDI/VDE VEREIN DEUTSCHER INGENIEURE/VERBAND DER ELEKTROTECHNIK ELEKTRONIK INFORMATIONSTECHNIK: VDI/VDE 3681, Classification and evaluation of description methods in automation and control technology (2005)

[Vdiv10] VDI/VDE VEREIN DEUTSCHER INGENIEURE/VERBAND DER ELEKTROTECHNIK ELEKTRONIK INFORMATIONSTECHNIK: Guideline VDI/VDE 3695-2, Engineering of industrial plants Evaluation and optimization - Subject processes (2010)

[Vdiv93] VDI VEREIN DEUTSCHER INGENIEURE: VDI 2221, Systematic approach to the development and design of technical systems and products (1993)

[Vdiv94] VDI VEREIN DEUTSCHER INGENIEURE: VDI 2422, Systematical development of devices controlled by microelectronics (1994)

[Vdma15] VDMA VERBAND DEUTSCHER MASCHINEN- UND ANLAGENBAU: IT und Automation im Maschinenbau – VDMA-Trendstudie (2015)

[VeDa00] VENKATESH, V. and DAVIS, F.D.: A Theoretical Extension of the Technology Acceptance Model: Four Longitudinal Field Studies. In: *Management Science* Vol. 46 (2000), No. 2, pp. 186–204

[VFFU17] VOGEL-HEUSER, B.; FISCHER, J.; FELDMANN, S.; ULEWICZ, S. and RÖSCH, S.: Modularity and architecture of PLC-based software for automated production Systems: An analysis in industrial companies. In: *Journal of Systems and Software* Vol. 131 (2017), pp. 35–62

[VFST15] VOGEL-HEUSER, B.; FAY, A.; SCHAEFER, I. and TICHY, M.: Evolution of software in automated production systems: Challenges and research directions. In: *Journal of Systems and Software* Vol. 110 (2015), pp. 54–84

[VoCu91] VOGWELL, J. and CULLEY, S.J.: A strategy for selecting engineering components. In: *Journal of Engineering Manufacture* Vol. 205 (1991), No. 12, pp. 11–17

[VoHe16] VOGEL-HEUSER, B. and HESS, D.: Guest Editorial Industry 4.0–Prerequisites and Visions. In: *IEEE Transactions on Automation Science and Engineering* Vol. 13 (2016), No. 2, pp. 411–413

[VoOc18] VOGEL-HEUSER, B. and OCKER, F.: Maintainability and evolvability of control software in machine and plant manufacturing — An industrial survey. In: *Control Engineering Practice* Vol. 80, Elsevier Ltd (2018), No. August, pp. 157–173

[VoWK05] VOGEL-HEUSER, B.; WITSCH, D. and KATZKE, U.: Automatic Code Generation from a UML model to IEC 61131-3 and system configuration tools. In: *IEEE International Conference on Control and Automation (ICCA)*, 2005, pp. 1034–1039

[VSFL14] VOGEL-HEUSER, B.; SCHÜTZ, D.; FRANK, T. and LEGAT, C.: Model-driven engineering of Manufacturing Automation Software Projects – A SysML-based approach. In: *Mechatronics* Vol. 24, Elsevier Ltd (2014), No. 7, pp. 883–897

[Vyat11] VYATKIN, V.: IEC 61499 as Enabler of Distributed and Intelligent Automation: State-of-the-Art Review. In: *IEEE Transactions on Industrial Informatics* Vol. 7 (2011), No. 4, pp. 768–781

[Vyat13] VYATKIN, V.: Software Engineering in Industrial Automation: State-of-the-Art Review. In: *IEEE Transactions on Industrial Informatics* Vol. 9 (2013), No. 3, pp. 1234–1249

[Wann10] WANNAGAT, A.: *Entwicklung und Evaluation agentenorientierter Automatisierungssysteme zur Erhöhung der Flexibilität und Zuverlässigkeit von Produktionsanlagen*, Dissertation, Technical University of Munich, 2010

[WaTT18] WANG, H.; THOMSON, V. and TANG, C.: Change propagation analysis for system modeling using Semantic Web technology. In: *Advanced Engineering Informatics* Vol. 35, Elsevier (2018), No. May 2017, pp. 17–29

[WBJF92] WHARTON, C.; BRADFORD, J.; JEFFRIES, R. and FRANZKE, M.: Applying cognitive walkthroughs to more complex user interfaces. In: *SIGCHI conference on Human factors in computing systems (CHI)*. Vol. 53, 1992, pp. 381–388

[Wern09] WERNER, B.: Object-oriented extensions for IEC 61131-3. In: *IEEE Industrial Electronics Magazine* Vol. 3 (2009), No. 4, pp. 36–39

[Winn93] WINN, W.: An Account of How Readers Search for Information in Diagrams. In: *Contemporary Educational Psychology* Vol. 18 (1993), No. 2, pp. 162–185

[Wits12] WITSCH, D.: *Modellgetriebene Entwicklung von SteuerungsSoftware auf Basis der UML unter Berücksichtigung der domänenspezifischen Anforderungen des Maschinen- und Anlagenbaus*, Dissertation, Technical University of Munich, 2012

[Wits13] WITSCH, M.: *Funktionale Spezifikation von Manufacturing Execution Systems im Spannungsfeld zwischen IT, Geschäftsprozess und Produktion*, Dissertation, Technical University of Munich, 2013

[WiVo11] WITSCH, D. and VOGEL-HEUSER, B.: PLC-Statecharts: An Approach to Integrate UML-Statecharts in Open-Loop Control Engineering – Aspects on Behavioral Semantics and Model-Checking. In: *IFAC Proceedings Volumes* Vol. 44 (2011), No. 1, pp. 7866–7872

[WKLW11a] WEYRICH, M.; KLEIN, P.; LAUROWSKI, M. and WANG, Y.: Mechatronic engineering of novel manufacturing processes implemented by modular and sensor-guided machinery. In: *IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, 2011, pp. 1–7

[WKLW11b] WEYRICH, M.; KLEIN, P.; LAUROWSKI, M. and WANG, Y.: A function-oriented approach for a mechatronic Modularization of a sensor-guided Manufacturing System. In: *Innovation in mechanical engineering - shaping the future: International scientific colloquium Ilmenau University of Technology*, 2011, pp. 12–16

[WoSU14] WOLLSCHLAEGER, M.; SCHRIEBER, R. and ULLEMEYER, M.: Life-cycle-management in automation-models and strategies. In: *Annual Conference of the IEEE Industrial Electronics Society (IECON)*, 2014, pp. 2578–2584

[WRLP94] WHARTON, C.; RIEMAN, J.; LEWIS, C. and POLSON, P.: The cognitive walkthrough method: A practitioner's guide – Technical Report #93-07 (1994)

[Zafi14] ZAFIROV, R.: Produktionsmodelle und Simulation (DiFa). In: EIGNER, M.; ROUBANOV, D.; ZAFIROV, R. (eds.): *Modellbasierte virtuelle Produktentwicklung*. Berlin, Heidelberg: Springer, 2014, pp. 197–226

[Zafi17] ZAFIROV, R.: *Model-based systems engineering methods for integrated product design, process planning, and production systems design*: Universität Kaiserslautern, 2017

[ZhGe03] ZHANG, Y. and GERSHENSON, J.: An initial study of direct relationships between life-cycle modularity and life-cycle cost. In: *Concurrent Engineering* Vol. 11 (2003), No. 2, pp. 121–128

10. List of figures

Figure 1-1: Applied research methodology 7
Figure 1-2: Structure of the thesis 10
Figure 2-1: Basic structure of a mechatronic system according to VDI 2206 13
Figure 2-2: Interrelation between technical process and technical system 14
Figure 2-3: Systematic mechatronic development process based on VDI 2422 15
Figure 2-4: Gaps in the interdisciplinary development 16
Figure 2-5: V-model according to VDI 2206 16
Figure 2-6: Project-independent tasks and project-related tasks in the lifecycle of aPS 17
Figure 2-7: Reuse 18
Figure 2-8: Classification of Model-Based engineering 19
Figure 2-9: Modeling domain 20
Figure 3-1: Survey results regarding modeling languages 24
Figure 3-2: Answers regarding the interdisciplinarity of teams 25
Figure 3-3: Survey results regarding the expected efficiency of Model-Based approaches 25
Figure 3-4: Survey results regarding modularity 26
Figure 3-5: Answers regarding the analysis of change impacts 27
Figure 4-1: Semantic layers of UML 39
Figure 4-2: Relationship between UML 2 and SysML 40
Figure 4-3: SysML diagram taxonomy 41
Figure 4-4: Example of a BDD (left) and IBD (right) in SysML 42
Figure 4-5: Four layer MOF-based MDA framework of the OMG 43
Figure 4-6: Example of a property and its property carrier 51
Figure 4-7: eCl@ss structure at the example of an inductive proximity switch 54
Figure 4-8: Three-level module model at the example of a fiberboard production system 58
Figure 4-9: Conflicting aspects of module-granularity 60
Figure 4-10: Comparison of existing approaches and resulting research gap in MDE of mechatronic systems 64
Figure 5-1: SysML4Mechatronics profile in the context of SysML and UML 67
Figure 5-2: SysML4Mechatronics metamodel – components and modules 69
Figure 5-3: SysML4Mechatronics metamodel – ports and interfaces 70
Figure 5-4: SysML4Mechatronics metamodel – properties and functionalities 72
Figure 5-5: SysML4Mechatronics metamodel – value types 73
Figure 5-6: Concrete Syntax of SysML4Mechatronics blocks and modules 74
Figure 5-7: SysML4Mechatronics visualization at the example of a cylinder module 75
Figure 5-8: Example for levels of detail for properties from different aspects 76

Figure 5-9: SysML4Mechatronics metamodel – properties in eCl@ss model library ... 80
Figure 5-10: SysML4Mechatronics metamodel – eCl@ss component library 81
Figure 5-11: Usage of SysML4Mechatronics during engineering of mechatronic systems, extension of the V-model 82
Figure 5-12: Integrated analysis of solution alternatives and their impacts 84
Figure 5-13: Overview of the compatibility analysis using SysML4Mechatronics and a formal model representation..... 85
Figure 5-14: Comparison of different degrees of compatibility 86
Figure 6-1: SEQUAL Framework for model quality..... 89
Figure 6-2: Screenshot of the SysML4Mechatronics modeling editor..... 91
Figure 6-3: Selection of an eCl@ss component and properties in the editor 92
Figure 6-4: Handling station (left) and three-way switch (right)..... 94
Figure 6-5: Schematic representation of the pallet transportation system..... 95
Figure 6-6: High-level HTA of “Design mechatronic system” task..... 96
Figure 6-7: HTA of “Develop detailed concept” without integrated modeling approach..... 97
Figure 6-8: Excerpt of the system with six three-way switches 98
Figure 6-9: HTA of “Develop detailed concept” with SysML4Mechatronics 99
Figure 6-10: SysML4Mechatronics model of the three-way switch 100
Figure 6-11: SysML4Mechatronics model of the connecting conveyor module 100
Figure 6-12: HTA of “Exchange module” without integrated modeling approach (discipline-specific models) 101
Figure 6-13: HTA of “Exchange module” with SysML4Mechatronics 102
Figure 6-14: Handling station with barcode-scanner in SysML4Mechatronics 103
Figure 6-15: Handling station with RFID-reader in SysML4Mechatronics..... 104
Figure 6-16: Excerpt of the SysML4Mechatronics ribbon for creating components ... 109
Figure 6-17: Specification of component properties in the SysML4Mechatronics editor a) original interface, b) revised interface after cognitive walkthrough 110
Figure 6-18: a) E/E components with ports from different disciplines, b) disabled ports of the other disciplines while connecting an E/E port 111
Figure 6-19: Extended Technology Acceptance Model 113
Figure 6-20: Silo system of the evaluation assignment 115
Figure 6-21: Participants’ knowledge in programming/modeling languages..... 117
Figure 6-22: Participants’ experience with the interdisciplinarity of mechatronic systems and their interest in developing machines..... 117
Figure 6-23: Results of the TAM-questionnaire regarding the perceived benefits of using SysML4Mechatronics..... 118
Figure 6-24: Results of the TAM-questionnaire regarding the perceived ease of use of SysML4Mechatronics 119

Figure 6-25: Results of the questionnaire regarding the usability of SysML4Mechatronics 120

11. List of tables

Table 1: Structure of survey participants 23
Table 2: MBE-approaches for mechatronic systems – comparison of existing modeling approaches 50
Table 3: Comparison of existing approaches focusing on properties in engineering 57
Table 4: Comparison of existing approaches focusing on modularization 63
Table 5: Distinction of cases to generate module properties from component properties 79
Table 6: Summary of the requirements fulfillment of SysML4Mechatronics 125

Appendix

Lehrstuhl für Automatisierung
und Informationssysteme
Prof. Dr.-Ing. B. Vogel-Heuser (Ordinaria)

Technische Universität München

Fragebogen zur Modellbasierten Entwicklung und Analyse von Änderungsauswirkungen

Vielen Dank, dass Sie bereit sind, an der folgenden Befragung teilzunehmen. Im Rahmen der Entwicklung von modellbasierten Ansätzen zur interdisziplinären Entwicklung möchten wir einige Informationen über Ihre Erfahrung und Vorgehen mit Modellen erheben. Im Fragebogen gibt es keine richtigen oder falschen Antworten. Ziel ist es Ihre persönliche Einschätzung zu Potentialen und Herausforderungen bei Modell-basierten Ansätzen zu erfassen, um diese bei der Entwicklung der Modellierungssprache SysML4*Mechatronics* berücksichtigen zu können. Ihre Angaben werden selbstverständlich anonym und vertraulich behandelt.

Zunächst sollen Sie einige Angabe zu Ihrer Person und Ihrer beruflichen Erfahrung machen.

1. Alter: _____

2. Geschlecht weiblich/ männlich

3. gelernter Beruf bzw. Studium _____

4. tätig als _____

Welche Aufgaben gehören zu Ihrer Tätigkeit?

_____

5. Welche Modellierungssprachen kennen Sie und wie häufig nutzen Sie diese?

	ist mir bekannt	gar nicht genutzt	selten genutzt	manchmal genutzt	oft genutzt	sehr oft genutzt
UML						
SysML						
BPMN						
Entity-Relationship-Modelle						

Sonstige:_____						

6. Wie schätzen Sie Ihre bisherige Erfahrung in der Modellierung mit den folgenden Notationen ein:

	keine Erfahrung	geringe Erfahrung	etwas Erfahrung	viel Erfahrung	sehr viel Erfahrung
UML					
SysML					
BPMN					
Entity-Relationship-Modelle					
Sonstige:_____					

7. Arbeiten Sie in einem Team mit unterschiedlichen Fachrichtungen?

gar nicht	selten	manchmal	oft	sehr oft

8. Welche außer den oben genannten Modellierungssprachen oder Spezifikationen nutzen Sie in der Systementwicklung und für welche Systemteile?

	Wird genutzt	Für welche Systemteile?
CAD (Mechanik)		
E-CAD		
IEC 61131-3		
IEC 61499		
TTCN-3		
Sonstige:		

9. Erachten Sie es für praktikabel, (Teile der) Qualitätssicherungsmaßnahmen in der Entwicklung weiter nach vorne zu verlagern?

ja	
nein	

Im Folgenden werden Ihnen einige Fragen zu Modell-basierten Entwicklungsansätzen, Modularisierung und der Umgang mit Änderungen gestellt.

Es gibt verschiedene Arten von Entwicklungstätigkeiten. Zum einen kann die Entwicklung von Systemen nur eine Disziplin (z.B. reine Softwaresysteme) beinhalten, zum anderen können auch interdisziplinäre Systeme im Fokus stehen, die verschiedene Disziplinen gemeinsam betrachten.

10. Welche Art von System steht im Fokus Ihrer Entwicklungstätigkeit?

Disziplinspezifische Systeme wie z.B. Software		→ Frage 13
Interdisziplinäre Systeme bestehend aus Komponenten mehrerer Disziplinen		→ Frage 11

In interdisziplinären Systemen gibt es verschiedene Arten der Modularisierung.

11. Welche Art von Modularisierung verwenden Sie?

In jeder Disziplin einzeln	
Disziplinübergreifende „reale“ Module, d.h. das mechatronische Modul besteht so auch in der Realität	
Disziplinübergreifende „virtuelle“ Module in der Entwicklung, d.h. die zusammengehörigen Komponenten bestehen als Modul im Modell, müssen jedoch in Realität nicht zusammenhängen	
Keine	

12. In wie weit sehen Sie Verbesserungspotenziale im systematischen Umgang mit Modellen, an denen mehrere Disziplinen (z. B. Mechanik, Pneumatik, Elektrik- und Elektronik, Software Engineering) beteiligt sind?

13. Im Gegensatz zu bisher gängigen Modelliersprachen ermöglicht SysML4*Mechatronics* die Darstellung disziplinspezifischer Komponenten (Mechanik, Elektronik, Software) in einem Gesamtsystem. Ziel ist dabei die Zusammenhänge, sowohl innerhalb einer Disziplin als auch Disziplinübergreifend, als Gesamtmodell darzustellen.

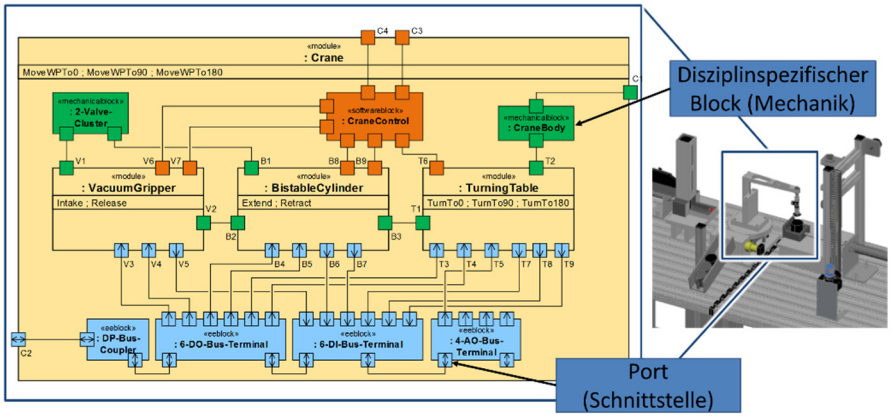
Folgende Darstellung zeigt das Modell eines Krans in SysML4*Mechatronics*.

a. Wie beurteilen Sie die Darstellung der verschiedenen disziplinspezifischen Komponenten im Gesamtsystem (Mechanik, Elektrik/Elektronik, Software)?

sehr unübersichtlich					sehr übersichtlich
- - -	- -	-	+	+ +	+ + +

b. Wie beurteilen Sie die Darstellung der Komponentenzusammenhänge über die Portdarstellung?

schwer verständlich					leicht verständlich
- - -	- -	-	+	+ +	+ + +



14. Wie schätzen Sie einen modellbasierten Entwicklungsansatz im Vergleich zu traditionellen Entwicklungsmethoden (in jeder Disziplin einzeln) ein?

viel ineffizienter					viel effizienter
- - -	- -	-	+	+ +	+ + +

15. Werden an Ihren Systemen Änderungen im Laufe des Lebenszyklus durchgeführt?

ja	
nein	

➔ Frage 16

16. Wie schätzen Sie die Auswirkung einer Änderung vor Implementierung der Änderung ab?

Im Modell	
Durch Expertenwissen	
Anders:_____	
Gar nicht	

Vielen Dank für die Teilnahme an der Befragung!

PC-Nummer:

_____

Fragebogen zu SysML4*Mechatronics* (Teil1)

Die Auswertung der Daten erfolgt selbstverständlich anonym.

Fragen zur Person

1. Alter: ☐
2. Geschlecht: ○ männlich ○ weiblich
3. Schulabschluss:
   ○ Abitur
   ○ Fachabitur
   ○ Sonstiges:
4. Studium:

   _____

5. Haben Sie bereits Erfahrung mit Programmierung?
   ○ ja ○ nein

6. Wenn ja, was haben Sie programmiert?

   _____

   _____

7. Mit welchen Programmiersprachen haben Sie bereits gearbeitet? Schreiben Sie maximal 5 auf und bewerten Sie jeweils, wie gut Sie die jeweilige Sprache beherrschen?

Programmiersprache	gar nicht	etwas	teils/ teils	gut	sehr gut
1. _____	○	○	○	○	○
2. _____	○	○	○	○	○
3. _____	○	○	○	○	○
4. _____	○	○	○	○	○
5. _____	○	○	○	○	○

8. Haben Sie bereits Erfahrungen mit Modellierungen?
   ○ ja ○ nein

9. Wenn ja, mit welchen Notationen haben Sie bereits gearbeitet? Schreiben Sie maximal 5 auf und bewerten Sie jeweils, wie gut Sie die Notation beherrschen?

Notation	gar nicht	etwas	teils/ teils	gut	sehr gut
1. UML	○	○	○	○	○
2. SYSML	○	○	○	○	○
3. _____	○	○	○	○	○
4. _____	○	○	○	○	○
5. _____	○	○	○	○	○

10. Haben Sie Erfahrung im Engineering von mechatronischen Systemen?

	gar keine	etwas	teils/ teils	gute	sehr gute
1. Wie viel Erfahrung haben Sie im Umgang mit interdisziplinären Engineering-Anforderungen in der Automatisierungstechnik?	○	○	○	○	○
2. Wie viel Freude bereitet Ihnen das Konzipieren von Maschinen?	○	○	○	○	○
3. Wie viel Erfahrung haben Sie bei der Erfassung des Zusammenspiels verschiedener Disziplinen (Mechanik/Elektrotechnik/Software) in mechatronischen Systemen?	○	○	○	○	○

PC-Nummer:

_____

Fragebogen zu SysML4*Mechatronics* (Teil2)

Dabei geht es nicht um eine Beurteilung Ihrer Person, sondern um Ihre persönliche Bewertung der Modellierung und Software. Die Auswertung der Daten erfolgt selbstverständlich anonym.

Fragebogen (Nutzen und Gebrauchstauglichkeit)

Bitte bewerten Sie folgende Aussagen über SysML4*Mechatronics*.

	Trifft gar nicht zu						Trifft voll zu
Die Anwendung von SysML4*Mechatronics* macht interdisziplinäre Modelle ausdrucksfähiger	❑	❑	❑	❑	❑	❑	❑
Durch SysML4*Mechatronics* können interdisziplinäre Abhängigkeiten der Komponenten effektiver abgeschätzt werden	❑	❑	❑	❑	❑	❑	❑
Merkmale (Attribute) der Komponenten und Module können in SysML4*Mechatronics* effektiver spezifiziert werden	❑	❑	❑	❑	❑	❑	❑
SysML4*Mechatronics* erleichtert die Wiederverwendung von Komponenten und Modulen	❑	❑	❑	❑	❑	❑	❑
Die Anwendung von SysML4*Mechatronics* würde die Entwicklung mechatronischer Systeme beschleunigen	❑	❑	❑	❑	❑	❑	❑
SysML4*Mechatronics*-Modelle würden auch in späteren Lebenszyklusphasen (z.B. zur Aufwandseinschätzung bei einer Änderung des Systems) nützlich sein	❑	❑	❑	❑	❑	❑	❑
Es ist einfach für mich SysML4*Mechatronics* zu erlernen	❑	❑	❑	❑	❑	❑	❑
Es ist einfach für mich die verschiedenen Disziplinen und Schnittstellen in SysML4*Mechatronics*-Diagrammen zu erkennen	❑	❑	❑	❑	❑	❑	❑
Es ist einfach für mich interdisziplinäre Module in SysML4*Mechatronics* zu erstellen	❑	❑	❑	❑	❑	❑	❑
Es wäre leicht für mich SysML4*Mechatronics* für das Engineering mechatronischer Systeme anzuwenden	❑	❑	❑	❑	❑	❑	❑
Durch SysML4*Mechatronics*-Diagramme wäre es leicht für mich mit anderen Fachdisziplinen (z.B. Elektrotechnik,Informatik) über das System zu sprechen	❑	❑	❑	❑	❑	❑	❑
Es wäre leicht für mich meine Kenntnisse in SysML4*Mechatronics* zu vertiefen	❑	❑	❑	❑	❑	❑	❑

Fragebogen (Usability)

Im Folgenden geht es um die Beurteilung der Usability oder Gebrauchstauglichkeit der Software auf Grundlage der Internationalen Norm ISO 9241/10.

Ziel ist es, herauszufinden, wie gut die Software aufgebaut ist und wie gut die Arbeit mit der Software durchgeführt werden kann.

Dabei geht es nicht um eine Beurteilung Ihrer Person, sondern um Ihre persönliche Bewertung der Software.

Noch ein Hinweis zur Beantwortung des Beurteilungsbogens:

Die einzelnen Normen werden über Beschreibungen konkretisiert. Diese Beschreibungen weisen immer folgende Form auf:

Beispiel:

Die Software	**---**	**--**	**-**	**-/+**	**+**	**++**	**+++**	**Die Software**
ist schlecht.	❑	❑	❑	❑	❑	☒	❑	ist gut.

Im ersten Beispiel wird danach gefragt, wie gut bzw. wie schlecht die Software ist. Der Benutzer beurteilt in diesem Fall die Software zwar als gut, sieht jedoch noch Verbesserungsmöglichkeiten.

Füllen Sie bitte den Beurteilungsbogen äußerst sorgfältig aus und lassen Sie keine der Fragen aus.

Die Auswertung der Daten erfolgt anonym.

Usability der verwendeten Modellierungssprache

Wie bewerten Sie die verwendete Modellierungssprache (**Notation)**?

Die Modellierung…

	---	--	-	-/+	+	++	+++	
ist kompliziert zu verstehen.	❑	❑	❑	❑	❑	❑	❑	ist einfach zu verstehen.
verwendet schwer verständliche, abstrakte Begriffe und Konzepte.	❑	❑	❑	❑	❑	❑	❑	verwendet leicht verständliche, anschauliche Begriffe und Konzepte.
ist ineffizient.	❑	❑	❑	❑	❑	❑	❑	ist effizient.
ist schlecht strukturiert und schwer nachzuvollziehen.	❑	❑	❑	❑	❑	❑	❑	ist gut strukturiert und leicht nachzuvollziehen.
erfordert viel Zeit zum Erlernen.	❑	❑	❑	❑	❑	❑	❑	erfordert wenig Zeit zum Erlernen.
erfordert, dass man sich viele Details merken muss.	❑	❑	❑	❑	❑	❑	❑	erfordert nicht, dass man sich viele Details merken muss.
Das Abbilden von Zusammenhängen ist schwierig.	❑	❑	❑	❑	❑	❑	❑	Das Abbilden von Zusammenhängen ist leicht.

Was fanden Sie an der Modellierungsweise gut, was hat sie gestört? Was würden Sie an der Modellierungsweise ändern / verbessern?

_____

_____

_____

_____

_____

_____

Aufgabenangemessenheit

Unterstützt die Software die Erledigung Ihrer Arbeitsaufgaben, ohne Sie als Benutzer unnötig zu belasten?

Die Software	---	--	-	-/+	+	++	+++	Die Software
ist kompliziert zu bedienen.	❑	❑	❑	❑	❑	❑	❑	ist unkompliziert zu bedienen.
bietet nicht alle Funktionen, um die anfallenden Aufgaben effizient zu bewältigen.	❑	❑	❑	❑	❑	❑	❑	bietet alle Funktionen, um die anfallenden Aufgaben effizient zu bewältigen.
bietet schlechte Möglichkeiten sich häufig wiederholende Bearbeitungsvorgänge zu automatisieren.	❑	❑	❑	❑	❑	❑	❑	bietet gute Möglichkeiten sich häufig wiederholende Bearbeitungsvorgänge zu automatisieren.
erfordert überflüssige Eingaben.	❑	❑	❑	❑	❑	❑	❑	erfordert keine überflüssigen Eingaben.
ist schlecht auf die Anforderungen der Arbeit zugeschnitten.	❑	❑	❑	❑	❑	❑	❑	ist gut auf die Anforderungen der Arbeit zugeschnitten.

Selbstbeschreibungsfähigkeit

Gibt Ihnen die Software genügend Erläuterungen und ist es in ausreichendem Maße verständlich?

Die Software	---	--	-	-/+	+	++	+++	Die Software
bietet einen schlechten Überblick über sein Funktionsangebot.	❑	❑	❑	❑	❑	❑	❑	bietet einen guten Überblick über sein Funktionsangebot.
verwendet schlecht verständliche Begriffe, Bezeichnungen, Abkürzungen oder Symbole in Masken und Menüs.	❑	❑	❑	❑	❑	❑	❑	verwendet gut verständliche Begriffe, Bezeichnungen, Abkürzungen oder Symbole in Masken und Menüs.
liefert in unzureichendem Maße Informationen darüber, welche Eingaben zulässig oder nötig sind.	❑	❑	❑	❑	❑	❑	❑	liefert in zureichendem Maße Informationen darüber, welche Eingaben zulässig oder nötig sind.
bietet auf Verlangen keine situationsspezifischen Erklärungen, die konkret weiterhelfen.	❑	❑	❑	❑	❑	❑	❑	bietet auf Verlangen situationsspezifischen Erklärungen, die konkret weiterhelfen.
bietet von sich aus keine situationsspezifischen Erklärungen, die konkret weiterhelfen.	❑	❑	❑	❑	❑	❑	❑	bietet von sich aus situationsspezifischen Erklärungen, die konkret weiterhelfen.

Steuerbarkeit

Können Sie als Benutzer die Art und Weise, wie Sie mit der Software arbeiten, beeinflussen?

Die Software	---	--	-	-/+	+	++	+++	Die Software
bietet keine Möglichkeit, die Arbeit an jedem Punkt zu unterbrechen und dort später ohne Verluste wieder weiterzumachen.	❑	❑	❑	❑	❑	❑	❑	bietet die Möglichkeit, die Arbeit an jedem Punkt zu unterbrechen und dort später ohne Verluste wieder weiterzumachen.
erzwingt eine unnötig starre Einhaltung von Bearbeitungsschritten.	❑	❑	❑	❑	❑	❑	❑	erzwingt keine unnötig starre Einhaltung von Bearbeitungsschritten.
ermöglicht keinen leichten Wechsel zwischen einzelnen Menüs oder Masken.	❑	❑	❑	❑	❑	❑	❑	ermöglicht einen leichten Wechsel zwischen einzelnen Menüs oder Masken.
ist so gestaltet, dass der Benutzer nicht beeinflussen kann, wie und welche Informationen am Bildschirm dargeboten werden.	❑	❑	❑	❑	❑	❑	❑	ist so gestaltet, dass der Benutzer beeinflussen kann, wie und welche Informationen am Bildschirm dargeboten werden.
erzwingt unnötige Unterbrechungen der Arbeit.	❑	❑	❑	❑	❑	❑	❑	erzwingt keine unnötigen Unterbrechungen der Arbeit.

Erwartungskonformität

Kommt die Software durch eine einheitliche und verständliche Gestaltung Ihren Erwartungen und Gewohnheiten entgegen?

Die Software	---	--	-	-/+	+	++	+++	Die Software
erschwert die Orientierung durch eine uneinheitliche Gestaltung.	❑	❑	❑	❑	❑	❑	❑	erleichtert die Orientierung, durch eine einheitliche Gestaltung.
lässt einen im Unklaren darüber, ob eine Eingabe erfolgreich war oder nicht.	❑	❑	❑	❑	❑	❑	❑	lässt einen nicht im Unklaren darüber, ob eine Eingabe erfolgreich war oder nicht.
informiert in unzureichendem Maße über das, was sie gerade macht.	❑	❑	❑	❑	❑	❑	❑	informiert in ausreichendem Maße über das, was sie gerade macht.
reagiert mit schwer vorhersehbaren Bearbeitungszeiten.	❑	❑	❑	❑	❑	❑	❑	reagiert mit gut vorhersehbaren Bearbeitungszeiten.
lässt sich nicht durchgehend nach einem einheitlichen Prinzip bedienen.	❑	❑	❑	❑	❑	❑	❑	lässt sich durchgehend nach einem einheitlichen Prinzip bedienen.

Fehlertoleranz

Bietet Ihnen die Software die Möglichkeit, trotz fehlerhafter Eingaben das beabsichtigte Arbeitsergebnis ohne oder mit geringem Korrekturaufwand zu erreichen?

Die Software	---	--	-	-/+	+	++	+++	Die Software
ist so gestaltet, dass kleine Fehler schwerwiegende Folgen haben können.	❑	❑	❑	❑	❑	❑	❑	ist so gestaltet, dass kleine Fehler keine schwerwiegenden Folgen haben können.
informiert zu spät über fehlerhafte Eingaben.	❑	❑	❑	❑	❑	❑	❑	informiert sofort über fehlerhafte Eingaben.
liefert schlecht verständliche Fehlermeldungen.	❑	❑	❑	❑	❑	❑	❑	liefert gut verständliche Fehlermeldungen.
erfordert bei Fehlern im Großen und Ganzen einen hohen Korrekturaufwand.	❑	❑	❑	❑	❑	❑	❑	erfordert bei Fehlern im Großen und Ganzen einen geringen Korrekturaufwand.
gibt keine konkreten Hinweise zur Fehlerbehebung.	❑	❑	❑	❑	❑	❑	❑	gibt konkrete Hinweise zur Fehlerbehebung.

Individualisierbarkeit

Können Sie als Benutzer die Software ohne großen Aufwand auf Ihre individuellen Bedürfnisse und Anforderungen anpassen?

Die Software	---	--	-	-/+	+	++	+++	Die Software
lässt sich von dem Benutzer schwer erweitern, wenn für ihn neue Aufgaben entstehen.	❑	❑	❑	❑	❑	❑	❑	lässt sich von dem Benutzer leicht erweitern, wenn für ihn neue Aufgaben entstehen.
lässt sich von dem Benutzer schlecht an seine persönliche, individuelle Art der Arbeitserledigung anpassen.	❑	❑	❑	❑	❑	❑	❑	lässt sich von dem Benutzer gut an seine persönliche, individuelle Art der Arbeitserledigung anpassen.
eignet sich für Anfänger und Experten nicht gleichermaßen, weil es der Benutzer nur schwer an seinen Kenntnisstand anpassen kann.	❑	❑	❑	❑	❑	❑	❑	eignet sich für Anfänger und Experten gleichermaßen, weil es der Benutzer gut an seinen Kenntnisstand anpassen kann.
lässt sich – im Rahmen seines Leistungsumfangs – von dem Benutzer schlecht für unterschiedliche Aufgaben passend einrichten.	❑	❑	❑	❑	❑	❑	❑	lässt sich – im Rahmen seines Leistungsumfangs – von dem Benutzer gut für unterschiedliche Aufgaben passend einrichten.
ist so gestaltet, dass der Benutzer die Bildschirmdarstellung schlecht an seine individuellen Bedürfnisse anpassen kann.	❑	❑	❑	❑	❑	❑	❑	ist so gestaltet, dass der Benutzer die Bildschirmdarstellung gut an seine individuellen Bedürfnisse anpassen kann.

Lernförderlichkeit

Ist die Software so gestaltet, dass Sie sich ohne großen Aufwand einarbeiten konnten und bietet es auch dann Unterstützung, wenn Sie neue Funktionen lernen möchten?

Die Software	---	--	-	-/+	+	++	+++	Die Software
erfordert viel Zeit zum Erlernen.	❑	❑	❑	❑	❑	❑	❑	erfordert wenig Zeit zum Erlernen.
ermutigt nicht dazu, auch neue Funktionen auszuprobieren.	❑	❑	❑	❑	❑	❑	❑	ermutigt dazu, auch neue Funktionen auszuprobieren.
erfordert, dass man sich viele Details merken muss.	❑	❑	❑	❑	❑	❑	❑	erfordert nicht, dass man sich viele Details merken muss.
ist so gestaltet, dass sich einmal Gelerntes schlecht einprägt.	❑	❑	❑	❑	❑	❑	❑	ist so gestaltet, dass sich einmal Gelerntes gut einprägt.
ist schlecht ohne fremde Hilfe oder Handbuch erlernbar.	❑	❑	❑	❑	❑	❑	❑	ist gut ohne fremde Hilfe oder Handbuch erlernbar.

Zum Schluss

Zum Schluss bitten wir Sie, noch folgende Fragen zu beantworten.

Wie oft haben Sie die Software schon benutzt?

heute das erste Mal ❑ öfter ❑ häufig ❑

Wie lange benutzen Sie durchschnittlich die Software? _____ Minuten

Wie lange arbeiten Sie überhaupt schon mit Computern? _____ Jahre

Wie viele Stunden arbeiten Sie pro Woche mit dem Computer? _____ Stunden

PRÜFEN SIE BITTE ABSCHLIEßEND, DASS SIE BEI JEDER AUSSAGE ETWAS ANGEKREUZT HABEN.

VIELEN DANK FÜR DIE BEANTWORTUNG DER FRAGEN !