

The constantly increasing complexity in automotive development requires data models, processes, and tools to address and handle this complexity by the documentation of information artifacts and their relationships to create traceability. Particularly, electrics/electronics (E/E) development including software and the corresponding information artifacts which are exchanged between engineering partners and have a high reciprocal dependency are crucial for traceability.

The developed data model addresses peculiarities of model-based systems engineering (MBSE) in alignment with product data management (PDM). The process model implements enhanced alignment during systems engineering through automatized synchronization of changes across IT systems and a consensus mechanism to identify discrepancies as early as possible.

As a technological solution, the Blockchain technology is implemented and serves as a product lifecycle management (PLM) backbone intermediating among multiple engineering partners' IT systems. Connecting the corresponding IT systems and tools within a company as well as providing interfaces to external engineering partners, the PLM Blockchain backbone fosters internal and external traceability.

A FRAMEWORK TO FOSTER TRACEABILITY OF E/E ARTIFACTS DURING AUTOMOTIVE DEVELOPMENT IN CONSIDERATION OF MODEL-BASED SYSTEMS ENGINEERING WITHIN DISTRIBUTED ENGINEERING COLLABORATION BY MEANS OF THE BLOCKCHAIN

Dominik T. Heber

A FRAMEWORK TO FOSTER TRACEABILITY OF E/E ARTIFACTS DURING AUTOMOTIVE DEVELOPMENT IN CONSIDERATION OF MODEL-BASED SYSTEMS ENGINEERING WITHIN DISTRIBUTED ENGINEERING COLLABORATION BY MEANS OF THE BLOCKCHAIN

Dominik T. Heber



**A FRAMEWORK TO FOSTER TRACEABILITY OF E/E
ARTIFACTS DURING AUTOMOTIVE DEVELOPMENT IN
CONSIDERATION OF MODEL-BASED SYSTEMS
ENGINEERING WITHIN DISTRIBUTED ENGINEERING
COLLABORATION BY MEANS OF THE BLOCKCHAIN**

Dominik Tobias Heber

A FRAMEWORK TO FOSTER TRACEABILITY OF E/E
ARTIFACTS DURING AUTOMOTIVE DEVELOPMENT
IN CONSIDERATION OF MODEL-BASED SYSTEMS
ENGINEERING WITHIN DISTRIBUTED ENGINEERING
COLLABORATION BY MEANS OF THE BLOCKCHAIN

DISSERTATION

to obtain
the degree of doctor at the University of Twente,
on the authority of the rector magnificus,
prof.dr.ir. A. Veldkamp,
on account of the decision of the Doctorate Board,
to be publicly defended
on Thursday the 8th of December 2022 at 16:45 hours

by

Dominik Tobias Heber

This dissertation has been approved by:

Supervisor:
prof.dr. M. W. Groll

Supervisor:
prof.dr. J. Henseler

Cover design: Own design. Car illustration: © Mercedes-Benz Group. Background: © andruxevich/Shutterstock.com

Printed by: Ipskamp Printing

ISBN: 978-90-365-5462-6

DOI: [10.3990/1.9789036554626](https://doi.org/10.3990/1.9789036554626)

© 2022 Dominik T. Heber, The Netherlands. All rights reserved. No parts of this thesis may be reproduced, stored in a retrieval system or transmitted in any form or by any means without permission of the author. Alle rechten voorbehouden. Niets uit deze uitgave mag worden vermenigvuldigd, in enige vorm of op enige wijze, zonder voorafgaande schriftelijke toestemming van de auteur.

GRADUATION COMMITTEE:

Chair/secretary	prof.dr.ir. H.F.J.M. Koopman (University of Twente)
Supervisor	prof.dr. M.W. Groll (University of Twente)
Supervisor	prof.dr. J. Henseler (University of Twente)
Members	dr.ir. G.M. Bonnema (University of Twente) Prof. Dr.-Ing. R. Dumitrescu (Paderborn University) prof.dr. I. Gibson (University of Twente) Prof. Dr. J. Oehmen (Technical University of Denmark)

"I remember clearly my parents' advice when, like all teenagers, I wondered what I would do when I grew up. They said, "Money is not important. It will never bring you happiness. [Strange advice to a future economist.] Use the brain God has given you, and be of service to others. That is what will give you satisfaction"."

Joseph E. STIGLITZ, 2010: p. 276

Recipient of "The Sveriges Riksbank Prize in Economic Sciences in Memory of Alfred Nobel 2001"

Acknowledgements

This thesis was written during my time as a PhD student at the University of Twente, *faculty of engineering technology, department of design, production & management*, and at Daimler AG, *CoC electrics/electronics & software, IT processes & methods*, and would not exist without the help and support of many people.

I would like to express my highest gratitude towards my supervisor Prof. Dr. Marco W. Groll. He ventured to accept me as a PhD student and I am deeply grateful for this opportunity. I highly appreciate the scientific freedom I was granted, fruitful discussions, with valuable input, helpful limitations in scope at the given time, and organizational support in this new phase for me. Without Prof. Dr. Groll's guidance, this thesis would not have been feasible.

Furthermore, I would like to thank my supervisor Prof. Dr. Jörg Henseler for his structured comprehension and scientific rigor which helped me to ameliorate this thesis as well as driving the official process. I also would like to extend my appreciation towards the members of the graduation committee dr.ir. Gerrit Maarten Bonnema, Prof. Dr.-Ing. Roman Dumitrescu, prof.dr. Ian Gibson, prof.dr.ir. H.F.J.M. Koopman, Prof. Dr. Josef Oehmen, for their valuable input, participation in the defense ceremony, and support.

I am exceptionally beholden to my internal supervisor at Daimler. Dr. Axel Lankenau, the team leader of *E/E documentation and testing*, who supported me scientifically, encouraged publications, served as enabler where necessary, gave me space, and helped me to focus where required. With his brilliance and yet nonchalance, respect, and motivation to others, Dr. Lankenau defines the embodiment of an excellent leader for me.

Dr. Frank Arbes, head of the department of *CoC electrics/electronics & software, IT processes & methods* at Daimler, I am very grateful not only for his monetary support during my PhD phase at his department, but also for his encouragement during rough patches. I am much obliged that he gave me the opportunity for this endeavor, to be able to learn and grow as part of this PhD. Prof. Alfred Katzenbach I am thankful for the introduction of me to Daimler and for shaping the beginning of my work. I appreciate Dr. Siegmund Haasis', CIO *R&D Mercedes-Benz Cars*, PhD-friendly environment and that I had the chance to combine research and application in his center.

I would like to thank the *Blockchain swarm* at the *Engineering IT Mercedes-Benz Cars* for their work for the prototype. Without their support, the prototype would not be what it is. I particularly thank Dr. Sebastian Handschuh, Jochen Heinkel, Harald Lichtenstein, Florian Michelbach, Melanie Paul, and Sebastian Sindermann for their awesome teamwork, with inspiring and deepening discussions which were – most of all – a lot of fun.

I would like to extend a great thank you to my colleagues at Daimler and especially my teammates in the team “Lankenau”. This includes my supervised students, Florian Michelbach and Furkan Karaoglu, I am grateful for their valuable work contributing to the overall topic, interesting discussions and fun despite the stress during their Master and Bachelor theses.

Dr. Barbara Heine I would like to thank especially for her strategic discussions, moral support, and helpful coaching.

Towards my family, relatives, and friends I want to express my special thankfulness. Constant support and understanding helped me to succeed. My brother Danilo always had my back during the last decades and particularly during my PhD time. Without his assistance in private life, I could not have undertaken this work. Infinite thanks to my parents for my good characteristic traits. I wish you still could live to see this.

Abstract

The constantly increasing complexity in automotive development requires data models, processes, and tools to address and handle this complexity by the documentation of information artifacts and their relationships to create traceability. This challenge exists both within the OEM's development organization as well as for each supplier. Particularly, electrics/electronics (E/E) development including software and the corresponding information artifacts which are exchanged between engineering partners and have a high reciprocal dependency are crucial for traceability.

Derived from these challenges, this dissertation has the objectives of the conceptualization and prototypical implementation of a solution framework that addresses internal traceability, i.e., within a company's IT systems, and external traceability, i.e., among multiple engineering partners. For this purpose, three distinct enablers for a framework for traceability are identified: i) data model, ii) process model, iii) technology.

Given the current state of science and technology, the enablers are assessed by means of the derived requirements. Therefore, the proposed solution framework also composes these three enablers.

The developed data model addresses peculiarities of model-based systems engineering (MBSE) in alignment with product data management (PDM) for early automotive E/E development and thereby fosters predominantly internal traceability but also external. Moreover, the data model includes universal identifiers for the promotion of external traceability and proposes a data integration mechanism for the exchange and synchronization of relevant data.

The process model implements enhanced alignment during systems engineering through automatized synchronization of changes across IT systems and a consensus mechanism to identify discrepancies as early as possible. This mainly addresses external traceability among engineering partners.

As a technological solution, the Blockchain technology is implemented and serves as a product lifecycle management (PLM) backbone which intermediates between multiple engineering partners' IT systems. Connecting the corresponding IT systems and tools

within a company as well as providing interfaces to external engineering partners, the PLM Blockchain backbone fosters internal and external traceability.

As this technological approach using the Blockchain technology for engineering IT in automotive engineering collaborations is completely new, the evaluation of the solution framework was conducted with an existing development use case as well as a potential future scenario. Conclusively, the elaborated solution framework addresses the research objectives adequately. Limitations are discussed and serve as basis for prospective work.

Title and summary in Dutch

EEN RAAMWERK OM DE TRACEERBAARHEID VAN E/E-ARTEFACTEN TIJDENS DE ONTWIKKELING VAN AUTO'S TE BEVORDEREN IN HET LICHT VAN MODELGEBASEERDE SYSTEMEN ENGINEERING BINNEN GEDISTRIBUEERDE ENGINEERING-SAMENWERKING DOOR MIDDEL VAN DE BLOCKCHAIN

De complexiteit in de ontwikkeling van auto's neemt voortdurend toe, en dat vereist gegevensmodellen, processen en hulpmiddelen om deze complexiteit aan te pakken en te verwerken. Dit gebeurt door het documenteren van informatieartefacten en hun relaties om traceerbaarheid te creëren. Deze uitdaging bestaat zowel binnen de ontwikkelingsorganisatie van de OEM als voor elke leverancier. Vooral de ontwikkeling van elektronica/elektronica (E/E) is cruciaal voor de traceerbaarheid. Onder deze E/E-artefacten vallen software en de bijbehorende informatieartefacten die worden uitgewisseld tussen engineeringpartners en een hoge wederzijdse afhankelijkheid hebben.

Deze uitdagingen leiden tot de doelstellingen van dit proefschrift: de conceptualisering en prototypische implementatie van een oplossingsraamwerk dat werkt aan interne traceerbaarheid (binnen de IT-systemen van een bedrijf) en externe traceerbaarheid (onder meerdere engineeringpartners). Voor deze doelen worden drie verschillende enablers geïdentificeerd voor een raamwerk voor traceerbaarheid: i) gegevensmodel, II) procesmodel, III) technologie.

De enablers worden beoordeeld aan de hand van de eisen die zijn afgeleid uit de huidige stand van wetenschap en technologie. Het voorgestelde oplossingskader vormt daarom ook deze drie mogelijkheden.

Het ontwikkelde datamodel richt zich op de specifieke kenmerken van model-based systems engineering (MBSE) in overeenstemming met product data management (PDM), voor de vroege ontwikkeling van E/E in de automobiellindustrie. Het bevordert daardoor voornamelijk interne traceerbaarheid, maar ook externe. Bovendien bevat het gegevensmodel universele identificatoren voor de bevordering van externe traceerbaarheid. Daarnaast stelt het model een mechanisme voor, voor data-integratie voor de uitwisseling en synchronisatie van relevante data.

Het procesmodel implementeert verbeterde afstemming tijdens systeemontwikkeling. Dit gebeurt door geautomatiseerde synchronisatie van veranderingen in IT-systemen en een consensusmechanisme om afwijkingen zo vroeg mogelijk te identificeren. Dit betreft vooral de externe traceerbaarheid onder engineeringpartners.

Als technologische oplossing wordt de blockchain-technologie geïmplementeerd. Deze fungeert als een PLM-ruggengraat (Product Lifecycle Management) die de IT-systemen van meerdere engineeringpartners bemiddelt. De PLM blockchain backbone bevordert de interne en externe traceerbaarheid door de overeenkomstige IT-systemen en -tools binnen een bedrijf te verbinden en interfaces te bieden aan externe engineeringpartners.

Deze technologische aanpak, waarbij de blockchain-technologie wordt gebruikt voor engineering-IT in samenwerking met automotive engineering, is volledig nieuw. Daarom werd de evaluatie van het oplossingskader uitgevoerd met een bestaande ontwikkelingsgebruiksscenario en een potentieel toekomstig scenario. Het uitgewerkte oplossingskader richt zich op afdoende wijze op de onderzoeksdoelstellingen. Beperkingen worden besproken en dienen als basis voor toekomstig werk.

Table of Contents

Acknowledgements	i
Abstract	iii
Title and summary in Dutch	v
Table of Contents	vii
Abbreviations	xi
List of Figures	xiv
List of Tables	xviii
List of source codes	xix
1 Introduction	1
1.1 Initial situation.....	1
1.2 Problem statement and delineation	7
1.2.1 <i>Traceability of information artifacts to address increasing complexity in automotive E/E architecture</i>	7
1.2.2 <i>Current deficiency in research</i>	9
1.3 Objectives of this thesis.....	14
1.4 Design research, research methodology, and structure of this thesis	16
2 Current state of science and technology, definitions, and general terms ..	22
2.1 Traceability	23
2.1.1 <i>Definitions, norms, and standards</i>	23
2.1.2 <i>Methods</i>	25
2.1.3 <i>Traceability in the context of automotive development</i>	27
2.2 Product development process	28
2.2.1 <i>Definitions, norms, and standards</i>	28
2.2.2 <i>Processes and methods for product development</i>	31
2.2.3 <i>Idiosyncrasies of automotive electric/electronic product development</i>	39
2.2.4 <i>Traceability in the context of product development</i>	40

2.3	Product data management and product lifecycle management.....	41
2.3.1	<i>Definitions, norms, and standards</i>	41
2.3.2	<i>Configuration management</i>	45
2.3.3	<i>Traceability in the context of PDM/PLM</i>	52
2.4	Model-based systems engineering.....	54
2.4.1	<i>Definitions, norms, and standards</i>	54
2.4.2	<i>Methods and languages</i>	56
2.4.3	<i>Traceability in the context of MBSE</i>	60
2.5	Automotive electrics and electronics including software.....	65
2.5.1	<i>Definitions, norms, and standards</i>	65
2.5.2	<i>Architecture, communication, hardware, and software</i>	65
2.5.3	<i>Traceability in the context of automotive E/E and software</i>	68
2.6	Distributed engineering collaboration	69
2.6.1	<i>Definitions, norms, and standards</i>	69
2.6.2	<i>Phenotypes</i>	71
2.6.3	<i>Traceability in the context of engineering collaboration</i>	73
2.7	Data base solutions.....	74
2.7.1	<i>Definitions, norms, and standards</i>	75
2.7.2	<i>Central data bases</i>	75
2.7.3	<i>Decentral data bases</i>	76
2.7.4	<i>Traceability in the context of data base solutions</i>	83
2.8	Ontologies	84
2.8.1	<i>Definitions, norms, and standards</i>	84
2.8.2	<i>Traceability in the context of ontologies</i>	87
2.9	Conclusion.....	88
3	Requirements for a solution framework and evaluation of current state....	90
3.1	Evaluation method.....	90
3.2	Requirements for internal traceability	92
3.3	Requirements for external traceability	93
3.4	Classification of the current state of science and technology	95
3.5	Concluding evaluation of current state of science and technology	99

4	Synthesis of a solution framework	102
4.1	Definition of a data model	104
4.1.1	<i>Definition of the relevant information artifacts</i>	<i>105</i>
4.1.2	<i>Relevant metadata for a linked data model</i>	<i>125</i>
4.2	Definition of a process model	133
4.2.1	<i>Alignment of the SPES method and PDM</i>	<i>135</i>
4.2.2	<i>New product creation process</i>	<i>139</i>
4.2.3	<i>Configuration and variant creation process</i>	<i>142</i>
4.2.4	<i>Version creation and change management process</i>	<i>144</i>
4.2.5	<i>Inactivation process</i>	<i>146</i>
4.2.6	<i>Processes for multiple engineering collaboration partners</i>	<i>148</i>
4.3	Definition of a technology	151
4.3.1	<i>Fundamental IT architecture of the IT solution</i>	<i>152</i>
4.3.2	<i>Consensus mechanism</i>	<i>160</i>
4.4	Solution framework and its satisfaction of requirements	163
5	Prototypical implementation	167
5.1	Goal and scope of the prototypical implementation	167
5.2	Implementation of a prototypical IT framework	168
5.2.1	<i>The structure of the prototype</i>	<i>172</i>
5.2.2	<i>GUI</i>	<i>178</i>
5.2.3	<i>Roles and permissions</i>	<i>184</i>
5.3	Implementation of process model	186
5.4	Implementation of data model	190
5.5	Alignment with legacy IT architecture	198
6	Evaluation of the solution framework	200
6.1	Evaluation approach	200
6.2	Use case 1: Door control module	202
6.2.1	<i>Technical problem statement</i>	<i>202</i>
6.2.2	<i>Use case description</i>	<i>204</i>
6.2.3	<i>Exemplary implementation of use case 1</i>	<i>204</i>
6.3	Use case 2: Centralized, server-oriented E/E architecture	207

6.3.1	<i>Technical problem statement</i>	207
6.3.2	<i>Use case description</i>	209
6.3.3	<i>Exemplary implementation of use case 2</i>	210
6.4	Evaluation of research objectives.....	212
6.5	Discussion.....	214
7	Summary and outlook	216
7.1	Summary	216
7.2	Ramifications and outlook for automotive engineering IT	218
7.3	Ramifications and outlook for further industries.....	221
7.4	Ramifications and outlook for the wider social context	222
8	Appendix	224
9	References	225

Abbreviations

ALM	Application lifecycle management
API	Application programming interface
ASPICE	Automotive software process improvement and capability determination
AUTOSAR	Automotive open system architecture
bdd	Block definition diagram
BOM	Bill of material
CAD	Computer-aided design
CAE	Computer-aided engineering
CAN	Controller area network
CASE	Computer-aided software engineering
CAX	Computer-aided X
CI	Configuration items
CM	Configuration management
CRM	Customer relationship management
CRUD	Create, read, update, delete
DAO	Decentralized autonomous organizations
DB	Data base
DBMS	Data base management system
DCM	Door control module
DLT	Distributed ledger technologies
DSM	Dependency structure matrices
E/E	Electrics/electronics
E-BOM	Engineering BOM

E-CAD	Electrics/electronics computer-aided design
ECU	Electronic control unit
EEPROM	Electrically erasable programmable read-only memory
ERP	Enterprise resource planning
FMI	Functional mock-up interface
FMU	Functional mock-up unit
GUI	Graphical user interface
HTTP	Hypertext transfer protocol
HW	Hardware
I/O	Input/output
ID	Identification, identifier
IP	Intellectual property
IT	Information technology
JT	Jupiter tessellation
LIN	Local interconnected network
M-BOM	Manufacturing BOM
MBSE	Model-based systems engineering
M-CAD	Mechanics computer-aided design
MOST	Media oriented systems transport
NCD	Network communication description
NoSQL	Not only structured query language
OEM	Original equipment manufacturer
OS	Operating system
OSLC	Open services for lifecycle collaboration
OVM	Orthogonal variability model
OWL	Web ontology language

P2P	Peer-to-peer
PDM	Product data management
PLM	Product lifecycle management
PPS	Production planning system
R&D	Research and development
RDF	Resource description framework
RDFS	RDF-scheme
REST	Representational state transfer
RFLP	Requirements, functional, logical, physical views
SCM	Supply chain management
SiL/HiL	Software/hardware in the loop
SPES	Software platform embedded systems
SPICE	Software process improvement and capability determination
STEP AP 242	Standard for the exchange of product data application protocol 242
SW	Software
SysML	Systems modelling language
TDM	Team data management
UML	Unified modelling language
URI	Uniform resource identifiers
W3C	World wide web consortium
XMI	Extensible markup language metadata interchange
XML	Extensible markup language

List of Figures

Figure 1-1: Comparison of production quantity, variability of possible combinations, and number of parts used for a final product for different industries (cf. SIEMENS PLM, n.a. according to KATZENBACH, 2015b: p. 47).....	3
Figure 1-2: Comparison of products' complexity, their variability, and the quantity of production units for different industries (cf. REUSCHER, n.a. according to KATZENBACH, 2015b: p. 48).....	3
Figure 1-3: Increasing complexity in a luxury automobile's E/E architecture represented by the growth of the total number of ECUs and communication busses (LANKENAU and HEBER, 2017: p. 4).....	6
Figure 1-4: The problem space of automotive E/E development in a heterogeneous IT tool and system landscape (in alignment to HEBER et al., 2018: p. 8; HEBER and GROLL, 2018a: p. 281).....	13
Figure 1-5: Generic system development process in automotive E/E development with multiple engineering collaboration partners. The objectives are marked yellow as potentials for improvement (cf. SCHÄUFFELE and ZURAWKA, 2016: p. 199).....	15
Figure 1-6: Framework of the design research methodology (DRM) including stages, basic means, and deliverables (in alignment to BLESSING and CHAKRABARTI, 2009: p. 39).	18
Figure 1-7: The problem space of this work (blue) and the enablers data model, process model, and technology.....	20
Figure 1-8: Structure and approach of the thesis.....	21
Figure 2-1: Association between source artifact and target artifact by means of a trace link (in alignment to GOTEL et al., 2012: p. 6).....	24
Figure 2-2: Different methods for traceability representation: (a) Traceability matrix, (b) cross referencing, (c) graph-based representation (in alignment to WINKLER and PILGRIM, 2009: p. 542).....	26
Figure 2-3: The main phases of a product lifecycle in differentiation to product development and to product creation (in alignment to VEREIN DEUTSCHER INGENIEURE, 2014: pp. 5–6; MÜLLER et al., 2012: p. 173).	29
Figure 2-4: Evolution of the product development process over time (in alignment to EIGNER and STELZER, 2009: p. 19; STEPHAN, 2013: p. 11).	30
Figure 2-5: General process model for product development and design (in alignment to PONN and LINDEMANN, 2011: p. 18; VEREIN DEUTSCHER INGENIEURE, 1993: p. 9; STEPHAN, 2013: p. 26; EIGNER, 2014d: p. 16).	33
Figure 2-6: Phase model of software development (in alignment to BOEHM, 1979: p. 4; EIGNER et al., 2012a: p. 162; EIGNER, 2014d: p. 33).....	35
Figure 2-7: Alteration of the term “mechatronics” (in alignment to EIGNER et al., 2012a: p. 34; STEPHAN, 2013: p. 17; GROLL and HEBER, 2016: p. 291; EIGNER, 2014d: p. 43; BERTSCHE et al., 2009: p. 3).	36
Figure 2-8: Basic structure of a mechatronic system (in alignment to VEREIN DEUTSCHER INGENIEURE, 2004b: p. 14; PONN and LINDEMANN, 2011: p. 12; STEPHAN, 2013: p. 18).	37
Figure 2-9: The V-model of mechatronic system development (in alignment to VEREIN DEUTSCHER INGENIEURE, 2004b: p. 29; BENDER, 2005: p. 45; GROLL and HEBER, 2016: p. 291; EIGNER et al., 2012b: p. 1670; ZAFIROV, 2014: p. 87).	38

Figure 2-10: Product lifecycle phases (in alignment to EIGNER and STELZER, 2009: pp. 16, 20, 28).	42
Figure 2-11: Location of PDM and PLM with respect to the product lifecycle (in alignment to EIGNER and STELZER, 2009: p. 37; EIGNER, 2014b: p. 270).	43
Figure 2-12: Typical four-layered PLM architecture with a central PLM backbone (in alignment to EIGNER and STELZER, 2009: p. 43; EIGNER, 2014b: p. 280).	44
Figure 2-13: The five major aspects of configuration management (in alignment to GRANDE, 2013: p. 16; INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2015c: p. 40; KIRSCH et al., 2017a: p. 157).	47
Figure 2-14: Effectivity in configuration and change management (in alignment to EIGNER and STELZER, 2009: p. 118; EIGNER, 2014c: p. 263).	48
Figure 2-15: Different variations of version control in software development (in alignment to KEYDEL and MEDING, 2008: pp. 230–231; CHACON and STRAUB, 2014: pp. 1–4; GIFT and SHAND, 2009: pp. 2–3; GRANDE, 2013: p. 106).	50
Figure 2-16: Connection between configuration, coexisting variants, and sequential versions.	52
Figure 2-17: System breakdown structure (in alignment to IEEE COMPUTER SOCIETY, 2007: pp. 4, 18).	55
Figure 2-18: Model-based systems engineering with a central system model (in alignment to ZAFIROV, 2014: p. 82; FRIEDENTHAL et al., 2012: p. 18).	56
Figure 2-19: The three modules of MBSE with the system model as its central artifact (in alignment to ALT, 2012: p. 9; EIGNER et al., 2018: p. 382; EIGNER et al., 2016a: p. 167).	56
Figure 2-20: Overview of SysML (lite) language features (in alignment to FRIEDENTHAL et al., 2012: p. 33).	57
Figure 2-21: Example of three different product descriptions in hierarchical representation in different phases of the lifecycle and the aim to foster traceability by connection of information artifacts (in alignment to MÜLLER and KIRSCH, 2017: p. 179).	61
Figure 2-22: Transformation of supplier structures across time (in alignment to FELDHOUSEN and GROTE, 2013: p. 7; EIGNER and STELZER, 2009: p. 15; STEPHAN, 2013: p. 71; KATZENBACH, 2015a: p. 626).	72
Figure 2-23: Generic structure of blocks in the Blockchain (in alignment to NARAYANAN et al., 2016: p. 33; BASHIR, 2018: p. 20).	80
Figure 2-24: Semantic web layer cake (in alignment to SAKR et al., 2018: p. 4).	86
Figure 4-1: Approach for the definition of a data model.	104
Figure 4-2: System concepts: i) functional, ii) structural, iii) hierarchical (in alignment to ROPOHL, 2009: p. 76).	106
Figure 4-3: SysML taxonomy (in alignment to OMG, 2015: p. 187; FRIEDENTHAL et al., 2012: p. 30).	107
Figure 4-4: Reference model for an automotive E/E system depicted as a package diagram in SysML. Gray packages are not in scope of this work.	109
Figure 4-5: Generic structure of the automotive E/E system model.	110
Figure 4-6: Generic structure of the ECU.	111
Figure 4-7: Generic structure of the E/E architecture.	112
Figure 4-8: Generic structure of the communication bus.	113
Figure 4-9: Generic structure of the function.	114
Figure 4-10: Generic structure of the configuration.	116

Figure 4-11: Connection between functional and logical viewpoints (in alignment to POHL et al., 2012: p. 45).	118
Figure 4-12: Generic structure of the viewpoints.	119
Figure 4-13: Generic structure of the abstraction layers.	120
Figure 4-14: Generic relationships between different viewpoints for one system element and between different abstraction layers (in alignment to POHL et al., 2012: pp. 38, 45).	120
Figure 4-15: Generic structure of the I/O definition.	122
Figure 4-16: Generic structure element of potential variants of an ECU.	123
Figure 4-17: Generic structure of IT systems and tools as well as their relevant structural aggregation models or elements.	125
Figure 4-18: Generic structure of an RDF triple as a graph.	129
Figure 4-19: Schematic data integration across different domains and between engineering partners.	133
Figure 4-20: The SPES development process with its main information artifacts (in alignment to POHL et al., 2012: pp. 51–105).	135
Figure 4-21: Alignment of a generic engineering process with the SPES viewpoints (in alignment to POHL et al., 2012: p. 153).	137
Figure 4-22: Alignment of the MBSE and PDM processes with their main information artifacts.	138
Figure 4-23: Generic process for the creation of a new product.	141
Figure 4-24: Generic process for the creation of a new configuration and variant.	143
Figure 4-25: Generic process for the creation of a new version as part of the change management.	145
Figure 4-26: Generic process for the creation of new versions as part of the change management.	146
Figure 4-27: Generic process for the inactivation of a version.	148
Figure 4-28: Generic process for the creation of a new configuration and variant with three engineering partners.	150
Figure 4-29: Generic IT architecture including a PLM Blockchain backbone simplified for one OEM and one supplier.	153
Figure 4-30: Generic IT architecture including the transfer of relevant information artifacts of the E/E development process according to the SPES methodology.	154
Figure 4-31: Generic IT architecture for multiple engineering partners with their own PLM Blockchain backbone.	155
Figure 4-32: Integration into generic IT architecture with multiple channels implemented by different suppliers.	159
Figure 4-33: Consensus mechanism: Initial creation and distribution of data by the OEM and approval by the engineering partners.	161
Figure 4-34: Consensus mechanism: Creation and distribution of data by a supplier and approval by the engineering partners.	162
Figure 4-35: Consensus mechanism: Creation and distribution of data by a supplier and rejection by one engineering partner.	163
Figure 4-36: Solution framework for traceability of E/E artifacts during automotive development in consideration of MBSE within distributed engineering collaboration by means of the Blockchain and the satisfaction of requirements.	166
Figure 5-1: Network deployment of the prototypical implementation of the Blockchain network with multiple suppliers.	171
Figure 5-2: The structure of the prototype.	172

Figure 5-3: Top level structure of the prototype.	172
Figure 5-4: Runnables including different, organization-specific settings.	173
Figure 5-5: Runnables for the administrator.	173
Figure 5-6: Blockchain network definition files.	174
Figure 5-7: Blockchain network explorer files.	174
Figure 5-8: Blockchain network setup files.	175
Figure 5-9: Blockchain network setup binaries.	175
Figure 5-10: Local database and Fabric binaries.	176
Figure 5-11: Base files for the organizations within the Blockchain network.	176
Figure 5-12: Cryptographic and configuration files for each organization.	176
Figure 5-13: Structure of cryptographic files for peers of each organization.	177
Figure 5-14: Structure of cryptographic files for one peer.	177
Figure 5-15: Structure for multi-channel setup including chain code.	178
Figure 5-16: Channel artifacts.	178
Figure 5-17: Supplier1 creates an artifact.	179
Figure 5-18: List of artifacts of Supplier1 with pending voting answers.	179
Figure 5-19: List of artifacts of OEM with pending voting actions.	180
Figure 5-20: Voting by OEM.	180
Figure 5-21: List of artifacts of OEM with different release status.	181
Figure 5-22: Details of rejection by the OEM of supplier's artifact creation.	181
Figure 5-23: List of artifacts of Supplier1 after voting.	182
Figure 5-24: Artifact details including transaction history.	182
Figure 5-25: Voting retry.	183
Figure 5-26: Responsive design of web interface.	183
Figure 5-27: Process model of the Hyperledger Fabric framework.	187
Figure 5-28: Generic ontology for an ECU and its associated information artifacts (blue) with additional MBSE views (green).	190
Figure 5-29: Generic positioning of the prototypical implementation of a PLM Blockchain backbone within a legacy IT architecture. For abbreviations, please refer to the description of Figure 2-12.	199
Figure 6-1: Door control module (in alignment to REIF, 2014: p. 246).	203
Figure 6-2: Development process for door control module.	206
Figure 6-3: Technology aspects for door control module.	207
Figure 6-4: Centralized ECU (in alignment to CONTINENTAL AG, 2021).	208
Figure 6-5: Use case centralized ECU.	209
Figure 6-6: Development process and technology aspects for centralized ECU.	211
Figure 7-1: Automobile as distinct node in the Blockchain network.	221
Figure 8-1: Digital Twin for the automotive lifecycle (HEBER et al., 2018).	224

List of Tables

Table 2-1: Three alternatives of data model alignment between MBSE and PDM/PLM (in alignment to MÜLLER and KIRSCH, 2017: p. 179; HEBER and GROLL, 2018b: p. 127).	62
Table 3-1: Assessment of objectives addressed by enablers.	91
Table 3-2: Generic depiction of the evaluation method (in alignment to ESTEFAN, 2008: p. 10; KÖNIGS, 2013: p. 52; GILZ, 2014: 51).	91
Table 3-3: Comparison of different MBSE methods (cf. Chapter 2.4.2 for the authors of the different methods) (own evaluation in alignment to HEBER and GROLL, 2018b: p. 127, 2018a: p. 284).	96
Table 3-4: Evaluation of different types of data bases with respect to peculiarities in collaborations (own evaluation in alignment to STIEFEL, 2011: pp. 57–62; HECKMANN et al., 2006: pp. 7–17).	99
Table 3-5: Evaluation of the current state of science and technology according to the defined requirements in alignment to the research objectives.	101
Table 6-1: Evaluation scheme for use cases.	202
Table 6-2: Evaluation of addressed objectives and requirements by use cases.	214

List of source codes

Source Code 4-1: Generic structure of the URI (in alignment to HITZLER, 2008: 27).	126
Source Code 4-2: Generic structure of the hardware variant and version scheme....	127
Source Code 4-3: Generic structure of the software variant and version scheme.	127
Source Code 4-4: Generic structure of the ECU variant and version scheme.	128
Source Code 4-5: Generic structure of an RDF triple using Turtle syntax.	129
Source Code 4-6: Generic structure of the OSLC namespace, domains, resources, and properties.	130
Source Code 4-7: Example of the alignment of product development data with the OSLC framework (in alignment to IBM KNOWLEDGE CENTER, 2020).....	132
Source Code 5-1: Recording of votes.....	184
Source Code 5-2: Authorized roles.....	185
Source Code 5-3: Initialization of configuration state.....	186
Source Code 5-4: Permitted voters.	186
Source Code 5-5: Basic operations.	188
Source Code 5-6: Create artifact operation.	188
Source Code 5-7: Artifact states.....	189
Source Code 5-8: Vote status "in progress".	189
Source Code 5-9: Vote status "accepted".....	189
Source Code 5-10: Vote status "not accepted".....	189
Source Code 5-11: Generic information artifact structure.....	191
Source Code 5-12: Generic metamodel structure.	192
Source Code 5-13: Generic ontology description.	193
Source Code 5-14: Generic object properties of the ontology.	194
Source Code 5-15: Generic data properties of the ontology.....	195
Source Code 5-16: Generic classes of the ontology.....	196
Source Code 5-17: Generic object properties of the information artifacts.....	197
Source Code 5-18: Generic data properties of the information artifacts.....	198
Source Code 6-1: Data model for door control module.....	205
Source Code 6-2: Data model for centralized ECU.	210

1 Introduction

1.1 Initial situation

INDUSTRIES WITH COMPLEX PRODUCTS

GERICKE et al. (2013) show that projects of complex products¹, such as aerospace, motor vehicles and software, have a higher coverage of all related lifecycle phases and their processes within them, such as all related quality processes needed to meet regulatory specifications etc. This can be argued by the tremendous risk a company engages in with the development of a complex product and the complexity of products themselves. On the other hand, companies building less complex products can skip certain aspects of processes, for instance when a few disciplines or organizational entities are involved during development (GERICKE et al., 2013: 7). Hence, for complex products most lifecycle phases and processes are relevant and therefore the seamless integration of those is necessary for efficiency and quality.

Particularly in aerospace, products have a lifecycle of over 50 years whereas a spaceship's or plane's applications only are used for about three years. This implies a constant adaption of applications, re-design, and re-development in collaboration with many disciplines, departments, suppliers, and external engineering partners (SINDERMAN, 2014: pp. 345–346; LOTAR INTERNATIONAL: LONG TERM ARCHIVING AND RETRIEVAL). In the automotive industry, lifecycles of products are within circa five to seven years, which is a lot shorter. However, automobiles are considered more and more to be consumer goods that have to adapt quickly to technological changes demanded by the customers. For that purpose, automobiles undergo frequent minor or major alterations to meet the market demand and to include improvements or technological novelties.

¹ Products or systems are called complex or synonymously complicated. However, those terms have to be distinguished. HABERFELLNER (2012) defines complexity of systems according to the amount, variety, or size of elements on one axis and the dynamic and volatility of elements' interfaces on the other. A simple system consists of little elements that have little dynamic or little intense relations, i.e., parts have little interfaces to other parts and these interfaces are mostly constant. Systems with many elements are called massive networked complicated systems. Systems with a high dynamic or volatility are called dynamic complicated systems. Complex systems have a high dynamic or volatility of elements' interfaces as well as a high amount, variety, or size of elements. Complex systems cannot be described, understood, or modeled entirely, whereas with complicated systems this might be, at least partially, feasible (HABERFELLNER, 2012: pp. 40–41; SCHUH, 2005: pp. 5–7). For more information about complexity in products and systems, please refer to ULRICH and PROBST (1988); LINDEMANN et al. (2009); LINDEMANN (2009); SCHUH (2005); DAENZER and HUBER (2002).

In comparison to other industries which produce complex products, such as truck or machineries, commercial aircrafts, or ships, the automotive industry produces the most units per day. In contrast, automobiles are not as big as most products of the other mentioned industries and hence the number of parts used is significantly smaller than for a ship, for example. With respect to variability, i.e., the possible combinations of different parts for one final product, for instance another engine or color for the same automobile model, the automotive industry has succeeded in standardization of platforms and common parts. Consequently, the variability in the automotive industry is not as high as for instance in the commercial aircraft industry (cf. Figure 1-1) (KATZENBACH, 2015b: p. 47). Yet, customers demand a highly configurable product, especially in the luxurious automotive segment. Thus, automotive manufacturers offer a high variability to satisfy this demand.

A relatively complex industry, such as aerospace with its small production quantities, does not have as much variability in its products as the automotive industry. Also, the consumer electronics industry does not offer such a high variability of one product as the automotive industry, albeit a high quantity of products is produced. Consumer electronic products often are less complex than automobiles or spaceships because they are made for mass production, therefore mostly have limited fields of application and shorter lifecycles. The automotive industry is an intermediary regarding complex products and high production quantities in comparison to aerospace and consumer electronics. However, in variability of products the automotive industry excels compared to these other industries (cf. Figure 1-2) (in alignment to KATZENBACH, 2015b: p. 48).

Due to a relatively intermediate complexity, based upon the amount and type of interfaces as well as the art and number of elements (cf. Footnote 1), but with a relatively high production output and also a significant variability, the automotive industry can be considered as one of the most challenging industries. This is in regards to the development and production of its products. Particularly, the congruency of complexity, quantity, and variability of automobiles yields a special foundation for the assessment of the accompanying development processes in a company and between different companies contributing to an automobile. The increasing importance of electric/electronics in automobiles and resulting challenges in the development of complex systems further accentuate this stance, as it will be discussed in the next section.

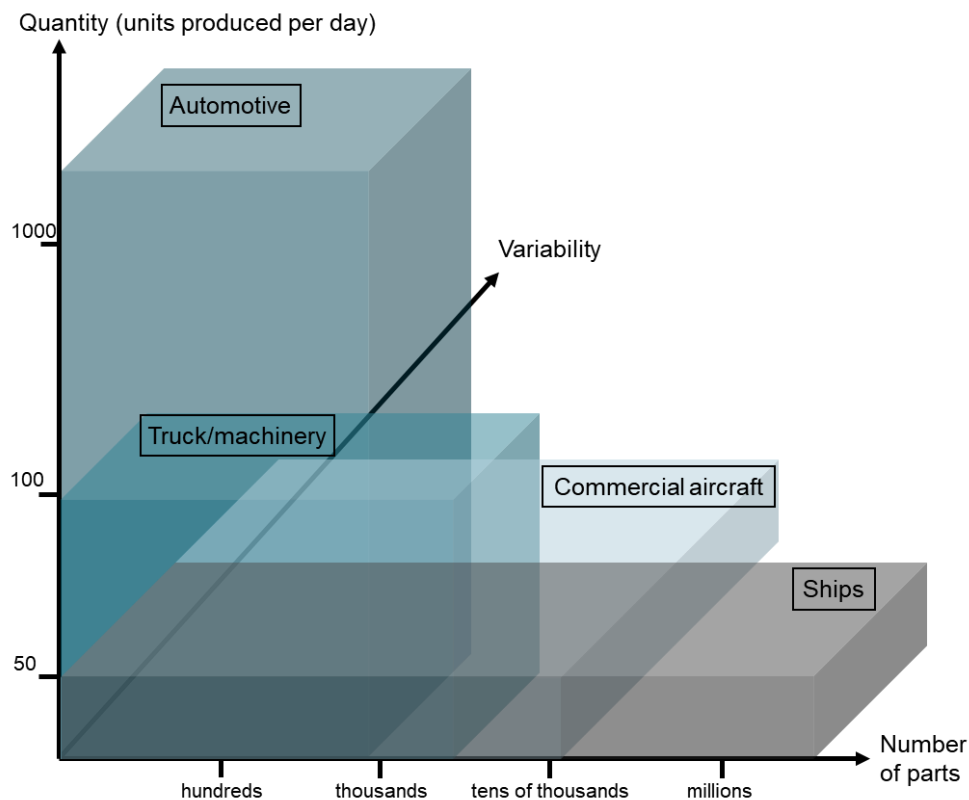


Figure 1-1: Comparison of production quantity, variability of possible combinations, and number of parts used for a final product for different industries (cf. SIEMENS PLM, n.a. according to KATZENBACH, 2015b: p. 47).

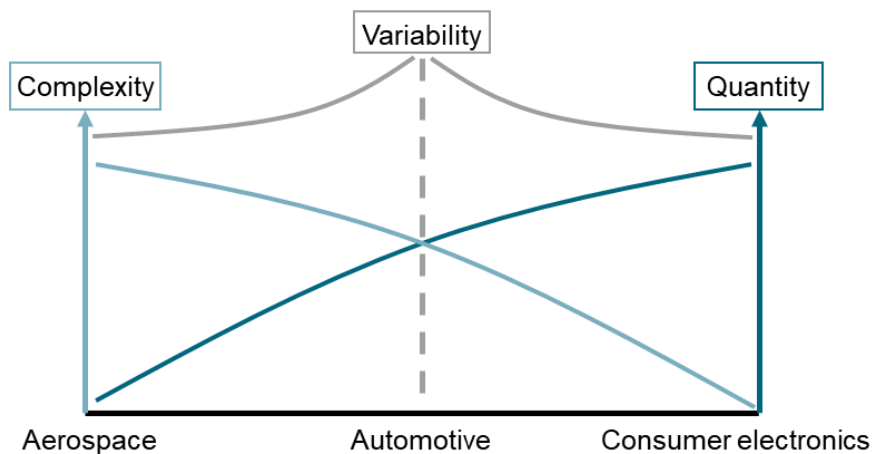


Figure 1-2: Comparison of products' complexity, their variability, and the quantity of production units for different industries (cf. REUSCHER, n.a. according to KATZENBACH, 2015b: p. 48).

VICISSITUDES OF THE AUTOMOTIVE INDUSTRY

In the last decades, a change from seller to a buyer market and resulting individualization of products shaped product variety in order to satisfy the different demands of customers. This is also true for the automotive market where automotive

manufacturers court customers with many possible combinations and hence individualization (BURMANN and KOTHES, 2014: p. 3). This results in a huge amount of variants (BURMANN and KOTHES, 2014: pp. 8–10; KATZENBACH, 2015a: p. 609). Of a total of 1.1 million Mercedes-Benz A-class models produced within two years, only two automobiles have been identical (SCHLOTT, 2005: p. 38). Today, even without color combinations, 25,000 possible variants of one automobile model are common and for a door panel there exist 18,000 possibilities (EHRENSPIEL and MEERKAMM, 2017: p. 864; SCHLOTT, 2005: p. 38). An increase in parts variety and product variety results in a complexity of products, which has to be handled organizationally as well as technologically.

Currently, the automotive industry faces profound vicissitudes based upon topics such as connectivity, autonomous driving, car sharing, and electric drive systems (DAIMLER AG, 2017: p. 24). Expectations by customers to connect their smartphone to their car and also control it partially, induce this change. Additionally, especially in major cities, a trend towards car sharing instead of purchasing an own car occurs. This is again induced by changing customer demands. This can be considered as a further pull factor in the view of automotive manufacturers. Opposing, there exist push factors. For instance, most major original equipment manufacturers (OEMs) pursue a strategy towards autonomous driving using assistance systems. This technology is becoming more mature and enables new business segments, e.g., robot taxis. Implementation of (partial) electric driving in vehicles is based on, one the one hand, a more efficient battery and power train technology. On the other hand, regulatory requirements foster electric vehicles. Amalgamation of push and pull factors yield new technological solutions in an automobile in the realm of electrics/electronics (E/E), i.e., actuators, sensors, electronic control units (ECUs), and software, in order to address these new requirements. Due to new functionalities more likely being implemented software-based and this software being more easily altered, the volatility as well as overall variance of the current state of construction of a vehicle can increase tremendously (TRIPPNER et al., 2015: p. 557; BEIHOFF et al., 2014: p. 12). BENDER (2005) states that 90 percent of innovations in manufacturing engineering are realized by information technology (IT) and therefore software can be considered as a business enabler (BENDER, 2005: pp. 7–8). In the automotive industry, software and electronics even constitute up to 90% of all innovations (BEECK, 2007: p. 205; BEUTNER et al., 2013: p. 19). A modern car from the year 2012 has about 100 million lines of code and therefore more code than a F-35

fighter jet from 2013 with ca. 24 million lines of code (NEWCOMB, 2012; AXE, 2012). Accordingly, on the one hand, software facilitates myriad of new product functionalities and hence increases functional complexity of products. On the other hand, a shift of variance from hardware to software partially reduces complexity of development and production of products (EIGNER et al., 2014: p. 2; BEECK, 2007: p. 205). Managing the vicious cycle of changeability and understandability is a challenge when designing products, particularly systems. Flexibility in dealing with changing information artifacts is needed, whereas this flexibility itself induces complexity in the development process (POMBERGER and PREE, 2004: p. 85; NEUMEYER et al., 2017: p. 29).

This vicissitude from hardware to software and their conjunction, so-called mechatronic products², has direct impact for automotive development. To enable new functionalities and hence address the arising complexity, the quantity of ECUs and communication busses within an automobile's E/E architecture rose intensely. In a modern luxury automobile, ECUs increased about 260% and communication busses 500% in the years from 1995 to 2013 (LANKENAU and HEBER, 2017: p. 4), as depicted in Figure 1-3. This steady augmentation of ECUs originates from the approach that for each new feature added, an additional ECU is added to the current E/E architecture, particularly in comfort electronics. However, in the power train domain a converse evolution is visible with more performant ECUs instead of quantitatively more due to a higher integration of functionalities in the power train domain (FROST & SULLIVAN, 2018: p. 10; BORGEESE, 2014: 90). As a fully autonomous automobile will require 40 to 120 ECUs solely to compute all autonomous applications and will generate approximately four terra bytes of data per day, there is a tendency for the automobile E/E architecture's complexity to rise (FROST & SULLIVAN, 2018: p. 10). FROST & SULLIVAN (2018) estimate that the average total amount of ECUs accumulates up to 178 in autonomous vehicles in Europe and North America in 2017 (FROST & SULLIVAN, 2018: p. 36).

² Mechatronic products today comprise mechanics, electronics, and informatics (software) and will be delineated in more detail in Chapter 2.2.2 (EIGNER et al., 2014: p. 43).

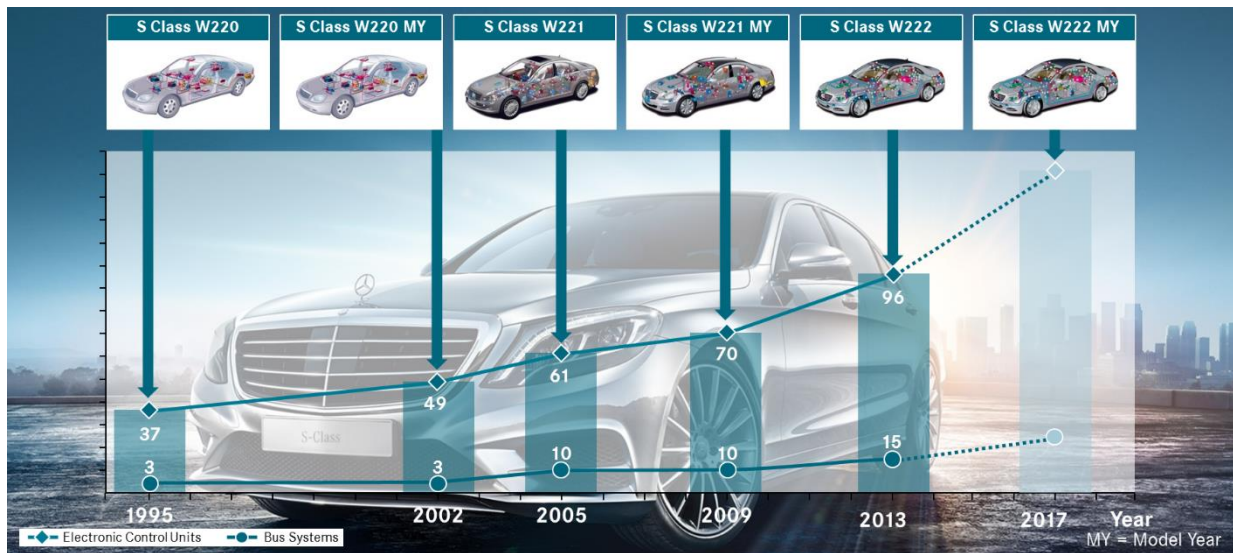


Figure 1-3: Increasing complexity in a luxury automobile's E/E architecture represented by the growth of the total number of ECUs and communication busses (LANKENAU and HEBER, 2017: p. 4).

Rampant product recalls of automobiles imply that the increased complexity of an automobile's E/E architecture is not yet appropriately managed (BERTSCHE et al., 2009: p. 5; RITTBERG, 2014: p. 63). Moreover, software already causes 15 percent of automobile recalls (HALVORSON, 2016). Not only does the customer demand a functioning automobile, regulations and laws also require high quality assurance, especially for safety-relevant applications (BEECK, 2007: p. 205; INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2011b: p. V; LÄMMER and THEISS, 2015: p. 463).

Due to an increased demand for variety, product variety grows, and therefore parts variety, and order variety augment. Hence, a higher variety in suppliers can result because the more diversified products are, the more knowledge is needed which cannot always be provided by the OEMs (EHRENSPIEL and MEERKAMM, 2017: pp. 863–868). An OEM alone cannot master such a complexity. Consequently, there exists a long tradition of supplier and engineering partner relationships with the OEMs in the automotive industry. In the last years, suppliers evolved from engineering partners to the developer or provider of entire systems. In this role, the engineering partners develop, produce, and deliver particular systems and the OEM sometimes only executes integration and assembly (KATZENBACH, 2015a: p. 610). However, the inclusion of new and more intertwined partners all over the world increases process and work organization complexity. This demands an organizational paradigm shift and hence yields a higher demand for reconciliation (EIGNER et al., 2014: p. 3; KATZENBACH, 2015a: pp. 607, 611).

1.2 Problem statement and delineation

1.2.1 Traceability of information artifacts to address increasing complexity in automotive E/E architecture

In order to address all vicissitudes of the automotive industry, from more product variance to innovations in E/E and their impact on the E/E architecture of an automobile, an automobile developing and producing company has to take measures. These measures could be to connect information artifacts. Also, their connections to distinct entities of a product, i.e., a configuration, and their changes across the lifecycle within a company. This is called traceability (EIGNER et al., 2014: p. 274).

A high variance of products is already established during development and it is much more costly to address issues and changes later in the product lifecycle (EIGNER and STELZER, 2009: p. 16). During development, 70% of total product costs are determined which will become effective to 94% in later lifecycles (VEREIN DEUTSCHER INGENIEURE, 1987: p. 3). Hence, the focus of this work will lie on the early development phases in order to address complexity there, where it has the highest impact on costs (EHRENSPIEL et al., 2014: p. 15). There exists a plethora of methodologies³ to support the product development process and to foster traceability mostly for one specific discipline but sometimes also for different disciplines, processes, and lifecycle phases (cf. EIGNER et al., 2014: pp. 15–52). A particular methodology for the development of mechatronic products, i.e., multidisciplinary products, is model-based systems engineering (MBSE). By means of digital and development-specific system models, an integration of information artifacts as well as modeling along the product development process occurs. The issue of integration and alignment of specific information artifacts during the development process can be alleviated in the early stages by such modeling approaches. For that purpose, correlations between system requirements, functions, behavior, and structure are defined explicitly (EIGNER et al., 2014: pp. 45, 77). The goal is to make documented information available to all different domains and organizational entities in a company. Product data management (PDM) particularly for development, and product lifecycle management (PLM), already exist for some decades and take a pivotal role in information and configuration management within companies during the

³ A methodology can be considered as a collection of related processes, methods, and tools (ESTEFAN, 2008: p. 10; EHRENSPIEL and MEERKAMM, 2017: p. 173). For further definitions of method and methodology, please refer to EHRENSPIEL and MEERKAMM (2017), EIGNER (2014d) and Chapter 2.

development and for the entire lifecycle, respectively. Moreover, PDM/PLM systems often provide application programming interfaces (APIs) for external engineering partners (cf. EIGNER and STELZER, 2009: pp. 27-42). However, the documents stored in PDM systems often do not represent connections and relations between information artifacts described in these documents or this information cannot be made visible without the distinct authoring tool (GILZ, 2014: p. 3). Thus, the more formalized approach of MBSE and its conceptual system design aligned to product data models and their management over the lifecycle is a solution approach which fosters traceability from the very start of the development process with the documentation of requirements (cf. GILZ, 2014: pp. 3-7).

As stated in Chapter 1.1, complex products often are developed conjointly between OEMs and engineering partners. Within such engineering collaborations, traceability of information artifacts, their respective changes, and configurations in a worldwide-distributed engineering and supply chain by means of potent IT technologies is crucial. Therefore, PLM concepts are prerequisites. Those PLM concepts foster ubiquitous information management (KATZENBACH, 2015a: p. 611; BEIER, 2014: p. 37). In order to understand better the connection of the mentioned complex single artifacts, it is helpful to map their interdependencies explicitly. Particularly, the heterogeneous methods of different disciplines require traceability of information artifacts. Hence, traceability is not autotelic but rather supports the comprehensive disciplinary understanding of systems of a product. Therefore, traceability is required by many norms and standards addressing both qualitative and statutory requirements towards traceability (cf. Chapter 2.1.1) (BEIER, 2014: p. 37; STARK, 2015: p. 45).

In this context, it can be distinguished between traceability within a company, i.e., internally, and externally, i.e., in the above-mentioned engineering collaborations. The former addresses the connection of information artifacts of diverse disciplines which develop parallelly or sequentially a product. The latter focuses on how to integrate information artifacts across many engineering partners that all face the challenges of internal traceability, too. This duality of internal and external traceability will be elaborated in more detail in the following chapter and generally occurs in all engineering collaborations for all involved partners, not only in the automotive but also in other industries (*vide supra*).

1.2.2 Current deficiency in research

INTERNAL TRACEABILITY

As already addressed above, complexity requires, inter alia, traceability in IT systems. MBSE and PDM/PLM are methods to provide traceability in specific disciplines and phases of the product lifecycle. MBSE fosters traceability commonly in the early phases of the lifecycle, PDM/PLM usually starting in the middle and extending towards the end. However, systems engineering⁴ is not stringently executed beyond the early product development phase. Hence, subsequent processes cannot use the information created during systems engineering. Consequently, data from manufacturing, support, and after sales cannot be used to enhance the product. Integration of MBSE into PDM/PLM requires plenty of alignment of different functionalities, such as to handle product lines, variability, design, simulation, and configuration⁵. However, this alignment is not yet fully achieved. This lack of alignment, preferably in one integrated platform, would address the increasing complexity and foster traceability (GRIEVES, 2012: pp. 236–241; PAVALKIS, 2016: pp. 2, 14; BIAHMOU, 2015b: pp. 225, 228, 231). GILZ (2014) addressed some of this alignment by the methodical integration of a functional product description. Though, some open points remain, for instance, the specific configuration management, specific workflows, organizational roles, as well as the application on the property level for electrics/electronics computer-aided design (E-CAD) and inclusion of software models (GILZ, 2014: pp. 183–184). Further research focused on the incremental integration of different stages of expansion of the model-based development process into the PLM according to the respective use case and how this integration of system models in the PLM environment could look like. Moreover, different possibilities of how IT systems could be linked in order to achieve this integration for the connection of MBSE and PDM/PLM were assessed and how the connection on data model level could look like. Restrictions of this joint research project so far is the lack of industrial application and the refinement of some aspects such as which is the optimal solution approach to document metadata in system models and distinct traceability schemes (LINDEMANN and KRSTEL, 2017: p. 150; KIRSCH et al., 2017b: pp. 161–162; MÜLLER and KIRSCH, 2017: pp. 178–179; MECPRO² ABSCHLUSSBERICHT, 2016d: p. 28; MÜLLER and HABE, 2017a: p. 185, 2017b: p. 228; BEIER, 2014: p. 256). Furthermore, the missing documentation of

⁴ For the definition of systems engineering and the distinction to model-based systems engineering, please refer to Chapter 2.4.

⁵ For more details please refer to Chapter 2.3 and 2.4 and PAVALKIS (2016).

dependencies caused by low transparency about changes and their impacts requires the explicit modeling of information artifacts within a system and across organization structures (KÖNIGS et al., 2012: pp. 926, 927, 930).

The availability of once created information artifacts to the other downstream or parallel processes in the development process is threefold⁶. Horizontal integration requires the integration of information artifacts along the development process regarding milestones and procedural sequence. Vertical integration shall ensure that information artifacts are modeled gradually with increasing level of detail. Interdisciplinary integration aims at integrated information artifacts of different disciplines, such as mechanics, E/E, and software. These diverse integration approaches foster traceability and still have to be fully achieved (TRIPPNER et al., 2015: p. 560).

EXTERNAL TRACEABILITY

Particularly, the exchange of system models beyond a company's boundaries is crucial for traceability as well as to reduce reconciliation and is not yet fully supported. Also, conventions of how the composition of system models in engineering collaborations and its processes could look like and whether libraries of system models across companies would support this, are still open issues. Research with scope on enabling technologies for engineering collaboration identify a high necessity for more flexible and more efficient solutions to transfer data amongst multiple engineering partners (STIEFEL, 2011: pp. 280–283; MECPRO² ABSCHLUSSBERICHT, 2016d: p. 28). Hence, the implementation of common system models and joint development processes could reduce reconciliation and error proneness and thus foster external traceability. However, so far, there do not exist sufficient traceability schemes for OEM and supplier communication, i.e. the transmission of supplier-specific traces of information artifacts in the context of the OEM's product (BEIER, 2014: pp. 80-81, 256). A better integration of IT systems and data models reduces coordination, especially in case of highly integrated system suppliers (KATZENBACH, 2015a: pp. 610–611, 632–633). Communication between departments, organizations, and suppliers involved in development increases and yields errors based on missing traceability (KÖNIGS et al., 2012: 926–927, 939).

Another aspect of external traceability, besides the reduction of reconciliation and coordination in engineering collaborations, is the transparent and safe documentation of changes made during development between multiple engineering partners. In this

⁶ The fourth dimension “modern workplace” is not in scope here (cf. TRIPPNER et al., 2015: p. 560).

context, transparent means that changes during product development are visible in due course to all parties involved. Safe means that changes are documented and stored with legal protection for the obligation to prove the accuracy of components developed and delivered as well as failure tolerant. This means that all data is available to all parties at any time regardless of whether one server is offline. KATZENBACH (2015a), LÄMMER and THEISS (2015), and STJEPANDIĆ et al. (2015a) highlight the dangers of sharing intellectual property (IP) in engineering collaborations due to constant exchange of data between partners who could later become harsh competitors (STJEPANDIĆ et al., 2015a: pp. 521, 526; LÄMMER and THEISS, 2015: pp. 464, 474; KATZENBACH, 2015a: pp. 611–612). STIEFEL (2011) proposes a peer-to-peer network approach to address traceability between multiple development partners. In a peer-to-peer network, the failure of a node is possible. If each engineering partner only hosts their own data in order to better protect IP and reduce bandwidth by not exchanging all data, the failure of one IT system or node induces the loss or unavailability of data⁷ (STIEFEL, 2011: pp. 51, 281). In order to avoid this scenario, a central node for data hosting would be necessary again what, in turn, contradicts the protection of IP and the need for tamper-proof documentation. Moreover, immediate transparency about change activities would alleviate exhausting communication and data search in today's distributed engineering collaboration using different data formats which is not yet fully addressed by recent concepts (BIAHMOU, 2015b: pp. 222–223; KÖNIGS et al., 2012: p. 926).

Due to more intertwined engineering collaboration (cf. Chapter 1.1), a simple connection of IT systems with engineering partners by means of standardized APIs also enables external traceability (LÄMMER and THEISS, 2015: p. 464). This is also required by the Code of PLM openness (CPO)⁸ and serves as the fundamentals for a joint PLM concept (BIAHMOU, 2015a: pp. 790–791; DEUTSCHES INSTITUT FÜR NORMUNG E. V., 2018c: pp. 5–7; KATZENBACH, 2015a: p. 611). This conceptual framework is normative and hence the actual implementation and execution of norms required by the CPO is incumbent on the company aligning with it. In collaborations with non-trustworthy partners, i.e., potential future competitors or ad hoc contributors, a sub-collaboration in a separate network alleviates onboarding of engineering partners. Additionally, many partners can suggest new product models and engineering approaches by joining the standardized

⁷ Cf. Chapter 2.7.3 for more information about peer-to-peer networks.

⁸ Code of PLM openness (CPO) is an initiative by the *prostep ivip* association focusing on interoperability, infrastructure, extensibility, interfaces, standards, architecture, and partnership in engineering collaboration (cf. DEUTSCHES INSTITUT FÜR NORMUNG E. V., 2018a, 2018b, 2018c).

engineering network (STIEFEL, 2011: pp. 106–108, 150). This alleviated connection of engineering partners fosters traceability by an easy, system-based exchange of data instead of data sent by email or spreadsheets. Reasons for this could be a long and tedious IT connection process or no connection at all due to costs. STIEFEL (2011) describes a framework for a peer-to-peer engineering collaboration network based upon the necessity of flexible and efficient solutions for collaborative engineering. However, this described network is not reliable enough in the sense that peers might fail and hence data is not available⁹. Also, standardized inclusion of new peers, i.e. engineering partners, is complex and, depending on the chosen overlay-network, centralized peers have to hold data (STIEFEL, 2011: pp. 149–160).

The research deficiency is illustrated generically in the problem space in Figure 1-4 where the entire product lifecycle is depicted at the top, underneath the V-model of development (cf. Chapter 2.2.2), and at the bottom the contemporary heterogeneous IT tool and system landscape of bigger enterprises. Information artifacts are created in each IT tool, often are domain-specific and, if not standardized, hardly can be connected with or transferred to another data model of other tools or systems. However, this connection of information artifacts is necessary due to the increasing complexity in automotive E/E development (*vide supra*) in order to align development results across disciplines as well as to track errors and changes in other phases of the lifecycle. This issue of traceability is augmented if there are multiple engineering partners and suppliers developing jointly for one final product. In such a case, the standardized exchange of data is crucial.

⁹ Please refer to Chapter 2.7.3 and STIEFEL (2011) for more information about peer-to-peer networks.

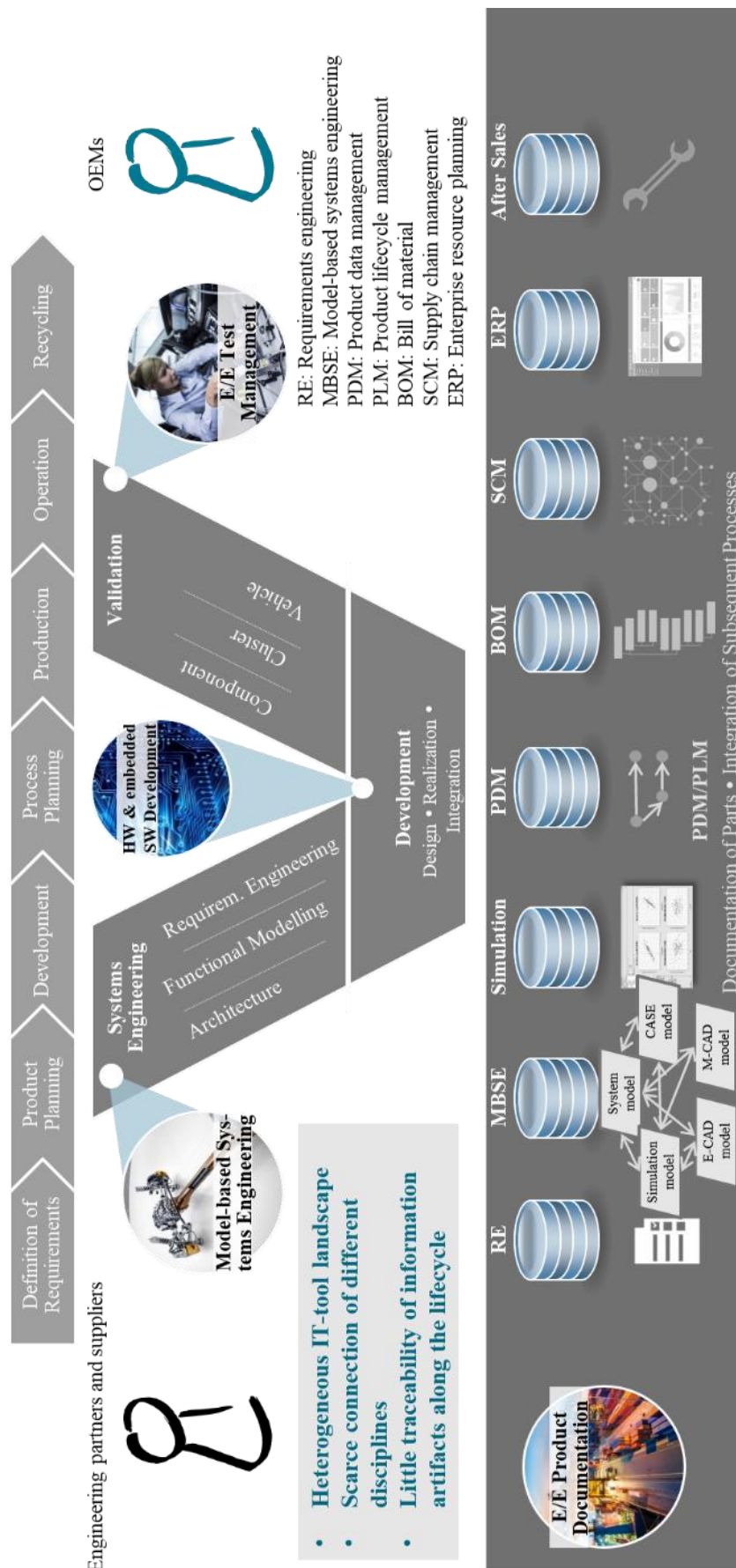


Figure 1-4: The problem space of automotive E/E development in a heterogeneous IT tool and system landscape (in alignment to HEBER et al., 2018: p. 8; HEBER and GROLL, 2018a: p. 281).

1.3 Objectives of this thesis

Taking into account the issues previously stated and the current deficiency in research, the objectives of this work is to describe the traceability in automotive E/E engineering collaboration generically, i.e., for all OEM-supplier-relations. Therefore, focus will be on:

1. Internal traceability i.e., within a company's IT systems:
 - a. Alignment of MBSE and PDM for E/E;
2. External traceability, i.e., among multiple engineering partners:
 - a. Reduction of reconciliation;
 - b. Transparent and safe product changes;
 - c. Alleviated connection of engineering partners.

The objectives are depicted schematically in Figure 1-5, where a generic system development process in automotive E/E engineering collaboration is shown. This generic process was described and analyzed with engineers and for the tool responsible persons from all different domains. The three different swim lanes describe the engineering collaboration partners, i.e., the OEM and suppliers for mechanical, E/E, and software parts. The development process is subdivided according to the requirements, functional, logical, and physical (RFLP) approach¹⁰. At the beginning (R-phase), the Engineer A, responsible for the product, e.g., an electronic control unit (ECU) at the OEM, formulates the requirements and sends them to the suppliers. The suppliers then, each separately, develop their subcomponents without standardized or only partially standardized and IT-supported exchange of data in the progress of development within the separate steps of RFLP, each with more details. Moreover, the OEM internally develops and integrates the E/E system or parts of it. However, the OEM also faces disruptions between IT systems of different domains, such as mechanics, E/E, and software. The whole E/E system with all its subcomponents will be integrated for testing purposes in the software or hardware in the loop (SiL/HiL). Therefore, data models have to be exchanged or suppliers have to get access to the OEM's IT systems. Often only then are errors identified due to a lack of traceability and transparency during the early development process. Then, the OEM will report the errors and resulting adaptations to the suppliers and a further reconciliation cycle starts. However, testing is not in scope of this work which will mainly focus on the early phase of development. The potentials for improvement and the objectives of this thesis are highlighted in yellow. The goal is

¹⁰ See Chapter 2.4 for more information about the RFLP approach.

to make development faster, more effective and efficient by an increased traceability internally as well as externally.

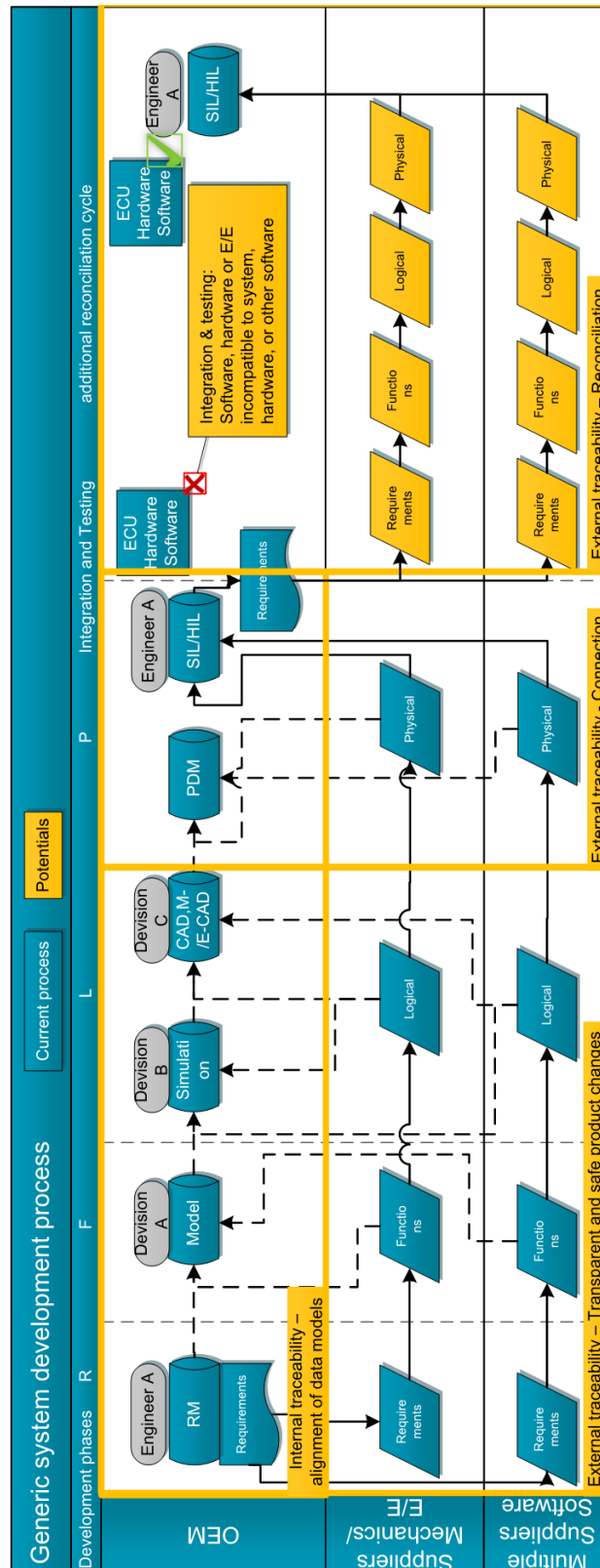


Figure 1-5: Generic system development process in automotive E/E development with multiple engineering collaboration partners. The objectives are marked yellow as potentials for improvement (cf. SCHÄUFFELE and ZURAWKA, 2016: p. 199).

1.4 Design research, research methodology, and structure of this thesis

Natural science and the thereof descended behavioral science can be considered as descriptive sciences, testing hypothesis with collected data and deducing theories. In contrast, design research focuses on problem-solving where the “goal is to produce an artifact which must be built and then evaluated” (HEVNER and CHATTERJEE, 2010: p. 5). SIMON and LAIRD (2019) denominate the latter, design research and science of engineering, as *science of the artificial* in contrast to *science of the existing*, which would be the natural sciences¹¹ (SIMON and LAIRD, 2019: pp. 4–5; HENSELER, 2021: pp. 27–28).

“Designers, are exploring concrete integrations of knowledge that will combine theory with practice for new productive purposes” (BUCHANAN, 1992: p. 6). Hereby, designers, such as engineers, deal with *wicked problems* and try to solve them. Immanent to wicked problems is their non-linear analysis, incomplete requirements specification, confusion, conflicting stakeholders, unclear consequences within the system, and lack of an own subject matter other than what the designer envisions¹² (CHURCHMAN, 1967: p. 141; BUCHANAN, 1992: pp. 15–16). Distinguishing the subject matter between a general and particular level, the designer generates a working hypothesis of the appropriate scope of its humanmade application or product on the general level. This can be considered as the *artificial* (vide supra). However, “design is fundamentally concerned with the particular, *and there is no science of the particular*” (BUCHANAN, 1992: p. 17). According to BUCHANAN (1992), the *particular* work of designers begins with a so-called *quasi-subject matter* or *placement*. This is the basis in design thinking in which the designer creates a working hypothesis using placements. Science and design thinking are congruent in regard to that placements in design thinking are what constitutes in science a subject matter. Hence, design can be considered as an integrative discipline (BUCHANAN, 1992: pp. 17–18). CROSS (2006) also states that *design science* can be considered as a scientific activity itself with an “explicitly organised [sic], rational and wholly systematic approach to design” (CROSS, 2006: p. 98). In contrast, *science of*

¹¹ For an overview, what constitutes the *science of the existing*, e.g., physics, economics, etc., and the *science of the artificial*, e.g., engineering, medicine, etc., please refer to HENSELER (2015); HENSELER (2021). For the connection of behavioral and design research and testing thereof, please refer to HENSELER (2017).

¹² For further definitions and properties of *wicked problems*, please refer to RITTEL and WEBBER (1973).

design concerns itself with the improvement of understanding of design by means of scientific methods (CROSS, 2006: pp. 98–99).

Furthermore, design activities postulate a future solution, i.e., a reality that does not yet exist. This future solution could be a composite of existing solution fragments which combined yield a surplus benefit. This is called emergence (HENSELER, 2015: pp. 16–17). Composition and assembly constitute emergence from a design perspective (NELSON and STOLTERMAN, 2003: p. 93).

Given the explorative, problem-solving, *wicked*, *artificial*, and *emergent* nature of this work and its underlying scope with the aim of creating, by composition and assembly, and evaluating a novel solution approach, a general design research approach will be followed to structure and conduct this work. This will be presented in the following.

RESEARCH METHODOLOGIES

The *design research methodology* (DRM) by BLESSING and CHAKRABARTI (2009) is a widely accepted research approach for supporting mainly engineering and industrial design research in order to ameliorate the design process. Here, design is defined as activities aligned to the development of a product from the initial requirements via a solution idea or technology, to the full documentation needed (BLESSING and CHAKRABARTI, 2009: pp. 1–2). The DRM can be considered as a distinct activity of design research rather than “a framework in which multidisciplinary methodological approaches are facilitated” (ECKERT et al., 2003: p. 254). The DRM strives to structure design research for the purpose of generating: a clear goal, methods, and tools as solutions for distinct problems that actually exist. The definition of success criteria is a very rigid concept for the definition of goals. This stems from the fact that DRM was designed to ameliorate industrial practice. In contrast, *the eight fold model of design research* by ECKERT et al. (2003) is a more agenda-driven framework for design research wherein the eight different research steps are more generic and allow for the inclusion of several research projects, such as PhD studies (ECKERT et al., 2003: pp. 253–255; BRAUN, 2013: pp. 129–130). Due to the DRM emphasizing a more industrial application and also aims at individual projects being executed towards practical outcome (ECKERT et al., 2003: p. 254), the DRM will be used as a blueprint for this thesis where suitable. However, the DRM will not be executed stoically step by step¹³. The DRM framework is presented in

¹³ Please refer to BLESSING and CHAKRABARTI (2009) and ECKERT et al. (2003) for an overview of more research methodologies regarding the design phase in engineering development.

Figure 1-6. Here, each stage's deliverables are used iteratively as input for the previous stages.

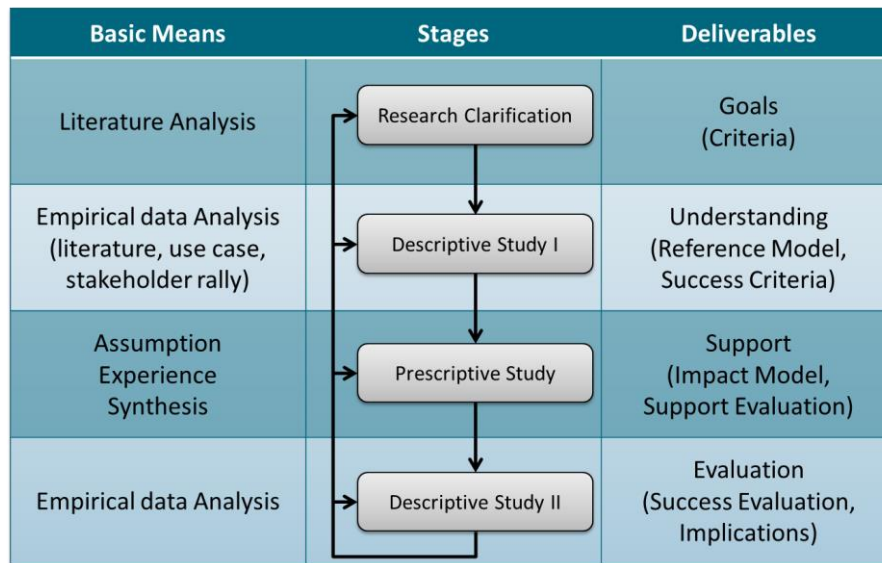


Figure 1-6: Framework of the design research methodology (DRM) including stages, basic means, and deliverables (in alignment to BLESSING and CHAKRABARTI, 2009: p. 39).

APPROACH FOR THE ASSESSMENT OF THE CURRENT STATE OF SCIENCE AND TECHNOLOGY

There exist distinct methods for development of E/E components in MBSE, the documentation of these components in IT systems along their lifecycle, engineering collaboration, and technologies connecting all these different domains. Hence, the research approach requires a precise assessment of each domain with respect to elements which could integrate them. Literature proposes certain elements for this purpose.

PAVALKIS (2016) suggests that a solution approach to connect MBSE with PLM should follow the three steps of agreement of a metadata model, identification of a process, and then selection of the technology (PAVALKIS, 2016: pp. 2466–2467). As crucial for the management of data and information streams within PLM, FELDHOUSEN and GEBHARDT (2008) identify the structure of a standard product (a data model), the structure of a standard process, and a knowledge repository, i.e. a specific technology that contains and connects this information (FELDHOUSEN and GEBHARDT, 2008: pp. 73–74). PFENNING (2017) also suggests a central link repository to connect MBSE and PLM data (PFENNING, 2017: pp. 156 ff.). This can be extended with a cooperation model (KÖNIGS, 2013: p. 40). STÖCKERT (2011) emphasizes the necessity that interfaces in distributed engineering processes have to be assessed according to objects (data model),

processes, and tools (technology)¹⁴ (STÖCKERT, 2011: pp. 102–103). A more technical view focusing on communication between models assesses languages and data representations (data model), ontologies (data model), tool, and interfaces (technology) (FISHER et al., 2014: p. 224). For the template-based systems engineering and based upon the systematics of product development, prerequisites are a model for the description of technical systems, a method to handle models efficiently, and a tool which implements both (KÖNIGS, 2013: p. 7). ESTEFAN (2008) differentiates the diverse aspects of MBSE accordingly (ESTEFAN, 2008: p. 9): i) process, ii) method, and iii) tool¹⁵. STARK (2016) mentions in his *PLM grid* the aspects which have to be addressed for the management of a product across its lifecycle and handle the inherent complexity and difficulty. Processes, methods, and tools as well as their interfaces are also part of the *PLM grid*. Additionally, STARK (2016) mentions on a lower level of granularity product data, such as CAD models and master data, as part of the grid (STARK, 2016: pp. 5–6).

This work focuses more on fostering traceability on the lower implementation level, i.e., a technical level. Therefore, the first step is to connect different IT tools by a defined technology. Afterwards, aligning the semantic level, which is the data model, between different IT tools has to be done. Then, a process model defines at which point in time which data shall be created and exchanged given the correct technology.

To subsume the above-mentioned approaches, which all aim at fostering traceability throughout the development by means of data, processual, and technological support, the current state of science in Chapter 2 will be assessed using a threefold approach (in alignment to ESTEFAN, 2008: p. 9; KÖNIGS, 2013: p. 7; GILZ, 2014: p. 8):

1. *Data model*: For the connection of different disciplines working in different IT tools and systems, a joint data model serves as a connecting factor and bridges the gap between MBSE and PDM. It is crucial for the exchange of data in an engineering collaboration and hence a key ingredient of traceability.
2. *Process model*: If many parties are involved in the development, it has to be defined at which step which data has to be handed over to whom in order to enable and maintain traceability.

¹⁴ Here, the fourth element would be *people* (STÖCKERT, 2011: pp. 102–103). However, this would be included in the fourth element *environment* in ESTEFAN (2008) and hence is not in scope (cf. Footnote 15).

¹⁵ The fourth element in ESTEFAN (2008) would be *environment* but it will not be considered here separately due to limitations of the scope and a vast variety of degrees of freedom this would induce, for instance social, organizational, cultural, or personal aspects.

3. *Technology*: A technology enabling the connection of MBSE and PDM, developing disciplines and their idiosyncrasies, alleviating the integration of engineering partners, and connecting heterogeneous IT systems on a technical level, is fundamental for intra- and inter-company traceability.

Figure 1-7 depicts the tripartite enabling elements, also called enablers, for traceability, data model, process model, and technology, by which each aspect of the current state of science and technology will be assessed in order to ensure that the aspired solution will suffice the postulated objectives. The centered triangle contains the main aspects of the problem space that are in scope, i.e., the early phase of automotive development, focus E/E, within engineering collaborations and shall be enabled by the surrounding elements which foster traceability.

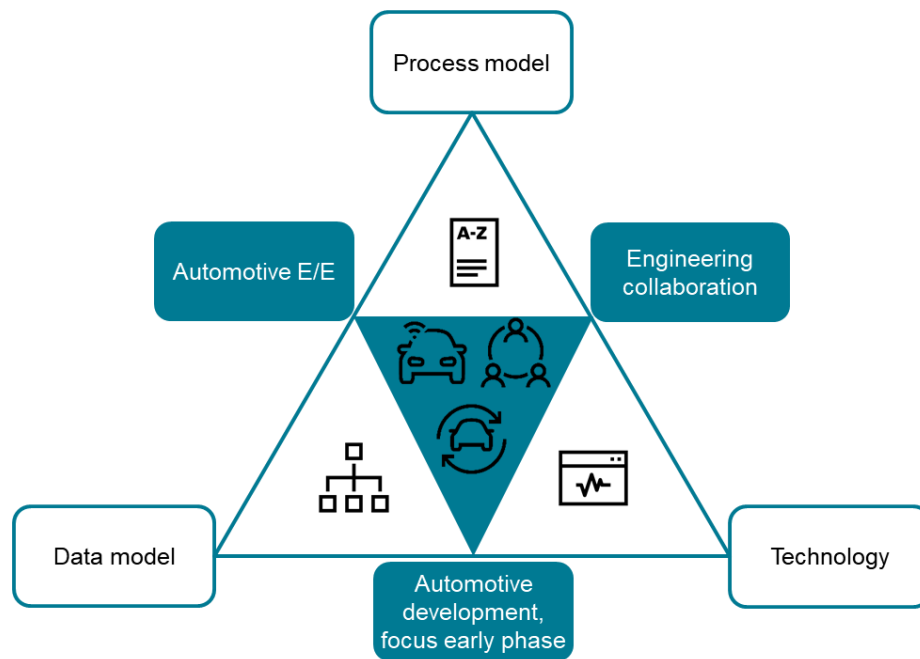


Figure 1-7: The problem space of this work (blue) and the enablers data model, process model, and technology.

STRUCTURE OF THIS THESIS

For the purpose of achieving the research objectives, this thesis is built according a structure which addresses all relevant aspects. In Chapter 2, the current state of science and technology will be elicited. Therefore, traceability, product development processes, PDM/PLM, MBSE, automotive E/E, engineering collaborations, data base solutions, and ontologies each are defined, investigated, and assessed by means of the enablers (vide supra) how they foster traceability. Chapter 3 derives requirements for a solution approach based upon the state of science and technology in Chapter 2. Moreover, the

fulfillment of these requirements by existing enablers will be assessed and the necessity for improvement will be derived. In Chapter 4, the synthesis of a solution framework according the enablers will be developed. The prototypical implementation of the solution framework will be presented in Chapter 5. A subsequent evaluation of the solution framework and accordingly selected use cases will be described in Chapter 6. This work will be finalized by a summary and an outlook in Chapter 7. This structure is shown in Figure 1-8. The deductive approach to assess the applicability of the solution framework will be discussed in more detail in Chapter 6.4.

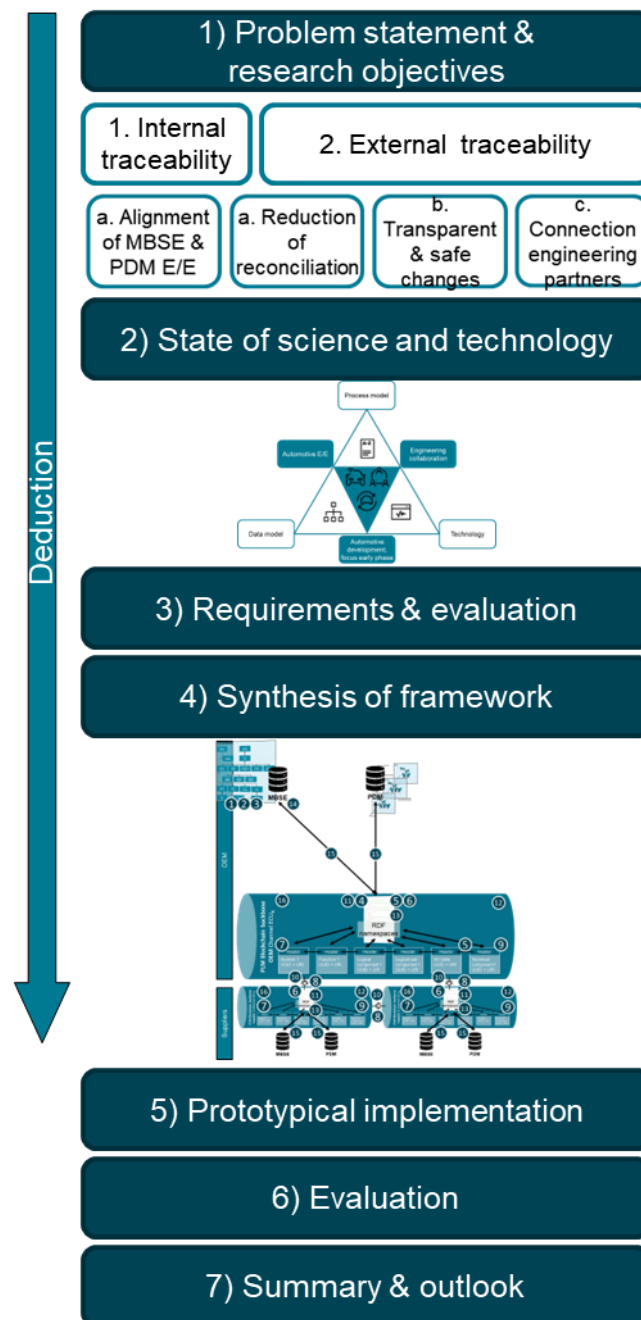


Figure 1-8: Structure and approach of the thesis.

2 Current state of science and technology, definitions, and general terms

This chapter depicts the current state of science and technology with respect to the fundamental theoretical frameworks and methods which later are necessary to build the own solution framework upon. As depicted in Chapter 1, the early phase of automotive E/E development is a complex endeavor and traceability within a company's IT landscape with external engineering partners is crucial more than ever. Against this background, all fundamentals within the scope of an early automotive E/E product development with external, distributed engineering partners will be stated in this chapter. The focus particularly lies on traceability and how it can be achieved within multiple disciplines.

The approach of this chapter follows a deductive approach ("cone logic"), where applicable. This means to start with the more generic, high-level theory and later advancing to the more idiosyncratic, specialized details (cf. Figure 1-8). To encompass the recent and relevant developments in each theory, at the start of each chapter the definitions, norms, and standards are described. Then, the decisive methods of the respective disciplines are elaborated on or the specific phenotypes and processes relevant to the later work are explained. Finally, traceability in each theory is assessed: How can traceability be achieved in the particular discipline, and what fosters traceability in it?

INITIAL TERMINOLOGY

The term *method* indicates a description of a rule-based and systematic approach in accordance of which certain tasks have to be executed to achieve a given goal (LINDEMANN, 2009: p. 57). A method is part of a methodology (cf. EHRENSPIEL and MEERKAMM, 2017: p. 173; HEYN, 1999 according to EVERSHEIM and SCHUH, 2005: p. 17). A method has a prescriptive character, i.e., it is an instruction. In distinction to a process model, which describes *what* has to be done according to which steps, a method describes in what manner (*how*) something has to be done and subsumes different techniques in order to perform a task (EIGNER et al., 2014: p. 47; LINDEMANN, 2009: p. 58; ESTEFAN, 2008: pp. 9–10). In this context, a process is defined as "a logical sequence of tasks performed to achieve a particular objective" (ESTEFAN, 2008: p. 9). By usage of an instrument or tool, the efficiency of a task can be enhanced by supporting the *what*

and *how*. Consequently, a methodology combines related processes, methods, and tools¹⁶ (cf. MARTIN (1996) according to ESTEFAN, 2008: pp. 9–10).

2.1 Traceability

2.1.1 Definitions, norms, and standards

Traceability is required by ISO 9001:2015 “Quality management systems – Requirements” to ensure quality across a product’s lifecycle which is controlled by dedicated management systems. Within this norm, traceability is required explicitly for production and service provision in case the organization wants to identify uniquely its outputs and, for that purpose, the organization shall preserve documentation that is necessary to foster traceability (INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2015a: p. 41).

Traceability in systems and software engineering is defined as the

1. “discernible association among two or more logical entities, such as requirements, system elements, verifications, or tasks” (INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2017b: p. 478);
2. “degree to which a relationship can be established between two or more products of the development process, especially products having a predecessor-successor or master-subordinate relationship to one another” (INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2017b: p. 478).

Here in this work, the terms *logical entities* and *products* are denoted as (*information*) *artifacts* in order to confine from the association between products and entities to ECUs or automobiles. GOTEL et al. (2012) specify traceability of or within software and systems generally as: “Traceability is simply the potential to relate data that is stored within artifacts of some kind, along with the ability to examine this relationship” (GOTEL et al., 2012: p. 4). Hence, traceability thereby is an attribute of one or more artifacts (GOTEL et al., 2012: p. 9). Here, GOTEL et al. (2012) also use the generic term of *artifacts* to describe generic units of data. Likewise, other studies use the term artifact as an entity of which a system can be composed (WINKLER and PILGRIM, 2009: p. 531; BROY, 2013: p. 83). A *trace artifact*, i.e., a traceable unit of data such as requirements, data model

¹⁶ Some authors deviate from the given definition of methodology. For instance, the *SPES methodology* POHL et al. (2012) does neither include a defined process nor an IT tool. However, as *SPES methodology* is a fixed term, it will be used accordingly. For more information about the *SPES methodology*, please refer to Chapter 2.4.2.

classes, or even persons, can be either a *source artifact* or a *target artifact*. Source artifacts are called accordingly because they depict the artifact from which the trace originates. In contrast, target artifacts denominate the destination. The connection or association between source and target artifacts is referred to as *trace link* and can have a primary direction, i.e. from the source to the target artifact, and a reverse direction (GOTEL et al., 2012: pp. 5-6). This is depicted in Figure 2-1. An operationalized trace link with a primary direction can, for instance, be the triple “A implements B”¹⁷. Conversely, the reverse direction of a trace link could have the semantics “B is implemented by A” (note the passive voice) (GOTEL et al., 2012: p. 6). This already shows that trace links can have different types with different syntax (structure) or semantics (purpose), such as *implements*, *tests*, *refines*, or *replaces*.

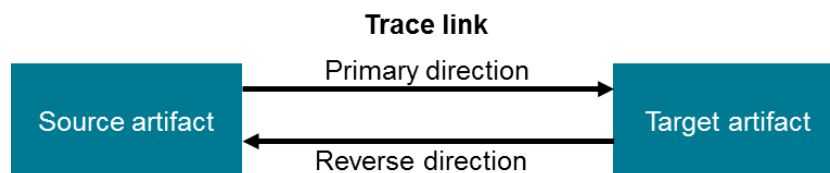


Figure 2-1: Association between source artifact and target artifact by means of a trace link (in alignment to GOTEL et al., 2012: p. 6).

Vertical traceability addresses the connections of different parts or components of the artifact, e.g., if the final product is denoted as the main artifact and has sub-components that evolve during the development process. In contrast, horizontal traceability captures the relations of these components across associated artifacts, for instance if a software belongs to an ECU (PFLEEGER and BOHNER, 1990: p. 323). However, literature disagrees upon the definitions of vertical and horizontal traceability (KÖNIGS et al., 2012: pp. 929–930). Here, the definition of PFLEEGER and BOHNER (1990) will be used as it aligns with the V-model of development (cf. Figure 2-9) where on one level, i.e. horizontally, different domains develop associated components simultaneously in time. The vertical axis also depicts a consecutive sequence of more and more detailed artifacts of one component or system. Connection of the consecutive development artifacts would induce vertical traceability; connection of the simultaneously developed artifacts of different domains would constitute horizontal traceability.

Many roles during the development process are confronted with a need for traceability. Due to this importance of traceability (cf. Chapter 1.2), there are plenty of norms and

¹⁷ For more information regarding triples, please refer to Chapter 2.8.

modeling guidelines from different disciplines that demand associations between information artifacts (BEIER, 2014: p. 37; ARKLEY, 2007: pp. 16–17). Moreover, in requirements traceability, a subset of requirements engineering, as well as traceability in general, research flourished. This is due to the necessity to trace and ultimately test requirements appropriately for the purpose of increasing quality of the final product¹⁸ (ARKLEY, 2007: p. 11).

For the purpose of managing traceability effectively and efficiently, it is crucial to provide appropriate means and tools to engineers to handle all relations between artifacts. Even for a medium-sized automobile with an already reduced effort of 65% due to a previous exclusion of relations on a higher aggregation level, about 700 million relations should be investigated for thorough documentation (KÖNIGS, 2013: p. 37). There exists a plethora of methods to manage and visualize traceability of complex engineering data during the product design phase. This will be shown in the next chapter.

2.1.2 Methods

One can distinguish between different representation techniques for traceability, such as matrices, cross referencing, entity relational models, or graph-based visualization (WIERINGA, 1995: pp. 12–13; ARKLEY, 2007: p. 21; WINKLER and PILGRIM, 2009: p. 542; KÖNIGS et al., 2012: p. 929).

MATRICES

A traceability matrix displays all information artifacts and which artifacts have an association with each other through trace links. Rows and columns of the matrix contain traceable items and do not have to be identical (GOTEL et al., 2012: p. 7; WIERINGA, 1995: p. 12). A special form of traceability matrices is dependency structure matrices (DSM). These matrices also are used in systems engineering. The DSM depict the connections between different development levels, such as which system requirement is satisfied by which system function and further executed by which component (GILZ, 2014: pp. 42-43, 95). It can be further differentiated between intra-domain matrices depicting vertical traceability, such as the DSM, and inter-domain matrices capturing

¹⁸ Requirements engineering, requirements management, and requirements traceability will not be in scope of this work explicitly. However, general concepts of traceability are used for requirements traceability as well as for other disciplines. Please refer to RAMESH and JARKE (2001), TORKAR et al. (2012), and ARKLEY (2007) for details about requirements traceability.

horizontal traceability, and further combinations of those¹⁹ (LINDEMANN et al., 2009: pp. 49–50).

CROSS REFERENCING

A cross reference can range from a trivial “see system specification version 2.3, section 4.8 interfaces”, via links within one document or tool, to outgoing and incoming hyperlinks stored in an artifact’s metadata. The surrounding context of the reference contains the semantics of the link. Given new languages and modeling methods, cross referencing augments in application (WINKLER and PILGRIM, 2009: pp. 542–543; WIERINGA, 1995: p. 13; ARKLEY, 2007: p. 22).

GRAPH-BASED

Whenever traceability links are understood as edges and information artifacts as nodes, such a traceability model can be represented similar to data models with a graph-based notation. This representation can include more information and hence increase interpretability. For instance, edges can have different meaning and nodes can also include metadata for the description of each artifact. Traceability matrices can be depicted as graphs²⁰ (WINKLER and PILGRIM, 2009: p. 543; KÖNIGS, 2013: p. 50; WIERINGA, 1995: p. 12).

The three different methods for traceability representation are depicted in Figure 2-2 (in alignment to WINKLER and PILGRIM, 2009: p. 542).

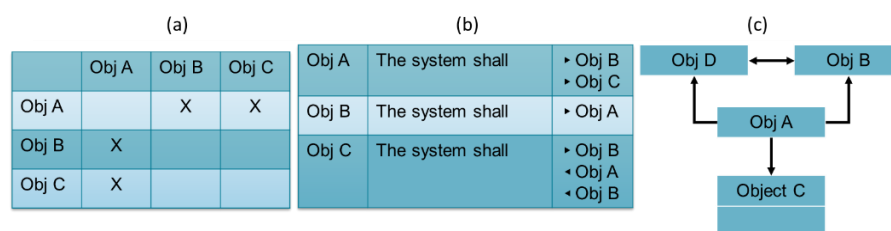


Figure 2-2: Different methods for traceability representation: (a) Traceability matrix, (b) cross referencing, (c) graph-based representation (in alignment to WINKLER and PILGRIM, 2009: p. 542).

¹⁹ Please refer to LINDEMANN et al. (2009), KÖNIGS (2013), and BEIER (2014) for an extensive overview of matrices to depict and manage traceability and dependencies.

²⁰ For an overview of different graph-based methods, please refer to SCHWARZ et al. (2010) and HERMAN et al. (2000).

2.1.3 Traceability in the context of automotive development

DATA MODEL

As described above, the necessity for traceability becomes more prominent in automotive development where many sub-systems have to realize multiple functions. Graph-based models can be implemented in any database technology and are often used in industry. Implementing a centrally managed data model connecting all different domains, their proprietary IT tools, and data models to foster traceability is also an issue for automotive development²¹ (WIERINGA, 1995: p. 12; BROODNEY et al., 2013: pp. 1176–1178; FIGGE, 2014: p. 33). How traceability through programming of domain-specific data models is enabled, will be explained in each of the following chapters and hence will not be outlined further at this point.

PROCESS MODEL

A huge impediment for integrated data models that allow traceability are decoupled activities in processes. Particularly defined processes can describe which information artifact or relation has to be documented at which development step in order to foster traceability. For that purpose, traceability process models depict all those relevant activities. FISHER et al. (2014) present a method for managing data models of systems engineering (cf. Chapter 2.4) in development scenarios. These are not specific to automotive development but are used to align the *create*, *read*, *update*, and *delete* (CRUD) operations amongst heterogeneous data models within the used tools and databases of the network. A joint management of activities, such as configurations, including versions, variants, and baselines (cf. Chapter 2.3) of models, enables traceability (FISHER et al., 2014: p. 214; GOTEL et al., 2012: pp. 10–11; KAUFMANN and SCHULER, 2017: p. 347). Each development domain has its own process models which have to be aligned with the process models of upstream, downstream, and parallel development domains and again their process models so as to allow traceability. This will be discussed in more detail in the following chapters.

TECHNOLOGY

There exist several tools that foster traceability and are applied to different domains of automotive development. The software *METUS* by *ID-Consult GmbH* enables the documentation of relations between requirements, functions, sub-functions,

²¹ Please refer to HAUSMANN (2010), SUTINEN et al. (2000), and SUTINEN et al. (2002) for more information about integrated data models which foster traceability during the product development.

components, modules, configurations, and suppliers. However, neither is it possible for relation types, i.e., edges, to be differentiated nor modeling up to the level of parameters. *METUS* also can be integrated into a PLM system. *ToolNet* by *EADS* and *DaimlerChrysler* was designed to manage traceability between different CAx tools centrally in a separate database on a qualitative level without parameters. The reuse and efficient handling of traceability relations in large companies is not within the scope of this tool. *LOOME* by *Teseon GmbH* mainly focuses on handling different types of traceability matrices. The software is non-integrative and hence requires copied data from other IT tools and systems. Therefore, traceability between data models from authoring tools, such as CAx, and PDM systems cannot be guaranteed²² (KÖNIGS, 2013: pp. 49–51; BEIER, 2014: pp. 55–57, 71–72). Above-mentioned are IT tools that visualize and manage traceability. Additionally, technology enabling traceability can also be found on a lower level of the IT infrastructure, such as special databases, interfaces, and networks. An elaboration will follow in the subsequent chapters.

2.2 Product development process

2.2.1 Definitions, norms, and standards

Most established process models for mechanics focus on the product development process in four major phases (EIGNER et al., 2014: p. 16): i) Clarification of requirements and tasks, planning; ii) Conceptual design; iii) Designing; iv) Elaboration, detailing.

In general, these phases apply to all sorts of products and their corresponding product development processes, regardless of their distinct domain, e.g., mechanics, software, or E/E. However, different literature includes or excludes different product lifecycle phases from the product development process (BURR, 2008: p. 35; STEPHAN, 2013: p. 6). In this elaboration, the above-mentioned view, limited to the actual requirements, conceptual design or the concept phase, design phase, and elaboration as well as detailing phase will be applied. Here, the product development process is considered a section of the product lifecycle and excludes, for instance, the strategy phase, ramp-up for production, production itself, aftersales and recycling (STEPHAN, 2013: p. 7; BURR, 2008: pp. 35–36). The product development process can also be distinguished from the product creation process as the product creation process also includes production planning and production (MÜLLER et al., 2012: p. 173; VEREIN DEUTSCHER INGENIEURE,

²² For more information about traceability tools, please refer to KÖNIGS et al. (2012) and FIGGE (2014).

2014: pp. 5–6). Product lifecycle management spans over the entire product lifecycle and will be discussed further in Chapter 2.3. The distinction between product development and product creation in regards to the entire product lifecycle is illustrated in Figure 2-3 (in alignment to VEREIN DEUTSCHER INGENIEURE, 2014: pp. 5–6; MÜLLER et al., 2012: p. 173):

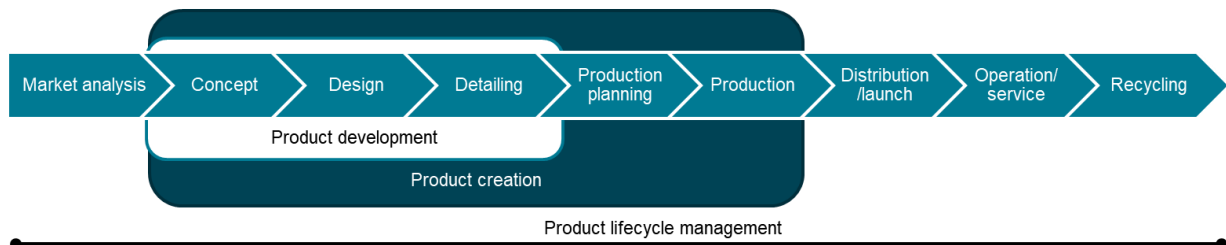


Figure 2-3: The main phases of a product lifecycle in differentiation to product development and to product creation (in alignment to VEREIN DEUTSCHER INGENIEURE, 2014: pp. 5–6; MÜLLER et al., 2012: p. 173).

The process of product development has significantly changed in the last few decades due to a parallelization, distribution, and interconnectedness of projects, product, and processes around the world; both intra-company and inter-company. The first evolution of product development is called serial engineering. Each phase of product development used the predecessor's output as input consecutively. Product development took place mainly in one company. The next evolution is called simultaneous engineering. Here, phases of product development, such as product design and production planning or purchasing of machinery, overlap. This is achieved by a better integration, organizationally as well as with computer-aided engineering (CAE). By those means, time to production can be reduced. This helps to address the requirements of markets and customers faster. Simultaneous engineering commonly describes product development within one company. Cross enterprise engineering is the latest evolution of product development where product development as well as production planning are interconnected between different locations of one company across the world as well as between separated companies. This means that a multidimensional collaboration and cooperation within a company and in the context of supplier and customer relationships takes place. This collaboration is neither limited by a company's boundaries, nor limited to one domain of product development, i.e., mechanics, E/E, and software. Collaborations can span over all product lifecycle phases (EIGNER and STELZER, 2009: pp. 18–20; STEPHAN, 2013: pp. 10–11). The evolution of the product development

process in the last few decades is depicted in Figure 2-4 (in alignment to STEPHAN, 2013: p. 11; EIGNER and STELZER, 2009: p. 19).

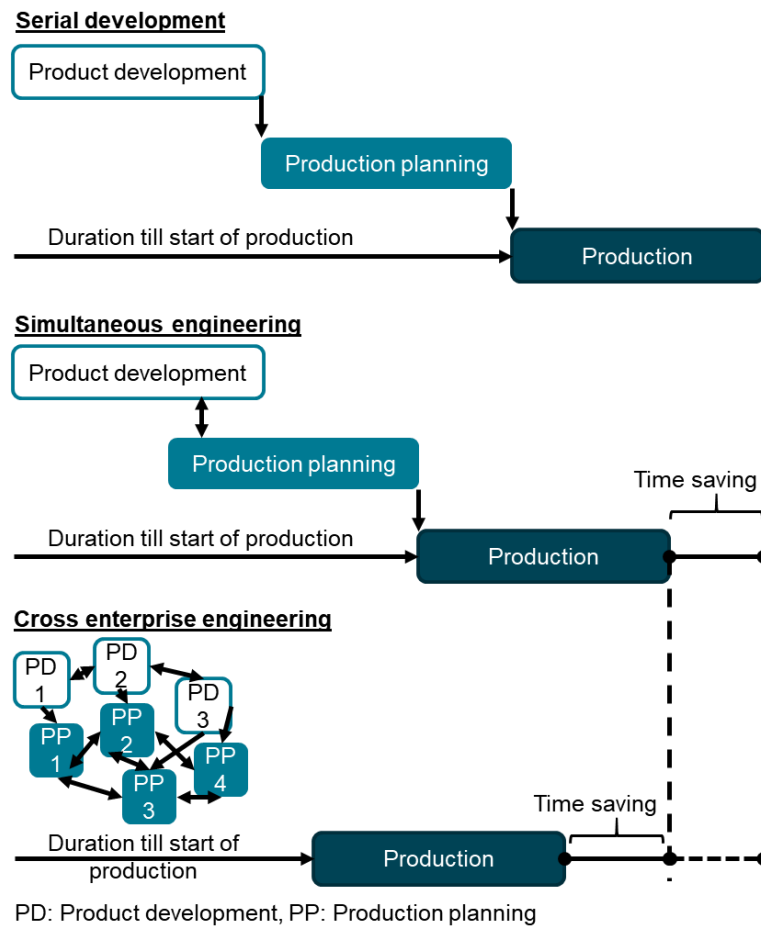


Figure 2-4: Evolution of the product development process over time (in alignment to EIGNER and STELZER, 2009: p. 19; STEPHAN, 2013: p. 11).

Due to the increasing interaction and connection between producers, international engineering locations, and suppliers across the world, requirements towards IT solutions have increased to address:

- The integration of IT solutions across the entire product lifecycle, i.e., from the first idea and conceptual solution until recycling. The integration of IT solutions across the product lifecycle has to be granted by means of APIs²³ or functional interfaces, for instance STEP AP 242²⁴ or JT²⁵. In practice this means the

²³ An example for a normative approach for APIs gives INTERNATIONAL ORGANIZATION FOR STANDARDIZATION (2011a).

²⁴ STEP AP 242: Standard for the exchange of product data application protocol 242 for managed model-based 3D engineering of the ISO standard 10303 (INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2014). For more details on data exchange standards, please refer to Chapter 2.6.1.

²⁵ JT: Jupiter tessellation is a standard primarily used for the exchange of 3D data. Due to the JT standard is not as performant regarding data exchange as STEP and is limited to 3D data (KATZENBACH et al.,

integration of a plethora of domain-specific IT solutions that are scattered across the product lifecycle.

- The federation of IT solutions in a decentral and distributed working company and, in the context with the supply chain, also beyond a company's boundaries. Federation of data and processes is required by the distribution across the entire product lifecycle, interdisciplinarity, as well as the increasing connected and interlinked development and production partnerships. Distribution can occur: i) within a company, e.g., between many locations and across several phases of the lifecycle; ii) between companies of a joint network of suppliers; and iii) between companies in a customer/supplier relationship (EIGNER and STELZER, 2009: pp. 20–21).

The integration of different IT solutions, i.e., systems and tools, across the product lifecycle will be discussed in more detail in Chapter 2.3.1.

Interdisciplinarity, as one requirement for the federation of IT solutions, describes the cooperation of different disciplines or domains. This interdisciplinary development will be elaborated further in Chapter 2.2.2.

2.2.2 Processes and methods for product development

In order to develop products, different phases in a product lifecycle have to be passed through (cf. Chapter 2.2.1). Scientists from different disciplines suggested for many years methods and process models to support the product development process. Mostly, these methods and process models have to be considered as procedural guidelines in which different phases of the product development process are defined as best practices and are traversed once or repetitively. Additionally, development results or deliverables of each phase in the process model are stipulated (EIGNER, 2014d: p. 15). Moreover, process models help the users to verify in which step they are currently within a process and which step has to be executed next. Reflection of one's own proceedings by means of the process model also helps the engineer to control their actions (PONN and LINDEMANN, 2011: p. 17). There has to be a distinction between domain- or discipline-specific²⁶ methods and process models. These disciplines are the

2015: p. 301), the JT standard is not in scope of this work. For more information, please refer to KATZENBACH et al. (2015).

²⁶ In the context of product development, *domain* and *discipline*, i.e., mechanics, E/E, software, systems engineering, are used synonymously in this work.

development of mechanics or hardware, E/E, software, and mechatronics as a combination of all of the afore mentioned.

MECHANICS

The guideline *VDI 2221* (VEREIN DEUTSCHER INGENIEURE, 1993) classifies the general approach of development and design beginning with the task of development until completion of design in seven distinct steps. Here, the focal point is on the deliverables, i.e., the result documents or development results, which each single step yields as a work result. The deliverables, for instance could be a requirements list, functional structure, or principle solution, which depict representations or partial models, respectively, of the product with increasing degree of detail and concretion. The representation of the process model in the guideline *VDI 2221* conveys a very sequential character. However, the necessity of recesses in the sense of iterations also is emphasized (PONN and LINDEMANN, 2011: pp. 17–18). The workflow with its steps and deliverables in the context of the four major development phases, is presented in Figure 2-5 (in alignment to PONN and LINDEMANN, 2011: p. 18; VEREIN DEUTSCHER INGENIEURE, 1993: p. 9; STEPHAN, 2013: p. 26; EIGNER et al., 2014: p. 16)²⁷.

ELECTRICS/ELECTRONICS

In E/E development, a diversity in development methods is more prevalent due to multiple reasons. On the one hand, E/E is a vast discipline concerning electrical installations, plant construction, automotive, aerospace, conductor boards, chips, microprocessors, flash memory, schematics, etc. On the other hand, there is a fundamentally different approach by engineers in E/E in contrast to mechanics. In E/E, there exists a design level of the schematical draft (circuit design) before the geometrical design (layout). Additionally, there is a rapid technological change particularly in the field of the circuit design as well as the evolution in automation technology with respect to logical and physical design and verification (EIGNER, 2014d: p. 21, cf. WEHN, 2013 according to EIGNER, 2014d: p. 21). It is noteworthy that for some methods of E/E development a step called “behavior” occurs. This step depicts the behavior of algorithms, registers, Boolean algebra and differential equations in E/E hardware

²⁷ For further methods of development in mechanics, please refer to PONN and LINDEMANN (2011), FELDHOUSEN and GROTE (2013), EVERSHEIM and SCHUH (2005), WINZER (2016), and EIGNER et al. (2014).

(EIGNER, 2014d: pp. 26–27). The development level “behavior” or “function” is highly important in a model-based approach for systems development (cf. Chapter 2.4)²⁸.

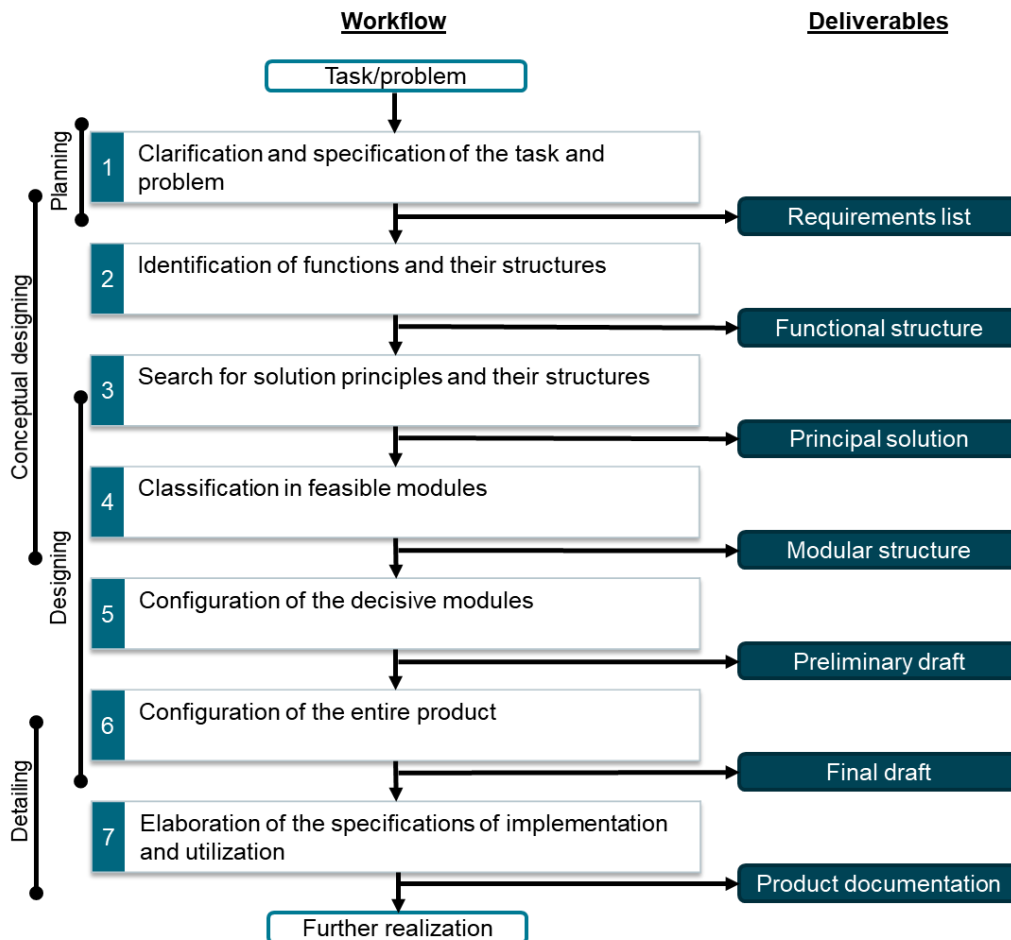


Figure 2-5: General process model for product development and design (in alignment to PONN and LINDEMANN, 2011: p. 18; VEREIN DEUTSCHER INGENIEURE, 1993: p. 9; STEPHAN, 2013: p. 26; EIGNER, 2014d: p. 16).

SOFTWARE

Software development has always been severely affected by fast changing customer requirements, and need to implement those new demands quickly into products. Usually, to mitigate development risks and to realize the demanded quick implementation, new software is built on basis of existing software components. Hence, fast reactions to new market demands were feasible and existing knowledge could be reused (FELDHUSEN et al., 2013: p. 808). In software development, this reuse is achieved using previously developed software components (BROWN, 2000: p. 8). As requirements became ever more volatile and had to be implemented faster, methods evolved likewise.

²⁸ For more information on recent methods in the field of (virtual) product development and design techniques please refer to STELZER (2014) and BEUTNER et al.(2013).

So-called *ponderous* or *heavy* process models are strictly phase-oriented models and are very formalized with respect to processes and the amount of associated (intermediate) results and (intermediate) products, respectively. Prominent phenotypes of the ponderous process models in software development are the classical sequential process model, the waterfall model, and the V-model. In contrast, the *lightweight* process models, also called agile process models, are more flexible, less formalized, iterative process models, e.g., *eXtreme* programming. *Scrum* and the *spiral model* can be considered as intermediate process models according to the above-introduced classification. In agile software development methods, complete specifications at the beginning of a project often are not available and are not considered to be important because they are assumed to change throughout the process anyway. To compensate the absence of neatly documented specifications, constant exchange between team members is crucial (cf. Figure 1-5) (POMBERGER and PREE, 2004: p. 45; STEPHAN, 2013: p. 32). The most prominent software development processes and methods will be described in the following, starting with the classical and advancing to the agile ones bearing in mind that some methods cannot be classified clearly.

The phase model describes the typical activities in software development, which is divided into four major phases and their immanent questions. The fundamental approach it is presented here (EIGNER et al., 2012a: pp. 161–162; EIGNER, 2014d: pp. 32–34):

1. Requirements analysis: The goal is to have completed as many as possible complete of the requirements.
2. System design: The software-oriented system design aims at describing an abstract solution plan for the problem. Based upon the requirements analysis, it is noted of which components the system is composed.
3. Detailed design: Itemization of the plan for the problem solution from the system design is in scope of this step.
4. Encoding and integration: The last phase's scope is the implementation of the complete solution.

The realized partial solutions of individual components will be integrated into an overall solution (EIGNER et al., 2012a: pp. 161–162; EIGNER, 2014d: pp. 32–34). The major phases of the phase model of software development, as described above, have an

apparent resemblance with the process steps of model-based systems engineering in Chapter 2.4 and are depicted in Figure 2-6.

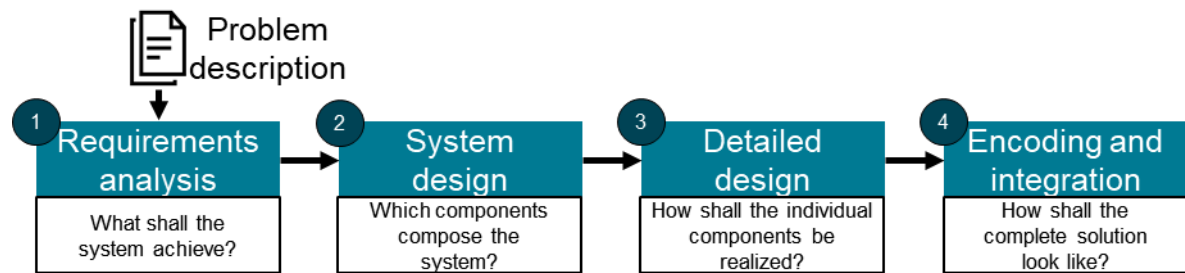


Figure 2-6: Phase model of software development (in alignment to BOEHM, 1979: p. 4; EIGNER et al., 2012a: p. 162; EIGNER, 2014d: p. 33).

The waterfall process model of software development is based on the assumption that a phase is not passed through anymore after its finalization. This implies that a phase shall not be started before the preceding phase has been terminated. This means that after finalization of the requirements analysis, in Figure 2-6 phase 1, requirements have to be specified faultless. Due to a high degree of volatility within the specification of requirements, in practice, the waterfall model proves to be improper. With this fundament, further iterative process models have evolved. A very famous one is the “spiral model of software development and enhancement” by BOEHM (1988). The *spiral model* aims at representation and mitigation of development risk in the software development process by depicting each development round (analysis, evaluation, development and tests, planning) as one round in a spiral, visualizing increasing cost and complexity (EIGNER, 2014d: pp. 34–36). The documents and product model of software development is based on the fundamental work of BOEHM (1979), who introduced a V-shaped view on the software development process, and this was extended and adapted frequently. This so-called *V-model* also was adopted by the VEREIN DEUTSCHER INGENIEURE (2004b) and adjusted to the development of mechatronic products (EIGNER et al., 2012a: p. 162). Therefore, the *V-model* will be discussed in more detail in the next section *MECHATRONICS*.

Against this background, agile methods for software development were created and first adopted by the software community, and to an increasing degree later on, by commercial software development (FELDHUSEN et al., 2013: p. 808). Most of these agile software development methods are based upon the *agile manifesto* and its twelve

principles²⁹ (BECK et al., 2001). An agile approach is commonly characterized by the separation of the project into several stages of configuration or iterations of the product, respectively. The focus is on the main criteria in order to quickly serve customer's demands³⁰ (FELDHUSEN and GROTE, 2013: p. 809).

MECHATRONICS

In 1969, Ko Kikuchi coined the term *mechatronics*, a combination of mechanics and electronics. Hereby, he meant the increased electrical and electrotechnical functionality of mechanical components and devices. Software only became relevant later, as depicted in Figure 2-7 (COMERFORD, 1994: p. 46; HARASHIMA et al., 1996: p. 1; STEPHAN, 2013: p. 16; EIGNER, 2014d: p. 42).

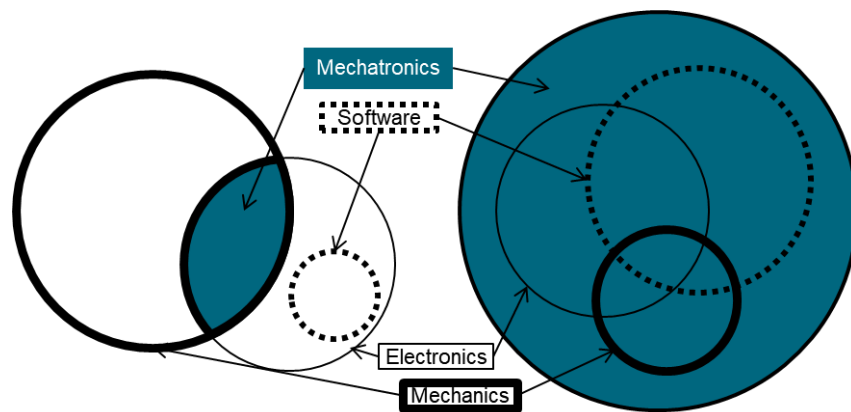


Figure 2-7: Alteration of the term “mechatronics” (in alignment to EIGNER et al., 2012a: p. 34; STEPHAN, 2013: p. 17; GROLL and HEBER, 2016: p. 291; EIGNER, 2014d: p. 43; BERTSCHE et al., 2009: p. 3).

Mechatronic systems consist of a mechanical basic system, sensors, actuators, and information processing (cf. Figure 2-8). Hence, all different domains discussed above, mechanics, E/E, and software, are present. The initial goal of a mechatronic system is to improve the functionality and behavior of the underlying mechanical basic system by means of sensors that register information of the environment and the system itself. This information is manipulated in processors which trigger the optimal reactions by means of actors in the respective context. Through intelligent software, mechatronic systems nowadays are capable of adapting themselves to changes in the environment, detect critical operational states, and optimize processes (VEREIN DEUTSCHER INGENIEURE, 2004b: p. 10).

²⁹ Please refer to BECK et al. (2001) for all twelve principles.

³⁰ For further literature regarding software development and a more detailed view on the above-mentioned process models please refer to EIGNER et al. (2012a), EIGNER (2014d), POMBERGER and PREE (2004), STEPHAN (2013), DÖRN (2018).

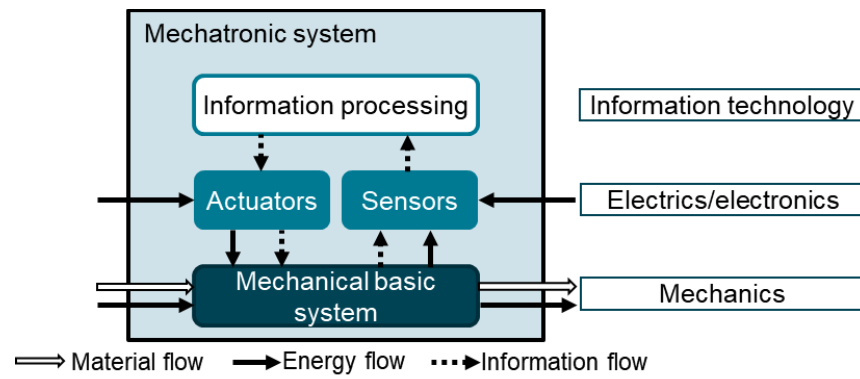


Figure 2-8: Basic structure of a mechatronic system (in alignment to VEREIN DEUTSCHER INGENIEURE, 2004b: p. 14; PONN and LINDEMANN, 2011: p. 12; STEPHAN, 2013: p. 18).

TOMIZUKA (2000) extends the mechatronic system by “complex-decision making in the design, manufacture and operation of industrial products and processes” (TOMIZUKA, 2000: p. 1). However, the guideline *VDI 2206* “Design methodology for mechatronic systems” limits the scope of mechatronics to the definition: “[Mechatronics is] the synergetic integration of mechanical engineering with electronic and intelligent computer control in the design and manufacturing of industrial products and processes”³¹ (HARASHIMA et al., 1996: pp. 1–2; VEREIN DEUTSCHER INGENIEURE, 2004b: p. 14).

There exists a plethora of different process models and methods to develop mechatronic products or systems (VEREIN DEUTSCHER INGENIEURE, 2004b: p. 14; STEPHAN, 2013: p. 21; EIGNER, 2014d: p. 44).

The *V-model*, designed as a process model for software development, which was frequently extended and adapted further, became a widely used process model for the development of mechatronic systems. The most common *V-model* for mechatronic systems is in the guideline *VDI 2206* (VEREIN DEUTSCHER INGENIEURE, 2004b). BENDER (2005) extended the *V-model* further by including three levels, namely the system level, subsystem level, and component level in the development process of mechatronic systems (BENDER, 2005: p. 45). If the automotive industry is in scope, then, additionally, the fourth level “vehicle level” as an overall system for other mechatronic systems is appended at the beginning and the end of the development cycle. The left wing of the *V-model* describes the interdisciplinary system development. The four different levels of the mechatronic system – vehicle, system, subsystem, component – each address differently the granularity of system description. Aligned are description elements –

³¹ Due to three involved disciplines, complex products, and a long evolution of mechatronic systems, there exist a myriad of definitions. Please refer to VEREIN DEUTSCHER INGENIEURE (2004b), EIGNER et al. (2014), STEPHAN (2013), and EIGNER et al. (2012a) for an overview.

requirement, function, element of the logical architecture, physical element (RFLP³²). The component level is dedicated to discipline specific itemization (mechanics, E/E, software). The right wing of the *V-model* describes the steps for system integration and testing by means of virtual, hybrid, or physical tests. On each level horizontally from the right to the left wing, there is validation of requirements and specifications as described in previous phases³³. The *V-model* by BENDER (2005) allows for an iterative proceeding, similar the *V-model* in the guideline *VDI 2206* (STEPHAN, 2013: pp. 44–46; VEREIN DEUTSCHER INGENIEURE, 2004b: pp. 26–31; ZAFIROV, 2014: pp. 85–87; PEARCE and HAUSE, 2012: p. 10). The *V-model* as described is depicted in Figure 2-9. It serves as fundament for model-based systems engineering in Chapter 2.4.

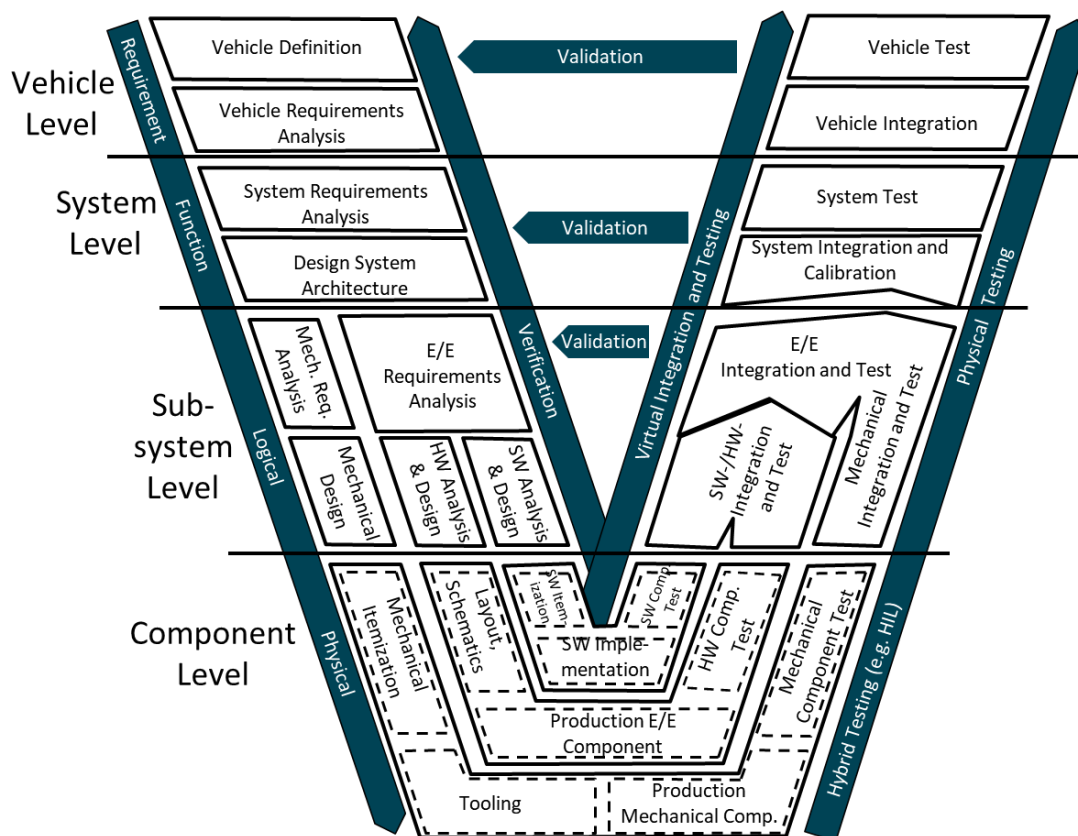


Figure 2-9: The V-model of mechatronic system development (in alignment to VEREIN DEUTSCHER INGENIEURE, 2004b: p. 29; BENDER, 2005: p. 45; GROLL and HEBER, 2016: p. 291; EIGNER et al., 2012b: p. 1670; ZAFIROV, 2014: p. 87).

³² Cf. Chapter 2.4, HORVATH (2017), and HORVÁTH and RUDAS (2015) for more information about the RFLP approach.

³³ Cybertronic systems consist of at least two cybertronic elements. A cybertronic element is, in turn, a mechatronic system with the ability for communication in open networks and the be part of a cybertronic system. Hence, simplified, cybertronic systems are mechatronic systems that are enabled to communicate via open networks for the purpose of joint cooperation (CADET and MEISSNER, 2017: pp. 19–20; ZAFIROV and ROUBANOV, 2014: p. 139). Cybertronic systems are not in scope of this elaboration.

2.2.3 Idiosyncrasies of automotive electric/electronic product development

As depicted in Figure 1-3, complexity in automotive development steadily increased in recent years. It can be distinguished between a risen complexity concerning the product, i.e., technical complexity, as well as complexity regarding the development process, i.e., organizational complexity. The number of E/E components, such as ECUs, actuators, and sensors, has increased. Due to this, the total length of automotive wiring harness has augmented drastically. Organizational complexity in automotive E/E development stems from the fact that an automobile is a complex product including multiple involved development domains that are highly integrated and interactive (cf. Chapter 1). Engineers in these development domains use different tools, processes, methods, and even specific vocabulary (BIAHMOU, 2015b: p. 222).

In order to alleviate and address technical and organizational complexity, the application of auxiliary means is indispensable. In E/E development, there has to be a distinction between three different classifications of development objects which differ by their degree of granularity. A high-level development object means a low level of granularity and vice versa. These classifications are relevant for specific IT tools and the field of application of computational E/E development, i.e., electrics/electronics computer aided design (E-CAD). These three levels of development objects align to the definition of mechatronic systems (cf. Figure 2-8). Hence, E/E design, conductor plate design, and chip design are the three relevant levels for E/E development. Inherent to these levels are different approaches, methods, and IT tools (ZAFIROV and ROUBANOV, 2014: pp. 138–141). Mastering this complexity, cost pressure, a desire to offer automobiles worldwide, and legal regulations, amongst others, forced automotive manufacturers and suppliers eventually to pursue standardized solutions for E/E development. This included communication bus systems, common software platforms for ECUs, and protocols for data exchange (ZIMMERMANN and SCHMIDGALL, 2014: p. 1; ROBERT BOSCH GMBH, 2014: p. 11). Additionally, in the automobile industry there is an increasing variance of variants. Simultaneously, there is a decreasing quantity per variant, economic necessity to develop these variants on common platforms, shorter lifecycles, and a high volatility in E/E and software due to a high demand from customers for new features (cf. Chapter 1). This also contributes to a relatively high and continuously increasing complexity in automotive E/E development (EIGNER and STELZER, 2009: pp. 11–13; HARMS, 2009: p. 39).

As an automotive manufacturer seldom is the producer of integrated circuits and conductor plates, the main task for engineers in E/E development for an automobile is the design of electrical wiring that is necessary, in order to display all connectors and pins of an ECU, as well as which communications messages are transported on which bus system (cf. Chapter 2.5). For that purpose, collaborative engineering (cf. Chapter 2.6) by usage of team data management (TDM) (cf. Chapter 2.3) is necessary to integrate different E/E development disciplines, different organizational departments, and development activities scattered around the globe. The fundamentals for engineering collaboration concerning IT architectures will be given in Chapter 2.3. The technical complexity of joint development by means of MBSE will be described in Chapter 2.4. A closer look will be taken on automotive E/E in Chapter 2.5, and a concise overview with of engineering collaboration in Chapter 2.6 in order to address organizational complexity.

2.2.4 Traceability in the context of product development

Product development is a challenging endeavor and differs from discipline to discipline by processes, methods, organization, data models, and IT tools. This makes the identification of traceable artifacts difficult so as to enable and foster traceability.

DATA MODEL

During product development, data models are created which can be modeled accordingly to implement traceability in product data across IT tools and IT databases throughout the entire lifecycle. Traceability in product development has a crucial significance and realization of it is decisive in modern development of complex, interdisciplinary products and processes. Additionally, the work results and deliverables of those processes have to be capable of being integrated and machine-readable across different IT tools, disciplines, and IT databases. This can be achieved by one or many collective data models for mechatronic systems. Depicting traceability in engineering is often done using graph-based modeling and graph theory³⁴.

PROCESS MODEL

Product development occurs in an early phase of the product lifecycle where changes in the design of a product does not yet affect costs as severely as in later phases (EIGNER and STELZER, 2009: pp. 15–16). However, if changes have to be made in later phases

³⁴ Please refer to ZAWIŚLAK and RYSIŃSKI (2017) for more information about graph-based modeling in engineering.

of the product lifecycle, as well as during the product development itself, ramifications with other (sub-)products have to be traceable. Furthermore, processes of different disciplines involved, as depicted in Figure 2-9, have to be aligned. This means that the organizational structure intra- and inter-company shall foster collaboration in development. When changes during development occur, the processes have to propagate them to all affected domains and relevant engineering partners.

TECHNOLOGY

Above it was mentioned that a distributed product development process states high requirements on the integration of IT solutions, i.e., the technology that enables distributed engineering collaboration. Often, the dedicated development tools for a specific domain offer very good traceability within their tool platform. This means, that information artifacts can be easily traced within a software development platform and across single instances of development platforms or tools of one vendor. However, some vendors use proprietary data formats and interfaces which impedes traceability in product development and engineering collaboration. For the development of 3D mechanical parts (CAD), the tool *CATIA* by *Dassault Systèmes* and *NX* by *Siemens* shall be mentioned, inter alia³⁵. However, as the focus of this elaboration is on E/E and software development as well as MBSE, only these specific tools will be highlighted further (please refer to Chapters 2.4 and 2.5).

2.3 Product data management and product lifecycle management

2.3.1 Definitions, norms, and standards

Already during the years of 1980, the first PDM IT systems were available. At that time, the focus of those IT systems was to provide an instrument for document management in CAD and enterprise resource planning (ERP). PDM is defined as the management of a product and process model with the aim to create distinct and reproducible product configurations (EIGNER and STELZER, 2009: p. 34). However, the typical area of application was restricted to department-specific activities of development and design (EIGNER and STELZER, 2009: p. 27). Consequentially, through usage of PDM across the entire product lifecycle, different specifications of product structures occurred inevitably along the individual product lifecycle phases. According to EIGNER and STELZER (2009),

³⁵ Please refer to SENDLER (2009), VAJNA (2009), and KÖNIGS (2013) for more information about CAD development tools.

the product lifecycle phases are depicted in Figure 2-10 (EIGNER and STELZER, 2009: pp. 16, 20, 28).



Figure 2-10: Product lifecycle phases (in alignment to EIGNER and STELZER, 2009: pp. 16, 20, 28).

In contrast, STARK (2016) only defines five phases of a generic product lifecycle. Those are *Imagine*, *Define*, *Realize*, *Support/Use*, and *Retire/Dispose* which can be considered as superordinate categories of the above mentioned phases (STARK, 2016: pp. 3–4).

The product model within the definition of PDM aims at the digital reproducibility of products and the information which is relevant for the lifecycle. A process and its model delineate the technical and organizational sequence of business. If one combines the functions of the product and process model, this yields the configuration model. The configuration model integrates all relevant information with regard to content, status, or version (EIGNER and STELZER, 2009: pp. 26–30). Configurations, their models, and their management will be discussed in Chapter 2.3.2. Configuration management is also one of the core functionalities of PDM systems. However, sometimes it is not fully implemented³⁶ (EIGNER and STELZER, 2009: p. 35).

Definitions of the functionalities of PDM systems, their names, and their location in one or more different IT systems differ (STARK, 2016: p. 233). STARK (2016) also gives an overview of eight more generic components of a PDM system that mainly align with the above mentioned (STARK, 2016: pp. 233–243).

In order to integrate data from PDM systems universally, i.e., along all lifecycle phases, across all organizational divisions, and all domains involved with the product, for instance development, production, after sales, the PLM concept was introduced. Moreover, legal requirements towards reproducibility and traceability demanded a higher degree of integration of all IT systems along the product lifecycle. By means of a continuous configuration management, PDM becomes the backbone of a complete lifecycle management. There is a plethora of names and definitions for this, yet PLM became accepted and prevails. What all definitions of PLM have in common is that the

³⁶ Please refer to EIGNER and STELZER, 2009: pp. 35–36 for more information about the common functionalities of a PDM system.

scope of PLM in comparison with PDM is a broader application. Also, a higher degree of integration of multiple IT systems across all phases of the product lifecycle and the process of the supply chain is part of PLM. This is to manage all parts and products and the entire portfolio of a company (EIGNER and STELZER, 2009: pp. 36–37; STARK, 2016: p. 2). The PLM concept combines functionalities to manage or execute for instance objectives and metrics, management and organization, people, methods, facilities and equipment, other PLM applications, PDM system, product data, processes, products, and the lifecycle³⁷ (STARK, 2016: p. 5). The distinction of PDM and PLM, their allocation within the product lifecycle, and major stakeholders and their corresponding IT systems are displayed in Figure 2-11 (in alignment to EIGNER and STELZER, 2009: p. 37; EIGNER et al., 2014: p. 270).

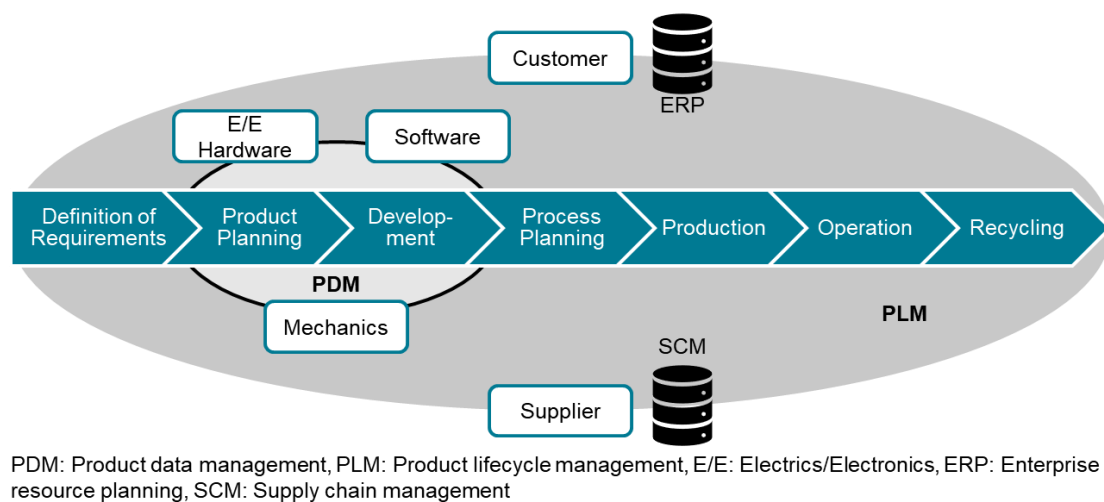


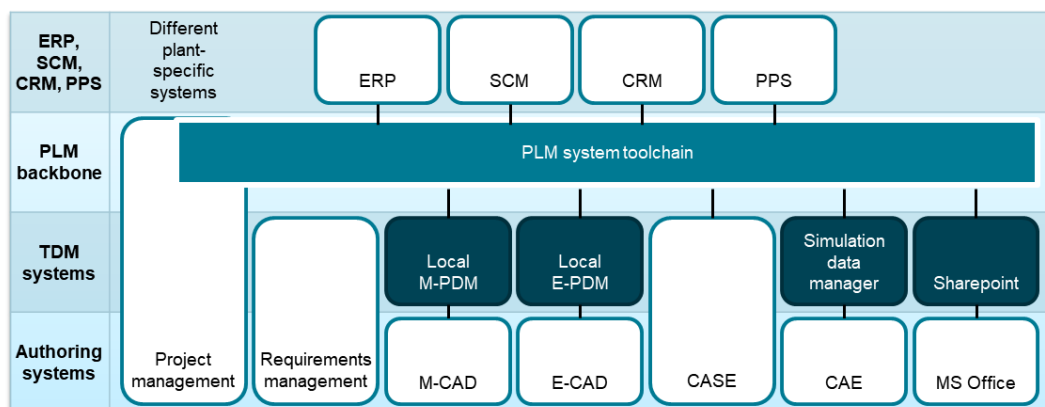
Figure 2-11: Location of PDM and PLM with respect to the product lifecycle (in alignment to EIGNER and STELZER, 2009: p. 37; EIGNER, 2014b: p. 270).

As stated above, IT systems executing PLM functionalities often serve as an integrational solution in terms of processes, data models, methods, organizational aspects, etc. in the interaction with multiple other IT systems across the entire product lifecycle in a company's IT infrastructure. Hence, these IT systems used for PLM often are referred to as so-called *engineering backbones* or *PLM backbones*. This denomination shall reflect the functional, technical, and process-related connection a company-wide PLM system toolchain enables (EIGNER and STELZER, 2009: pp. 43–44; MÜLLER et al., 2017: p. 193; EIGNER, 2014b: p. 280).

³⁷ For more information regarding functionalities of a PLM system, please refer to STARK (2016), EIGNER and STELZER (2009), KIRSCH et al. (2017a), BUHL et al. (2001).

In a typical four-layered PLM architecture (cf. Figure 2-12), as encountered in automotive, aerospace, and high-tech industries, the PLM backbone is situated between the ERP layer and the TDM layer (EIGNER et al., 2014: p. 280). As the functionalities of PLM systems also comprise PDM functionalities or systems (cf. Figure 2-11), hereafter, PDM/PLM will be used together where a separation of terms is not necessary, e.g., in case when the management of bill of materials (BOMs), technical master data, change, release, and configuration management is considered which is also part of the PLM backbone (cf. EIGNER and STELZER, 2009: p. 44). Also, focus in this work is on the early engineering phase where PDM functionalities and processes dominate as downstream processes for PLM sometimes have not started yet.

At the very bottom of the four-layered PLM architecture in a company's IT landscape, there usually are the domain-specific authoring systems. In the case of simultaneous engineering, those authoring systems can be used in parallel as well as sequentially. Data is created within the authoring tools, e.g., data such as CAD models, schematic layouts, source code, calculations, simulations, etc. Examples for domain-specific authoring systems can be for the support of requirements management, M-CAD, E-CAD, computer aided software engineering (CASE), CAE, as well as Microsoft Office programs and project management tools (EIGNER and STELZER, 2009: p. 43; EIGNER et al., 2014: p. 280).



ERP: Enterprise resource planning, SCM: Supply chain management, CRM: Customer relationship management, PPS: Production planning and control system, PLM: Product lifecycle management, TDM: Team data management, M-PDM: Mechanics product data management, E-PDM: Electric/Electronics product data management, M-CAD: mechanics computer aided design, E-CAD: electric/electronics computer aided design, CASE: computer aided software engineering, CAE: computer aided engineering, MS Office: Microsoft Office

Figure 2-12: Typical four-layered PLM architecture with a central PLM backbone (in alignment to EIGNER and STELZER, 2009: p. 43; EIGNER, 2014b: p. 280).

The next layer manages the created data close to the authoring systems. Tools and systems installed at this layer commonly are referred to as TDM tools or systems and handle data in the native format of the authoring systems. Separation of disciplines, product lines, and organizational entities by means of this layer contributes to the reduction of overall complexity of big, worldwide operating companies' IT landscapes. In the development phase, TDM systems can be local PDM systems. If authoring systems are simple, the TDM layer can be omitted and authoring systems are connected directly to the PLM system. Often, TDM and authoring systems are combined by tool vendors (EIGNER et al., 2014: p. 280; EIGNER and STELZER, 2009: pp. 43–44).

The PLM backbone is the next layer. The PLM backbone usually comprises of multiple IT systems and includes the mechatronic product structure and all corresponding documents, which are commonly available in neutral data formats to improve transfer of those across all tethered IT systems. Most importantly, the PLM backbone includes the configuration and change management and by that, is the actual PLM solution layer. Additionally, further central processes, for instance release management, visualization, and archiving, usually are implemented in this layer. This guarantees a worldwide access to all technical master and structural data with all configurations. From this layer occurs the transfer of information which is relevant for production to the plant-specific ERP systems (EIGNER et al., 2014: p. 280; EIGNER and STELZER, 2009: p. 44).

The uppermost layer commonly is an ERP, supply chain management (SCM), production planning system (PPS), or customer relationship management (CRM) system. There, the logistical and production-related parts of change and configuration management are executed. Plants often have their own local ERP and PPS systems to adapt flexibly to local production conditions (EIGNER et al., 2014: p. 280; EIGNER and STELZER, 2009: p. 44).

As stated above, configuration management is essential to the PLM backbone systems as a solution layer to connect product structures and documents in a native data format across many IT systems and along the product lifecycle. Therefore, the following chapter will scrutinize configuration management further.

2.3.2 Configuration management

A configuration is defined in *ISO 10007:2017* as “interrelated functional and physical characteristics of a product or service” and is delineated in the configuration information, which are “requirements for product or service design, realization, verification, operation

and support” (INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2017a: p. 1; INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, 1983: p. 13). This norm aligns to *ISO 9001:2015* in which traceability is required to “control the unique identification of the outputs” (INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2015a: p. 41) and to meet traceability requirements and product identification (INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2015a: p. 60). Hence, a configuration is a description of a product or output at a certain point in time or in a defined status of delivery, respectively, and includes all relevant information, such as the product structure including software components in the form of bills of materials (EIGNER and STELZER, 2009: p. 113). Accordingly, configuration management (CM) is a discipline, which has the scope to track and monitor a product’s functional and physical characteristics across its lifecycle. CM serves to establish integrity, reproducibility, traceability, availability, and consistency of configuration items (CIs), can be considered as the logical consequence of an integrated implemented product and process management, and is essential to systems and software engineering³⁸ (GRANDE, 2013: pp. 8–10; SCHULTE et al., 2017d: p. 326; EIGNER and STELZER, 2009: pp. 33, 112–113; INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2017a: V; INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, 2012: VIII; FELDHUSEN and GROTE, 2013: pp. 792 ff.; EIGNER and STELZER, 2009: p. 115; SCHULTE et al., 2017d: p. 328; WALLMÜLLER, 2011: pp. 339–340). The *CMII Standard for Enterprise-Wide Configuration Management and Integrated Process Excellence* (CMII) strives for an efficient CM process as well as for integrated process excellence. CMII is an incremental approach for the improvement of business processes (MECPRO² ABSCHLUSSBERICHT, 2016c: p. 170; INSTITUTE OF CONFIGURATION MANAGEMENT AND CMII RESEARCH INSTITUTE, 2014: pp. 2, 10). In comparison with a change process in software development, the CMII, aiming at mechanical development, can be considered rather cumbersome. This is due to many different roles, workflows, and a change review board are involved (PFENNING, 2017: pp. 29–30). This is in contrast to objective 2.a. which postulates the reduction of reconciliation, to have earlier and faster reconciliation of changes.

Additionally to manage changes to a configuration, it is decisive to align specific versions and their appurtenant configurations (INTERNATIONAL ORGANIZATION FOR

³⁸ For further classification of configuration items, such as technical, contractual, and serialized configuration items, please refer to EIGNER and STELZER (2009) and EIGNER et al. (2014). Here in this elaboration, the above-mentioned, generic definition of configuration item is used.

STANDARDIZATION, 2003: p. 14). Hence, version control or version management is part of CM. Also, CM can be considered as a superordinate concept for or advancement of release and change management, as well as build, baseline, version including variant, and audit management (KIRSCH et al., 2017d: p. 334; GRANDE, 2013: pp. 14–16; EIGNER and STELZER, 2009: p. 112). A configuration baseline is a “frozen”, dedicated state of product description in development status that is used for an internal or external transition³⁹. The five major components of CM are displayed in Figure 2-13. Moreover, configuration management supports the intra- and interdisciplinary collaboration and builds a foundation for communication between design engineers and all other involved parties in the engineering process (WATTS, 2011: p. 4).

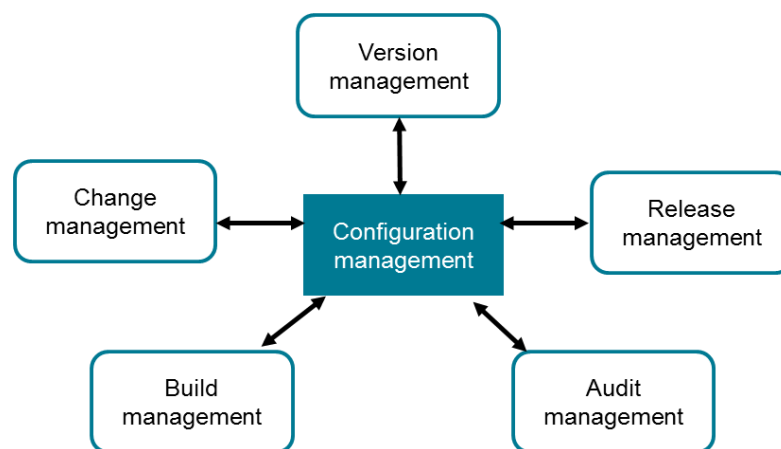


Figure 2-13: The five major aspects of configuration management (in alignment to GRANDE, 2013: p. 16; INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2015c: p. 40; KIRSCH et al., 2017a: p. 157).

Change, version, and variant management will be described in more detail as they are relevant for understanding of the solution framework and its building blocks.

CHANGE MANAGEMENT

Change management results from necessary fixes or new requirements to optimize the product with the purpose of evaluation and decision, whether changes should be included and if so, which artifacts will be affected at which costs and expenditure of time (INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2017b: p. 66; GRANDE, 2013: p. 15; EIGNER, 2014c: pp. 253, 255). Sub-steps of change management also prevail outside of a company, e.g., at a supplier or engineering partner, if this particular change requires a consensus (STARK, 2016: p. 330; EIGNER, 2014c: 253).

³⁹ Please refer to EIGNER (2014c) and GRANDE (2013) for more information regarding baselines.

According to a taxonomy⁴⁰, there exists at each point in time one or more assemblies of items, i.e., configurations of a product, that are effective (EIGNER, 2014c: pp. 256, 259, 260). This so-called *effectivity* is a major part not only of change management but also of CM. Effectivity in the context of CM describes a validity period of a configuration or the over the time changing product status, respectively, and can be modeled by an CI (INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2012b: p. 97; EIGNER, 2014c: pp. 262–263; ŞENALTUN and CANGELIR, 2012: p. 371). Effectivity enables the distinct identification of a product and documentation configuration that can be restored in a system at any point in time in order to foster traceability (EIGNER, 2014c: p. 263). An example for effectivity is shown in Figure 2-14.

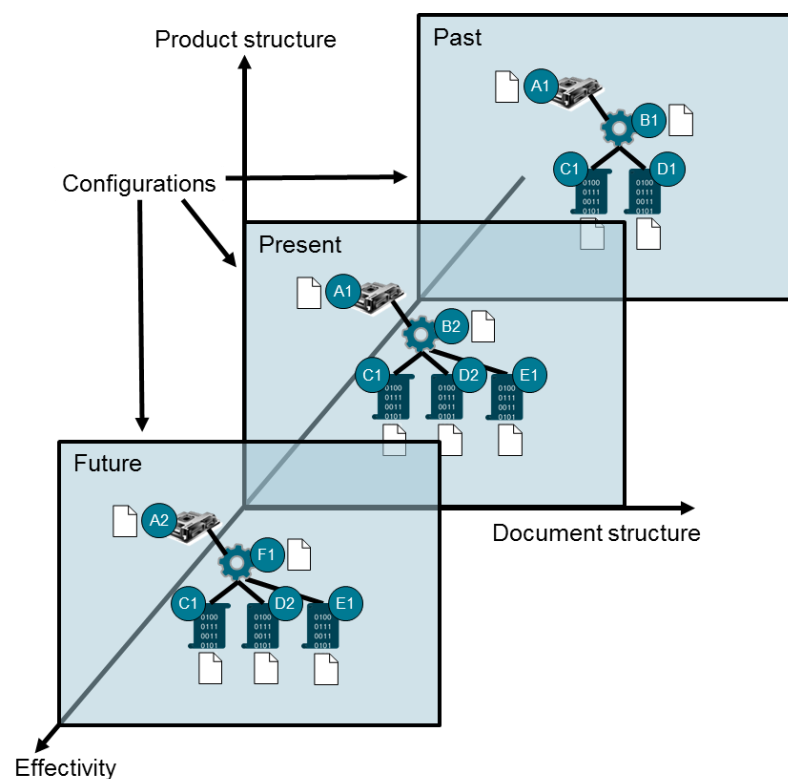


Figure 2-14: Effectivity in configuration and change management (in alignment to EIGNER and STELZER, 2009: p. 118; EIGNER, 2014c: p. 263).

VERSION MANAGEMENT

Version management enables capture of all changes to objects in a lifecycle in its various different versions and, if necessary, to reproduce them (GRANDE, 2013: p. 15; KIRSCH et al., 2017d: p. 334). In software configuration and its version management

⁴⁰ A taxonomy is a directed model, i.e., terms at the top commonly are generic terms for or aggregations of terms at the bottom, that hierarchically orders terms and hence describes relations between superordinate and subordinate terms (DORSCHER, 2015: p. 317). For more information, please refer to Chapter 2.8.

there has to be a distinction between central (using centralized server to store version database) and distributed (storing repository of version database at client) version control systems (SCHULTE et al., 2017d: p. 326; GIFT and SHAND, 2009: p. 2). For the sake of completeness, the local version control on only one computer is mentioned here. This yields six different variations of version control systems (in alignment to KEYDEL and MEDING, 2008: pp. 230–231; CHACON and STRAUB, 2014: pp. 1–4; GIFT and SHAND, 2009: pp. 2–3):

1. Sole local version control system: This is the classical local version control system on one computer, which is very error prone and has its limitations in collaboration.
2. No local client and no common file system: The simplest variation for distributed development is to access a file server remotely, perform a check-out of a file, transfer it to your local computer, modify the software, transmit it back to the file server, and then, at last, check the altered software in again.
3. No local client but a common file system: This variation is a so-called centralized version control system where a version database is stored centrally at a file server and each computer or user transfers single files back and forth.
4. A local client but no common file system: Only very few version control systems' clients are capable of retrieving data from a server directly over the internet, e.g., with a network protocol or file transfer protocol and without proprietary software also on the file server side. Hence, this scenario is seldom used.
5. A local client and a common file system: Here, the client accesses the software repository of a common file system on the central file server. This facilitates providers of version control software implementation on local computers and file servers and fosters distributed software development and its version control. A major downside of this variation with a centralized version control system is the single point of failure with respect to the central file server that stores the only version database. In case of downtime of the file server or a corrupted database, further development might temporarily not be possible.
6. Distributed version control system: This variation recently prevails most often. Here, clients do not only check out the latest files, but they completely mirror the entire version database repository of the server. This makes the entire version control system resistant towards server failure, compromised data, and errors in the version database. Hence, those advantages are used by the today's most

popular distributed version control systems, for instance *Git* (KEYDEL and MEDING, 2008: pp. 230–231; CHACON and STRAUB, 2014: pp. 1–4; GIFT and SHAND, 2009: pp. 2–3).

The six different variations of version control in software development are displayed in Figure 2-15⁴¹. The insights from variation 6 are particularly relevant for distributed engineering collaborations (cf. Chapter 2.6) and the underlying data bases (cf. Chapter 2.7).

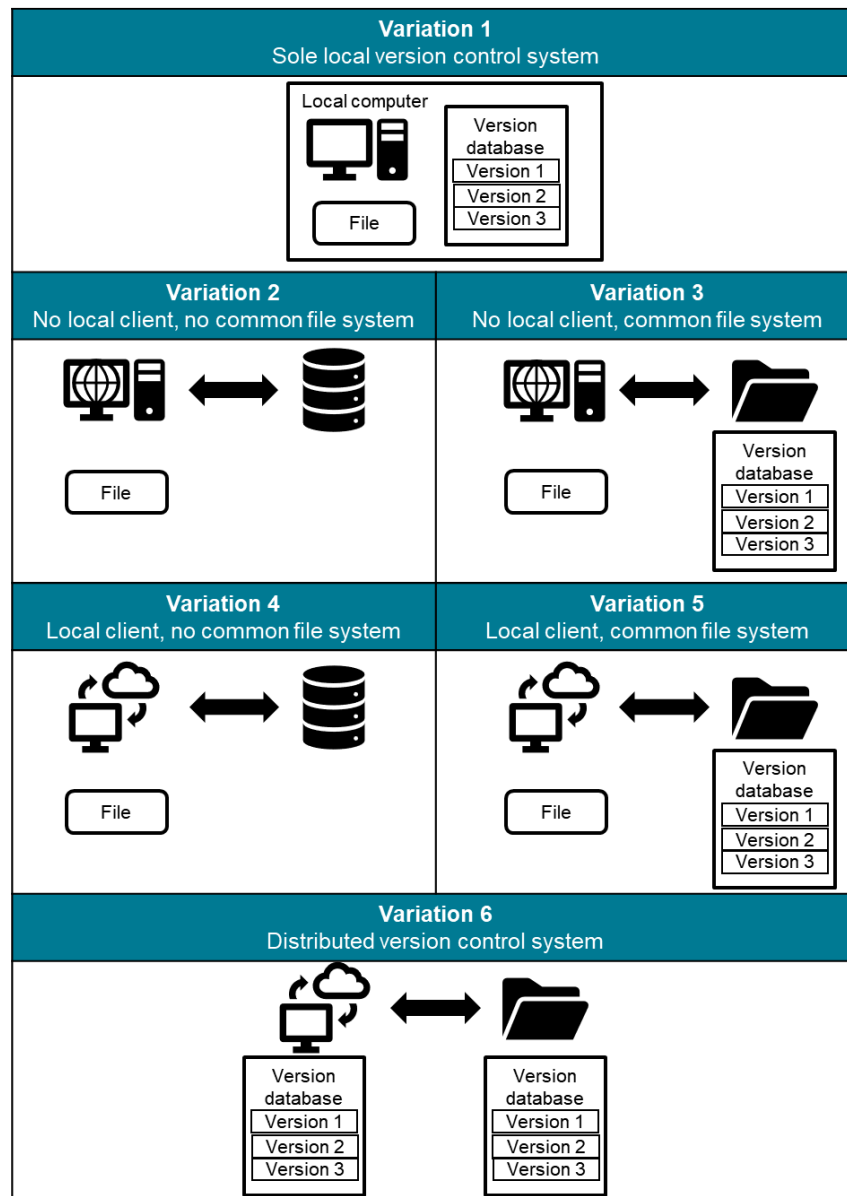


Figure 2-15: Different variations of version control in software development (in alignment to KEYDEL and MEDING, 2008: pp. 230–231; CHACON and STRAUB, 2014: pp. 1–4; GIFT and SHAND, 2009: pp. 2–3; GRANDE, 2013: p. 106).

⁴¹ For more information about version control in software development, please refer to CHACON and STRAUB (2014) and BRICOGNE et al. (2012).

VARIANT MANAGEMENT

As stated above, variant management can be considered part of version management within configuration management. A variant is a version of a product that is intended to coexist with other versions (BRUEGGE and DUTOIT, 2010: p. 562). Variant management is in the discrepancy between the economic necessity of as much common parts for many similar products as possible on the one hand and the desire to meet customers' requirements on the other (FELDHUSEN and GROTE, 2013: pp. 793–794; AVAK, 2006: p. 22).

There is an outer variance, which is noticeable as product variance by the customer. Conversely, the inner variance is the needed or used variance of parts, assemblies, and products as well as processes and resources for the realization of the outer variance by means of variation points (SCHULTE et al., 2017c: p. 262; KIRSCH et al., 2017a: p. 157; INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2015b: p. 6).

A reference or template product, i.e., a so-called “150% product” based upon a company's knowledge repository, marketing, product management, or strategy, often builds the basis or “platform” for the deduction of variants which are built from standard components, options, and individual solutions for customers⁴² (FELDHUSEN and GROTE, 2013: pp. 795–796; POHL et al., 2012: 170). The “150% product” describes all elements of a platform (main configuration) or, in other words, includes all standard features whereas only a subset of 100% of elements is used for one variant (CHADZYNSKI, 2022a: p. 309, 2022b).

The connection between configuration, version, and variant is displayed in Figure 2-16 using an example from the automotive industry showing coexisting variants and consecutive versions⁴³.

⁴² For more information about variant management, development of the reference product structure for reference variants for a product group, an implementation approach according the reference product structure, and its processes, please refer to FELDHUSEN and GROTE (2013). AVAK (2006), HARMS (2009), HASS (2003), WOSS (1997), and BRUEGGE and DUTOIT (2010) give more details on variant management. POHL et al. (2005) and DALGARNIO and BEUCHE (2007) show the distinction of problem space and solution space for each domain engineering and application engineering, where variability occurs, and is managed. Variant management drivers, complexity, cost, and examples from automotive industry are presented in ELMARAGHY et al. (2013).

⁴³ Also cf. SCHÄUFFELE and ZURAWKA, 2016: p. 199.

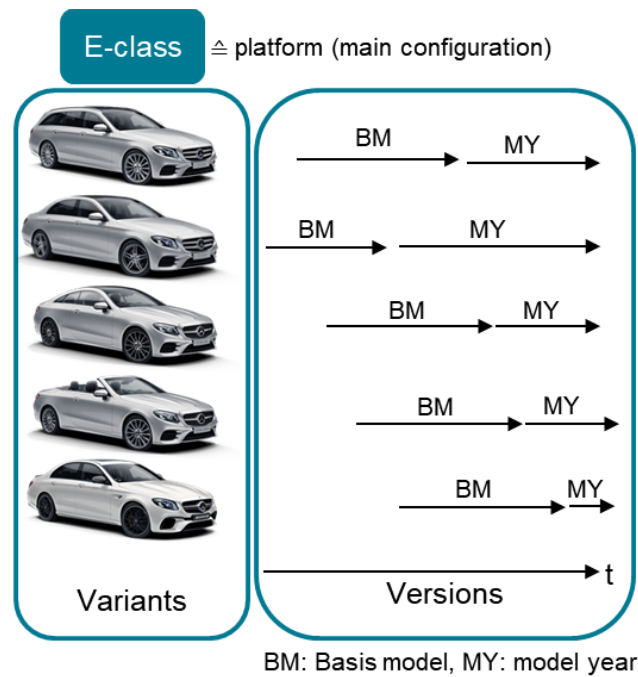


Figure 2-16: Connection between configuration, coexisting variants, and sequential versions.

2.3.3 Traceability in the context of PDM/PLM

When considering PDM and PLM, traceability is the fundament in today's product development and lifecycle for complex products in the realm of E/E and mechatronics.

DATA MODEL

Due to different disciplines develop and process data in a PDM/PLM IT landscape, a common language, i.e., a shared data model for the main business objects, is crucial. There exist different peculiarities of which data is stored where. For instance, a PLM system can hold all technical data, or only construction data, or intermediate data handling in reciprocity with an ERP system. The extent of PLM functionalities define where and how the different model structures and BOM, e.g., CAD model structure, the engineering BOM (E-BOM), and the manufacturing BOM (M-BOM), are handled (EIGNER and STELZER, 2009: pp. 301–309). Standardized data exchange formats as well as standardized APIs are key for a PDM/PLM integrated network (SINDERMAN, 2014: pp. 327–347).

PROCESS MODEL

As PDM and PLM are management methods to administrate product data, distinct process models are inherent. Without these process models, for example configuration management and all its subprocesses where certain items of a product are managed in order to monitor their characteristics (cf. Chapter 2.3.2), traceability would not be

feasible. Particularly, traceability would not be feasible for complex products in distributed engineering with multiple parties involved. Hence, the capability of configuration management and its appurtenant aspects (cf. Figure 2-13) has to be ensured in the context of PDM/PLM processes to foster traceability.

TECHNOLOGY

An interconnected and linked technology, i.e., IT systems, authoring tools, and TDMs in the context of PDM/PLM, is decisive. As described in Chapter 2.3.1, PLM backbone systems between the IT systems in product development and further downstream processes serves as a pivotal integration point with dedicated APIs for different data and process models in discipline-specific IT systems and tools. TRIPPNER et al. (2015) show the complexity in IT architecture at BMW AG. Figure 2-12 and STARK (2015) show the complexity generically (STARK, 2015: pp. 184–185). Standardized and open APIs and the reduction of IT systems are the main approaches in industry to manage or master this kind of complexity (PROBST, 2010: p. 13; EIGNER et al., 2016b: p. 59; BITZER et al., 2018: p. 351). Additionally, especially with respect to software development but also generally for distributed engineering and the respective PDM/PLM systems, it is decisive to have a jointly accessible version control system (cf. Figure 2-15). KÖNIGS et al. (2012) mention popular PDM/PLM IT systems which provide basic means to establish traceability of information artifacts. V6 by *Dassault Systèmes* offers the RFLP approach⁴⁴ to foster traceability by means of trace links across all development phases that are stored in one single PDM system (KÖNIGS, 2013: pp. 45–46; KÖNIGS et al., 2012: p. 930; SENDLER, 2009: p. 153). *Dassault Systèmes' 3DEXPERIENCE platform* is the successor of V6 and also combines all relevant PDM/PLM functionalities and, additionally, provides the open services for lifecycle collaboration (OSLC)⁴⁵ (DASSAULT SYSTÈMES, 2014, 2018a: pp. 11–12, 2018b: p. 6, 2020). The IT system *Teamcenter* by *Siemens* provides interconnected PDM/PLM functionalities to enable traceability across development phases and domains using URL hyperlinks. However, OSLC only is supported partially (KÖNIGS, 2013: p. 48; KÖNIGS et al., 2012: p. 930; SIEMENS INDUSTRY SOFTWARE INC., 2019, 2020a, 2020b; SENDLER, 2009: pp. 189–190; PROSTEP IVIP E.V., 2020b). *PTC's Windchill PDMLink* is an extensive PDM/PLM tool, offering OSLC link creation as well as OSLC link inclusion in its latest version, handling of system elements within the BOM, and fosters traceability via trace links between parts in *PDMLink* and

⁴⁴ Cf. Chapter 2.4 for more information about the RFLP approach.

⁴⁵ Please refer to Chapter 2.8 for more information about OSLC.

model system blocks in *Windchill Modeler*⁴⁶ (SODIUS CORP., 2020; PTC INC., 2020a, 2020c; OLLERTON, 2016: p. 4). The *Aras Innovator*, a PDM/PLM platform by Aras, applies the RFLP approach (REARDON, 2016; PFENNING, 2017: p. 94). In combination with an adapter, OSLC can be used together with *Aras Innovator* (PROSTEP INC.).

2.4 Model-based systems engineering

2.4.1 Definitions, norms, and standards

A system constitutes of a quantity of elements or sub-systems with specific properties that are linked to each other. System boundaries and the environment confine a system and, in case of an open system, the system interacts with its boundaries and environment (EHRENSPIEL and MEERKAMM, 2017: p. 28). A system's elements interact and are organized to achieve a purpose (INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2017b: p. 449). Systems often are depicted using an architecture description. An architecture in the system's context describes general properties of a system through its elements, their relationships, and embedded in its environment (INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2011f: p. 2).

In order to develop systems, MBSE enhances previous methods by the creation of a system model in the early concept phase. Here, collaboration between involved disciplines shall not occur via documents, as in systems engineering⁴⁷, but rather by means of centrally available and up to date, semantically rich models. Hence, MBSE is the formalized application of model creation to support activities of requirements engineering, development, verification and validation⁴⁸ of a system, from the conceptual design phase throughout later phases in the lifecycle^{49,50} (TECHNICAL OPERATIONS INTERNATIONAL COUNCIL ON SYSTEMS ENGINEERING, 2007: p. 15; ZAFIROV, 2014: p. 81). According to IEEE COMPUTER SOCIETY (2007), the basic building blocks of a system are

⁴⁶ For more information about PTC's *Windchill Modeler*, please refer to Chapter 2.4.3.

⁴⁷ For an overview of the transition from systems engineering as document-based discipline to model-based systems engineering as model-driven discipline please refer to ESTEFAN (2008), TECHNICAL OPERATIONS INTERNATIONAL COUNCIL ON SYSTEMS ENGINEERING (2007), WALDEN et al. (2015), BUEDE (2009), and NASA (2007).

⁴⁸ The purpose of verification is to provide an objective confirmation that specifications are met, i.e., that the "product is built correctly". Validation on the other hand means that it has to be clarified whether a product in use meets the customer's requirements, i.e., that the "right product was built" (BOEHM, 1979: p. 3; INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2015c: pp. 70–75).

⁴⁹ An engineering process is model-based when its description is based upon a formal language. A formal language is an abstract language with focus on the mathematical or physical application, such as a programming language (EIGNER, 2014a: p. 7).

⁵⁰ The conceptual design phase is located at the phase "concept" in the product lifecycle (cf. Figure 2-3).

elements of the product hierarchy as well as of the lifecycle processes (IEEE COMPUTER SOCIETY, 2007: p. 4). A system can be divided further into the so-called system breakdown structure, which reflects the three major phases of systems development: i) system definition, ii) preliminary design, iii) detailed design (cf. Figure 2-17) (IEEE COMPUTER SOCIETY, 2007: pp. 17–18).

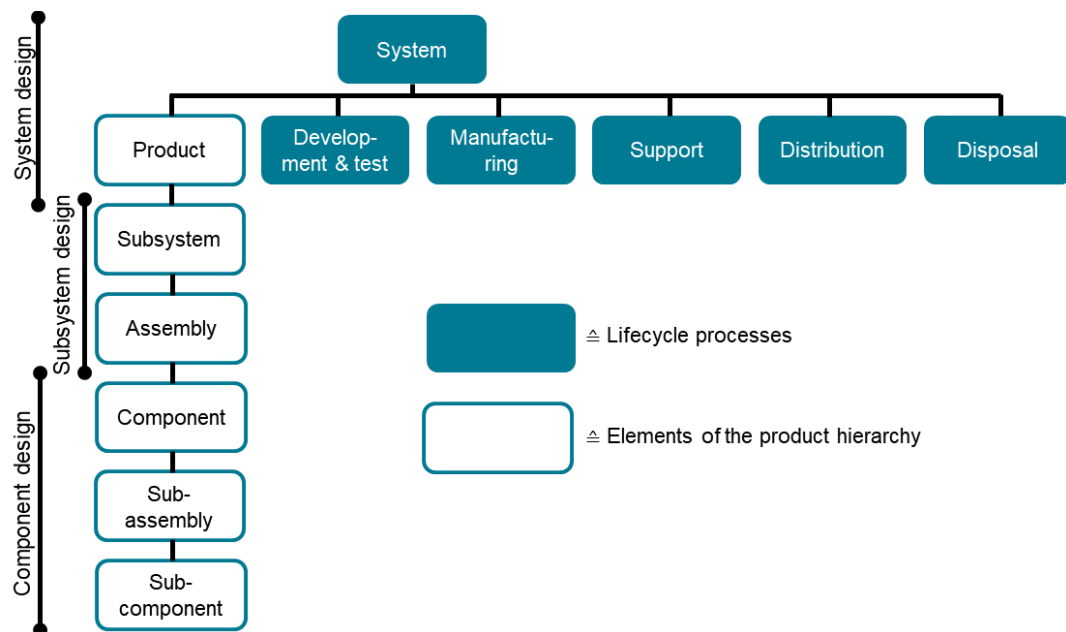


Figure 2-17: System breakdown structure (in alignment to IEEE COMPUTER SOCIETY, 2007: pp. 4, 18).

By implementation of a central system model that connects all the other discipline-specific models, for instance simulation, CASE, E-CAD, and M-CAD models, MBSE improves complexity management, collaboration, quality, productivity, and reuse, amongst others. The central system model and its connection to the discipline-specific models is depicted in Figure 2-18. Due to a model only being a partial representation of reality, it can only contain aspects that are relevant for its purpose, whether synthesis or analysis⁵¹. Therefore, the content of the system model, its application along the development process, and which languages and IT tools that will be used, are focal questions (ZAFIROV, 2014: pp. 82–83). Different methods for MBSE define different artifacts, different steps, and modeling approaches. These will be discussed in the next chapter⁵².

⁵¹ Synthesis describes the conception of new solutions including the specification of new goals. Contrarily, the identification or prediction of actual behavior based on these specifications by means of tests or simulations are inherent to analysis (ZAFIROV, 2014: p. 80).

⁵² Please refer to FRIEDENTHAL et al. (2012) for an overview of a partial systems engineering standards taxonomy including further standards, architecture frameworks, modeling methods, and interchange standards.

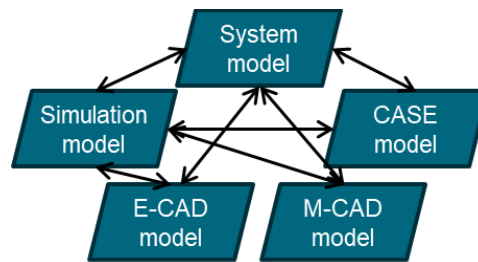


Figure 2-18: Model-based systems engineering with a central system model (in alignment to ZAFIROV, 2014: p. 82; FRIEDENTHAL et al., 2012: p. 18).

MBSE commonly consists of an underlying method of how the modeling has to be executed, a modeling language, and a tool in which the method according to the syntax and semantics of the language that has been applied. Those three parts of MBSE have in common that the system model is crucial, as depicted in Figure 2-19 (in alignment to ALT, 2012: p. 9; EIGNER et al., 2018: p. 382; EIGNER et al., 2016a: p. 167).

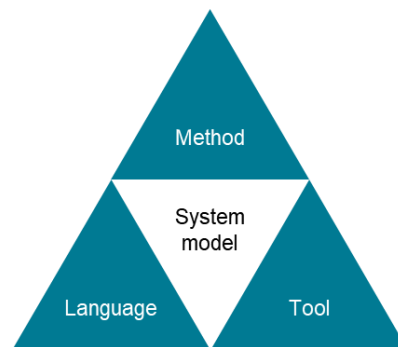


Figure 2-19: The three modules of MBSE with the system model as its central artifact (in alignment to ALT, 2012: p. 9; EIGNER et al., 2018: p. 382; EIGNER et al., 2016a: p. 167).

The most prominent modeling language is *systems modeling language* (SysML)⁵³. MBSE methods using SysML, and current IT tools which support the introduced methods and apply SysML are presented in the next chapter.

2.4.2 Methods and languages

LANGUAGE

MBSE is an approach which uses a formal language to describe connections and creates connections between different disciplines. There are several languages used to graphically model systems. However, SysML, a successor of unified modeling language (UML), evolved to an industry standard and is wide spread (ALBERS and ZINGEL, 2013: p. 82; KLEINER and KRAMER, 2013: p. 102; ZAFIROV, 2014: p. 89). SysML provides

⁵³ SysML is standardized by the Object Management Group (OMG). Please refer to OMG (2015) for a holistic overview of SysML and its components.

structure diagrams (block definition, internal block), parametric diagrams, package diagrams, behavior diagrams (activity, use case, state machine, sequence), structure and behavior models, and cross-sectional diagrams (requirement, stereotype, data exchange formats) (WEILKIENS, 2008: pp. 226–227; KÖNIGS, 2013: p. 28; ZAFIROV, 2014: p. 90; FRIEDENTHAL et al., 2012: pp. 17, 30; KORDON, 2013: p. 49). An exemplary depiction of the diagrams in SysML (lite) language features and some highlights for each type of diagram are displayed in Figure 2-20⁵⁴ (in alignment to FRIEDENTHAL et al., 2012: p. 33).

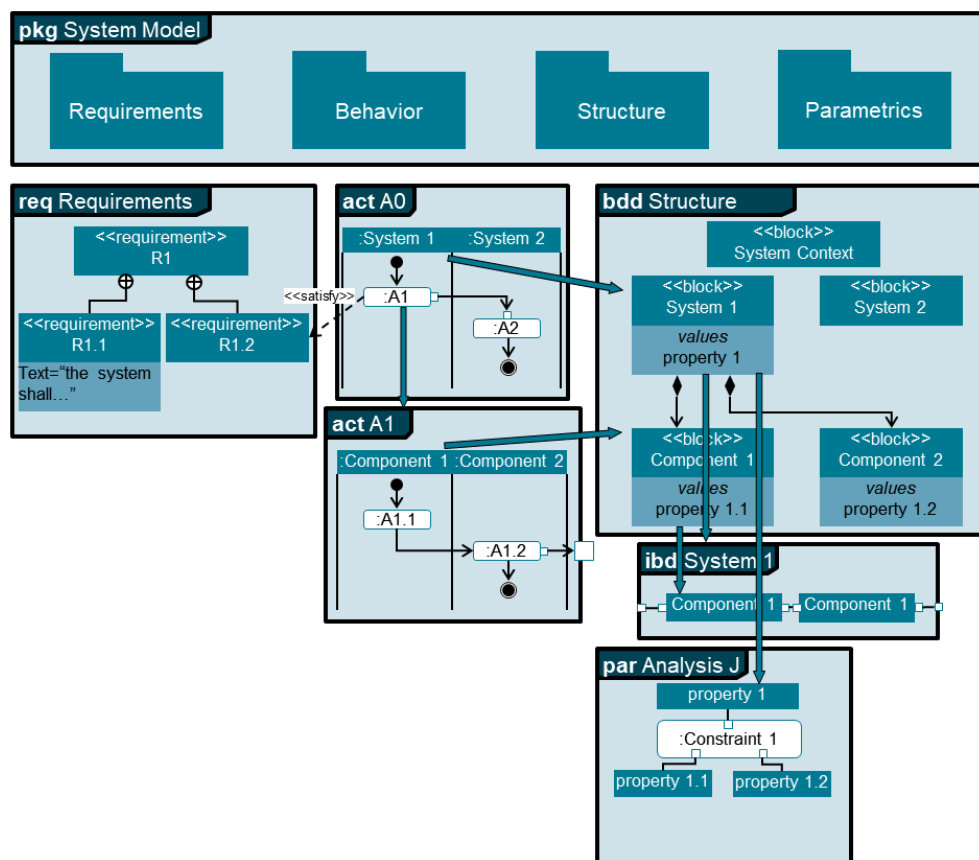


Figure 2-20: Overview of SysML (lite) language features (in alignment to FRIEDENTHAL et al., 2012: p. 33).

METHODS

As depicted in Figure 2-9, the RFLP (requirements engineering, functional design, logical design, physical design) approach describes the systematic product development process for systems in alignment with the *V-model*. Commonly, PLM systems support the RFLP breakdown structure. It is also the most mature systems engineering metamodel in PLM. Hence, compatibility on this metalevel of data between

⁵⁴ SysML lite is a simplified subset of the SysML notation and only used here for exemplary reasons (cf. FRIEDENTHAL et al., 2012: pp. 31 ff.)

MBSE authoring tools and PDM/PLM systems is often provided, although some MBSE methods define the RFLP approach differently⁵⁵ (KLEINER and KRAMER, 2013: p. 95; PAVALKIS, 2016: pp. 2466–2469, 2479; ZAFIROV, 2014: p. 88; HORVATH et al., 2015: p. 85; ZAFIROV, 2017: pp. 31–32).

Due to SysML being considered as the de facto standard modeling language for MBSE (vide supra), only MBSE methods using SysML are in scope of this work. In alignment to EIGNER et al. (2016a), MECPRO² ABSCHLUSSBERICHT (2016b) and DICKOPF et al. (2017), the following MBSE methods use SysML⁵⁶:

Alt, Oliver (ALT (2012))

This method focusses on the model-based top-down description of technical systems starting at the requirements and use cases. Chains of effects following input, processing, and output are established. Allocation relations span connections between different levels of abstraction.

FAS method (LAMM and WEILKIENS (2010))

The *functional architectures for systems* (FAS) method deduces a functional systems architecture from use cases and detailed activities. A functional architecture already on a system level shall enable a preferably solution-neutral and hence technology-independent depiction.

FAS4M (GRUNDEL et al. (2014))

This method extends the FAS method towards mechanics. FAS4M (*functional architecture of systems for mechanical engineers*) aims at the connection of abstract functional models with shape describing CAD models by using SysML as a fundament for further depictions of the distinct mechanical characteristics at the logical level.

Harmony SE (HOFFMANN (2011))

This method provides an integrated development process for systems and software by a combination of the Harmony software engineering and systems engineering processes. Iterative and incremental steps in the requirements phase, system function analysis, and design synthesis are based upon use cases.

⁵⁵ For more information, please also refer to LOPER (2015).

⁵⁶ The method *CONSENS* is not in scope due to its proprietary modeling language and SysML is only included by means of an additional profile (IWANEK et al., 2013: pp. 337–346; DICKOPF et al., 2017: p. 67).

Holt and Perry (HOLT and PERRY (2008))

This method's foundation is the elements ontology, MBSE framework, and viewpoints. The ontology describes concepts, terms, and relations in the system's context. The framework defines the application of the ontology by viewpoints. Viewpoints delineate extracts of a system that refer to a specific part of the ontology⁵⁷.

OOSEM (FRIEDENTHAL et al. (2012))

The *object-oriented systems engineering method* (OOSEM) supports the specification, analysis, design process, and verification of a system. This method again is a top-down approach and aims at an easily adaptable model-based systems architecture.

SE-VPE (GILZ (2014))

This method focuses on the model-based functional and logical breakdown (cf. Figure 2-17) in an early phase of development derived from the requirements. In doing so, vertical and horizontal traceability⁵⁸ is ensured. Very important in this method is the ability to transfer elements of the generated system elements to a system lifecycle management IT system.

SPES (POHL et al. (2012))

The *software platform embedded systems* (SPES) methodology⁵⁹ combines different modeling approaches from diverse disciplines. It uses abstraction layers to further increase the level of detail. Viewpoints are implemented to distinguish between different aspects of different stakeholders. Viewpoints are similar to the RFLP approach: requirements, functional, logical, and technical viewpoint.

SYSMOD (WEILKIENS (2008))

The method *system modeling process* (SYSMOD) serves as a tool kit to model systems without a predetermined sequence of activities. It supports modeling of requirements as

⁵⁷ A *view* is defined as a "collection of entities and assigned attributes (domains) assembled for some purpose" (INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2017b: p. 502). Consecutively, a *viewpoint* on a system is defined as a "form of abstraction achieved using a selected set of architectural concepts and structuring rules, in order to focus on particular concerns within as system" (INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2017b: p. 502). Please refer to INTERNATIONAL ORGANIZATION FOR STANDARDIZATION (2017b) and POHL et al. (2012) for more definitions on views and viewpoints.

⁵⁸ Vertical traceability means the connection of precedent and subsequent phases of the systems development. Horizontal traceability is enabled by linking different disciplines in one phase of development. Cf. also Chapter 2.1 for further definitions of vertical and horizontal traceability.

⁵⁹ Please refer to Footnote 16 (p. 23) for an explanation why the *SPES methodology* is called like this.

well as the functional and physical structure of complex systems (in alignment to DICKOPF et al., 2017: pp. 66–68).

2.4.3 Traceability in the context of MBSE

Traceability should be intrinsic to all MBSE methods between domains involved, e.g., mechanics (CAD), E/E (E-CAD and CASE), and simulations, in the early systems engineering process because MBSE methods claim to model relations explicitly. Hence, MBSE methods enable traceability in the first place. Therefore, this chapter will focus on how MBSE methods foster traceability with further downstream processes in the development phase as well as with supplementary processes of documentation (PDM/PLM). Further upstream processes and methods, such as Modelica, Matlab, Simulink, functional mock-up unit/interface (FMU/FMI) are not in scope as they also often occur in separate author tools.

DATA MODEL

Within the discipline of MBSE and each MBSE method, traceability is fostered by the system model and explicit modeling of relations between artifacts, their values, etc. (cf. Figure 2-20). However, due to MBSE occurs in the early development phase, these relations and connections to downstream processes and disciplines are not yet fully implemented. For that purpose, WEILKIENS et al. (2016) included a criterion “connectivity” into their “framework for the evaluation of MBSE methodologies for practitioners” to assess this essential feature of MBSE tools. In this study it is assessed whether information can be exchanged easily with other tools, which standard API are provided or can be added, and if open protocols are used for import and export. The criterion “connectivity” is weighted with the highest possible value, indicating its relevance for the assessment in practice (WEILKIENS et al., 2016: pp. 2–3).

Hence, a lot of research focuses on enabling traceability of relations created in early development between many different disciplines and providing this information to subsequent processes and disciplines throughout the product lifecycle. Figure 2-21 shows a schematic representation of different hierarchical product descriptions in different phases of the lifecycle and the aim to foster traceability between the product descriptions and between lifecycle phases by means of linkage of information artifacts that refer to artifacts of other phases (in alignment to MÜLLER and KIRSCH, 2017: p. 179). However, the requirements phase is not in scope of this work and is displayed here for mere exemplary reasons. BIAHMOU (2015b) emphasizes the necessity of traceability of

changes on the different abstraction levels of development, i.e., functional, logical, and physical, by means of a common metadata model of the system. This could be achieved by the central systems model within MBSE (BIAHMOU, 2015b: pp. 222, 231). Additionally, an integrated, ontology-based (meta) data model enhances connections and traceability between different disciplines in MBSE and between MBSE and PDM/PLM (HOOSHMAND et al., 2018: pp. 106, 108; HOOSHMAND et al., 2016: pp. 246, 253).

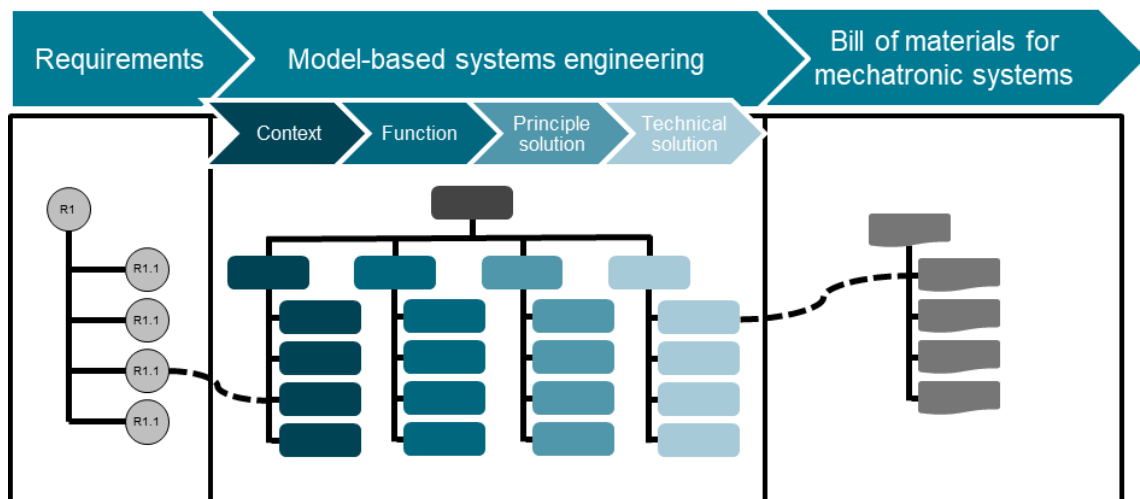


Figure 2-21: Example of three different product descriptions in hierarchical representation in different phases of the lifecycle and the aim to foster traceability by connection of information artifacts (in alignment to MÜLLER and KIRSCH, 2017: p. 179).

For the purpose of link creation between the phases of MBSE and PDM/PLM, the distinct data models have to be aligned. Due to a fundamentally different approach in generating data models in the involved disciplines and their IT tools and systems which grew over decades or are relatively novel, alignment of data models is a major endeavor. On data level, there are three different variations of how to generate this alignment of data models (in alignment to MÜLLER and KIRSCH, 2017: pp. 178–180; HEBER and GROLL, 2018b: p. 127):

1. Equivalent information artifacts are used within the data model structure of MBSE and PDM/PLM that are identical except in regards to their name. By this, asynchrony of content and metadata is ensured, for instance to describe different lifecycles, due to metadata being able to diverge. Hence, lifecycle information as well as rights and roles can deviate. The connection is achieved via a global connection and enables traceability. Regular synchronization of information artifacts is necessary.

2. Globally valid information artifacts with the same name are used for the data models of both disciplines. Therefore, lifecycle information as well as rights and roles cannot deviate.
3. If PDM/PLM information artifacts include a link in their metadata to artifacts of MBSE, then this is called linked information artifacts. Links can be URL, such as the OSLC approach, where one information artifact can merely hold metainformation and a link, while the other includes all relevant properties. Lifecycles and rights and roles should be identical.

Which variation of data model alignment in the form of information artifact connection is superior or optimal is a moot point. This is because it depends on many factors such as which IT systems prevail, how the data models are implemented, and how responsibilities as well as rights and roles are shaped (MÜLLER and KIRSCH, 2017: pp. 178–180). Table 2-1 illustrates the three options of data model alignment.

Table 2-1: Three alternatives of data model alignment between MBSE and PDM/PLM (in alignment to MÜLLER and KIRSCH, 2017: p. 179; HEBER and GROLL, 2018b: p. 127).

	MBSE	PDM/PLM
Equivalent information artifacts	Artifact „sysml::A“ Property X Property Y	Artifact „plm::A“ Property X Property Y
Global information artifacts		Artifact „global::A“ Property X Property Y
Linked information artifacts	Artifact „sysml::A“ Property X Property Y	Artifact „plm::A“ Link

To store and manage links, a central link repository is suggested as OSLC does not support a centralized management of links. The advantage of a central link repository can be that there is the update of links and prevention of broken links, in case of altered resources. In some cases, the PLM system itself can function as a central link repository (PFENNING, 2017: pp. 105, 118, 156). However, for distributed engineering collaborations such a centrally managed link repository might hold some impediments regarding data sovereignty, trustworthiness, availability, etc. (cf. Chapter 2.7).

Depending on which variation of data model alignment is chosen and hence how integrated lifecycles of the different disciplines have to be, it might be necessary to introduce a common variability or variant management (cf. Chapter 2.3.2). This has to be based upon a joint data model. If variability is present, a system model alone is not capable of completely ensuring traceability due to variability or variant management is

interdisciplinary and hence different data models of different disciplines have to be aligned. There are many variability concepts, standalone, embedded, or enabling techniques. However, in practice this still remains an issue at which level of granularity to install variability management (DUMITRESCU et al., 2014: pp. 130–131; BIAHMOU, 2015b: 231). A configuration item (CI) and a linkable item (LI) as distinct metadata allocated to the actual information artifact can create different configurations/variants of multiple items by links. This allows for a separation of disciplines' lifecycles and yet linking them together with a joint CI in a light weighted manner, such as the variation of linked information artifacts in Table 2-1 (SCHULTE et al., 2017a: pp. 88, 91; SCHULTE et al., 2017b: pp. 179–180; SCHULTE et al., 2017d: pp. 328–329, 331). This approach of a separate information artifact to handle variability in MBSE in order to foster traceability by alignment of MBSE and PDM/PLM data models is in line with the *orthogonal variability model* (OVM). In OVM, the variability of a software product line is modeled explicitly in a separate model as metadata and can connect various different models. (SCHULTE et al., 2017c: pp. 263–264; SCHULTE et al., 2017a: 90; POHL et al., 2005: pp. 72 ff.).

PROCESS MODEL

GILZ (2014) describes in the SE-VPE method an approach how a change workflow for the lifecycle of system elements by means of a voting mechanism is shaped. In the process, engineers can vote if changes affecting their systems or components are valid (compatible change) or not (incompatible). A majority accepting the change, promotes the system element in scope along its release lifecycle in the PLM backbone: preliminary, in review, released, in change, superseded. Change management can also occur in MBSE authoring tools or TDMs, but the focus here is on the change process on PDM/PLM level. In order to avoid broken links between information artifacts of MBSE and PDM/PLM when a change request is released, links and relations have to be maintained. Floating relations, conversely to fixed ones, always point to the latest version of a connected information artifact. By those means, explicit maintenance of linked relations between information artifacts in different data models can be circumvented. A floating relation automatically creates a modified copy whenever an artifact changes and links it with the previously related artifact. GILZ (2014) suggests not to transfer unreleased MBSE information from the specific authoring tools or TDMs to the PLM backbone and rather only transfer released information (cf. Figure 2-12). The release generates a version number and a unique identifier in the PLM backbone. By

mapping version numbers after a change occurred, SysML models in the MBSE TDM or authoring tool are kept updated (GILZ, 2014: pp. 115–123, 145-151). For regulatory purposes as well as traceability, storage of baselines of configurations of information artifacts as well as their links is necessary (PFENNING, 2017: pp. 160–161). Additionally, the integration between these MBSE information artifacts and processes with organizational processes is required (BRETZ et al., 2016: p. 8).

TECHNOLOGY

Some MBSE authoring tools are able to provide a hierarchical structure of model-based systems that is called containment tree. This hierarchical structure can be transferred to a PDM/PLM structure. However, today this still often requires proprietary APIs (KIRSCH et al., 2017b: pp. 161–167). Often, the replacement of the entire IT landscape to implement a holistic, model-driven set of IT tools and systems is not feasible for companies. For that purpose, a federative, integrated, and interdisciplinary backbone concept with links connecting different IT tools and systems could be an alternative (EIGNER et al., 2016b: pp. 59–61). *IBM's Engineering Systems Design Rhapsody - Model Manager (Rational Rhapsody)* is part of *IBM's Jazz platform* and uses SysML, which connects requirements, simulation, as well as PDM/PLM via OSLC, if applicable (BRUSA et al., 2018: pp. 335–336; IBM CORPORATION, 2020a, 2020b). The *Windchill Modeler (Integrity Modeler)* by *PTC* also supports the integration via OSLC and can describe models using SysML. Moreover, it also offers direct OSLC integration to *PTC's* own PDM tool, *PDMLink* (cf. Chapter 2.3.3) (BRUSA et al., 2018: pp. 336–337; PTC INC., 2019: p. 4, 2020b; OLLERTON, 2016: p. 4; NORFOLK, 2015: pp. 12–13). *Sparx Systems' Enterprise Architect (EA)* enables graphical depiction by usage of SysML. *EA* serves as an OSLC provider only. This means, that artifacts in *EA* can be addressed by their distinct URL for CRUD operations (cf. Chapter 2.1.3) using HTTP commands (SPARX SYSTEMS PTY LTD., 2020b, 2020a). SysML also is supported by *NoMagic's* (acquired by *Dassault Systèmes*) *Cameo Systems Modeler* and by the installation of additional plugins, OSLC resources can be used. However, only using an additional IT tool called *DataHub*, the *Cameo Systems Modeler* can handle OSLC links⁶⁰ (NO MAGIC, 2015: 48 ff., 2020a, 2020b).

⁶⁰ *iQUAVIS* by *ISID* can be considered a niche product and hence is not in scope here (HEIHOFF-SCHWEDE et al., 2017: p. 43). *Capella* by *Eclipse PolarSys* does not use SysML as modeling language and hence also is not in scope here (ECLIPSE FOUNDATION, 2020).

2.5 Automotive electrics and electronics including software

2.5.1 Definitions, norms, and standards

A vast quantity of electrical systems in an automobile address different requirements. Powertrain, comfort, security, and infotainment partially evolved independently and use their dedicated technologies of communication, actuators, and sensors (REIF, 2016: p. 2). There prevail many standards for E/E. Some examples of ISO norms for communication bus systems are given in REIF, 2014: p. 14⁶¹. Generically, ECUs consist of an input, e.g., plug with pins, which are connected with sensors that deliver input signals. Those signals are processed internally by means of a microcontroller and memory. The output signal addresses actuators according the calculated results. Communication interfaces build the connection to the communication bus systems and enable message delivery between different components (ECUs) in one or different systems (REIF, 2014: p. 136).

Software often already is embedded in ECUs (embedded system) as part of measuring, steering, and control functions as well as for communication. Hence, software also is considered to be part of E/E in this work as it is done in BORGEEEST (2014) (cf. BORGEEEST, 2014: pp. 213 ff.). The *automotive software process improvement and capability determination* (ASPICE), derived from ISO/IEC 15504 (SPICE), assesses the capability of development processes and their output of suppliers for ECUs according given criteria (MECPRO² ABSCHLUSSBERICHT, 2016c: pp. 65–66). ASPICE uses process performance indicators, such as best practices and work products, for such an assessment. Traceability is demanded normatively by explicitly stating that traceability between system artifacts shall be established, e.g., as a result of system integration tests (VDA QUALITY MANAGEMENT CENTER, 2017: pp. 21, 43). Hence, ASPICE is not suitable to implement traceability but rather request and assess it.

2.5.2 Architecture, communication, hardware, and software

The physical E/E architecture of an automobile consists of ECUs, actors, and sensors which are connected via cables, plugs, and pins⁶². An automotive E/E architecture with

⁶¹ Please refer to REIF (2014), particularly Chapters 1 and 2, and Chapter 2.5.2 here in this work for further E/E definitions and standards such as ISO/OSI reference model, communication principles, bus topologies, software, etc.

⁶² The power supply by means of a vehicle electrical system to transmit energy for electrical consumption is not in scope here.

focus on communication consists of (communication) bus systems. It enables the transport of messages (layer 3 and 4 of the OSI layer model) by means of transfer of bits (layer 1 and 2)⁶³.

There exist various communication bus systems in the automotive industry. However, former automobile manufacturer-specific bus systems were superseded by standardized solutions, such as the *controller area network* (CAN) or *local interconnected network* (LIN). Whereas the LIN bus as a cheap alternative is used for instance for switches with a low data rate, CAN, FlexRay, *media oriented systems transport* (MOST), or ethernet, amongst others, were developed for different requirements such as higher data rates, sequential communication times per peer on the bus, fault tolerance, star, bus, or ring topologies (cf. ZIMMERMANN and SCHMIDGALL, 2014: p. 8; REIF, 2014: pp. 7, 14; SCHÄUFFELE and ZURAWKA, 2016: p. 124). Multiple communication bus systems which use different technologies are linked together by central or decentral gateways. Commonly, functional systems, such as powertrain or infotainment, share their bus system with the same technology.

The complexity in the automotive E/E architecture tremendously increased in the last decades (cf. Figure 1-3). The usage of different technologies for communication busses and hence the yielding complexity from this approach is one reason for efforts of standardization and actions to cope with this complexity. Additionally, many solutions reflect organizational structures of automotive manufacturers and their suppliers. Historically, engine, gear unit, chassis, and body are developed in different departments. Striving for an optimal solution for the respective department itself and then handling the integration of the interfaces is a common phenomenon. Those isolated applications hinder the transdisciplinary integration of functions, which is the reality in today's automotive development. A high reuse of bus systems due to cost reasons enforces upward compatibility. Hence, new technologies do not fully replace older solutions and ECUs have to provide all different kinds of connectors and the integration during distributed development becomes more and more complex (REIF, 2014: p. 34; ZIMMERMANN and SCHMIDGALL, 2014: pp. 9–11; BOSCHERT and ROSEN, 2016: p. 62).

⁶³ For more information on the different layers of the ISO standard including the *open system interconnection* (OSI) model, please refer to ZIMMERMANN and SCHMIDGALL (2014), BERGEEST (2014), REIF (2014), VAJNA (2009), and PETERSON and DAVIE (2012).

HARDWARE

Although actuators and sensors play an important role in a modern automobile, e.g., for autonomous driving, here the focus regarding hardware will be on the ECUs due to their relevant role as interface components in E/E systems. Hence during development, different departments intra- and inter-company have to align their interface specifications for ECUs. Often, actuators and sensors are highly standardized and match the ECU they are connected to (REIF, 2016: p. 7).

SOFTWARE

Usually, software has to be changed more frequently during its application than other products. This might be due to newly added software that has to be integrated or processes in which the software supports have changed. One approach to alleviate the number of changes necessary is the parametrization of software to give software components yet another degree of freedom in customization without the need to alter huge parts of it. For that purpose, configuration parameters are stored in parametrization files⁶⁴ (POMBERGER and PREE, 2004: p. 85). An automotive example would be the parametrization of the same engine ECU for two different power levels where the parametrization file stores different characteristic curves, injection masses, air volume, etc. Additionally, ECUs commonly have a bootloader software and a firmware (STRINGHAM, 2010: p. 6).

Many functionalities in modern automobiles are enabled by software. Therefore, automobile manufacturers recognized the exigency of standardization for software. The *automotive open system architecture (AUTOSAR)* standard addresses this complexity by the development and implementation of standard software components suitable for reuse and exchange. This is accomplished by designing hardware-independent application software (OEM-specific) and hardware-oriented basic software (OEM-independent) connected by a flexible runtime environment. By those means, software can be developed without specific knowledge of the planned hardware and software components can be implemented flexibly on different ECUs. During development, the system configuration, i.e., network topology, is described explicitly from communication bus systems up to the communication matrix of single channels⁶⁵ (WINNER et al., 2015: pp. 106–115). The *AUTOSAR adaptive* platform enhances the classical *AUTOSAR* by

⁶⁴ See Chapter 2.3.2 for more information about software parametrization and its use for the configuration of software.

⁶⁵ See Chapter 2.5.3 for more information on the (network) communication matrix or network communication description.

offering functional clusters with services and APIs as interfaces which are linked dynamically during runtime. A virtual function bus connects joint functions across ECUs⁶⁶ (AUTOSAR, 2019; REIF, 2014: p. 58).

2.5.3 Traceability in the context of automotive E/E and software

DATA MODEL

Besides the requirements towards traceability in the E/E development for hardware and software, the above-mentioned communication matrix or *network communication description* (NCD) describes dependencies on the signal level. There is no standardized format of what the NCD has to compose. Sometimes the NCD names the signals on a communication bus in its rows and further attributes in its columns. Also, the NCD can have ECU names in rows and columns indicating which ECU consumes or provides which signal (BORGEEEST, 2014: 131–132; WINNER et al., 2015: p. 115; ZIMMERMANN and SCHMIDGALL, 2014: pp. 416–417). Hence, the NCD is one possibility to foster traceability in the context of automotive E/E by the possibility of the creation of a machine-readable file that indicates physical connections (hardware) among ECUs as well as their communication by means of messages (software).

PROCESS MODEL

As stated above, the complexity in automotive E/E, hardware and software, becomes hard to handle. This is particularly relevant when it comes to safety-relevant functions. Therefore, norms address this issue. The norm *ISO 2626* for road vehicles, one of the most important norms for automotive E/E, requires functional safety for systems with E/E components. The norm stipulates different analysis, required documentation of product information during the development process along nested *V-models*. Traceability has to be documented and maintained explicitly for safety-relevant relations between systems and elements on hardware and software level. Changes during the development as well as during configuration management have to be traceable (INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2011c: p. 9, 2011d: p. 10, 2011e: pp. 8, 11, 14, 37; KÖNIGS, 2013: p. 27; BORGEEEST, 2014: pp. 332 ff.).

TECHNOLOGY

Usually, automotive E/E architectures are developed discretely, i.e., not car-specific, and if a development project for an automobile model line starts, the desired E/E

⁶⁶ For more information on the *AUTOSAR (adaptive)* standard, please refer to AUTOSAR (2019), REIF (2014), ZIMMERMANN and SCHMIDGALL (2014), and WINNER et al. (2015).

architecture will be adapted to the automotive platform in scope. Without computer-aided modeling of harnesses, E/E architectures, and their message transmission, modern E/E development would not be practicable. Therefore, there prevail plenty IT tools to support the development process. Often, engineering disciplines are separated into the design of communication busses and message transmission, E/E architecture design, harness wiring layout, etc. Examples for common IT tools are *PREEVision*, *CANoe* and *CANape* by *Vector Informatics*. *PREEVision* is a widely used E/E development tool for the communication bus design in the automotive sector. It uses a centralized approach for an IT collaboration platform. The integration of automotive E/E hardware and closely related software development, particularly among IT tools of the same vendor, is relatively high. However, traceability with other disciplines or direct integration of E/E hardware or software into a BOM in a PDM/PLM system are currently scarce⁶⁷ (cf. ZIMMERMANN and SCHMIDGALL, 2014: pp. 415–433; BECK et al., 2016: pp. 6–7).

With respect to software development and its decisive version control, particularly in distributed development scenarios, the open-source tool *Git* is acknowledged widely. *Git* holds a complete replication of databases from the server at each client (cf. Chapter 2.3.2 and Figure 2-15) (CHACON and STRAUB, 2014: p. 4; GIT, 2020b). *Atlassian's* tool *Bitbucket* for the management of *Git*-code extends the sole software version control by means of project planning, testing and deployment, as well as collaboration. Hence, *Bitbucket* is in scope here instead of the stand-alone *Git*⁶⁸ (ATLASSIAN, 2020a, 2020b, 2020c).

2.6 Distributed engineering collaboration

2.6.1 Definitions, norms, and standards

Due to ever shorter innovation cycles, companies see themselves urged to decrease the duration of development projects. An approach to do so is the parallelization of processes so as to reduce engineering time and enhance quality of results, i.e. simultaneous engineering (cf. Figure 2-4) (EVERSHEIM and SCHUH, 2005: p. 8). A high

⁶⁷ For further information about software development tools and the necessary version control, please refer to Chapter 2.3.

⁶⁸ Here, *Apache Subversion (SVN)* by *CollabNet* is not in scope due to a centralized approach for a version control system and hence differences in performance and fault tolerance towards *Git* in a distributed engineering network approach (cf. Chapter 2.3.2 and Figure 2-15) (CHACON and STRAUB, 2014: p. 3; GIT, 2020a).

interdependency with suppliers already in the early development phase, increased globalization, and more product knowledge at the supplier, demands automotive manufacturers to execute this simultaneous engineering across their suppliers. By this so-called cross-enterprise engineering, frontloading and handling of complexity is fostered (cf. Figure 2-4) (STEPHAN, 2013: pp. 67–68; EIGNER and STELZER, 2009: pp. 14–15, 18; KATZENBACH, 2015a: p. 611). The necessity resulting from cross-enterprise engineering to be in a position to manage distributed and federative information and processes requires solutions across companies. Engineering collaboration denominates solution approaches for distributed development by means of product data exchange or direct collaboration in virtual project rooms (EIGNER and STELZER, 2009: pp. 182–183; EIGNER et al., 2012a: p. 27). Collaboration is understood as a process where companies share information, resources, and responsibilities in order to strive for a joint goal. For that purpose, short-term virtual enterprises can be founded to achieve a common project (BORSATO and PERUZZINI, 2015: pp. 168–169; WOGNUM and CURRAN, 2013: p. 6). Moreover, a company has to collaborate with external sources of knowledge, such as start-ups, to promote open innovation (WOGNUM and CURRAN, 2013: p. 7). Additionally, relationships in engineering collaboration are temporary, limited to a project and a current partner can become tomorrow's competitor. A flexible IT infrastructure is needed to address this volatility as well as the protection of intellectual property (IP). Moreover, mechanisms to protect IP, such as watermarking, are enforced to confront this insecurity and mistrust⁶⁹ (LIESE et al., 2013: pp. 270-272, 277; cf. HEYN, 1999 according to STEPHAN, 2013: p. 69).

For engineering collaboration, the exchange of data is mandatory (cf. Chapter 2.1.1). For that purpose, many standards for the exchange of data have been established. The most prominent standards in the realm of PDM/PLM are:

- Extensible markup language (XML) metadata interchange (XMI) which extends XML by means of inclusion of metaobjects.
- ISO 10303 STEP and here particularly AP 242 (cf. Footnote 24 on p. 30) that describes 3D engineering data whereas AP 233 focuses on systems engineering⁷⁰.

⁶⁹ Please refer to STJEPANDIĆ et al. (2015b) for more information on engineering collaboration, concurrent engineering, and further forms of joint development.

⁷⁰ For more information about ISO 10303 AP 233, please refer to GILZ (2014), KÖNIGS (2013), and INTERNATIONAL ORGANIZATION FOR STANDARDIZATION (2012a).

- OSLC aims at defining specifications to foster integration of IT tools during development and uses web technologies, such as uniform resource identifiers (URI), resource description framework (RDF), and representational state transfer (REST)⁷¹ (STIEFEL, 2011: pp. 40–42; GILZ, 2014: p. 34; KÖNIGS, 2013: pp. 26–27; KATZENBACH, 2015a: pp. 627–632).

Engineering collaboration platforms, i.e., dedicated IT systems and tools to exchange engineering data in engineering relationships, shall include aspects for communication, product data, processes, and organization that are relevant for collaborations⁷² (KATZENBACH, 2015b: p. 189; STIEFEL, 2011: p. 30).

2.6.2 Phenotypes

Engineering collaboration transformed itself in alignment to the transformation of the OEMs' and suppliers' relationships. Traditionally, an OEM had centralistic purchasing and each supplier developed and delivered a specific product which the OEM integrated into its final product. Later, as products became more complex, different tiers of suppliers formed, each tier specialized in one activity, such as engineering of components and single parts, sub-systems and modules, or entire systems. Today most prevalent is the engineering and supplier structure, which is separated out by production, developing systems, and integrating systems for an OEM. Already today there are engineering and supply networks that are highly integrated and connected. System integrators and system specialists offer standard systems to OEMs which adapt these systems according to their own preferences. Moreover, OEMs and the other suppliers develop highly innovative components and systems together to share costs, profits, risks, and opportunities. Therefore, the sole OEM-supplier relationship changed to joint engineering partners, mostly for dedicated projects or products such as autonomous driving. This relationship can extend to virtual enterprises amongst different OEMs, their engineering partners, and suppliers (DAIMLER AG, 2018; FELDHUSEN and GROTE, 2013: pp. 6-8, 31–33; EIGNER and STELZER, 2009: pp. 14–17; LIESE et al., 2013: p. 270). This vicissitude is depicted in Figure 2-22 (in alignment to FELDHUSEN and GROTE, 2013: p. 7; EIGNER and STELZER, 2009: p. 15; STEPHAN, 2013: p. 71).

⁷¹ See Chapter 2.8 and KATZENBACH (2015a), GILZ (2014), STIEFEL (2011), and KÖNIGS (2013) for more information about these standards.

⁷² Please refer to EIGNER and STELZER (2009) and STIEFEL (2011) for more details about different IT systems for engineering collaboration and how they are connected.

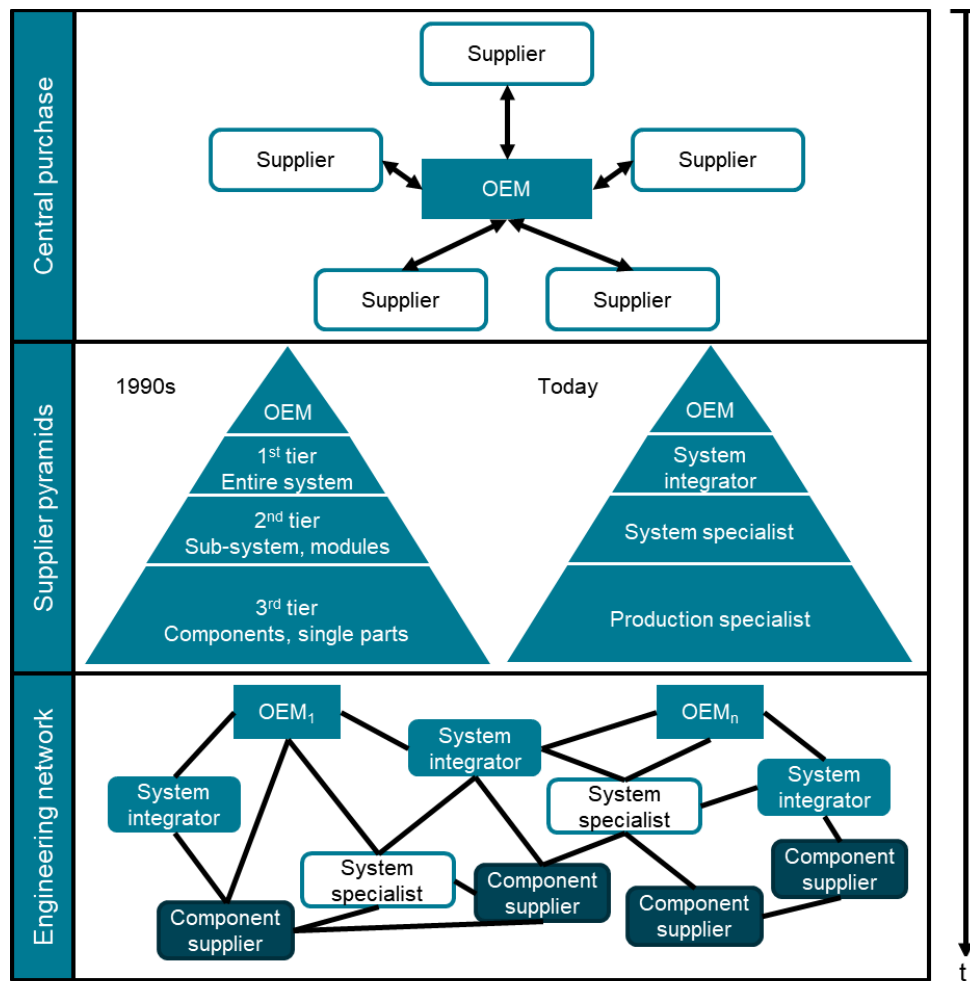


Figure 2-22: Transformation of supplier structures across time (in alignment to FELDHOUSEN and GROTE, 2013: p. 7; EIGNER and STELZER, 2009: p. 15; STEPHAN, 2013: p. 71; KATZENBACH, 2015a: p. 626).

STIEFEL (2011) differentiates between product data-oriented, project-oriented, and process-oriented collaboration (cf. KRAUSE et al., 2007 according to STIEFEL, 2011: pp. 21–26). Product data-oriented collaboration is defined according to the necessary exchange of engineering data between partners in an engineering collaboration in order to make this data available at the right point in time and at the right venue. The data exchange occurs by means of transfer of neutral data formats, e.g., STEP, between different PDM systems. These aspects are aligned to the Sections *Data Model* and *Technology* in Chapter 2.6.3. Project-oriented collaborations focus on coordination of schedules, controlling, planning, etc. and are not within the scope of this work. An intra-company development process characterizes process-oriented collaborations. The goal is to align process steps in each phase of the development process between the involved engineering partners in order to increase traceability and velocity, as well as to

reduce manual work and errors (STIEFEL, 2011: pp. 21–26). The process aspects of engineering collaboration contribute to the Section *Process Model* in Chapter 2.6.3.

Another relatively young form of engineering collaborations are start-ups, developing products jointly with incumbent OEMs or their suppliers and integrating their products in automobiles in an early stage or later as a service. Often, start-ups and emergent new entrants into the automotive market deliver software or other E/E components to OEMs and suppliers. Hence, OEMs heavily invest into start-ups associated with products for the automotive industry (KAAS et al., 2016: pp. 13–14; KÄSSER et al., 2017: p. 4; HOLLAND-LETZ et al., 2019: pp. 1–3; WIEHMEIER, 2017: p. 1; GLASNER, 2018: pp. 1–6; KRIEG et al., 2018: pp. 15–16). In regards to this connection, alignment of development processes, product lifecycles, and production pace are challenging. Additionally, sharing IP with start-ups in a secure manner is, likewise to traditional engineering collaborations, crucial (STAREPRAVO, 2019: p. 6).

2.6.3 Traceability in the context of engineering collaboration

DATA MODEL

In engineering collaboration, the data exchange according to a collective data model is crucial for traceability and reduces manual rework or intensive API programming and maintenance. Hence, standards for the exchange of data (cf. Chapter 2.2.1) in a multiple partners engineering collaboration facilitates automated data exchange. Thus, APIs do not have to be adapted frequently whenever new suppliers or collaboration partners enter the joint development. If the data model within each IT tool or system already matches data exchange standards or these tools and systems provide accordingly standardized APIs, then the adaptation effort for each participant is minimized. Otherwise, authoring tools often provide converters to convert the proprietary data format into a neutral data format, e.g., in the case of CAD models into JT. Only collaboration partners that have IT tools including a converter or are already capable of handling native data can participate in the engineering collaboration (EIGNER and STELZER, 2009: p. 186). In order to share knowledge in an engineering collaboration, a common taxonomy for the description of product data models is necessary. For that purpose, ontologies are used (STIEFEL, 2011: p. 36; VACHER et al., 2007: pp. 314–316). This will be discussed in Chapter 2.8.

PROCESS MODEL

Usually, engineering partners only exchange data at distinct milestones during the development process. This approach has disadvantages, namely that product data often is not up-to-date, reactions to constructive or technological changes of product models occur too late, and therefore this yields unnecessary changes, long development cycles, incomplete traceability, and high costs (STIEFEL, 2011: p. 27; FERREIRA et al., 2017: pp. 1478–1480; SCHÄUFFELE and ZURAWKA, 2016: p. 199). Usually, PDM/PLM systems have strong workflows implemented. The alignment of these workflows across multiple PDM/PLM systems as well as authoring tools of many engineering partners is essential. Today, this is often done with emails as a reminder for the partner that the previous engineer finished their task and the consecutive task can start (EIGNER and STELZER, 2009: p. 189).

TECHNOLOGY

OEMs and suppliers use up to eight different tools within one process phase during development. Hence, traceability and re-use of engineering data is necessary (BEIER, 2014: p. 31; SCHÄUFFELE and ZURAWKA, 2016: pp. 198–200). Moreover, many systems induce incompatibility of data types and hence released data which is often held redundantly at distinct collaboration partners, leading to errors and delays (STIEFEL, 2011: p. 22). Some vendors specialize in offering engineering collaboration IT platforms for the connection of PDM/PLM systems of the major automotive OEMs, SAP systems, authoring tools, simulation software, requirements management, etc. The example of *OpenPDM* by *PROSTEP* further offers APIs to a data exchange platform to transfer huge amounts of data, for instance, with suppliers who do not have direct connectors to *OpenPDM*. A high degree of collaboration for the maintenance and development of *OpenPDM* is required due to the definition of data models, APIs, process models, amongst other things (STIEFEL, 2011: p. 31). *OpenPDM* is a mere integrational facilitation platform in which you cannot manage information artifacts directly to the extent of a proper PDM/PLM IT system (PROSTEP IVIP E.V., 2020a).

2.7 Data base solutions

Particularly in Chapters 2.2, 2.3, 2.6, and in Figure 2-12, the importance of compatible IT systems for traceability in engineering activities and their accompanying documentation within a company and in engineering collaborations, was highlighted. In this chapter, different technologies for those IT systems or data bases will be examined.

This is in order to assess the adequacy of different types of data bases for fostering traceability during development within one and across multiple companies.

2.7.1 Definitions, norms, and standards

A data base is a collection of data which have logical relations among one another and are administered by one's own data base management system (DBMS). The DBMS has one or many logical external APIs which allow users and IT tools to access the data in the data base by translation of logical to physical access (SCHICKER, 2017: p. 3). Above, the distinction between the logical and physical layers of IT systems or data bases has not been made. Hence, below a data base or IT system will be considered to include both layers as well as the DBMS⁷³.

In the following, different types of data bases will be discussed in order to assess their aptitude to foster traceability in distributed engineering collaborations.

2.7.2 Central data bases

The central data base is the traditional IT system. It comprises the following type of data bases (SCHICKER, 2017: pp. 12–16; DORSCHER, 2015: pp. 288–293):

- Relational data bases: They consists of tables, called relations, which also store the relations between different tables. Their simplicity of use and program contributes to their popularity.
- Object-oriented data bases: An object can be a real or an abstract entity. Objects also can be stored in tables. Therefore, object-oriented data bases often are considered as extensions to relational data bases. Programmers and designers have to invest more effort in the creation of these more complex object-oriented data bases.
- Hierarchical data bases: The access via the uppermost node and the successive descent to the node of interest makes these data bases archaic and hence obsolete.
- Key/Value-oriented data bases: Those data bases gain their flexibility and velocity via the allocation of unique keys to values.

⁷³ Please refer to SCHICKER (2017) for a detailed overview of general capabilities of a data base.

- Document-oriented data bases: Similar to key/value data bases, documents (values) are assigned distinct names (keys). Hence, the storage of documents and their association with other data is feasible.
- Column-oriented data bases: Instead of reading row by row when looking for the values of interest as in traditional relational data bases, the column-oriented data bases invert the table. This yields quicker searches due to there only being one column, which stores the value of interest, that has to be searched.
- Graph-oriented data bases: Here, data is structured according to graph theory using nodes and edges and is particularly performant for geographical or social data.

The latter four data base types are so-called NoSQL (not only structured query language) data bases and are often used for use cases where performance of data access and calculations are crucial (SCHICKER, 2017: pp. 12–16; DORSCHER, 2015: pp. 288–293).

2.7.3 Decentral data bases

An increasing amount of data, as well as access to it anytime and anywhere, demands more performant and flexible data bases than the relational, central data bases described above. Central data bases are limited in their vertical scalability, i.e., the increase of processors, disk storage, and main memory for the purpose of higher performance. For that purpose, decentral data bases were created where the data processing is distributed among multiple IT systems (horizontal scalability). Decentral data bases are defined by data which is stored at least at two computers or IT systems within the same network. Validity or consistency in case of redundant data are just two more examples of how the complexity increases with multiple IT systems handling data in one network. It has to be figured out which set of data is the most recent and at which IT system the data is. On the other hand, decentral data bases offer advantages in comparison to central data bases such as faster access and higher performance. This is also advantageous for distributed companies with, for instance, multiple plants, subsidiaries, or development offices. Consequently, a frequent local access to data across the world can be facilitated. Moreover, availability, i.e., the reciprocal of an IT system downtime, can be ameliorated and hence the failure of one IT system does not compromise the entire network. The failure of the central data base then again is its Achilles' heel (SCHICKER, 2017: pp. 307–309; DORSCHER, 2015: p. 278). DATE (1990)

postulates twelve principles towards decentral data bases so that decentral data bases appear to the user as if they would be a central data base. The most important principles in the context of distributed engineering collaboration and traceability for engineering data across the lifecycle are⁷⁴ (cf. DATE, 1990 according to SCHICKER, 2017: pp. 309–312):

- No central administration instance,
- Permanent availability,
- Independent of fragmentation,
- Independent of data replication,
- Decentral transaction administration.

Not all principles can be satisfied concurrently due to an immanent contradiction of some of them. Additionally, consistency (C), availability (A), and tolerance of network partitions (P) partially are disjunct and only two of them can occur simultaneously. This is called the *CAP theorem* and is important for decentral data bases in the following (SCHICKER, 2017: pp. 312–315).

LINKED DATA

Linked data is characterized as a data base, multiple data bases, or IT systems which hold data from different domains that are integrated and linked semantically⁷⁵. The world wide web also became a web of linked data with highly integrated semantical links between contents instead of solely linking documents (SAKR et al., 2018: p. 5). Integrated content and semantical links in the context of linked data implies specific data relationships and machine-processable data. Herewith, isolated data silos shall be overcome and an interconnection of data fosters global data integration. The World Wide Web Consortium (W3C) standardized all fundamental concepts of linked data (SAKR et al., 2018: p. 9). The four most important principles are (BERNERS-LEE, 2006; SAKR et al., 2018: pp. 9–10):

- Denominate objects with uniform resource identifiers (URIs)⁷⁶.
- For transfer of data the hypertext transfer protocol (HTTP) shall be used⁷⁷.

⁷⁴ Please refer to SCHICKER (2017) for the full list of the principles by DATE (1990).

⁷⁵ Cf. Chapter 2.8 for more information about semantics in data bases and IT.

⁷⁶ For more information, please refer to <https://www.w3.org/Addressing/>.

⁷⁷ For more information, please refer to <https://www.w3.org/Protocols/>.

- Data identified by URIs and transferred via HTTP shall be provided using standards such as the resource description framework (RDF)⁷⁸.
- The inclusion of further links to other URIs into the content shall promote the integration of data.

Commonly, linked data is a static snapshot of information. So-called triples form the semantical interconnection of data that yields large RDF graphs. This will be explained further in Chapter 2.8.

PEER-TO-PEER NETWORK WITH STRUCTURED OVERLAY NETWORK

A peer-to-peer (P2P) network is a special kind of decentral data base where peers, i.e., participants, of a computer network or network of IT systems allows direct access to data stored at a peer's computer or data base. Resource sharing, such as content, bandwidth, processing power, etc., amongst millions of nodes⁷⁹ without contacting a centralized authority is a distinct feature of P2P networks. The absence of a central authority to control resources and the capability to scale up to millions of nodes makes a P2P network to a P2P network with an overlay network. The overlay network spans over the physical network of, e.g., the internet and creates the actual P2P network with nodes operating as hosts for their content, offering it to share their data, and often with their own address space. In contrast to unstructured overlay networks, structured overlay networks comprise a specific graph structure which allows for efficient search of data objects within the network. On the one hand, structured overlay networks require higher maintenance effort as well as induce further complexity during installation. On the other hand, the structured network allows for routing of queries due to nodes holding the information of other peers and which data they offer, so-called routing tables, which makes directed searches feasible. In an unstructured overlay network peers or nodes do not hold information of data of other peers inducing random searches, without specific forwarding⁸⁰ (PETERSON and DAVIE, 2012: pp. 769–772; STIEFEL, 2011: pp. 43–51).

IDIOSYNCRASIES OF THE BLOCKCHAIN TECHNOLOGY AS A PEER-TO-PEER NETWORK

In 2008, an unknown author with the pseudonym Satoshi Nakamoto (NAKAMOTO, 2008) proclaimed a digital P2P electronic cash system called *Bitcoin*. This uses the *Blockchain*

⁷⁸ For more information, please refer to <https://www.w3.org/RDF/>.

⁷⁹ Here, nodes and peers are used synonymously for computers, IT systems, or data bases in one or multiple networks that exchange data or have some sort of contact.

⁸⁰ Please refer to STIEFEL (2011), BOHN (2007), ÖZSU and VALDURIEZ (2011), and PETERSON and DAVIE (2012) for more information about peer-to-peer networks and their peculiarities.

technology, a special kind of a P2P network, as the fundamental technology to establish trust with distributed, public transactions in an otherwise centralized, restricted monetary system. Although all technical components of the Blockchain and Bitcoin emanate from academic research and literature of the 1980s and 1990s, Nakamoto's achievement was the particular combination of the underlying components in a very complex manner. The main (business) features of the Blockchain technology are:

1. **Decentrality:** The Blockchain is a distributed data base, or distributed ledger, holding redundant data with a P2P network of all nodes. There does not exist an intermediary or central authority to control transactions in the P2P network⁸¹. It is an unstructured P2P network due to the absence of routing tables. Searches in the network occur using a flooding protocol called gossip protocol, i.e., search requests are randomly forwarded⁸². Transactions are validated by means of a consensus mechanism employing a lot of computational power for the purpose of solving mathematical issues, for instance the so-called *proof of work*.
2. **Publicity:** Each node within the network has the same state of knowledge regarding the transactions executed. Data is stored completely redundant, meaning that each (full) node holds the entire transaction history ever made. Each new transaction between or among nodes is spread after some time across the whole network in order to keep all nodes' data bases up to date. This update only can be performed once the majority of nodes have agreed via a consensus upon the result of the mathematical problem.
3. **Irreversibility:** Blocks of multiple transactions include a time stamp and the hash value of a cryptographical hash function of the previous block of transactions. This concatenation of append-only blocks yields an irreversible transaction history and ensures the integrity of data. Due to blocks of transactions being interlinked irrevocably via hashes, this technology was named *chain of blocks*

⁸¹ BARAN (1964) distinguishes between decentralized and distributed networks. Decentralized networks (many connected stars) can be considered as many connected centralized networks (star) with a hierarchical structure whereas in distributed networks (mesh or grid) all nodes are connected, at least to those in proximity BARAN (1964: pp. 1–2). This definition goes in line with the above-provided descriptions of P2P networks with different types of overlay networks. Hence, in this work decentralized and distributed networks are considered to be synonymous.

⁸² ERCIYES (2013) describes the flooding algorithm and its composition in more detail.

and later *Blockchain*⁸³ (BASHIR, 2018: pp. 16–18, 24; NARAYANAN and CLARK, 2017: pp. 1–3).

As mentioned, blocks within the Blockchain contain transactions. These transactions can be of monetary nature in case of Bitcoin or merely a record of an event. A block does not only contain the payload, i.e., transactions, but also a block header including a pointer to previous block's hash value (not in case of the so-called first *genesis block*), a nonce, time stamp, and Merkle root. This is displayed in Figure 2-23. The components of the Blockchain and its blocks will be shortly described in the following.

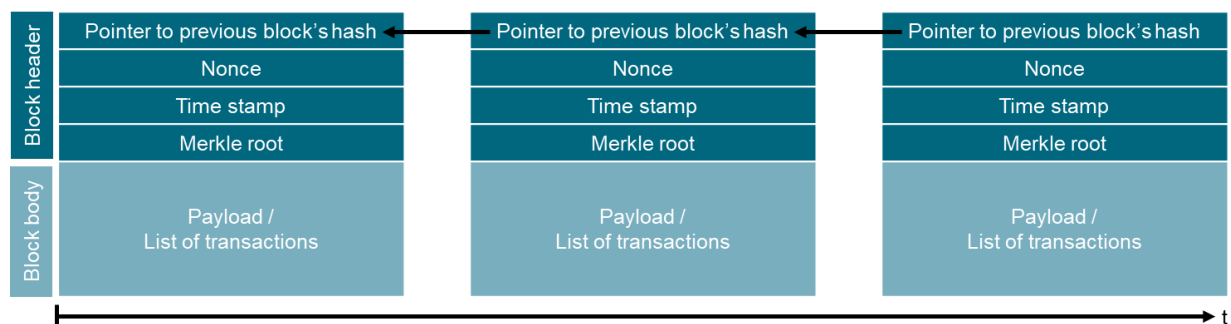


Figure 2-23: Generic structure of blocks in the Blockchain (in alignment to NARAYANAN et al., 2016: p. 33; BASHIR, 2018: p. 20).

A cryptographic hash function calculates a fixed sized output given a string of any size. Knowing the result of the hash function, i.e., the hash value or just hash, does not give one any feasible way to become aware of the input. Using the same input and the same hash function always yields the same result. Hence, if a transaction in one block of the Blockchain would be altered, the calculated hash value would differ and not match the successive block, which includes the previous block's hash value, anymore⁸⁴ (NARAYANAN et al., 2016: pp. 23, 27). Public and private keys are used to sign and verify transactions and are created upon entering the network (NAKAMOTO, 2008: p. 2). The Merkle tree is another cryptographic instrument using hash pointers to previous data. The hash values of transactions are paired in groups of two and two of those again serve

⁸³ If one node or a group of nodes gains more than 51% of computing power within the entire network, then transaction blocks indeed can be altered and hence are not irreversible anymore. However, for practical purposes, this scenario is not relevant due to whenever one interest group gains the majority of computing power, trust in the entire network will be lost and its crypto-currency will be worthless (BASHIR, 2018: pp. 17, 177).

⁸⁴ This shows that the change of a block requires recalculating not only the altered block but all successive blocks as well and solve the immanent mathematical problem of finding the nonce (see next paragraph). As this requires a lot of computational power and now blocks are appended approximately at the same speed, a catch up in the sense of re-calculating the entire transaction history can be considered as impossible. Based upon the computational power, i.e., the hashing rate or the calculation speed of hashes per second, within the entire Bitcoin network and to counter technological advancement, the difficulty of the mathematical problem can be adapted accordingly (BASHIR, 2018: p. 172).

as input for a new hash value which aggregates graphically to a tree shape. This technique fosters traceability because all hashes can be verified and data tampering can be precluded⁸⁵ (NARAYANAN et al., 2016: pp. 34–35; BASHIR, 2018: pp. 19, 111).

“A nonce is a number that is generated and used only once” (BASHIR, 2018: p. 19; DEUTSCHES INSTITUT FÜR NORMUNG E. V., 2018d: p. 10) and provides protection against repetition in cryptographic operations. In the Blockchain network the nonce is used for the consensus algorithm, i.e., the mathematical problem nodes or so-called miners must solve. Consensus is essential in the Blockchain network and particularly for cryptocurrencies such as Bitcoin. Consensus is a necessary process for the agreement about a final state of data among distributed and distrusting nodes and ensures, amongst other, validity, fault tolerance, and integrity. For that purpose, several consensus mechanisms prevail that can be selected according to the specific type of Blockchain (see last paragraph of this subchapter). The most prominent consensus mechanism, due to being implemented in the Bitcoin network, is the *proof of work*⁸⁶. Consensus mechanisms in distributed data bases, particularly in P2P networks, are an enabler to achieve traceability and integrity of data for peers who need due to, for instance, for legal reasons, a verified transaction history (BASHIR, 2018: pp. 35–37; SIXT, 2017: p. 13; VEREIN DEUTSCHER INGENIEURE, 2004a: pp. 8–11).

Smart contracts are decentralized, secure programs which denote an automatically executable and self-enforcing agreement. In the case of the Blockchain technology, smart contracts are small programs placed in the Blockchain code with a certain business logic agreed upon, i.e., consensus protocols for publicly specified programs (BASHIR, 2018: pp. 53–54, 261–262; NARAYANAN and CLARK, 2017: p. 20). In case of financial transactions, smart contracts could, for instance, trigger an event in the case of incoming payment. A smart contract can be considered an *if-then-relation* as known from programming. In automotive engineering IT, smart contracts could encompass all

⁸⁵ Please refer to NARAYANAN et al. (2016), BASHIR (2018), ANTONOPOULOS (2015), and ANTONOPOULOS (2017) for further information on public private key infrastructures, digital signatures, and Merkle trees.

⁸⁶ In the *proof of work* consensus mechanism, originally suggested as an anti-spam mechanism, so-called miners invest computational power (“work”) to seek or guess the nonce. This yields, out of the given hash function, a hash value small enough, i.e., with a varying number of leading zeros, to satisfy the difficulty level given by the Blockchain code according to the network’s immanent computing power. When the computing power increases, the difficulty level correlates positively and hence miners must test more nonces in order to get a small enough hash value. Miners are rewarded for their effort – in case of Bitcoin with Bitcoins. This reward incentivizes the protection of the integrity of the distributed ledger or otherwise double-spending and hence worthlessness would occur (NARAYANAN et al., 2016: pp. 64–65; NARAYANAN and CLARK (2017: pp. 11–17). For further consensus mechanism and details about the *proof of work* mechanism, please refer to BASHIR (2018), FRANCO (2015), and BOHN (2007).

Boolean expressions of the feasibility of components' combinations. Hence, the entire configuration management could be integrated in automatically executed smart contracts that check whether a change of a component is compatible or not. This is not in scope of this work due to the tremendous complexity and will be left for future research⁸⁷ (cf. Chapter 7.2).

There prevail different types of Blockchains that have been developed and adapted for diverse purposes. The initial and most common Blockchains are public due to the aim for circumvention of a financial intermediary (vide supra). This kind of Blockchains is not possessed by anyone and anyone can set up a peer and join the network. Hence, the Blockchain is called *unpermissioned* as all peers have a copy of the ledger on their local nodes and via a distributed consensus mechanism the eventual content of transaction blocks and hence the eventual state of the Blockchain is decided upon. Conversely, a private, permissioned or consortium Blockchain is restricted to a dedicated circle of peers who have agreed to utilize this distributed ledger⁸⁸. Access has to be granted and hence data of transactions, participants, and the network are secured amongst members. A consensus mechanism, such as the *proof of work*, is not necessary because the truth of the ledger also can be found using a simpler agreement protocol. In the case of an agreement protocol, all verifiers are known and preselected. In this work, consensus mechanism and agreement protocol are used synonymously due to both aim at the decision on the truth of data in the distributed ledger and differ only by the underlying mechanism to execute and achieve this. Completely private and proprietary Blockchains contradict the underlying principles of decentrality, publicity, and irreversibility. Hence, there are scarce applications for this kind of Blockchain. However, there might be intra-company use cases where data has to be shared and details of this data transfer have to be guaranteed legally or for traceability purposes (BASHIR, 2018: pp. 30–34; GLASER, 2017: p. 1548).

⁸⁷ Please refer to BASHIR (2018), PRUSTY (2017), and NARAYANAN and CLARK (2017) for further information about smart contracts, oracles, Ricardian contracts, and decentralized autonomous organizations (DAO).

⁸⁸ A permissioned Blockchain does not have to be private, given that a public Blockchain using an access control layer also can regulate participation of peers (BASHIR, 2018: p. 33). However, the introduction of any access control implies a certain authority deciding upon the requirements of access control etc. and hence a fraction of privacy is induced. Therefore, in this work, private and permissioned Blockchains are considered to be congruent.

2.7.4 Traceability in the context of data base solutions

Taking into account the context of data base solutions that are assessed here, namely in the automotive E/E development in distributed engineering collaborations, the following assessment of traceability for data base solutions will also be limited to this scope. There also exists partial congruency with traceability in the context of PDM/PLM IT systems in Chapter 2.3.3 due to those systems also using one of the above-mentioned data base solutions.

DATA MODEL

The data model implemented in data bases has to enable traceability of data objects and artifacts by offering the necessary references, dependencies, and to which model they belong (FELDHUSEN and GEBHARDT, 2008: pp. 79, 133). This is usually done in the first step, on an atomic level. An attribute is atomic if the corresponding attribute entry is only assigned one element. Transactions in data bases are considered atomic if either the entire transaction is executed completely, or not at all⁸⁹ (SCHICKER, 2017: pp. 18, 28). The next level where traceability in data models has to be ensured is the sub-assembly or module level where one engineering discipline models its product in scope, such as in E/E one ECU or one software function. The overlying level of data models is the assembly or final product level, with multiple intermediate levels where required. A data model on the metalevel, where configurations and their subsets across the lifecycle are handled, has to be enabled in order to foster traceability in and between data base solutions.

PROCESS MODEL

Data base solutions in automotive E/E development in distributed engineering collaborations have to have the capability to represent the necessary process model for PDM/PLM (cf. Chapter 2.2) (FELDHUSEN and GEBHARDT, 2008: p. 79). Especially when multiple data bases are involved, the holistic integration of processes becomes challenging, but is crucial.

TECHNOLOGY

When considering a central data base, which is controlled by the OEM, data is available to engineering partners and suppliers either via direct access to the data base or via transfer of data. The technology of the data base has to ensure basic traceability

⁸⁹ For idiosyncrasies regarding the data model of transactions in case of the Blockchain in comparison to traditional transaction systems, please refer to KRUIJFF and WEIGAND (2017).

mechanisms. The documentation of read/write actions of dedicated users and the documented history of an artifact can be taken for granted nowadays. In the case of distributed engineering collaboration, a central data base has to provide a standardized API for the up- and download of data for suppliers. Hence, traceability in a central data base mainly depends on the data model. When decentral data bases are in scope, the technological aspects become more prominent. Coherence, i.e., the connection of related artifacts, as well as consistency, i.e., that replications of data which have the same version, and correctness, whereby changes of data are propagated, have to be ensured to foster traceability. By mastering this complexity, decentral data bases can perform their advantages, especially when data is replicated completely (HECKMANN et al., 2006: pp. 7–17; STIEFEL, 2011: pp. 57–62; JOHNSIRANI and NATARAJAN, 2015: p. 120). Particularly, between P2P systems where data is often replicated completely, interoperability is a paramount challenge. If P2P systems were standardized, migration and development of P2P would be facilitated and hence traceability of data due to standardized technology would be fostered (BOHN, 2007: p. 287).

2.8 Ontologies

2.8.1 Definitions, norms, and standards

Comprehension in terms of communication based upon knowledge about the real world or fractions of the real world, respectively, require a common knowledge model. This must be understood by all participating humans and machines. The unique replication and illustration of the knowledge to be transferred is enabled by a knowledge model. For that purpose, ontologies describe the conceptional formalization of artifacts and their relations to each other. Ontologies represent hierarchies of terms in context-specific knowledge structures by means of connected, disjunct taxonomies⁹⁰. Furthermore, categories and rules to depict immanent connections are modeled. Ontologies are mostly depicted as undirected graphs, in contrast to taxonomies, meaning there is no interpretable hierarchy from top to bottom but rather connections and relations of terms within in a knowledge area. A node illustrates the terms and an edge depicts the relation between nodes. These undirected graphs with given semantics can stand for a semantic net and show the meaning of the data and its relations (GAUSEMEIER et al., 2014: pp. 57, 59–60; DORSCHER, 2015: pp. 317–318).

⁹⁰ Cf. Footnote 40 on p. 48 for the definition of taxonomy.

So that ontologies are computer interpretable, they have to be formally specified. For this, there prevail a set of formal description languages which can be serialized in XML and hence be processed further (GAUSEMEIER et al., 2014: p. 60). The main description languages for ontologies will be described in the following. These standards originally stem from the vision of the semantic web, the extension of the world wide web by means of structure and meaning, which was assigned to the data. These standards become more and more relevant in other disciplines. This is due to most devices are connected to the internet today and each internet-compatible device is equipped with technology based upon these ontological standards (SAKR et al., 2018: pp. 1–2; GAUSEMEIER et al., 2014: pp. 59 ff.)

RESOURCE DESCRIPTION FRAMEWORK (RDF)

The RDF provides a model for the representation of metadata, e.g., information about websites and other objects, and thus enhances automated information processing in internet-based information systems. For that purpose, resources are identified by an URI and are delineated by a triple consisting of a subject, a predicate, and an object, where the subject and object form the nodes and the predicate is a labeled edge between the nodes⁹¹. RDF is a language for knowledge representation and hence an ontology. RDF-scheme (RDFS) extends RDF by means of implementation of a semantic scheme or a vocabulary. In principle, it is feasible to exchange product models between different applications by means of RDFS without losing the original meaning. However, RDFS is not expressive enough and, therefore, the web ontology language (OWL) was developed (GAUSEMEIER et al., 2014: p. 60; SAKR et al., 2018: p. 4; STIEFEL, 2011: pp. 39–40; HITZLER, 2008: pp. 35, 38).

WEB ONTOLOGY LANGUAGE (OWL)

In order to describe terms of one discipline and make them machine-readable, the OWL was developed. It depicts ontologies by means of a formal descriptive language and enables the publication and distribution of these ontologies. The W3C standardizes OWL and expands the meaning of RDF by further constructs. On the one hand, for instance to increase the mightiness of expression, on the other hand also to limit the mightiness to avoid ambiguity. OWL uses classes, properties, and individuals. Object properties describe relations between individuals, and datatype properties depict individuals' properties. Hence, OWL provides a more enhanced framework than RDFS

⁹¹ Triple graph grammars already are in scope of research for many years. For an overview and more technical details, please refer to SCHÜRR (1995).

to depict complex knowledge (GAUSEMEIER et al., 2014: pp. 60–61; STIEFEL, 2011: pp. 39–40; SAKR et al., 2018: p. 4).

Together, OWL and RDF/RDFS are part of the so-called semantic web stack or layer cake by the W3C consortium, as well as the already above-mentioned standards (cf. Chapter 2.7.3). The semantic web layer cake is depicted in Figure 2-24. Not all of its components and standards are in scope here.

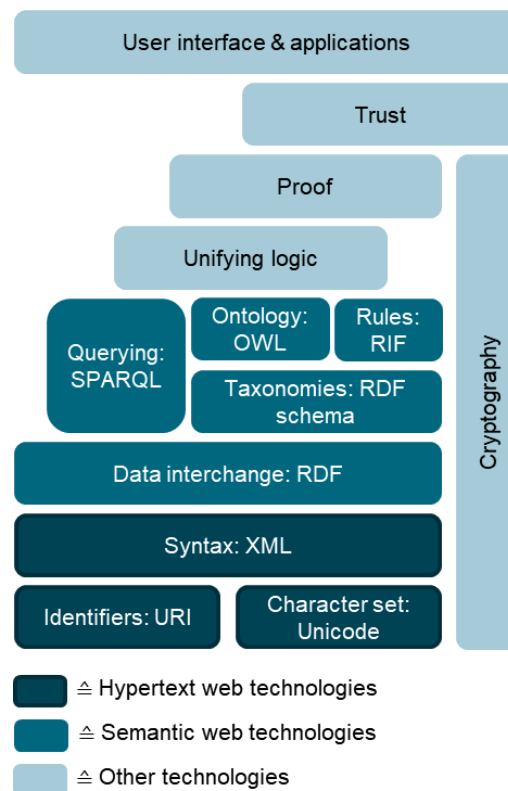


Figure 2-24: Semantic web layer cake (in alignment to SAKR et al., 2018: p. 4).

OPEN SERVICE FOR LIFECYCLE COLLABORATION (OSLC)

The open service for lifecycle collaboration (OSLC) provides a uniform infrastructure for interfaces between distinct systems. OSLC offers direct and neutral interfaces, i.e., data is directly linked and not exchanged between systems and interface descriptions are neutral and public. OSLC links data by means of an URI, as also depicted in Table 2-1. OSLC was primarily developed for application lifecycle management (ALM) but also gains more importance in the realm of PLM. However, due to immaturity of some specifications of OSLC, penetration of this integration standard in industrial practice is not very advanced yet. Loose coupling of heterogeneous tools and systems are prone to the implementation of OSLC because OSLC combines standardized interfaces with an overlying metadata. Internet technologies and linked data principles serve as the basement for OSLC. The interfaces use common HTTP commands such as *get*, *put*,

post, *delete*, etc. Representational state transfer (REST) serves as an enabler mechanism for distributed, loosely coupled APIs by means of stateless transactions, i.e., all REST messages include all information for clients and servers to understand the message. Therefore, messages are self-contained and neither node has to save information about the state. On top of this basic internet technologies, OSLC implements metamodels for specific domains which face a high necessity of inter-discipline collaboration, e.g., requirements, change, and quality management as well as ALM and PLM. For this purpose, OSLC also makes use of STEP AP233 (cf. Chapter 2.6.1). However, the OSLC PLM reference model is only a draft since 2011 (BACHELOR, 2011). The standardized interfaces and already partially standardized metamodels for different disciplines within the OSLC standard can contribute to an interoperability between IT systems and tools as well as their data models. Hence, OSLC fosters traceability in engineering from a data perspective⁹² (SINDERMAN, 2014: pp. 331–332; KIRSCH et al., 2017c: p. 171; MECPRO² ABSCHLUSSBERICHT, 2016c: pp. 139–140; OASIS, 2019: pp. 4–5; HOOSHMAND et al., 2018: p. 109; RYMAN, 2013: p. 2). OSLC does not standardize business processes which, in turn, have to be considered separately (PFENNING, 2017: p. 118).

2.8.2 Traceability in the context of ontologies

DATA MODEL

SELLGREN (2009) emphasizes the importance of modeled interfaces between (3D) components and argues that interfaces also have to be managed as a data artifact likewise other components where the interface function is stored as an attribute (SELLGREN, 2009: pp. 8, 11). Additionally, a common metadata model with dedicated ontologies in RDF and OWL within collaborations enables knowledge management in collaborative R&D, as already described in Chapter 2.6.3⁹³ (VACHER et al., 2007: pp. 315–317). PIMMLER, T. U., EPPINGER, S. D. (1994) suggest that for system analysis in system engineering, a system shall be decomposed into its physical elements and components (PIMMLER, T. U., EPPINGER, S. D., 1994: pp. 343–346). This also has to reflect in the ontologies when working across disciplines and within engineering collaborations. This is in order to describe variability in MBSE, for instance within the configuration and variant management, and to foster traceability across a

⁹² Please refer to <https://www.oasis-open.org/standards> for more information about OSLC.

⁹³ *Protégé* is a well-known and widely-used tool to build and manage ontologies with standards such as RDF and OWL (NOY and MCGUINNESS, 2001: p. 2).

heterogeneous IT landscape. Therefore, a federalized IT backbone is required (cf. Chapters 2.2 and 2.3). For this purpose, OSLC can be considered as an enabler and it is sufficient to link configuration items with models and artifacts in the respective IT system and data model, i.e., create a specific ontology (HOOSHMAND et al., 2018: pp. 108–109). Additionally, interfaces can be modeled explicitly using their own ontology to align MBSE, PDM/PLM, and simulation in engineering collaboration in order to foster traceability (VOSGIEN et al., 2012: pp. 612–622).

PROCESS MODEL

Initially, partners in an engineering collaboration have to agree upon a joint ontology describing the product they are all working on. This implies there is transformation of tacit into explicit knowledge and later modeling of this information into RDF and OWL. These ontologies can be updated dynamically in order to develop with the engineering processes (VACHER et al., 2007: pp. 315–317). Particularly when relations of different components or any of their properties alter, a process has to ensure that all disciplines or owners are notified of the change and, consequently, maintain traceability across disciplines and companies (SELLGREN, 2009: p. 9). For the integration of domain-specific data models into a system model, appropriate interfaces have to exist and a model-based process model has to be developed (MECPRO² ABSCHLUSSBERICHT, 2016a: p. 48).

TECHNOLOGY

In order to enable the full possibilities of ontologies, each discipline and company within an engineering collaboration has to employ the same ontology and has to have access to changes in it. Thus, a dynamic exchange and storage of ontologies has to be assured by means of appropriate data bases. This can be achieved by a federative approach (cf. Chapter 2.6) and linking artifacts directly according to the OSLC standard (EBELING and EIGNER, 2018: p. 261; ALVAREZ-RODRÍGUEZ et al., 2014: pp. 995–996).

2.9 Conclusion

An automotive industry in vicissitude demands solutions to address the resulting product complexity. Therefore, early engineering methods, tools, and processes have to address this complexity which increases if development occurs in distributed engineering collaborations. One way of doing this is to foster traceability of information artifacts across involved domains intra-company and across participating engineering partners inter-company.

For this purpose, the current state of science and technology was assessed focusing on automotive E/E development and the enablers data model, process model, and technology.

After defining traceability in this context, the product development processes and methods which are relevant for automotive development, were described. Afterwards, PDM/PLM set the focus on existing process and data base solutions to handle complexity and foster traceability during development and beyond. MBSE combines data models and process models for a holistic system view already in the early development phase. Peculiarities of automotive E/E development were described afterwards leading over to distributed engineering collaborations, in which E/E development often occurs. Highlighting technological solutions, i.e., data bases, was done in the following chapter. Ontologies were examined in the last chapter as the basis for common data models and traceability within them.

The gained insights from each chapter with respect to traceability in each domain now will be transferred to the formulation of requirements for a synthesis of a solution framework.

3 Requirements for a solution framework and evaluation of current state

3.1 Evaluation method




The objectives of this research are to foster internal and external traceability in different facets, which can already be matched to the enablers of a solution for traceability, a data model, a process model, and technology, in the early automotive E/E development within distributed engineering collaborations, as motivated in Chapter 1.3. This is in order to obtain a better understanding of what the final solution comprises and what has to be addressed during the evaluation. This is illustrated in Table 3-1. The internal traceability is addressed by the alignment of MBSE and PDM for E/E. This mainly concerns data and process model, as not only the data model has to be aligned, but the development and documentation processes having to be aligned, too. Despite technology here focuses more on the external aspects of engineering collaboration, also internal traceability is in scope as it connects different IT systems and tools. The external traceability among multiple engineering partners is addressed in a threefold manner. For the purpose of the reduction of reconciliation, all three enablers have to be aligned to foster traceability among engineering partners due to data, processes, and technologies have to work hand in hand. This is to allow for transparent and safe product changes, where again all enablers are crucial. The same holds true for the ad hoc and easy connection of new engineering partners. This assessment of which objectives are addressed by which enablers shows the complexity as all aspects have to be considered during the elaboration of a potential solution framework.

According to the tripartite enabling elements, the current state of science and technology was assessed in Chapter 2 at the end of each subchapter. Thereof, requirements for a solution will be deduced contingent on the current state of science and technology. These requirements will then be aligned with the research objectives motivated in Chapter 1.3. Then, the fulfillment of the requirements by the current state of science and technology will be measured qualitatively pursuant to the scale “not fulfilled” (○), “partially fulfilled” (◐), and “fulfilled” (●). The gray squares mean that at least one of the dimensions of the matrix does not apply. The evaluation method is depicted generically in Table 3-2.

Table 3-1: Assessment of objectives addressed by enablers.

X addressed by		Enablers		
Objectives		Data model	Process model	Technology
1. Internal traceability	a. Alignment of MBSE & PDM for E/E	X	X	X
2. External traceability	a. Reduction of reconciliation	X	X	X
	b. Transparent & safe product changes	X	X	X
	c. Alleviated connection of engineering partners	X	X	X

Table 3-2: Generic depiction of the evaluation method (in alignment to ESTEFAN, 2008: p. 10; KÖNIGS, 2013: p. 52; GILZ, 2014: 51).

			Enablers					
			State of the art: Data models		State of the art: Processes		State of the art: Technologies	
			Data Model 1	...	Process 1	...	Technology 1	...
1. Internal traceability	a. Alignment of MBSE & PDM for E/E	1. Requirement						
		...						
2. External traceability	a. Reduction of reconciliation	m. Requirement						
		...						
	b. Transparent & safe product changes	n. Requirement						
		...						
	c. Alleviated connection of engineering partners	o. Requirement						
		...						

3.2 Requirements for internal traceability

A) ALIGNMENT OF MBSE AND PDM FOR E/E

Based upon the first objective of this research, to enhance the alignment of MBSE and PDM for E/E, it is decisive to provide exactly the information necessary during early development phases. Therefore, the crucial information artifacts which are prone to changes in the E/E development have to be modeled already in the MBSE period and have to be found also in the PDM system and not only in domain-specific E/E development tools. Hence, the availability of relevant information artifacts has to be ensured across all existing IT systems and tools (cf. Chapters 2.2, 2.3, 2.4, inter alia). As an ECU's pins transfer the electrical messages in the form of charges over the communication bus systems between other ECUs, each pin has to be modeled discretely. The network communication description (NCD) serves as sort of a map for which ECU communicates with which or which ECU needs the signal of another ECU. Due to the crucial importance of the knowledge of an automobile's E/E communication and its inherent complexity, tracing this information during the development between multiple engineering partners is essential. Software realizes a lot of functionalities in a modern automobile, it is built across many departments and suppliers, and is tremendously complex. The single types of software within an ECU have to be traceable and hence must be modeled in a data model for MBSE and PDM. As mentioned above, different organizational departments in an OEM contribute to the automotive E/E system, using many different IT systems and tools. The smart connection of these systems, particularly to an IT backbone system, by a linked data approach is key for internal traceability. This leads to requirements 1 to 4.

1. **Requirement:** The solution framework for internal traceability shall comprise information artifacts for ECUs' pins (E-CAD).
2. **Requirement:** The solution framework for internal traceability shall comprise information artifacts for NCDs, including the respective communication bus systems, signals, and interfaces.
3. **Requirement:** The solution framework for internal traceability shall comprise information artifacts for ECUs' software versions, including their parametrization files.
4. **Requirement:** The solution framework for internal traceability shall comprise a linked data model in order to connect all legacy IT systems, e.g., for MBSE, E/E development, and PDM, with a common IT backbone data base.

3.3 Requirements for external traceability

A) REDUCTION OF RECONCILIATION

As elaborated in Chapters 1.2, 2.2, 2.3, 2.6, inter alia, a major issue for traceability with external engineering partners is an elevated effort for reconciliation during development. Hence, the reduction of effort for reconciliation is objective 2.a. It has been argued that there is still a plethora of deficiencies in the reconciliation process in distributed engineering collaborations, whether regarding data models, processes, or technologies (cf. entire Chapter 2). Particularly, trace links for dedicated information artifacts have to be available for all engineering partners (requirement 5). Additionally, a potential solution framework has to foster a distributed engineering collaboration and not focus on a centralized approach for the purpose of mitigating the bottleneck of communication between the OEM and suppliers (requirement 6). Requirement 7 addresses the necessity of a formalized consensus mechanism for the purpose of a faster decision-making process in engineering collaborations which, in turn, fosters traceability by transparency of changes. This is more than the release process that commonly takes place at each supplier and finally at the OEM separately. The required consensus mechanism here aims at the affirmation of all involved engineering partners for all decisive changes. Thereby, components affected by changes immediately can be identified, reconciliation circles can be reduced or shortened, and overall traceability is fostered. Accompanying this, changes and consent, or dissent about those have to be propagated automatically and instantaneously across the involved engineering partners for quicker reaction and intervention of engineers regarding their affected parts as well as reduced effort. Hereby, not a mere transmission of the information that a change has occurred is in scope. More, the focus lies on the actual propagation of metadata regarding the latest version of a product, which of the components have altered, and the documentation in all affected engineering partners' IT systems (requirement 8).

- 5. Requirement:** The solution framework for external traceability shall provide universally unique trace links to identify information artifacts across IT systems among multiple engineering partners.
- 6. Requirement:** The solution framework for external traceability shall foster distributed engineering collaboration with the focus on MBSE and E/E.
- 7. Requirement:** The solution framework for external traceability shall consist of a consensus mechanism for development changes.

- 8. Requirement:** The solution framework for external traceability shall include an automatic change propagation across all involved engineering partners.

B) TRANSPARENT AND SAFE PRODUCT CHANGES

In addition to the reduction of reconciliation during development, changes have to be transparent as well as safe (cf. Chapters 1.2, 2.6, inter alia). In case that suppliers deliver parts and software where there might be a dispute about the actual content of such a delivery, the tamper-resistant documentation about the content of information artifacts has to be provided. Therefore, an immutable product history for purposes of liability is one aspect of external traceability (requirement 9). Commonly, one supplier synchronizes its information artifacts solely with the OEM. This hinders transparency of changes across other affected suppliers. Hence, requirement 10 addresses a multi-directional synchronization of data among all involved engineering partners. A traceability scheme between an OEM and suppliers focuses explicitly only on the information artifacts' traceability that are relevant for one supplier but embeds this also on suppliers' side in the overall product structure of the OEM. Thus, information artifacts across all IT systems and tools on OEM's as well as all suppliers' sides are congruent and changes can be traced transparently (requirement 11). Fraud-save data becomes particularly relevant in the case of ad hoc contribution by start-ups. It must not be possible to compromise data integrity (requirement 12). In contrast to the above-mentioned immutable product history, data integrity emphasizes all information artifacts in an IT system, whereas an immutable product history focuses on the traceability of each change. Moreover, data integrity has to be guaranteed also if another engineering partner hosts the data base.

- 9. Requirement:** The solution framework for external traceability shall contain an immutable product history.
- 10. Requirement:** The solution framework for external traceability shall allow for multi-directional synchronization of data among all involved engineering partners.
- 11. Requirement:** The solution framework for external traceability shall include a traceability scheme for OEM and suppliers.
- 12. Requirement:** The solution framework for external traceability shall foster data integrity among multiple engineering partners.

C) ALLEVIATED CONNECTION OF ENGINEERING PARTNERS

The development of automobiles engages multiple engineering partners. As software's importance increases, also new engineering partners have to have access to the joint development data granting quicker development and cost reduction by the increase of traceability, which would not be possible in case of non- to low-integrated partners (cf. Chapters 1.2, 2.2, 2.5, 2.6, inter alia). Requirement 13 concerns a standardized data model for the exchange of information artifacts. Moreover, processes have to be aligned between engineering partners (requirement 14). The ad hoc technical integration of new engineering partners can be alleviated by standardized APIs and a connection to the established IT infrastructure (requirement 15). Data always has to be available and robust towards failure of nodes or the exit of an engineering partner, i.e., a peer in the network. Also, data has to be traceable in the network. Requirement 16 addresses this.

13.Requirement: The solution framework for external traceability shall include a standardized data model for the exchange of information artifacts.

14.Requirement: The solution framework for external traceability shall prescribe a standardized development process for all engineering partners.

15.Requirement: The solution framework for external traceability shall include standardized APIs and an integration into the legacy IT systems.

16.Requirement: The solution framework for external traceability shall guarantee the availability of data and its robustness.

3.4 Classification of the current state of science and technology

For the purpose of assessment of the above-deduced requirements, the current state of science and technology has to be classified and pre-sorted according to the most relevant prevailing solutions. Due to the vast range of topics and existing data models, process models, and technologies, the classification in advance ensures a more focused assessment below.

DATA MODELS

ISO 10303 STEP AP 233 focuses on systems engineering data, whereas *AP 242* describes the exchange of 3D engineering data (cf. Chapter 2.6.1, GILZ, 2014: p. 139). Hence, *AP 233* will be evaluated further on the basis of the requirements as state of the art regarding systems engineering and PDM data models. *AUTOSAR* is a standard for many companies for E/E and E/E-related software development and, therefore, will be

in scope (cf. Chapter 2.5.3). Due to *OSLC* aims at an improved collaboration by means of a joint linked data model, it will be evaluated hereinafter (cf. Chapter 2.6.1).

PROCESS MODELS

As the E/E development is also part of mechatronics in automotive development (cf. Chapter 2.2.2), the *V-model* of mechatronic system development (*VDI 2206*) as a widely spread development approach will be under scrutiny.

Due to E/E being within scope of this work, it is necessary to assess MBSE methods for their inclusion of the physical layer, according to the RFLP approach. This is partially a data model concern but also mainly belongs to a process model. As these MBSE methods prescribe specific processes, traceability is fostered by each consecutive step. Stopping the development process before modeling the physical layer explicitly will hinder traceability with respect to automotive E/E. The evaluation of the above-mentioned MBSE methods (cf. Chapter 2.4.2) with respect to their support of traceability and their inclusion of the physical layer is presented in Table 3-3.

Table 3-3: Comparison of different MBSE methods (cf. Chapter 2.4.2 for the authors of the different methods) (own evaluation in alignment to HEBER and GROLL, 2018b: p. 127, 2018a: p. 284).

Method	Supports traceability via	Includes physical layer how
Alt, Oliver	Allocation relationships (<<allocate>>)	Analogous to SysML (system breakdown, system elements)
FAS-Method	Relationship matrix	"allocate" link according SysML
FAS4M	Federative system model	Explicit modelling of physical elements in connection to CAD-models
Harmony-SE	Trace links within/between tools, <<satisfy>>	Analogous to SysML (system breakdown, system elements)
Hold & Perry	Traceability view: trace, refine, validate	Analogous to SysML (system breakdown, system elements)
OOSEM	Process, relationships in matrix (derive, satisfy, verify, refine, trace, copy), functional allocation	Explicit modelling of physical node architecture incl. SW-HW-data-architecture
SE-VPE	Allocation matrices; approach to integrate functional SysML artifacts in PLM systems	Only M-CAD, no dependencies for E-CAD and CAE, no dependencies on property level
SPES	Relations between different viewpoints and abstraction layers by dint of allocation links.	Technical viewpoint incl. SW and HW (physical architecture)
SYSMOD	Trace relationship <<TracedTo>> and <<TracedFrom>>	Explicit modelling of physical product architecture, behavior, and relations

It is noteworthy that the *SE-VPE* method includes a dedicated approach of how to transfer MBSE information artifacts to PLM (cf. Chapter 2.4.2). However, a distinct

integration of E/E in the physical layer lacks on lower level of detail, such as regarding pins. On the other hand, the *SPES* method has a very pronounced physical layer including detailed descriptions of software, hardware, and their connections on the level of ports and signals and, therefore, will be assessed regarding the fulfillment of the specified requirements.

TECHNOLOGIES

On the basis of a plethora of technologies, i.e., tools and IT systems for systems as well as E/E and SW development, traceability, PDM/PLM, and data bases, the scope has to be classified in advance, as mentioned at the beginning of this chapter.

Tools which primarily focus on the visualization of traceability, such as *LOOME*, *METUS*, and *ToolNet*, are not in scope due to their limitations regarding the ad hoc management and modification of data and integration with other tools (cf. Chapter 2.1.3).

For PDM/PLM, *3DEXPERIENCE platform* is not considered further due to *Cameo Systems Modeler*, also by *Dassault Systèmes*, is suboptimal regarding the ability to handle OSLC (cf. Chapters 2.3.3 and next paragraph). Hence, the integration between MBSE and PDM/PLM only can be achieved via extra tools. The partial support of OSLC disqualifies *Teamcenter* to be evaluated furthermore. The availability of OSLC, full PDM/PLM functionalities, and the full integration with the *Windchill Modeler* for MBSE makes *Windchill PDMLink* a valid candidate to come under further scrutiny (cf. Chapter 2.3.3). As *OpenPDM* mainly provides features for data exchange and serves as an API but lacks further PDM/PLM data management possibilities, it will not be in scope any further as a technical solution (cf. Chapter 2.6.3). *Aras Innovator* can only handle OSLC with the help of an adapter (cf. Chapter 2.3.3). In contrast to *PTC* and its *Windchill* platform, the *Aras* company does not provide a MBSE authoring tool from the same vendor. *Aras Innovator* does not fully integrate the system model but creates a separate so-called system architecture model only linking user-defined elements (PFENNING, 2020: pp. 4–5). The lack of complete integration of the MBSE and PDM data models, both MBSE and PDM tool do not come from one vendor, as well as non-native OSLC support excludes the *Aras Innovator* from further scrutiny in comparison to *Windchill PDMLink*.

Regarding MBSE development tools, the *Rhapsody Model Manager* is not nominated for further assessment given that *IBM* does not provide any PDM/PLM tools and solely
































































relies on partners such as *Dassault Systèmes* or *PTC* (SENDER, 2009: pp. 203–204). The *Cameo Systems Modeler* can only handle OSLC via plugins and additional tools, hence it is not in scope anymore. The tool *Enterprise Architect (EA)* only supports the provision of OSLC artifacts but for optimal engineering collaboration also the consumption would be required. Hence, *EA* does not qualify further. Due to OSLC compatibility and integration with *PDMLink*, the *Windchill Modeler* will be evaluated as the preferred MBSE tool (cf. Chapter 2.4.3).

Bitbucket includes software development as well as collaboration functionalities and is based on the widely used *Git*. Therefore, *Bitbucket* will be under further evaluation as a software development tool.

PREEVision supports MBSE as well as *AUTOSAR* and is very common in automotive E/E development. However, *PREEVision* does neither support a decentralized approach nor OSLC. Nevertheless, *PREEVision* is included in the further evaluation due to its high prevalence.

A data base technology serves as the fundamental pivot in engineering collaborations. As presented in Chapter 2.7, central and decentral technologies can be differentiated. Due to the exchange of data in engineering collaborations is in scope, a central data base also is evaluated according to how it is suited to exchange data and according the corresponding criteria. Furthermore, decentral data bases, e.g., linked data and peer-to-peer in different facets, are compared in Table 3-4 according to data base properties relevant in distributed engineering collaborations. In the case of the Blockchain technology, consistency of data is achieved eventually. This is due to the consensus or validation mechanism by multiple nodes which need a certain time for solving the mathematical puzzle and agree upon its result (BASHIR, 2018: p. 40). The evaluation reveals that the Blockchain technology might be advantageous for distributed engineering collaborations.

Table 3-4: Evaluation of different types of data bases with respect to peculiarities in collaborations (own evaluation in alignment to STIEFEL, 2011: pp. 57–62; HECKMANN et al., 2006: pp. 7–17).

Properties of data bases in collaborations		Central data base (to exchange data)	Decentral data base			
 not fulfilled	 partially fulfilled		 fulfilled	Linked data (single sources w/ APIs)	Peer-to-Peer (single sources w/ overlay network, structured)	Blockchain (P2P, redundant data, unstructured)
Data transmission		transfer	ad hoc, linked	ad hoc, transfer	continous, transfer	
Data sovereignty		centralized	co-centralized	co-centralized	distributed	
Adaptivity	Scalability (quantitatively)					
	Stability (quantitatively)					
	Flexibility (qualitatively)					
Trustworthiness	Reliability	Availability				
		Dependability				
		Robustness				
	Security	Confidentiality				
		Integrity				
		Availability				
		Accountability				
Efficiency	Performance/Effort					
Validity	Localizability					
	Coherence					
	Consistency					
	Correctness					

3.5 Concluding evaluation of current state of science and technology

The qualitative evaluation of the current state of science and technology is depicted in Table 3-5. The evaluation method has been described in Chapter 3.1. The evaluation has been executed given that the motivated requirements stated in Chapters 3.2 and 3.3 have been deduced from the research objectives in Chapter 1.3 and the current state of science and technology in Chapter 2. The selection and classification of the current state of science and technology, or “state of the art” for reasons of length, was elaborated in Chapter 3.4.

It is apparent that the available solutions do not fully address traceability of E/E artifacts during automotive development with respect to the connection of model-based systems engineering in distributed engineering collaborations. Some specific data models address the MBSE as well as E/E development to some extent. Existing processes partially include E/E specifics but, of course, hardly address peculiarities of distributed engineering. Certainly, modern IT development tools strive for traceability of information

artifacts in distributed engineering collaborations. However, not all tools have the capabilities to deal with special E/E data. Moreover, some tools use a centralized data base approach which might have advantages if working exclusively within one company. However, such a tool reaches its limits regarding data integrity or availability if external engineering partners want to contribute to the development. The lack of emphasis on collaboration-specific functionalities, such as a consensus mechanism and an automatic change propagation, becomes apparent. A major requirement for distributed engineering, also with ad hoc contributions by new partners, the immutable product history, is only guaranteed by the Blockchain technology as it is designed as a distributed ledger.

Finally, it can be stipulated that few approaches unify a congruent intersection of the defined objectives and requirements regarding traceability in the early automotive E/E development within distributed engineering collaborations.

Consequently, a solution approach as a framework consisting of the three enabling elements, data model, process model, and technology has to be conceptualized and evaluated for the purpose of addressing the deficiencies of the above-mentioned approaches and solutions.

Table 3-5: Evaluation of the current state of science and technology according to the defined requirements in alignment to the research objectives.

			Enablers								
			State of the art: Data models			State of the art: Processes		State of the art: Technologies			
			SE	E/E & SW	Collaboration	Development	MBS E	PDM/PLM	MBS E	Software	E/E
			STEP AP 233	AUTOSAR	OSLC	VDI 2206	SPES	Windchill PDMLink	Windchill Modeler	Bitbucket	PREVEVision
Objectives	Requirements										
1. Internal traceability	a. Alignment of MBSE & PDM for E/E	1. Include ECU pins									
		2. Include NCD									
		3. Include ECU SW									
		4. Linked data model									
2. External traceability	a. Reduction of reconciliation	5. Trace links available									
		6. Foster distributed engineering									
		7. Consensus mechanism									
		8. Automatic change propagation									
	b. Transparent & safe product changes	9. Immutable product history									
		10. Multi-directional synchronization									
		11. Traceability scheme OEM-supplier									
		12. Data integrity									
	c. Alleviated connection of engineering partners	13. Standard data model for exchange									
		14. Stand developm. processes									
		15. Standard APIs/integration in legacy IT									
		16. Availability of data & robustness									



not fulfilled



partially fulfilled



fulfilled



not applicable

4 Synthesis of a solution framework

In order to foster traceability in engineering collaborations it has been suggested that there have to be three prevailing enabling elements: data model, process model, and technology (cf. Chapter 1.4). Pursuant, the current state of science and technology has been investigated and classified according to which existing solutions shall be considered for a potential solution approach (cf. Chapters 2 and 3). Deduced from the current state of science and technology and aligned with the research objectives, the requirements have been elaborated. Opposing, the enablers, represented by the current state of science and technology, have been evaluated regarding their fulfilment by the research objectives, characterized by the requirements, and the potential for enhancement has been derived. Following this analysis, the synthesis (cf. Footnote 51 on p. 55) strives at the description of a solution approach alleviating the above-mentioned shortcomings by the creation of a framework consisting of the enablers and their operationalization on a practical level, for instance as a dedicated data model, process model, and IT tool or system.

In Chapter 4.1, the data model for enhanced traceability in the early automotive E/E development with focus on MBSE and PDM will be elaborated. Here, the different aspects of the internal traceability (objective 1) between IT tools for MBSE and PDM, that often serves as the linchpin for successive processes, is in scope. Without the thorough integration of these two domains, traceability hardly can be achieved as PDM information artifacts are often built upon by other domains during development as well as downstream processes, such as production. Therefore, first the relevant information artifacts from a system theoretical point of view with emphasis on E/E, have to be defined. Secondly, the required specifics for automotive E/E development, i.e., those information artifacts particularly relevant for traceability in distributed engineering collaboration as described by requirements 1 till 4, have to be modeled and aligned with the information artifacts from the system theory for MBSE. Consecutively, information artifacts for PDM/PLM in alignment to the previously defined information artifacts from system theory and automotive E/E development will be elaborated. Afterwards, the definition of a fundamental data model and its construction as an ontology will be performed. Coincidentally, the data model for the final product, i.e., an automotive ECU including its different types of software, and the data exchange-related aspects of

collaboration have to be addressed. This means that not only the information artifacts describing the product and its distinct aspects of each development step but also the product documentation has to be included in the data model. Additionally, it has to be considered which metadata, in case of a distributed engineering approach, shall be transferred in order to enable a linked data approach (requirement 4).

In a distributed engineering collaboration, the information of what someone does and when during development is decisive. In Chapter 4.2, a process model including these consecutive steps will be defined. These individual tasks must be known to each partner of the engineering collaboration for the purpose of a joint development of one final product. Subsequently, the information artifacts of each new process step have to be shared within the engineering collaboration's IT network so that each partner is aware of the most recent status of development. This is in case of relevant changes which might also affect one's own sub-product, making it possible to investigate these changes in a timely manner, in order to circumvent incompatibility at a later point, leading to further reconciliation circles and potential delays. All the relevant process steps of MBSE in alignment with PDM/PLM have to be outlined. This includes the initial MBSE process, creation of a new product in PDM, a process for the creation of a configuration and a variant, as well as versioning and a change management process for existing information artifacts. The identification of incorrect or outdated information artifacts has to be addressed by an inactivation or deletion process.

Crucial for traceability within one company and between many companies, too, is an interconnected and integrated IT landscape. As deduced from the evaluation of the current state of science and technology on basis of the requirements, for a distributed engineering collaboration a decentral data base can be considered advantageous. Therefore, a fundamental IT architectural framework will be motivated. Following, this conceptual IT architecture has to be aligned with a generic IT architecture in automotive E/E development and PDM/PLM for the purpose of enabling of an integrational approach into existing IT landscapes, so-called "brownfield" (WADE et al., 2018: p. 1176). This will be explained further in Chapter 4.3.

Chapter 4.4 will present the final framework to foster traceability of E/E information artifacts during automotive development in consideration of model-based systems engineering within distributed engineering collaboration and where the requirements are satisfied.

4.1.1 Definition of the relevant information artifacts

GENERAL SYSTEM DEFINITION AND CONCEPTS

In order to define the relevant information artifacts for the respective system within the final automobile, it has to be defined what constitutes the system and which are its boundaries, physical or functional connections, and interfaces within or with other systems. At this juncture, it can be distinguished between

- i). a functional concept,
- ii). a structural concept, and
- iii). a hierarchical concept of a system.

These concepts are not disjunct but rather conjunct views of one system with different foci⁹⁴ (see Figure 4-2). The functional concept depicts the system as an aggregation of properties and their correlation. These properties are mainly inputs and outputs and corresponding states of the system. This depiction is known as a “black box”. The relationship of elements within the system is in scope of the structural concept. Such elements could be different components of a system. The structural view emphasizes the interdependencies between parts of a system which shall not be examined in isolation but rather in their context. The hierarchical concept can be considered as cascading in the sense that elements of a system can be systems and the system itself again can be regarded as part of a more comprehensive system. Hence, a system is a model of an entity which i) possesses relations between attributes (e.g. inputs, outputs, states, etc.), ii) consists of concatenated parts or sub-systems, respectively, and iii) is confined from its environment or from a super-system (ROPOHL, 2009: pp. 75–77).

Bearing these different representation concepts with their complementary foci in mind, the next section will incorporate the alternative depictions of a system by explicitly modeling them by means of the modeling language SysML to create a reference model for an automotive E/E system.

⁹⁴ The socio-technical dimension, wherein the technical system interacts with humans, is not in scope of this work (cf. ROPOHL, 2009: pp. 58, 135 ff.).

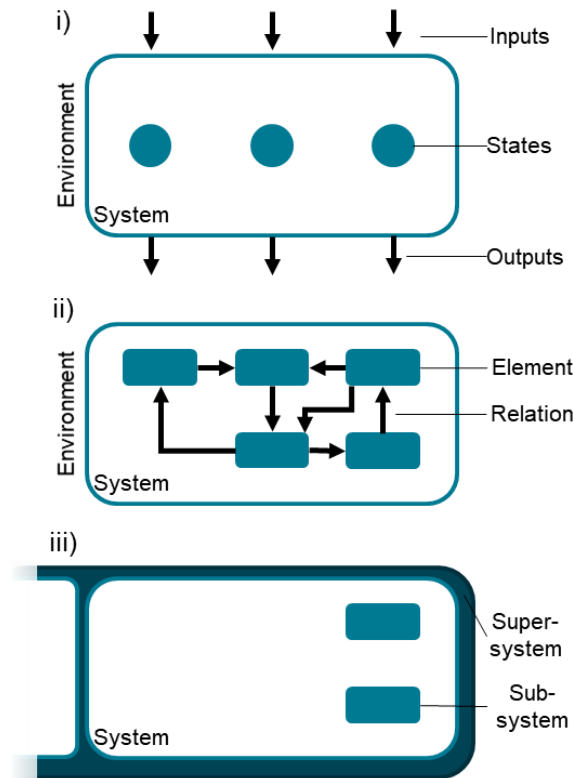


Figure 4-2: System concepts: i) functional, ii) structural, iii) hierarchical (in alignment to ROPOHL, 2009: p. 76).

COMPOSITION OF A REFERENCE MODEL OF AN AUTOMOTIVE E/E SYSTEM

A metamodel is a model of itself which is implemented to describe a modeling language. The modeling language SysML deploys this logic of a self-describing metamodel (cf. Chapter 2.4.2) (ALT, 2012: p. 24). For the purpose of the description of an automotive E/E system and due to a de facto standard in the systems development (MBSE) (cf. Chapter 2.4), the modeling language SysML is used to depict the reference model. The generic SysML taxonomy or metamodel is depicted in Figure 4-3 and consists of the *activity diagram*, *sequence diagram*, *state machine diagram*, *use case diagram*, *block definition diagram*, *internal block diagram*, *package diagram*, and *parametric diagram*. The foremost four diagrams are used to model the system's behavior. The latter four diagrams represent the structure. Additionally, there is a diagram to model the requirements. Hence, each type of the nine unique diagrams considers a special aspect of the system. The structure diagrams are commonly used to describe the system architecture which primarily is in scope here (HOOSHMAND, 2015: p. 59). The arrow with a white triangle ("△") in Figure 4-3 stands for a generalization of blocks⁹⁵ (OMG, 2015:

⁹⁵ For more information about SysML notation, please refer to DORI and CRAWLEY (2016), VAN RANDEN et al. (2016), OMG (2015).

p. 38; DORI and CRAWLEY, 2016: p. 33). For conformity reasons for consecutively following data model descriptions, not all SysML notations are used.

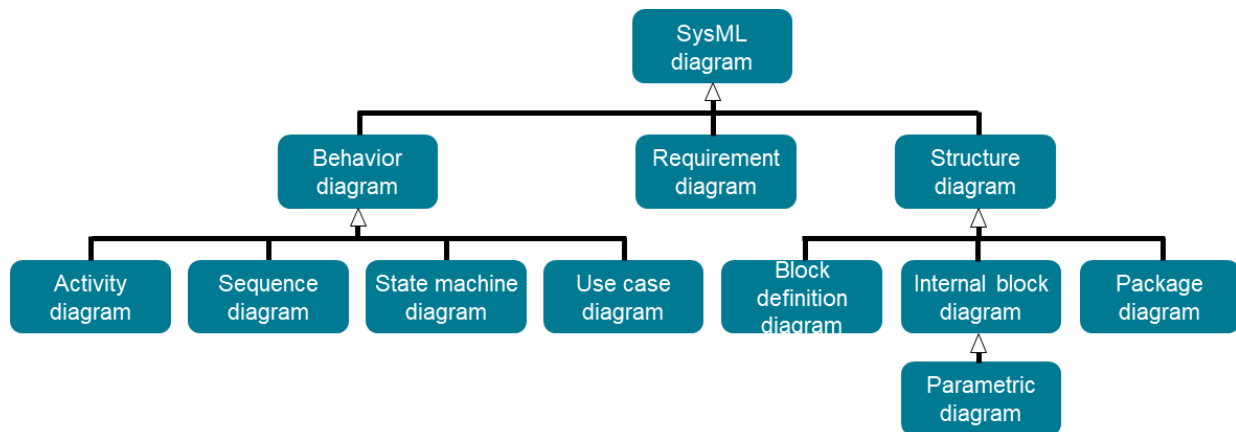


Figure 4-3: SysML taxonomy (in alignment to OMG, 2015: p. 187; FRIEDENTHAL et al., 2012: p. 30).

The SPES method, which is used to model the data model for MBSE, also extends to a metamodel. However, the metamodel has its scope on an architectural view of a system and does not include the relevant specifications of automotive E/E development that are pivotal here. Albeit, the metamodel includes the modeling of dedicated pins as a point of interaction, also for communication with transmission of signals, further information artifacts necessary for traceability in the automotive E/E development are not explicitly modeled. For instance, a NCD matrix is not represented distinctly within the communication data; neither are the three main ECU software components bootloader, functional, and parametric software. The description of software remains on a higher granularity level and only refers to it as a *system artifact*, on an abstract, coarser level, or a *rich component*, on a concrete, finer level that still can be distinguished into a function, logical, or technical component, inter alia⁹⁶ (WEBER et al., 2012: pp. 17, 27–30, 65–67, 126–127).

The SPES method, obviously, focuses on software-intensive systems and hence this approach is obvious, yet too generic for traceability discussed in this work here where certain information artifacts have been identified to be crucial for traceability in automotive E/E development in distributed engineering collaborations. Therefore, the elaborated data model here takes into consideration the information artifacts, which are not explicitly in scope of the SPES method and its architecture metamodel, and implements a generic data model specifically aligned to automotive E/E development.

⁹⁶ For more information about the SPES metamodel, please refer to WEBER et al. (2012).

By those means, a greater traceability particularly within the automotive E/E development is fostered.

For the purpose of holistic modeling of the system architecture, system requirements, and system behavior for a continuous and traceable MBSE, commonly all types of diagrams of the SysML modeling language are necessary (HOOSHMAND, 2015: p. 60). However, here in this work the pivotal attention lies upon those system elements that are essential for traceability in engineering collaboration as well as with a later documentation in PDM/PLM systems. Consequently, not all diagram types will be implemented.

The reference model is modeled in the package diagram which serves as an organizational storage location and captures all relevant model elements contained in the system. The package diagram again includes several packages with further elements (FRIEDENTHAL et al., 2012: pp. 53–55, 104-105). Figure 4-4 shows the reference model for a generic automotive E/E system with the relevant packages in scope of this work. *Behavior*, *use cases*, and *requirements* packages are in grey as these models will not be elaborated on in more depth. Due to an emphasis on automotive E/E, input/output (I/O) definitions and parametrics are added separately wherein the first encompasses model elements required for the specification of interfaces including ports as well as their inputs and outputs (FRIEDENTHAL et al., 2012: p. 55). The latter contains parametric data particularly necessary for automotive software development (cf. Chapter 2.5). A model library package, denoted with `<<modelLibrary>>`, includes all system elements and structural elements, respectively, such as hardware and software components, as well as the associated properties and information. These elements can be referenced and imported by other diagrams. This also applies to *IT systems* (HOOSHMAND, 2015: p. 60; FRIEDENTHAL et al., 2012: p. 369). The development process, which shall foster traceability in MBSE among multiple engineering partners, will be modeled in the package *processes*. Configuration elements can be found in the homonymous package. The symbol “⊕” in Figure 4-4 means that the packages have a hierarchical relationship and denotes a containment (FRIEDENTHAL et al., 2012: p. 55).

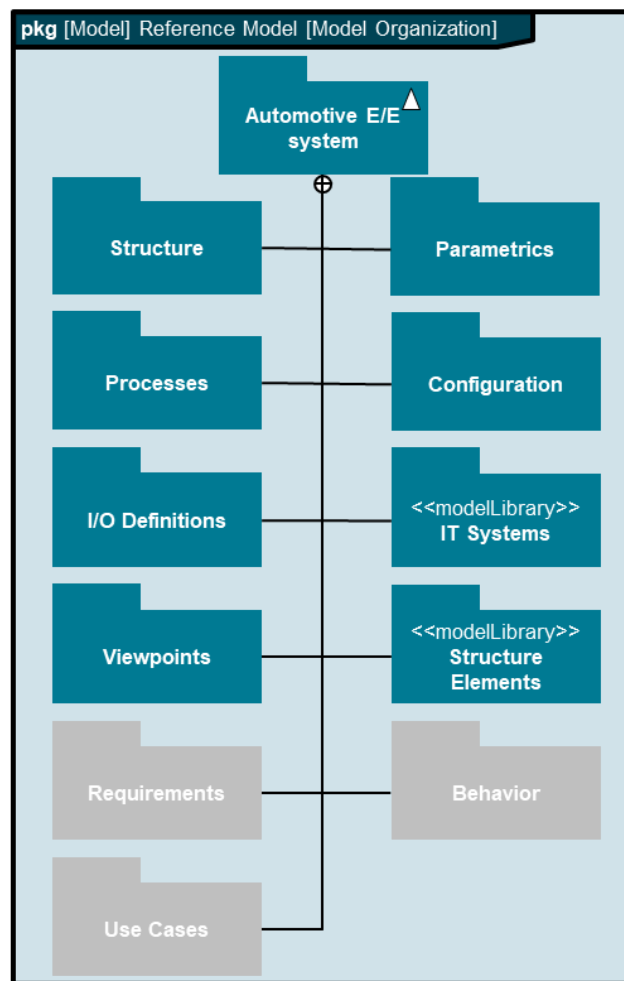


Figure 4-4: Reference model for an automotive E/E system depicted as a package diagram in SysML. Gray packages are not in scope of this work.

In the following sections, the building blocks for each package will be described to depict the reference model. Moreover, their relations will be described, too. First, the system structure will be modeled, then development process specifics including the viewpoints, which address the RFLP approach, will be considered. Afterwards, peculiarities of automotive E/E including I/O definitions as well as parametrics will be further detailed. Information artifacts for configuration management will be addressed in the following. In a final step, a dedicated data model for the involved IT systems will be depicted for the model library. The model library for structure elements does not have to be described separately due to all elements already being addressed in the previous packages.

DEFINITION OF THE GENERIC STRUCTURE OF THE AUTOMOTIVE E/E SYSTEM MODEL

The generic structure of the automotive E/E system model is depicted in Figure 4-5. For the purpose of visualization, in the SysML logic a so-called *block definition diagram* (bdd) is used to formalize structural relations between different blocks. A block in the SysML is an entity or element with distinct properties and features (FRIEDENTHAL et al.,

2012: pp. 57, 120). The arrow with one black lozenge-shaped end (“◆”) describes a compositional relationship between the “whole”, i.e., then end with the “◆”, and the part of the composite, i.e., the end with the arrow (“→”) (OMG, 2015: p. 38). A composite characterizes a relationship between the child and parent, where the child cannot exist independently from the parent. Of course, an ECU or a communication bus can exist without the relation to an E/E system. However, in automotive E/E this rarely will be the case. Hence, for practical relevance, the relation of the E/E system with other blocks within the bdd will be a composition each. Further attributes and additional details are not yet included here.

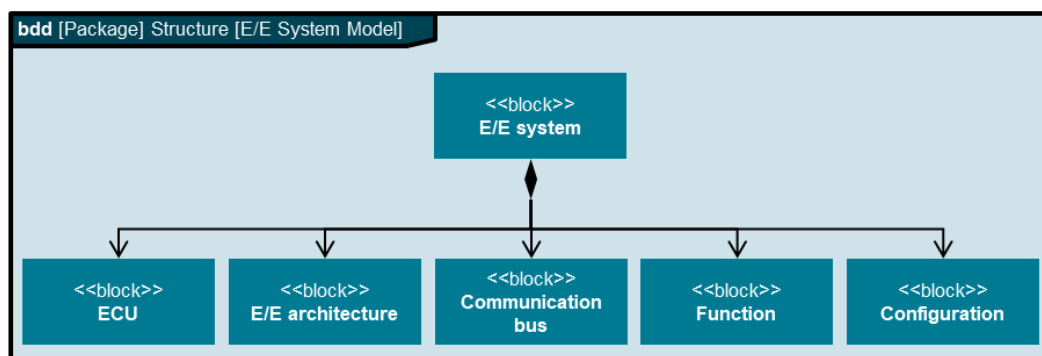


Figure 4-5: Generic structure of the automotive E/E system model.

Figure 4-5 shows the automotive E/E system at the top level with distinct blocks that each is part of the structure, referenced in other packages of the reference model, and in the model library for structure elements. The core of an automotive E/E system builds one or multiple ECUs which, in turn, again consist of different hardware and numerous software. The E/E architecture is commonly developed independently of the car model line or car platform and is only adapted during the development process. The E/E architecture defines the overall, high-level topology of an automobile’s E/E configuration whereas the communication bus delineates the 1..n connected busses in a specific E/E system. Due to the fact that distinct functions are spread across many E/E systems and hence several ECUs, modeling the functional association of an E/E system is essential. Moreover, the E/E system can exist in multiple discrete configurations, for instance if certain features are enabled or if different stages of functionalities are available and customizable. Each block will be described separately in the following.

DEFINITION OF THE GENERIC STRUCTURE OF THE ECU

The ECU is embedded in the E/E system, denoted by a composite relationship in Figure 4-6 (vide supra for nomenclature). As well as the E/E system, the ECU correlates with

one or many communication busses and functions. The configuration block subsumes the lifecycle management by means of variants and versions and will be discussed below. As a main criterion, the ECU commonly consists of hardware and software in their different characteristics. Hardware is further fragmented into its geometry for the purpose of calculations of the designed space contingent on the ECU's case. The E/E-related hardware composes of memory, a processor, and plugs serving as physical interfaces. The plugs again have a relation with the NCD, indicated by the symbol "→", fostering traceability of each signal between ECUs down to the level of individual pins that are used to transmit the currency for the signals. Also associated with the hardware is its schematic in different level of granularity.

Bootloader, functional, and parametric software commonly constitute what is called a software component, i.e., a given baseline of the relevant software artifacts, that is deployed in an ECU with a certain specification for a particular configuration of the automobile.

The blocks *E/E architecture*, *communication bus*, *function*, and *configuration* will be discussed in the following packages of the structures of the reference model.

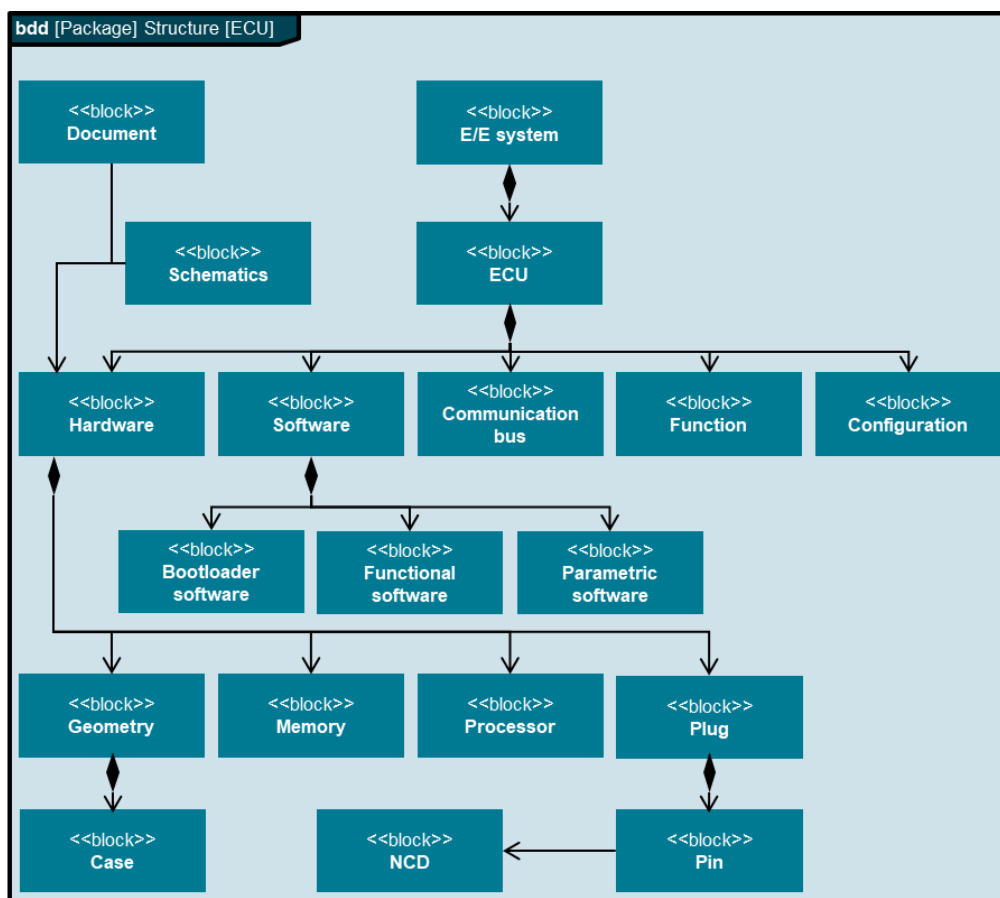


Figure 4-6: Generic structure of the ECU.

DEFINITION OF THE GENERIC STRUCTURE OF THE E/E ARCHITECTURE

The E/E architecture is developed independently of the car platform (cf. Chapter 2.5) and only adapted to the specific car platform and later model series in due course. This is done under the consideration of cost saving. Hence, in the context of a dedicated model series of a car platform⁹⁷, the E/E architecture has a specific configuration, which will be depicted in consecutive packages. Often, a 150% model of the wiring of the entire E/E architecture is used to depict the overall concatenations of all ECUs and communication busses (cf. Chapter 2.3.2). The data model for the structure of the E/E architecture is illustrated in Figure 4-7.

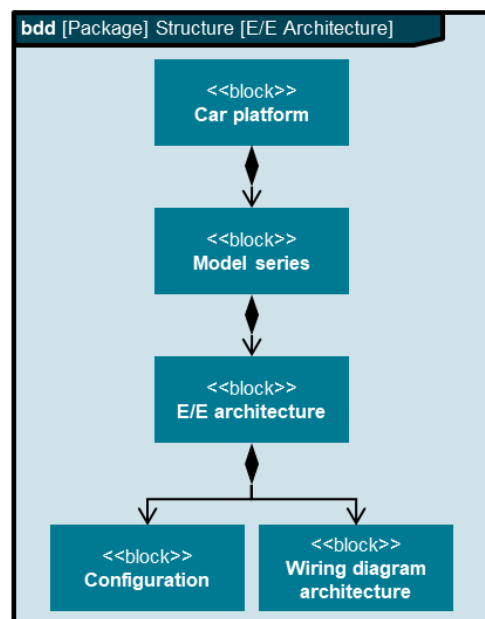


Figure 4-7: Generic structure of the E/E architecture.

DEFINITION OF THE GENERIC STRUCTURE OF THE COMMUNICATION BUS

The communication bus or busses are the backbone of communication among multiple ECUs. Hence, during development it is crucial to trace changes to the bus and its associated data defined for the exchange of messages within an engineering collaboration because a false specification will yield a deficiency in reliability. The communication bus has a simple aggregation relationship to the ECU due to the communication bus being able to exist without the ECU. This is particularly the case, if a communication bus connects many ECUs. The simple aggregation relationship is denoted by the symbol “◇” combined with the same arrow “→” as mentioned previously (OMG, 2015: p. 38). The NCD is a direct composite of the block *communication bus* due

⁹⁷ Analog to Figure 2-16, a Mercedes E-class denotes the car platform, whereas the model series would be a derived variant, such as the E-class convertible.

to it describing the messages on the communication bus and hence cannot exist without its parent. This is depicted in Figure 4-8.

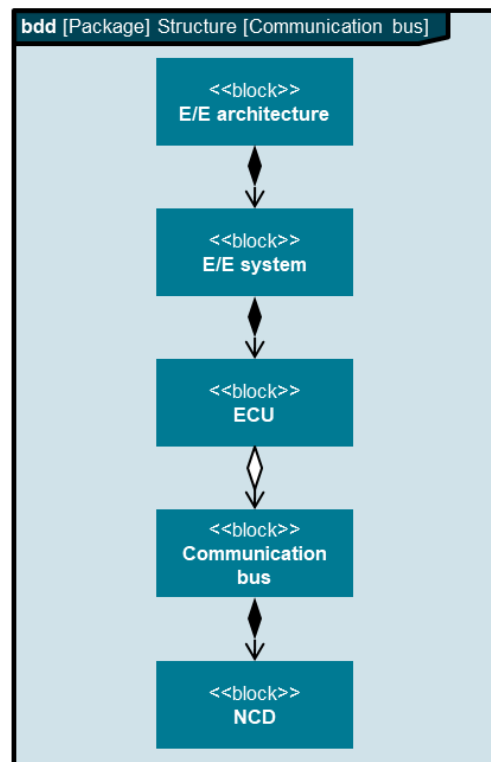


Figure 4-8: Generic structure of the communication bus.

DEFINITION OF THE GENERIC STRUCTURE OF THE FUNCTION

In the RFLP approach (cf. Chapter 2.4), functions are deduced from requirements. Afterwards, functions serve as a connector to the logical viewpoint and later, more granular, to the technical viewpoint. This breakdown on the basis of different aggregation levels is essential for traceability during development. Therefore, the block *function* has to be modeled explicitly for the purpose of generating data artifacts that can be traced by association with the E/E system and its components.

Here in this case, one or multiple functions are modeled to have only a reference association to the ECU, the E/E system, and the software; as represented in Figure 4-9. These simple reference associations stem from the fact that functions are often spread across many ECUs, hardware-related, and multiple software components in case the function is software-based. Moreover, functions frequently are implemented in different E/E systems. The relation to the block *Viewpoint* via the E/E system ensures the feasibility of different views which often have some correlation to functions and described coherently for a specific person, such as a systems architect (FRIEDENTHAL et al., 2012: p. 115).

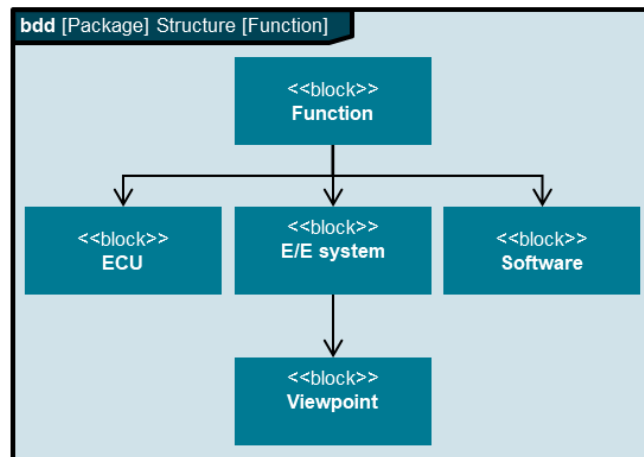


Figure 4-9: Generic structure of the function.

DEFINITION OF THE GENERIC STRUCTURE OF THE CONFIGURATION AND VARIANTS

By means of a generic configuration model, the topology of a product, the relationships between different components, and their association with the respective IT system, where the configuration will be stored and managed, will be modeled and implemented (cf. KÖNIGS, 2013: p. 91).

The inclusion and alignment of both MBSE and PDM, as required to foster traceability in the early development phase, is modeled based upon a dedicated configuration data model. For that purpose, a separate or orthogonal data model for the description of variability in a product and resulting configurations will be defined (POHL et al., 2005: pp. 57 ff; SCHULTE et al., 2017c: pp. 264–265). However, in contrast to POHL et al. (2005), the variability yielding different possible configurations will be modeled explicitly within the given data model, such as done by KÖNIGS (2013). This has the advantage that in an engineering collaboration, where data models have to be exchanged, the configuration data model is already included within the entire data model and is not implemented in a separate metamodel. This might alleviate incompatibilities as well as reduce issues or errors during the exchange of data among multiple engineering partners⁹⁸. As here a separate bdd solely for the configuration, i.e., variability, is defined, this also can be considered as “orthogonal”.

The definition of variants for elements of the E/E system and ECU, depicted by distinct blocks, can be implemented using a top-down or a bottom-up approach. The bottom-up approach requires the specification of discrete, variant-building attributes. In contrast, the top-down approach primarily addresses elements on system level and then

⁹⁸ Please refer to POHL et al., 2005: pp. 74 ff for disadvantages of modeling the variability model within the actual development models, in that case for software.

proceeds successively to elements on lower levels to elaborate variants (HOOSHMAND, 2015: p. 71). Here in this work, the complete description of the E/E system and ECU, besides focus on E/E, is not a goal and therefore the level of granularity of attributes is not in scope. Hence, the top-down approach will be used to form variants for the ECU on closer inspection.

The newly introduced block *variation point* describes the actual representation of variability, i.e., the possibility to vary and hence form configurations and variants, within the data model itself and its domain artifacts (POHL et al., 2005: 62). For instance, the processor or memory of an ECU are variation points. Depending on the requirements, the concrete instances of a processor or the size of memory represent different variants for this ECU within the modeled data⁹⁹.

Figure 4-10 shows the generic structure of the configuration of an E/E system including its relevant information artifacts. A configuration of an ECU comprises hardware and software and has a direct relationship with the block *ECU* itself. Via the ECU, a connection to the engineering BOM (E-BOM), manufacturing BOM (M-BOM), and their corresponding, but also other, IT systems is modeled. Both, hardware and software are combined each at a certain state of development and with specific characteristics to a variant. This variant, in turn, exists in a dedicated version at a given point in time (cf. Chapter 2.3). Such a version will be addressed if a deployment date was planned. This is the case when the ECU in scope will be included at the assembly line with a specific start date and a given end date at a dedicated plant. Additionally, an engineering change includes a precise version for the purpose of adaptation. A release can be considered as a major baseline for the entire E/E architecture that shall be built in combination at an explicit date. Hence, the release directly addresses an ECU's version and is a child to the model series in which it shall be implemented.

⁹⁹ Please refer to POHL et al. (2012) for variant handling with respect to E/E and software development in alignment of the SPES method and the IT tool *PREEvision*.

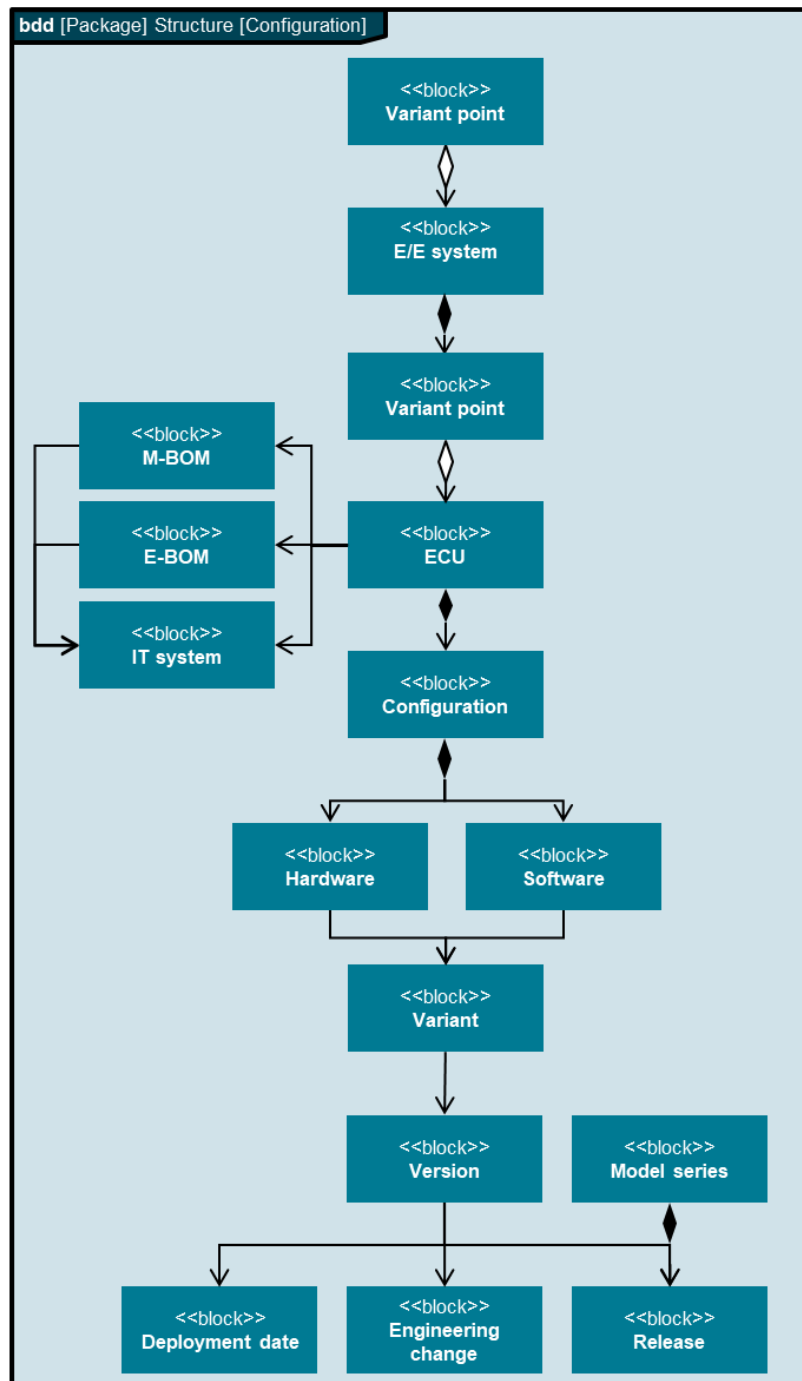


Figure 4-10: Generic structure of the configuration.

Subsequent to the definition of the reference model for the automotive E/E system and the elaboration of the concrete data models of a generic structure of the E/E system, ECU, E/E architecture, communication bus, function, and configuration, the description of process-relevant information artifacts follows.

DEFINITION OF PROCESS-SPECIFIC INFORMATION ARTIFACTS

The question of which information artifacts have to be generated in order to precisely address the required information in an engineering collaboration at a given point in time, will be addressed here in this section.

The SPES method according to POHL et al. (2012) describes an elaborated manner for MBSE (cf. Chapter 2.4.2). However, the detailed process steps for a distributed engineering collaboration framework in alignment to the SPES method will be developed in Chapter 4.2, due to there being no detailed process description inherent to the SPES method (DAUN et al., 2016: p. 3; POHL et al., 2012: p. 151). Consequently, the scope here in this section will be on the information artifacts defined by the SPES method.

The SPES method focuses on model-based and continuous documentation as well as seamless engineering. The former focal point is achieved by metamodels which define structures of information artifacts. The latter focal point is enabled via formal semantics that clearly define the relationships between different categories of information artifacts. By means of model-based and seamless engineering, models additionally can be used for the execution of automatized analysis and model transformation and not only for documentational purposes (POHL et al., 2012: pp. 34–35).

By means of abstraction layers, the SPES method implements the logic of engineering to increase the level of detail successively when advancing in someone's work. Moreover, abstraction layers can also be used as a transfer from one organizational department or domain to the other. Abstraction layers are not a fixed concept for granularity. In contrast, depending on the industry the SPES method is applied to, abstraction layers differ. For instance, in the automotive industry the abstraction layers would be *supersystem*, *system*, *subsystem*, and *hardware/software component* (POHL et al., 2012: pp. 35–38).

The SPES method defines the already previously introduced viewpoints as major clustering objects. With each viewpoint, which itself is a work product, a different perspective of a system and the stakeholder's current concern, are highlighted. For that purpose, the system is modeled distinctively to represent relevant information for the viewpoint. The requirements viewpoint serves as a basis for consecutive viewpoints as the satisfaction of those requirements in the following viewpoints have to adhere to the initial requirements. For the purpose of relating the viewpoints, functional requirements have to be modeled explicitly within the functional viewpoint and hence, fulfilled by a

user function. The user function can be distinguished from the realization function, as the latter is required to fulfill a user function. This means that several realization functions have to be implemented, commonly highly integrated, for the purpose of enabling one user function. Sequentially, the functional viewpoint is mapped with a logical component via a $n:m$ relation, i.e., a logical component can realize multiple user functions and one user function can be implemented within many logical components. In the following, the logical and technical viewpoints have to be correlated with each other. This is achieved by means of the so-called deployment mapping which specifies on which hardware, e.g., ECUs or communication busses, which software functions of the logical viewpoint are implemented (INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2011f: p. 2; POHL et al., 2012: pp. 36, 40–41, 43–45). The actual interaction point on a logical and technical level is depicted by a pin within the SPES metamodel¹⁰⁰ (WEBER et al., 2012: pp. 65–66). The relation between the functional and logical viewpoints and their actual connection is done by linkage of a user function with a logical component as depicted in Figure 4-11.

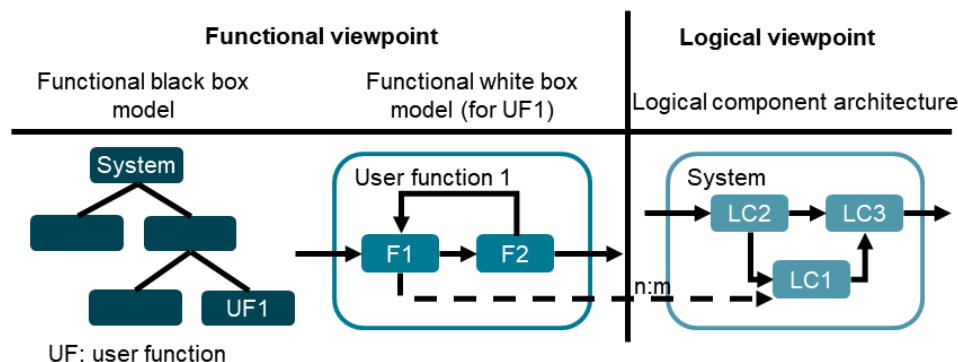


Figure 4-11: Connection between functional and logical viewpoints (in alignment to POHL et al., 2012: p. 45).

According to the above-mentioned definition of abstraction layers and their usage within the SPES method in combination with the different viewpoints, further information artifacts can be added to the reference model and its immanent generic structures modeled in SysML prior to this. In this connection, the focus still excludes the explicit consideration of requirements (cf. Footnote 18 on p. 25). However, the requirements viewpoints are already included in the reference model rudimentarily (cf. Figure 4-4). The connection of the viewpoints with the E/E system is depicted in Figure 4-12. The abstraction layers are modeled in Figure 4-13. Both, the viewpoints and the abstraction

¹⁰⁰ For more information about the SPES metamodel, please refer to WEBER et al. (2012).

layers are linked to the E/E system as an integrated and central information artifact, enabling traceability across the entire data model. Each block in the depiction of the generic structure of the abstraction layers in Figure 4-13 can inherit the different viewpoints described in Figure 4-12 and hence the individual instances of the E/E system. This is broken down from coarse to fine and can be examined according to each viewpoint. This enables a holistic overview of the entire E/E (super-) system, its sub-systems, as well as components. Moreover, explicitly modeling these dependencies fosters the required traceability.

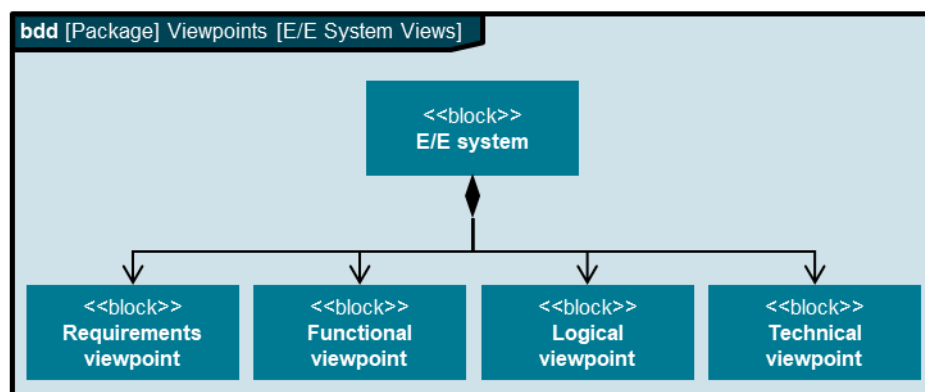


Figure 4-12: Generic structure of the viewpoints.

Figure 4-14 describes the generic relationship between the different elements of the E/E system, from the super-system to the hardware or software component, as well as the relationship between the different viewpoints and their immanent modeling elements. According to the structure of the abstraction layers (cf. Figure 4-13), the distinct system elements are connected and hence traceability throughout the different layers of granularity is ensured. The linkage of the system elements as well as the viewpoints across separate abstraction layers is depicted schematically by a bright blue arrow and connects the vertical axis in Figure 4-14. The horizontal layers describe the commonly successively developed viewpoints of one system element¹⁰¹. The viewpoints are linked by referencing of specific information artifacts (cf. Figure 4-11), which are depicted by black arrows in Figure 4-14. Due to all viewpoints being also connected to their respective system element, such as the E/E system (cf. Figure 4-12), traceability is therefore ensured along the horizontal axis.

¹⁰¹ Naturally, the development process is a highly iterative approach and therefore information artifacts and their relationships are modeled gradually. However, the main approach follows a sequential refinement from coarse to fine with respect to the level of granularity (cf. Chapter 2.2).

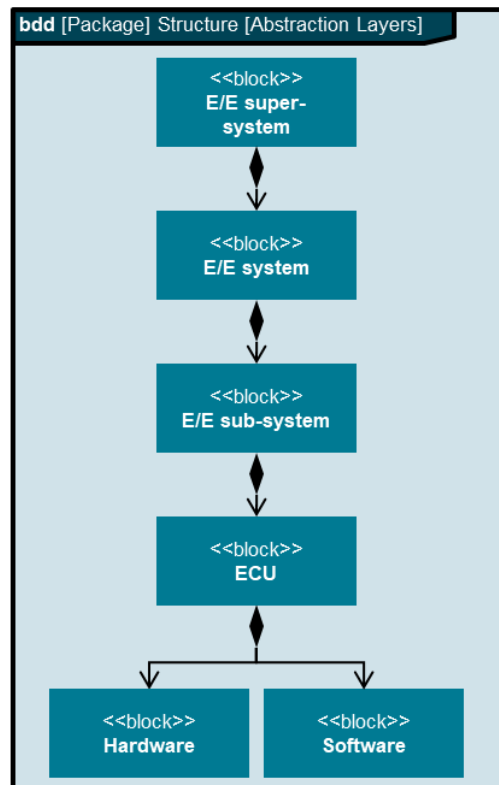


Figure 4-13: Generic structure of the abstraction layers.

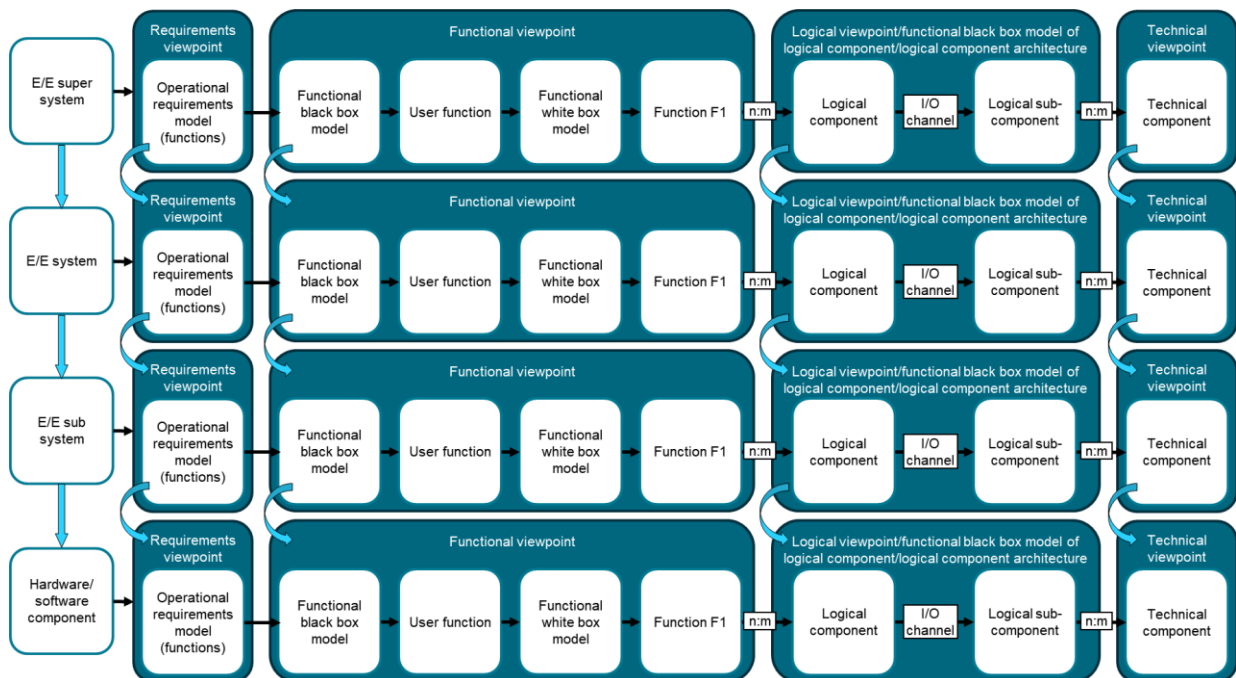


Figure 4-14: Generic relationships between different viewpoints for one system element and between different abstraction layers (in alignment to POHL et al., 2012: pp. 38, 45).

By integration of the relevant information artifacts deduced from the SPES method for the abstraction layers and viewpoints into the overall reference model, the data model now fosters traceability for the early MBSE phase. The dedicated interfaces between the different viewpoints, in Figure 4-14 denoted for instance by “I/O channel”, will be

described in the following section. The description of each process step follows in Chapter 4.2.1.

AUTOMOTIVE E/E DEVELOPMENT INCLUDING I/O DEFINITIONS AND PARAMETRIC SOFTWARE

In extension to the generic structure of an ECU (Figure 4-6), the relations between specific information artifacts within the realm of automotive E/E development have to be modeled in order to foster traceability for the main characteristics, yielding a higher complexity in today's automotive industry. Particularly, E/E-specific inputs and outputs and their form of transmission of signals as well as the medium of transmission is a major hurdle for traceability in an engineering collaboration. Subsequently, product quality is also impeded in mastering the complexity already in the IT systems (cf. Chapter 1).

The SPES method stipulates so-called *input/output channels* for the connection of logical components and their subcomponents, and the achievement of communication between them¹⁰² (POHL et al., 2012: pp. 88–89, 92). Modeling these relations explicitly, physical, e.g., a communication bus, as well as digital, e.g., a signal, traceability is enabled. This information is carried over to the technical viewpoint. Granularity levels, as fine as decomposing signals into messages, frames, or even bits, is not in scope of this work and hence not displayed in Figure 4-15. The same applies to sensors, actuators, and single ports.

The parametric software, or short parametrics, has already been modeled in the structure of the ECU in combination with the other major prevailing software types on an ECU (cf. Figure 4-6). Due to parametrics being very relevant with regards to input and output of an E/E system and its components, they also have been included here and being enriched by the connection to their signals, organized in the NCD and sent via the communication bus.

The generic structure of the input and output definitions is depicted in Figure 4-15. Relations between the blocks *pin*, *I/O channel*, *signal*, *NCD*, and *communication bus* have been complemented given the goal of traceability, the scope, and the level of granularity described above.

¹⁰² Dedicated input and output ports are modeled for logical components in the SPES method and then serve as connection point for the I/O channels (POHL et al., 2012: 92). However, this level of detail is not in scope of this work and hence it is refrained from modeling these ports explicitly. Additionally, the object *mapping*, *mapping block*, and further mapping artifacts are not in scope here due to dedicated connection points, such as pins, are modeled distinctly (cf. WEBER et al., 2012: pp. 73 ff; POHL et al., 2012: pp. 103 ff).

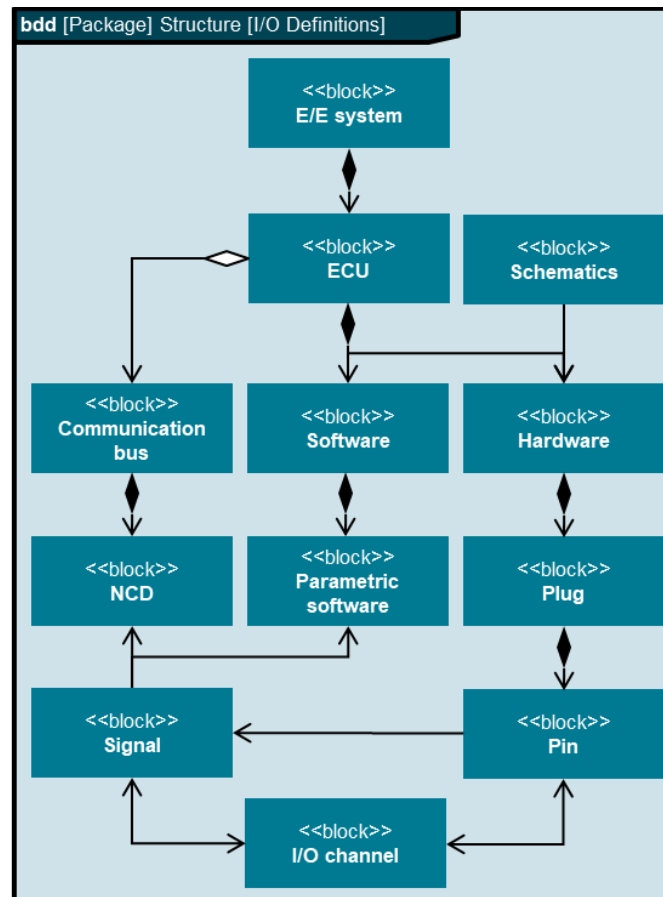


Figure 4-15: Generic structure of the I/O definition.

CONCRETIZED SPECIFICATION OF VARIANTS AND VERSIONS

Traceability of information artifacts across the lifecycle and different domains is one major purpose of PDM/PLM (cf. Chapter 2.3). Hence, this also has to be considered in the data model. The foundation for the for PDM/PLM relevant specifications of information artifacts was already described in the generic structure of the configuration in Figure 4-10. There, the blocks *Variant*, *Version*, and further temporally structural blocks, such as *Deployment date*, *Engineering change*, and *Release*, have been modeled, since these are crucial for PDM/PLM in general and for automotive development particularly.

Exemplary structure elements of variants of an ECU and their versions are depicted in Figure 4-16. There, hardware and software variants are itemized into different versions. Moreover, metadata for these dedicated versions has been added generically, for instance, identifiers are depicted as well as metadata where the different versions could be disjunct. Commonly, there exists a standard variant aiming at cost savings of a component and a high-end variant aiming at the best possible performance. With respect to hardware, the altering technical parts within the component are usually CPUs

or memory. Software components often distinguish themselves by a different parametrization or parametric software. This may yield a different service offered to other software components, or enables higher performance in combination with hardware parameters. Theoretically, the referenced blocks of versions could be endless and, here, it is only depicted generically with its first iteration.

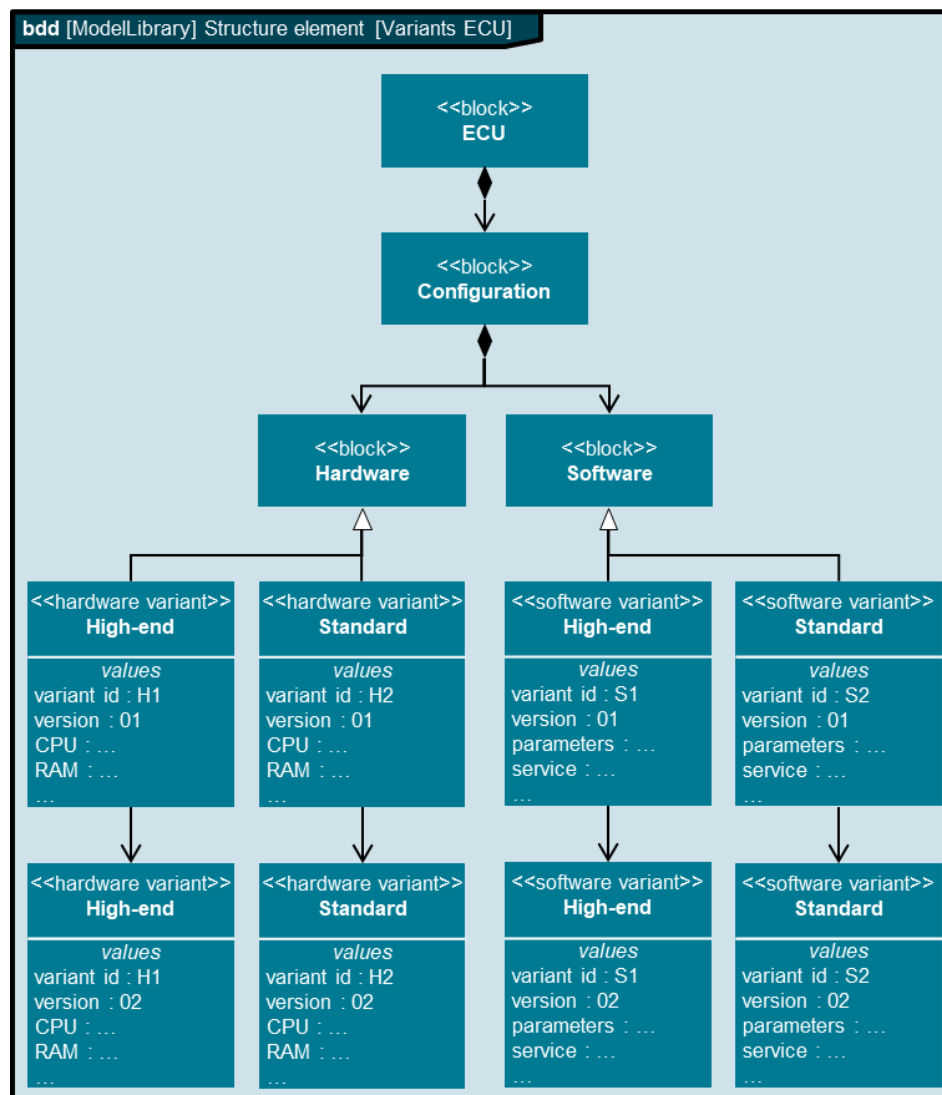


Figure 4-16: Generic structure element of potential variants of an ECU.

Dedicated metainformation regarding PDM/PLM, configuration and change management will be described in Chapter 4.1.2. For this purpose, identifiers such as for versions, variants, smaller engineering changes, and more extensive releases will be defined.

DEFINITION OF IT SYSTEMS-SPECIFIC AND COLLABORATION-SPECIFIC INFORMATION ARTIFACTS

In Figure 4-10, the generic structure of a configuration and the necessity to link the block *ECU* to an IT system was already modeled. This is due to different IT systems being

used in different domains, which define and handle configurations distinctly. For the purpose of generating traceability across numerous IT tools and systems, this relationship between the *ECU* and *IT system* will be defined more accurately in the following. Therefore, stereotypes of IT tools and IT systems that are commonly stored within the model library *IT systems* (cf. Figure 4-4), will be associated with the block *ECU* in order to make this information artifact traceable throughout the development process, towards PDM/PLM, and further along the product lifecycle. HOOSHMAND (2015) suggests to model blocks for each IT tool and then only reference norms and standards as metadata within the block of the IT tool (HOOSHMAND, 2015: pp. 80–81). Given that the explicit representation of norms and standards is not in scope of this work, it is deemed to be advantageous to directly model the relationship between ECU and IT tool or IT system.

In this context it is important to emphasize that not only different document types related to the ECU, such as a specification sheet, schematics, CAD drawing, etc. shall be linked to IT systems but rather the superordinate information artifact of the ECU itself. This is in contrast to KÖNIGS (2013), yet has the advantage that the relationship of information artifacts with IT systems can be identified, analyzed, and managed, regardless of the existence of respective documents and also aligns with the approach presented in the previous passage (KÖNIGS, 2013: p. 87).

ISO 10303 STEP AP 242, as a standard for data exchange in engineering collaborations (cf. Chapter 2.1.1), allows for different kinds of mapping of the BOM structure, such as the assemblies based upon the part occurrences, the part view, or the breakdown structure of the system¹⁰³. Here, the approach of directly linking the relevant information of the BOM and other IT systems with the MBSE data model will be implemented, as already presented in Table 2-1 and modeled by GILZ (2014) (GILZ, 2014: pp. 139–140). This approach ensures linking and hence traceability by usage of URIs (cf. Chapters 2.6 and 2.8) and will be addressed in the following chapter.

For purposes of distributed engineering collaboration, the transmission of trace links, which have been created between IT systems or tools at one engineering partner or the OEM, has to be ensured. Otherwise, only the engineering partner who created the trace links enables traceability within its own IT architecture while every other engineering

¹⁰³ Please refer to GILZ (2014) for more information about the mapping of BOMs in *ISO 10303 STEP AP 242*.

partner has to model trace links anew. In combination with the generic structure of the ECU (cf. Figure 4-6), hierarchical transitivity can be achieved. By means of explicit modeling the ECU's main components, the respective hierarchical transitivity can be transferred from one engineering partner to another. The connection of the ECU to its relevant IT tools and systems, in order to preserve and transmit links between information artifacts, is also achieved by explicit modeling (cf. BEIER, 2014: pp. 80–81). During development, this part of the data model also can be handed over and, therefore, via the block *ECU*, traceability including hierarchical transitivity among various engineering partners can be fostered.

The modeled connection of the ECU with the main IT systems or tools is depicted generically in Figure 4-17. Additionally, for each IT system or tool the main, relevant structural aggregation model or element is referenced. For example, for MBSE the relevant aggregation model is the system model (cf. Chapter 2.4), and for PDM/PLM this would be the engineering or manufacturing BOM.

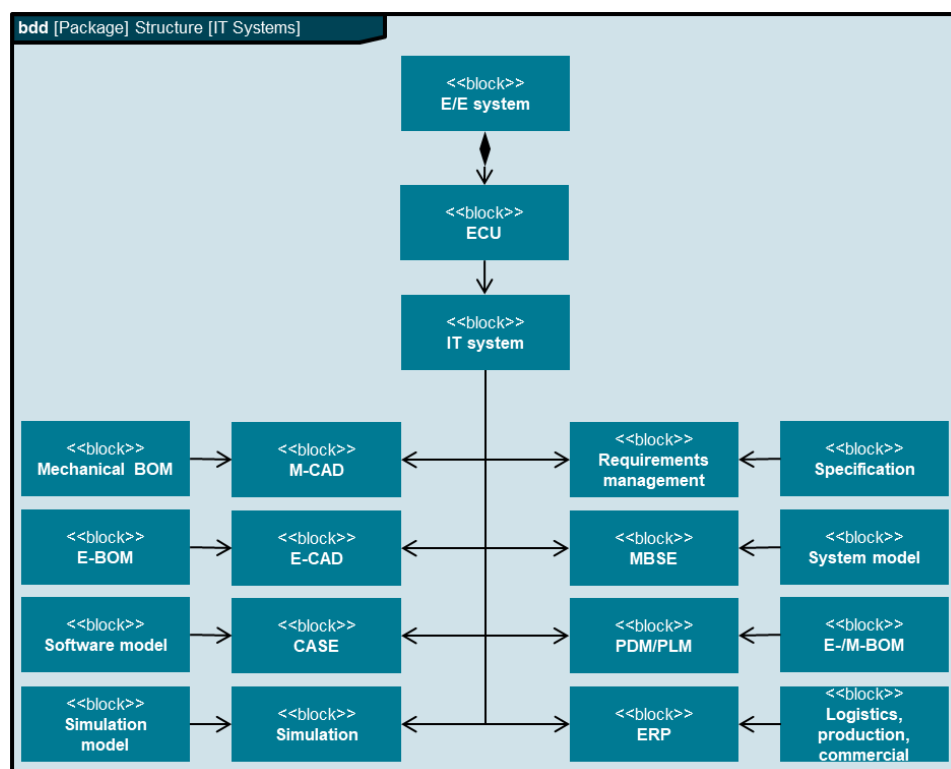


Figure 4-17: Generic structure of IT systems and tools as well as their relevant structural aggregation models or elements.

4.1.2 Relevant metadata for a linked data model

In order to satisfy requirement four, including a linked data model that is decisive for distributed engineering collaboration into a prospective solution framework, dedicated

metadata for the global identification of information artifacts have to be defined (cf. Chapters 2.6 and 2.8). This necessity stems from the heterogeneously deployed IT landscape, both within a company as well as externally with other engineering partners. The identification, integration, and management of knowledge must be feasible. Therefore, all relevant product-related data has to be easily accessible and department-overlapping. Also, the reduction of search time and processes for data management, maintenance, and access have to be enabled (HOOSHMAND, 2015: p. 79). For this purpose and analogous to the semantic web layer cake (cf. Figure 2-24), URIs, a syntax for data interchange, such as RDF, and a superordinate ontology, here according to OSLC, have to be defined. In this work, basic metadata, such as a time stamp of creation and modification, information artifact owner, description, etc. will not be addressed explicitly.

METADATA FOR THE LOCATION AND IDENTIFICATION OF RESOURCES

As URIs can identify objects and resources universally in a standardized manner, they build the foundation for a holistic data management and data exchange in distributed engineering collaboration across multiple domains¹⁰⁴. The exemplary URI scheme for the engineering context is depicted in Source Code 4-1¹⁰⁵. There, the first part denotes the scheme applied within the dedicated URI. Afterwards, the server follows. Subsequently, a self-defined path follows. In this case, the path aligns exemplarily with the generic structure of the E/E architecture as depicted in Figure 4-7. In Source Code 4-1, *Mercedes modular rear architecture 2* (*mra2*) denotes the car platform, *223* resembles the model series *BR223 Mercedes-Benz S-class*, *etherstar* stands for the specific E/E architecture, followed by the E/E system and the included ECU. This information will be included in the form of metadata within the information artifacts that will be transferred among engineering partners.



Source Code 4-1: Generic structure of the URI (in alignment to HITZLER, 2008: 27).

¹⁰⁴ Please refer to HITZLER, 2008: pp. 26 ff. for more information about URIs, their composition scheme, the definition and distinction between uniform resource locators (URLs) and uniform resource names (URNs).

¹⁰⁵ All source codes have been improved solely in their appearance for the purpose of better readability using the website <https://carbon.now.sh/>.

Additionally, universally unique identifiers (UUIDs) enable linking of data objects across systems by means of only one primary key (KIRSCH et al., 2017b: p. 165). UUIDs are mostly applied where there are no requirements for a speaking identification key and a sole technical mechanism of identification suffices. UUIDs are introduced additionally to URIs to provide an everlasting technical solution for the identification of data objects. Time, clock sequence, and a node identifier form a UUID¹⁰⁶.

METADATA FOR THE IDENTIFICATION OF INFORMATION ARTIFACTS IN PDM/PLM

As already introduced in Figure 4-16, information artifacts in PDM/PLM require specific identification for traceability across the product lifecycle.

Configurations, encompassing variants and versions, commonly are denoted using a self-selected scheme. Sometimes, this scheme has a speaking logic implemented. In Source Code 4-2 it is depicted that hardware variants are numbered consecutively alphanumerically with $H1...Hn$, where versions are numbered numerically $01...n$, and depicted jointly this yields $H1.01...Hn.n$.



```
Hardware variant id : String={H1,...,Hn}  
Hardware version id : String={01,...,n}  
Hardware variant version id : String={H1.01,...,Hn.n}
```

Source Code 4-2: Generic structure of the hardware variant and version scheme.

The same logic applies to software and the ECU as a final product, which combines hardware and software variants and versions, and is depicted in Source Code 4-3 and Source Code 4-4.



```
Software variant id : String={S1,...,Sn}  
Software version id : String={01,...,n}  
Software variant version id : String={S1.01,...,Sn.n}
```

Source Code 4-3: Generic structure of the software variant and version scheme.

¹⁰⁶ Please refer to LEACH et al. (2005) for the definition and specification of UUIDs by the Networking Working Group.



```
ECU variant id : String={E1,...,En}  
ECU version id : String={01,...,n}  
ECU variant version id : String={E1.01,...,En.n}
```

Source Code 4-4: Generic structure of the ECU variant and version scheme.

As change management is also considered part of configuration management (cf. Chapter 2.3.2), the above-mentioned logic of IDs for versions can be applied likewise to the change management for the purpose of traceability of individual information artifacts in case of changes. Figure 4-16 already depicts the documentation of changes for each variant of hardware as well as software denoted by an increase of the version ID in a simplified manner. Commonly, it is left to the engineer to decide when to create a new version of a component and when to change the old one. As a rule of thumb in practice, the *form, fit, function* assumption is used. If either one of the aforementioned are changed, then a new version or even part number is required in order to avoid confusion or errors.

METADATA FOR THE DATA EXCHANGE

Before, metadata for the identification across IT systems and within PDM/PLM have been addressed. Metadata has to be exchanged for the purpose of collaborative work. As described in Chapter 2.8.1, RDF was designed to represent and exchange metadata and is an internet standard today. The RDF uses graphs for the depiction of relationships of information artifacts. So-called triples are used for this purpose (cf. Chapter 2.8.1). Nodes, i.e., the information artifacts, can have multiple triples, i.e., relations, and hence a graph of relations can be built. This is particularly relevant in the case where the above-introduced SysML model cannot be implemented. This might be the case where there are other programming and modeling languages which prevail and the SPES method is not applicable. As described above, the heterogeneous IT landscape with numerous, domain-specific modeling languages is one major hindrance of traceability of information artifacts. Despite the focus in this work is on the early phase of development including MBSE where SysML prevails as a formal language, bridging the gap to other domains and their dedicated modeling languages must also be facilitated. Therefore, RDF and its triples are the chosen approach for a standardized connection and exchange of data among multiple engineering partners across the internet.

Figure 4-18 depicts the generic structure of an RDF triple for the relationship between the software `SW1` of the door control module (DCM) and the ECU, the `DCM` itself. Software development and E/E development are usually executed in different IT tools. There, SysML might not be the standard modeling language. Hence, the RDF triple connecting the software and the ECU by means of `is part of` offers this linking possibility independent of any IT tool and language. The blocks for software and ECU in Figure 4-18 also include the URI in the form of an `http` link.

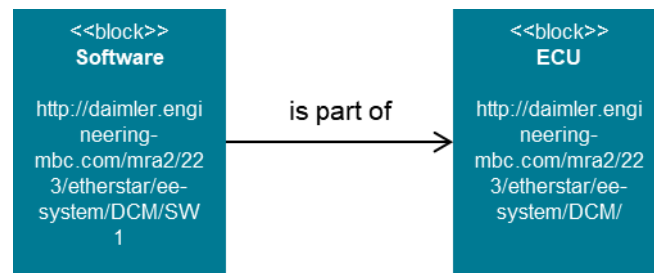


Figure 4-18: Generic structure of an RDF triple as a graph.

The triple in Figure 4-18 can also be depicted in source code using the *Turtle* syntax¹⁰⁷. The graphical depiction of relationships between information artifacts facilitates humans' work, whereas the depiction in source code fosters the interpretability for machines. The latter is depicted in Source Code 4-5¹⁰⁸.

```
<http://daimler.engineering-mbc.com/mra2/223/etherstar/ee-system/DCM/SW1>
<http://daimler.engineering-mbc.com/mra2/223/etherstar/ee-system/DCM/SW1
/ispartof>
<http://daimler.engineering-mbc.com/mra2/223/etherstar/ee-system/DCM>
```

Source Code 4-5: Generic structure of an RDF triple using Turtle syntax.

ONTOLOGY FOR THE EXCHANGE OF RELEVANT DATA FOR PRODUCT DEVELOPMENT

As OSLC includes dedicated identifiers for data, addresses resources in a standardized manner, and further offers an ontology for product development, OSLC will serve as the basis for the provision of an ontology. The first two included points have already been

¹⁰⁷ Please refer to HITZLER (2008) for more information about different RDF syntaxes.

¹⁰⁸ The RDFS, as a means of introducing a taxonomy into RDF, will not be addressed here due to OWL extends the content of RDFS further. However, OSLC already introduces certain ontologies specifically developed for the product development, sometimes described also in OWL, and is based upon RDF logic, too. Hence, there is no need for the dedicated specification of an additional ontology using OWL.

addressed above. Therefore, OSLC will be used to connect information artifacts and hence foster traceability (cf. Table 2-1).

The in Chapter 4.1.1 defined information artifacts will be aligned with the OSLC nomenclature. Therefore, the main E/E components of the generically defined E/E system have to be aligned with the OSLC resources, as these resources commonly denote a PLM or ALM artifact, change request, or requirement (JOHNSON and SPEICHER, 2013). The OSLC resources' names are appended via an # to the namespace URI. The exemplary namespace (ns) URI in our case was already mentioned above (`<http://daimler.engineering-mbc.com/ns/{domain}>`), where domain denotes, e.g., engineering domains such as MBSE and PDM. A resource would be ECU. Resources can have property definitions and constraints. Each property can also be addressed as an appendix to the core's namespace, for instance `#EEcomponent`. Moreover, appendices attached via # also denote classes (CROSSLEY, 2019). This generic structure of the used namespace including domains, resources, and properties is depicted exemplarily in Source Code 4-6.



```
<http://daimler.engineering-mbc.com/ns/MBSE#ECU>
<http://daimler.engineering-mbc.com/ns/MBSE#EEcomponent>
```

Source Code 4-6: Generic structure of the OSLC namespace, domains, resources, and properties.

According to the above-mentioned nomenclature, the namespace in OSLC terminology will be aligned given the information artifacts defined in Chapter 4.1.1. This is depicted exemplarily in Source Code 4-7. At the beginning, the different public namespaces for RDF, OSLC, and DCterms¹⁰⁹ are defined. In this example, the namespace of MBSE is connected to the namespace of PDM¹¹⁰, following the rule of `subject → predicate → object`, where `rdf:about` resembles the subject and `rdf:resource` the object. The predicate in this example is `describes`. In Source Code 4-7, the properties `DCM`, `Periphery-EE-System`, and `DCM_SW1` are defined for the distinct namespace denoted by opening with `<oslc:property>`, denoting the property's name with `<oslc:name>propertyname</oslc:name>`, and closing the argument

¹⁰⁹ For more information about the Dublin Core (DC) Metadata Initiative's terms, please refer to <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/>.

¹¹⁰ The namespace PDM solely is used exemplarily for demonstrative purposes.

with `</oslc:property>`. Further attributes of the properties are defined as an example, such as for `DCM` how often it might occur (`Zero-or-many`), a specific definition within the namespace `MBSE`, what value type it is assigned, or what its full title is. The property is defined locally, i.e., within the own namespace and only in addition to standard attributes. The property `Periphery-EE-System` is defined within the OEM's namespaces and can occur `Zero-or-one` times. It can be found in both namespaces of `MBSE` and `PDM` and hence this information artifact can be addressed out of a PDM IT system as well as out of a MBSE tool within the development IT architecture of a company. Similarly, the property `DCM_SW1` is defined both within the OEM's and within the supplier's namespaces and hence enables engineering collaboration by jointly using the same information artifact not only across IT systems but also between and among companies.

OSLC enables the connection of the definition of the reference model and generic structures in SysML for a product description in the early development phase with a standardized possibility for exchange of this development data. Hence, the intra- and inter-company engineering collaboration across different domains, IT systems, and engineering partners is enabled from the data perspective. This connection of data models is depicted schematically in Figure 4-19. In this figure, the OEM creates an early SysML data model during the early MBSE activities. For purposes of engineering collaboration, the Supplier 1 likewise generates a SysML data model, which is a subset of the entire data model found at the OEM, considering only these development parts, the supplier was contracted to deliver. If only considered one domain, such as MBSE, the data exchange between these two engineering partners could be implemented more easily. However, both engineering partners also have to take into account their respective subsequent processes, domains, and IT systems, such as PDM, which have to be connected to MBSE and with the other engineering partners. For this purpose, the data integration is implemented using separate RDF namespaces for each domain at each engineering partner. Of course, if the same namespaces are already implemented for different domains, this will facilitate data model alignment. For dedicated interdisciplinary, joint development activities, these RDF namespaces can then be addressed in an integrational routine written in OSLC, adding metadata in the form of attributes and values to the information artifact in focus. This allows for distinct pulls of all information artifacts of each RDF namespace, such as it is depicted exemplarily in Source Code 4-7.



```

<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:oslc="http://open-services.net/ns/core#"
  xmlns:dcterms="http://purl.org/dc/terms/"
  <oslc:ResourceShape rdf:about="http://daimler.engineering-mbc.com/ns/MBSE">
    <oslc:describes rdf:resource="http://daimler.engineering-mbc.com/ns/PDM">
      <oslc:property>
        <oslc:Property>
          <oslc:name>DCM</oslc:name>
          <oslc:occurs rdf:resource="http://open-services.net/ns/core#Zero-or-many"/>
          <oslc:propertyDefinition rdf:resource="http://daimler.engineering-mbc.com
/ns/MBSE#DCM"/>
          <oslc:valueType rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
          <dcterms:title rdf:datatype="http://www.w3.org/1999/02/22-rdf-syntax-
ns#JSON">
            DoorControlModule(DCM)</dcterms:title>
          </oslc:Property>
        </oslc:property>
      <oslc:property>
        <oslc:Property>
          <oslc:name>Periphery-EE-System</oslc:name>
          <oslc:occurs rdf:resource="http://open-services.net/ns/core#Zero-or-one"/>
          <oslc:propertyDefinition rdf:resource="http://daimler.engineering-mbc.com
/ns/MBSE#Periphery-EE-System"/>
          <oslc:valueType rdf:resource="http://open-services.net
/ns/core#LocalResource"/>
          <oslc:representation rdf:resource="http://open-services.net
/ns/core#Inline"/>
          <oslc:valueShape rdf:resource="http://daimler.engineering-mbc.com/ns/MBSE
/Periphery-EE-System"/>
          <oslc:range rdf:resource="http://daimler.engineering-mbc.com
/ns/PDM#Periphery-EE-System_range"/>
          </oslc:Property>
        </oslc:property>
      <oslc:property>
        <oslc:Property>
          <oslc:name>DCM_SW1</oslc:name>
          <oslc:occurs rdf:resource="http://open-services.net/ns/core#Zero-or-one"/>
          <oslc:propertyDefinition rdf:resource="http://daimler.engineering-mbc.com
/ns/MBSE#DCM_SW1"/>
          <oslc:valueType rdf:resource="http://open-services.net/ns/core#Resource"/>
          <oslc:representation rdf:resource="http://open-services.net
/ns/core#Reference"/>
          <oslc:valueShape rdf:resource="http://supplier1/oslc/shapes/DCM_SW1"/>
          <oslc:range rdf:resource="http://supplier1.com/ns/MBSE#DCM_SW1"/>
          <dcterms:title rdf:datatype="http://www.w3.org/1999/02/22-rdf-syntax-
ns#JSON">
            DoorControlModule(DCM)_Software1</dcterms:title>
          </oslc:Property>
        </oslc:property>
      </oslc:property>
    </oslc:describes>
  </oslc:ResourceShape>

```

Source Code 4-7: Example of the alignment of product development data with the OSLC framework (in alignment to IBM KNOWLEDGE CENTER, 2020).

Each engineering collaboration partner has their own data within their IT systems and tools, MBSE as well as PDM, and additionally there are the RDF namespaces for each

domain and partner. Moreover, these RDF namespaces are then combined to one routine, specific to this development endeavor. Given this separation of configuration items (cf. Figure 4-10) among domains, engineering partners, as well as IT systems, this approach for configuration management is close to the orthogonal variability model (OVM), as presented in Chapter 2.4.3 (cf. POHL et al., 2005: pp. 72 ff.). A metamodel links domains and IT systems and therefore also engineering partners' activities.

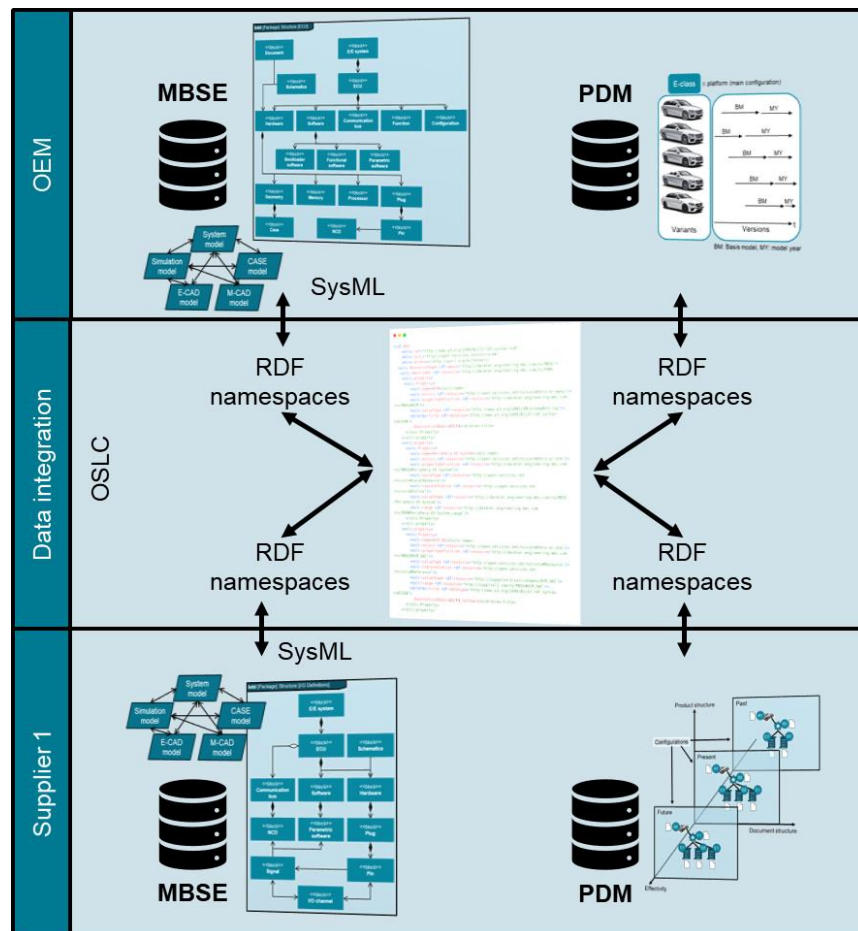


Figure 4-19: Schematic data integration across different domains and between engineering partners.

4.2 Definition of a process model

The process model depicts *what* is done *when* in time (cf. Chapters 2.2 and 2.3). It shall describe which information artifact (*what*) is created at which development step (*when*). The SPES method (cf. POHL et al., 2012) with its development steps, describing which view and which information artifact (*what*) is successively modeled (*when*), serves as a fundament (cf. Chapter 2.4.2). However, the SPES method does not explicitly define a process for the creation of the information artifacts depicted in Chapter 4.1 (DAUN et al., 2016: p. 3; POHL et al., 2012: p. 151). Therefore, this development process has to be

outlined here under the assumptions and conditions relevant for distributed engineering and in the context of the required information artifacts.

Additionally, basic aspects of PDM/PLM have to be incorporated for the process model to foster traceability of information artifacts in a distributed systems development. Therefore, the PDM/PLM processes of Chapter 2.3 shall be aligned to the SPES method. Consequently, it is described in the next chapters how the SPES systems development artifacts are created and linked to their respective PDM/PLM artifacts (Chapter 4.2.1). For that purpose, these information artifacts are written into the PLM Blockchain¹¹¹ and are made available to all involved engineering partners, which have access to the data of a particular system and its respective ECUs. This is done for all steps that occur in the product development within the engineering phase, i.e., all different use cases during engineering. Hence, the initial process describes which information artifacts are created just at the beginning of the development process, at the instant of time when a system is developed for the first time (Chapter 4.2.2). Following, configurations and variants have to be created and this information has to be aligned across IT systems, domains, and engineering partners (Chapter 4.2.3). Subsequently, the history of a product and its information artifacts has to be captured by the creation of versions and with a change management process (Chapter 4.2.4). Eventually, unused information artifacts shall be prevented to be reused what is demanding an inactivation process (Chapter 4.2.5). These use cases during product development cover the major processes in scope.

As described in Chapter 2.1.3, the CRUD (*create, read, update, delete*) operations, stemming from persistent database technologies, are essential for a process model to foster traceability in an engineering collaboration. Due to the data model integration of different domains across multiple engineering partners being implemented by means of OSLC (cf. Chapter 4.1.2), the CRUD operations have to be aligned with the OSLC syntax. As OSLC uses standard `http` commands, `post`, `get`, `put`, and `delete` (cf. Chapter 2.8.1) which are the counterparts of CRUD and will be used to implement and execute the different use cases of the processes during engineering in the respective databases.

¹¹¹ In analogy to the PLM backbone in Chapter 2.3.1, the PLM Blockchain serves as the IT system connecting other authoring tools, as depicted in Figure 2-12. For more information about the PLM Blockchain, please refer to Chapter 4.3.

4.2.1 Alignment of the SPES method and PDM

As the SPES method does not explicitly address a procedural step-by-step description of which information artifact has to be created in each development step (cf. Chapter 2.4.2), Figure 4-20 depicts the here applied process when using the SPES method. This process is derived from POHL et al. (2012) where only dedicated information artifacts, which have to be created, are described but not the chronological sequence of their creation¹¹².

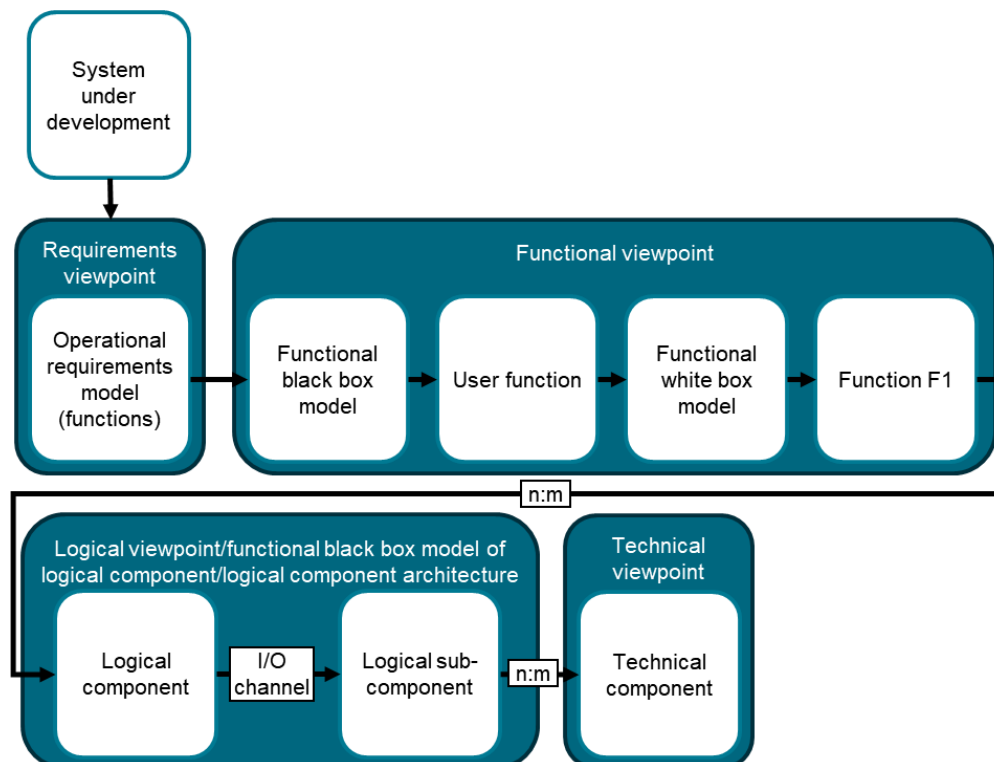


Figure 4-20: The SPES development process with its main information artifacts (in alignment to POHL et al., 2012: pp. 51–105).

As already depicted in Figure 4-14 with focus on the linkage of different viewpoints and different abstraction layers, each system under development is developed according to different viewpoints and abstraction layers, given the required level of granularity. Commonly, the engineer starts within the *requirements viewpoint* by modeling different aspects of requirements. As the description of requirements is not in scope of this work, this viewpoint is depicted in a simplified manner. Hence, only the operational requirements model, i.e., a high-level model of distinct functions of the product and how they might be connected by means of interfaces as well as inputs and outputs, is

¹¹² POHL et al. (2012) only maps the different viewpoints and abstraction levels to a generic engineering process but without either addressing distinct information artifacts nor the level of mechatronics (POHL et al., 2012: pp. 151–153).

modeled as an information artifact within this process step. This information artifact then can be used to connect the consecutive, functional viewpoint. The next process step is the modeling of the functional black box model and already belongs to the *functional viewpoint*. In this step, each function is denoted including its inputs and outputs. The functional black box model consists of distinct user functions, i.e., on a higher, user-perceived level, which are modeled separately (cf. Figure 4-11). Following, a functional white box model decomposes high-level user functions into functions on a technical level. For instance, the user function “accelerate car” is broken down into the different actions of actuators and sensors of the automobile’s powertrain to enable acceleration, e.g., the function “F1”. In turn, this function F1 has a $n:m$ relation to logical components, such as the above-mentioned actuators and sensors. The *logical viewpoint* consists of the logical component architecture which again can be denominated as a functional black box model of logical components due to it describes the generic relation of logical (sub-) components by means of I/O channels. Again, technical components have a $n:m$ relation to logical sub-components. The technical components within the *technical viewpoint* describe the system under development with its physical architecture including hardware and software, varying in level of granularity according to the given abstraction layer (POHL et al., 2012: p. 43).

Given a SPES development process, as depicted in Figure 4-20, the next step will be to align this with the generic PDM process, its single steps, and its main information artifacts. As the generic PDM process stipulates the creation of PDM-specific information artifacts (cf. Chapter 2.3), such as BOMs and other artifacts which are mostly associated with the technical viewpoint, these information artifacts will be briefly highlighted (cf. Figure 4-10). Due to the focus of this work being on systems engineering, only relevant information artifacts for both, systems engineering as well as PDM, will be considered. Therefore, PDM also has to include these relevant information artifacts which traditionally is not the case (cf. Chapter 2.3). For that purpose, the different viewpoints with their abstraction layers can be aligned with the generic engineering process. The conceptual phase during engineering is addressed by the requirements and functional viewpoint. In the phase of basic or desing engineering, all viewpoints are represented and aligned. Then, the detailing is modeled in the functional, logical, and technical viewpoint as well as following engineering phases, such as installation & commissioning or production planning. This is depicted in Figure 4-21 and is also in alignment with Figure 2-3 and Figure 2-10. Given that the SPES method stipulates an

iterative modeling, engineers will have to refine their models constantly by moving back and forth between the different engineering phases refining the distinct viewpoints as knowledge increases.

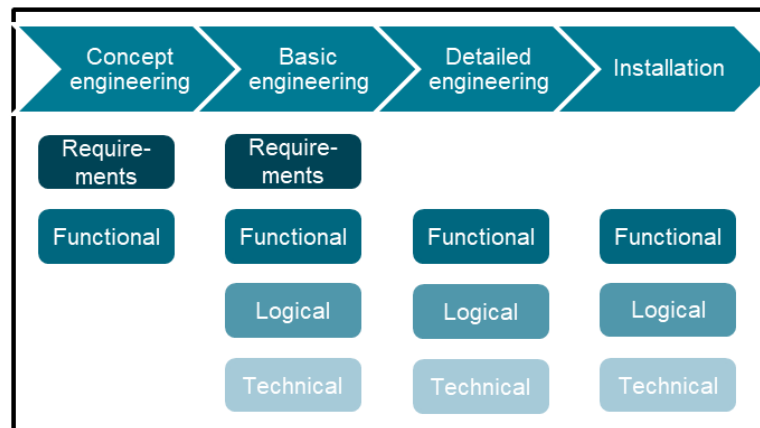


Figure 4-21: Alignment of a generic engineering process with the SPES viewpoints (in alignment to POHL et al., 2012: p. 153).

The generic alignment of the MBSE process with the PDM process, including the aforementioned relevant information artifacts, is depicted in Figure 4-22. Therefore, the information artifacts for MBSE and PDM from Chapter 4.1 have been considered. The E/E system serves as a starting point as well as a high level bundler for engineering activities across IT systems. Hence, its creation as a dedicated information artifact is relevant for both MBSE and PDM. Subsequently, the next relevant information artifact that is in scope is a specific function. Commonly, functional black box models etc. are not documented in PDM IT systems. However, not documenting any functions in PDM systems, as it often is still the case today, yields to a documentational gap as ECUs can execute different functions of different E/E systems and thus these relationships would not be captured. Hence, the distinct instantiation of a function, e.g. “F1”, also has to be documented within the PDM system. In order to distinguish an ECU from its inherent components, which are separate information artifacts that have to be addressed during development, the ECU will be aligned with the logical viewpoint and within this is a logical component. As depicted in Figure 4-10 and Figure 4-15, the ECU data model stipulates the important distinction between hardware and software. This process step is modeled with the creation of a logical sub-component and is documented likewise on the PDM side. The most detailed modeling level, the technical viewpoint, aligns with the highest granularity also in the PDM system where, during this process step, plugs, pins, signals, NCDs, parametric software, etc. will be documented.

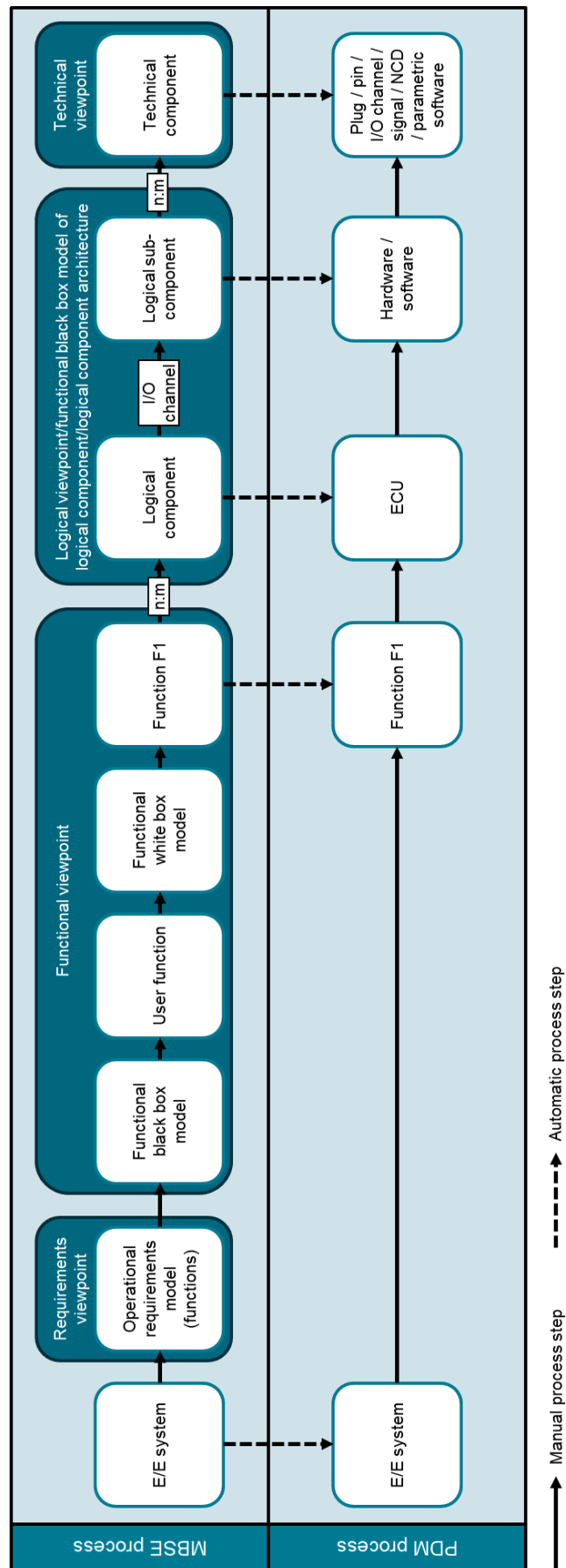


Figure 4-22: Alignment of the MBSE and PDM processes with their main information artifacts.

4.2.2 New product creation process

The initial process designates how the development process for a newly developed E/E system and its components occurs. In doing so, at each process step a new information artifact is written into the IT systems for MBSE, PDM, both mirrored to a certain extent on the side of the OEM and the supplier, as well as in data integration (cf. Figure 4-19). For the latter, these process steps are automated.

At the beginning, the common RDF namespaces for MBSE and PDM will be implemented in the data integration layer. This can be done by an initial upload of these RDF namespaces and the implementation of a work routine that aligns them automatically across connected IT systems. Considering one or many engineers on the OEM side responsible for modeling of an E/E system and its components in MBSE, denoted with “OEM MBSE”, and documenting this information in a PDM system, denoted with “OEM PDM”. Similarly, there are one or many engineers who model and document for the Supplier 1, denoted with “Supplier 1 MBSE” and “Supplier 1 PDM”. In the first process step it is assumed that the OEM engineer responsible for MBSE starts by modeling the E/E system under development. This assumption stems from the fact that the OEM determines the rough outlines including the requirements of an E/E system. Yet, also the supplier could start modeling. However, in this case it is presumed that the OEM has sovereignty over the E/E system and its functions and the supplier contributes the ECU including its hardware and software components. The OEM’s MBSE IT tool, which is connected to the data integration layer, automatically updates the respective RDF namespace in the data integration layer. Thus, these information artifacts are available for other IT systems within the engineering collaboration. These information artifacts will be synchronized with the MBSE IT tool from Supplier 1, where the E/E system under development will be “modeled”, i.e., the data will be transferred from the data integration layer to the MBSE IT tool. Instantaneously, the RDF namespaces for MBSE and PDM will be synchronized by means of the work routine implemented in the data integration layer. Then, the updated RDF namespaces for PDM will be synchronized with the respective PDM systems on each side of the engineering collaboration partners.

The second process step starts with modeling of function *F1*. Of course, considering the MBSE modeling process, there are interim process steps as depicted in Figure 4-22 that are not shown here for simplicity and due to relevance for PDM. After modeling of

function $F1$, the RDF namespace for MBSE in the data integration layer will be updated once more. This, in turn, triggers the automatic update of the supplier's MBSE IT tool as well as the alignment with the RDF namespaces for PDM. Consequently, an update of the PDM RDF namespaces propagates these changes to the PDM documentation of both the OEM's and the Supplier 1's PDM systems.

As Supplier 1 takes the lead of development of the ECU and its components from the consecutive process step onwards, modeling takes place at first in the MBSE IT tool of the supplier. The propagation of changes takes place symmetrically to the process steps initiated by the OEM via the data integration layer, which still serves as an intermediary between engineering partners as well as different domains. This logic also applies to the following process steps for modeling the logical sub-components for hardware and software and the technical components, e.g., the NCD. This generic process for the creation of a new product is depicted in Figure 4-23.

In order to satisfy requirement 7, that external traceability is fostered through the reduction of reconciliation effort by means of a consensus mechanism, the new product creation process also includes such a consensus mechanism for the case that a new information artifact will be created. Therefore, each time an engineering partner creates a new information artifact, the receiving engineering partner has the obligation to consent to or to decline the data transfer including the relevant metadata. This step is a manual affirmation step that, in turn, will again be transferred automatically through IT systems. In Figure 4-23, consensus is depicted by a green solid (manual process step) or dashed (automatic process step/transfer of data) arrow and check mark. Contrarily, decline is marked in red crosses and solid or dashed arrows. The technological aspects of the consensus mechanism will be described in more detail in Chapter 4.3.2, after the technological framework was presented.

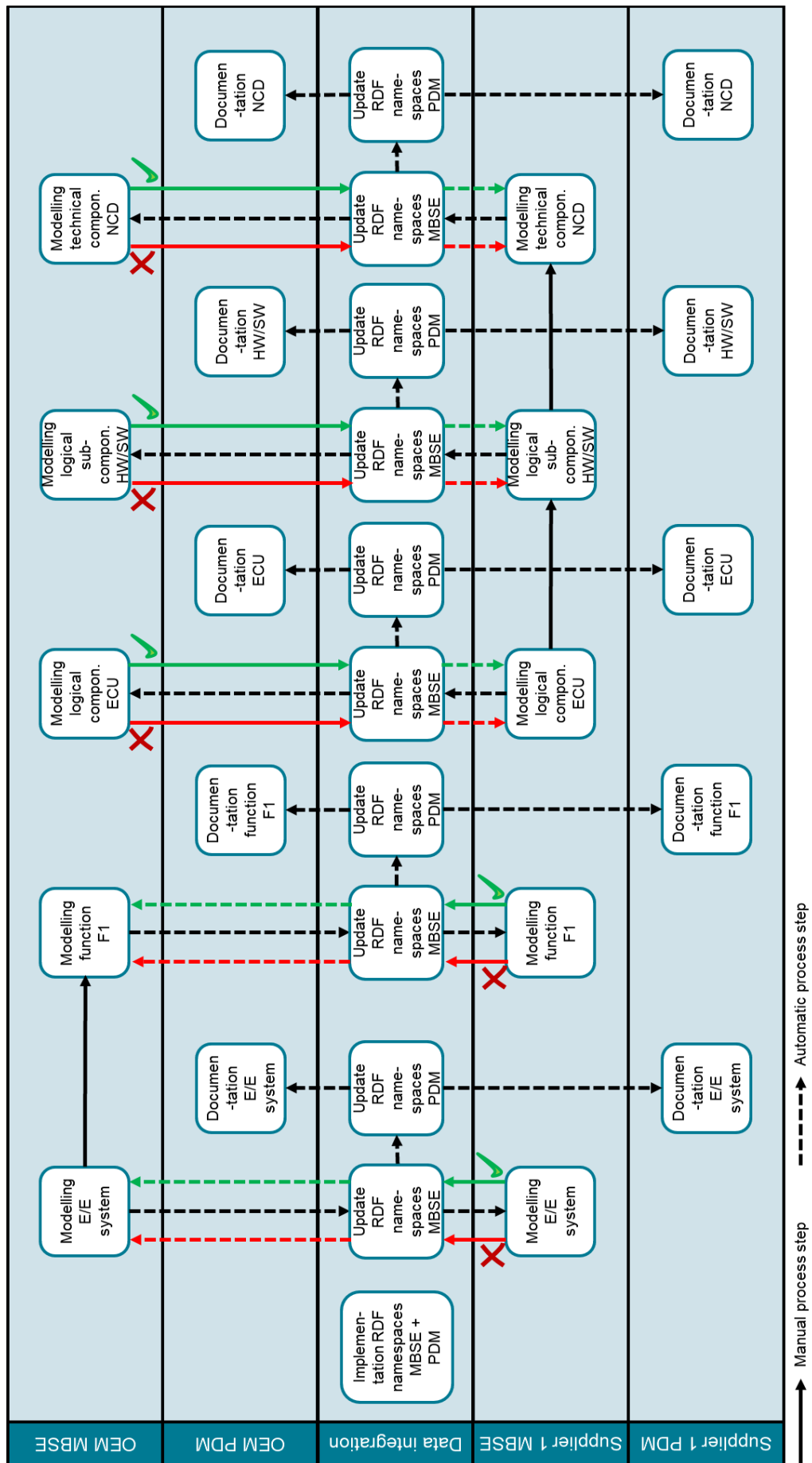


Figure 4-23: Generic process for the creation of a new product.

4.2.3 Configuration and variant creation process

For the creation of different configurations, and therefore distinct variants (cf. Chapter 2.3.2), the generic process is depicted in Figure 4-24. In analogy to the process for the creation of a new product, the OEM starts with the modeling of the E/E system and its variant points and, subsequently, the supplier follows with the residual development of the ECU and its components. Both engineering partners follow the same interchange of information artifacts as well as consent or decline as before. Again, the process is aligned with the data model for configurations in Figure 4-10 where variant points for the E/E system are defined. For instance, variant points are defined for separate car lines using similar specifications for an E/E system or an ECU, which is slightly adapted for the dedicated car model. At first, the OEM models these variant points in their MBSE IT tools from where the information will be propagated, via the data layer updating the RDF namespaces for MBSE, to the supplier's MBSE IT tools. The supplier's consent or decline of this update will follow and, in turn, will automatically trigger the update of the RDF namespaces for PDM in case of consent. It results an automated update of the documentation of the E/E system with its newly created variant point in the respective PDM systems.

Consecutively, the OEM models the variant point of the ECU in case Supplier 1 only develops this particular variant of the ECU. Otherwise, this variant point would be modeled by the supplier for the case that all different variants are developed by the supplier and hence they would be in charge of this process step. The consensus mechanism and the update of the RDF namespaces as well as the other IT systems happens likewise.

It follows the actual modeling of the logical component, i.e., the ECU, by the supplier who takes over development starting with this process step. Before continuing with the modeling of a dedicated configuration of hardware and software components, the OEM has the opportunity to affirm or reject the modeling results of the logical component by the supplier and, given a positive outcome of the consensus mechanism, RDF namespaces and IT systems will be updated accordingly. For simplicity, the hardware and software configuration are subsumed in one modeling process step, albeit the modeling of hardware and software are commonly separate steps, sometimes in discrete tools by distinct engineers. The consensus mechanism as well as the IT system updates will be executed in the known procedure, followed by a variant of the ECU.

Again, this process step is combined for succinctness and data is updated according to the previously depicted process steps.

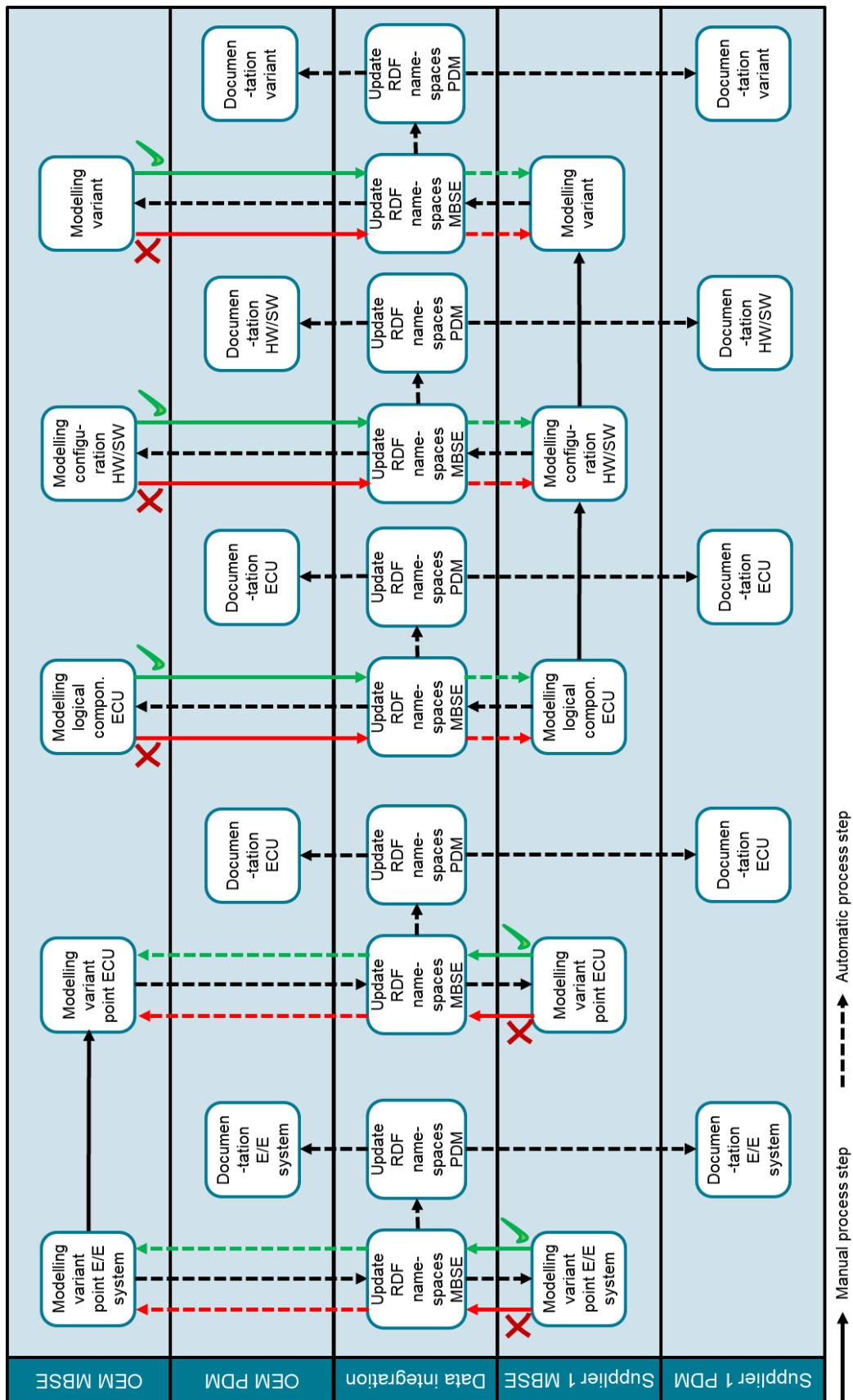


Figure 4-24: Generic process for the creation of a new configuration and variant.

4.2.4 Version creation and change management process

For the purpose of ensuring that a precisely defined configuration, i.e., a variant, can be identified and addressed in an IT system, the creation of a version and its effectivity over time is crucial (cf. Chapter 2.3.2 and Figure 2-14). In the definition for the version creation process and its change management process, the assumption is to have a distributed version control system, as depicted in Figure 2-15 in variation 6. This is due to the circumstance that the Blockchain technology is the objective of investigation which inherently is a distributed data base and hence is technologically closest to a distributed version control system. Therefore, the process has to reflect this technological premise.

The identification of the latest version of an information artifact within an IT system is essential for the embodiment of the version creation process. Given that the relationships between versioned information artifacts also have to be managed in case of versioning of an element, this has to be considered when the effectivity of a configuration and its version is defined. These relationships can be floating or fixed. In case of floating relationships, they will always point automatically to the latest version of an information artifact. Contrarily, a fixed relationship does not update to the latest version in case of alteration (GILZ, 2014: p. 121). For the definition of the version creation process it will be assumed that the relationships are maintained in a floating manner and hence a manual or explicit process step for management of relationships is not required. This ensures traceability to the always latest information artifact as well as a traceable history of changes due to artifacts will not be overwritten but the old versions will be archived in the background instead.

As the versioning of information artifacts commonly starts after the creation of a variant (cf. Figure 2-16), here the depicted process definition builds upon the variant creation process and presumes the existence of a variant. Thus, the process definition starts with the supplier creating a new version due to the OEM which already defined a variant and handed the development of this specific variant over to the supplier which creates different versions during the development process. The generic process for the creation of a new version is depicted in Figure 4-25.

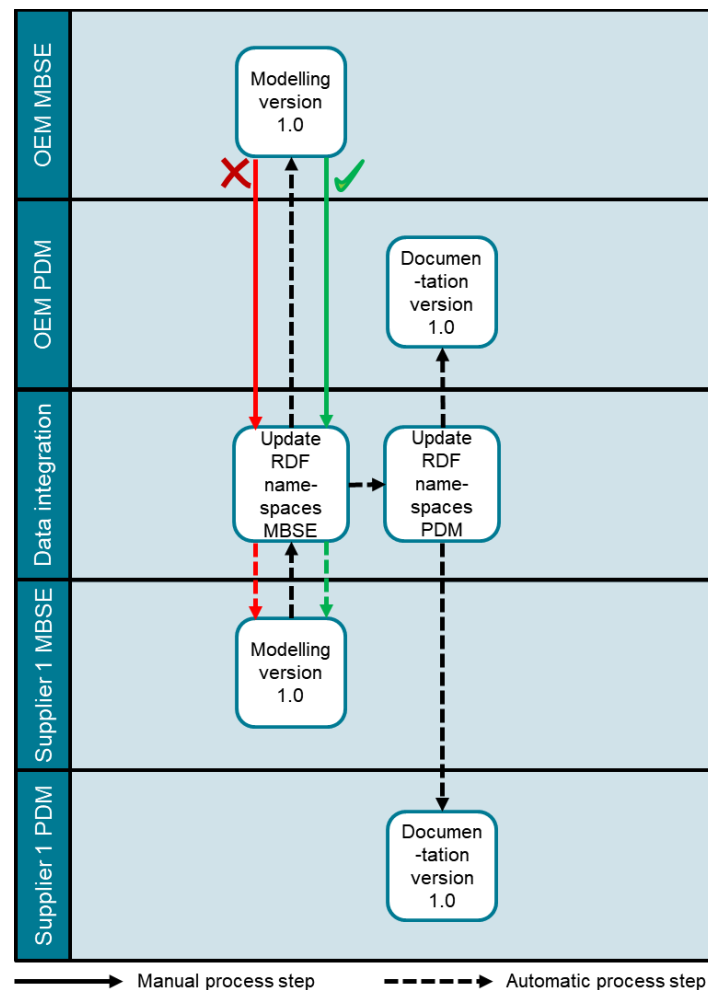


Figure 4-25: Generic process for the creation of a new version as part of the change management.

The different lifecycle states of an information artifact, for instance *preliminary*, *in review*, *released*, *in change*, or *superseded* (cf. GILZ, 2014: p. 123), are implicitly included in the consensus mechanism while a response is still pending. Hence, they are not addressed here explicitly for reasons of simplicity.

Figure 4-26 depicts the generic process for the change management, particularly the creation of a new version of information artifacts and how these updates are transferred process-relatedly across IT systems of the engineering partners. As described above, updated data models will always automatically refer to the latest version due to floating relationships and the separate lifecycle status of the information artifacts undergoing change are included in the consensus mechanism. Supplier 1 starts with modeling version 1.5, hence updating a previous version. This information will be transmitted to their own RDF namespace and then routed to the OEM's MBSE tool. Preconditioned the affirmation of the change by the OEM, changes will be propagated to the other RDF namespaces and into the documentation of the PDM systems in the same manner as

already described above. The same process repeats for further increments of information artifacts' versions.

The process for change management, as designed here, does support the automatic change propagation, given a suitable data base, such as the Blockchain technology (cf. Chapter 4.3), which satisfies requirement 8.

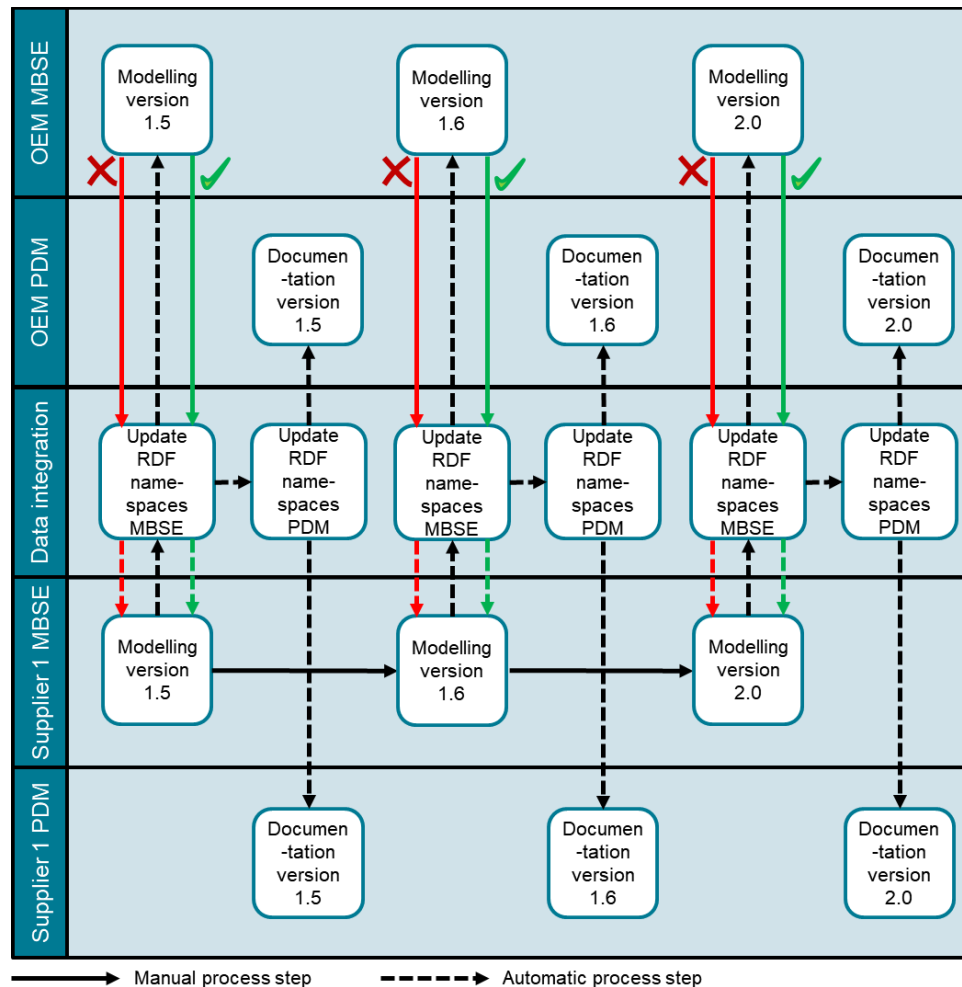


Figure 4-26: Generic process for the creation of new versions as part of the change management.

4.2.5 Inactivation process

The effectivity in configuration management, presented in Chapter 2.3.2, addresses timely aspects of information artifacts and their relationships between each other. For this purpose, obsolete information artifacts must also have the capability of being cancelled. This inactivation is required to prevent the usage and implementation of superseded information artifacts. This capability is particularly important for engineering collaborations due to the knowledge of a product is not concentrated within one

development department but dispersed across multiple different engineering partners who might have little to no contact.

The inactivation of information artifacts requires the definition of an obsolete item as well as the investigation of affected relationships in the case of the latest version of this item for the assessment of compatibility. The latter step would be inherent to the generic process for the creation of configurations and variants, as depicted in Chapter 4.2.3. Furthermore, the superseded information artifact has to be tagged as obsolete. Involved engineering partners must have the ability to consent or reject an inactivation by means of a consensus mechanism, in case of concordance or discrepancies. It is assumed that the inactivation of one single information artifact does not inactivate the entire product structure, i.e., hierarchical transitivity for inactivation does not apply (cf. Chapter 4.1).

Figure 4-27 depicts the generic process for the inactivation of a version of an information artifact. The inactivation of other information artifacts, for instance an entire variant, configuration, ECU, or E/E system, can be implemented analogously. Again, it is assumed that Supplier 1 is in charge of the development of the ECU and its components. Hence, in general it is the supplier's duty to deactivate obsolete components and inform all other engineering partners that certain information artifacts shall not be used anymore in the ECU and the associated E/E system. Therefore, first the supplier initiates the inactivation of a dedicated version, here in this case version 3 of a variant, which, in turn, is a specific configuration of hardware and software at a given point in time within the MBSE IT tool (cf. Figure 4-10). Afterwards, the already above-presented updates of the RDF namespaces for MBSE are triggered that again transfers the request for inactivation to the OEM's MBSE IT tool. The OEM now again has the opportunity to affirm or reject the inactivation which, in the first case, prompts the updates of the RDF namespaces for PDM and, consecutively, the actualization of the engineering partners' PDM systems.

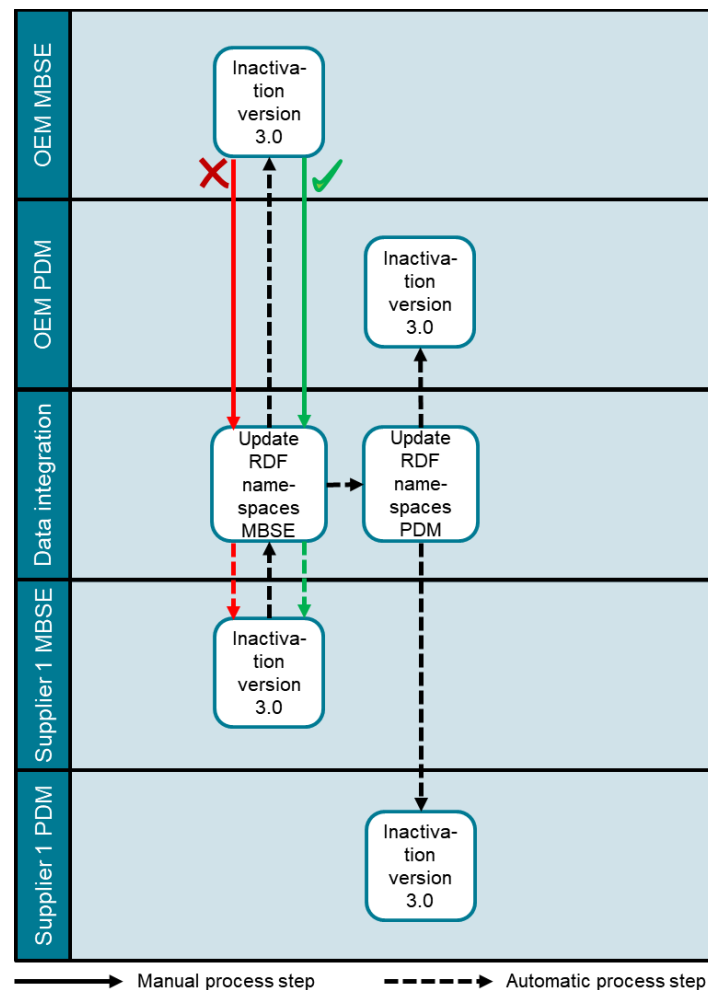


Figure 4-27: Generic process for the inactivation of a version.

4.2.6 Processes for multiple engineering collaboration partners

So far in this chapter, the generic engineering processes for only two partners have been depicted and described. This was mainly for reasons of simplicity. Of course, the entire framework to foster traceability within distributed engineering collaboration aims at the connection and inclusion of $n > 2$ engineering partners, as it often is the case for modern E/E development in the automotive industry. Therefore, the generic process for the creation of a new configuration and variant for the involvement of three engineering partners, one OEM and two suppliers, is depicted exemplarily in Figure 4-28. Scaling to more than three engineering partners complies with the same process steps as presented below with only three engineering partners. Hence again for simplicity, the depiction is limited to the basic scenario of $n = 3$ engineering partners. It is imaginable that the OEM contracts two different suppliers for the development of two ECUs for one E/E system or one supplier for the hardware of an ECU and one for the software. In both cases, all three engineering partners have to be informed about changes as early as

possible and shall have the possibility to confirm or reject changes for the purpose of fast and reliant development. This addresses requirements 7 and 8 in particular.

Once again, it is assumed that the OEM starts modeling a dedicated variant of an E/E system and its corresponding variant point in their MBSE IT tool. This update of information artifacts is synchronized with the RDF namespaces for MBSE of which there are as many distinct as there are engineering partners involved in the development of this particular E/E system; hence here in this case there are three. All RDF namespaces for MBSE include the new information artifacts and trigger the data transfer to the connected MBSE IT tools of Supplier 1 and Supplier 2. In addition to the previous depicted processes with only one supplier, now also Supplier 2 has the opportunity to affirm or decline the variant point of the E/E system modeled by the OEM. Via the RDF namespaces for MBSE, the consensus or dissension of the suppliers will be transferred to the OEM. It follows the update of the three RDF namespaces for PDM from where the local PDM systems of the engineering partners are refreshed.

The next process step for modeling a variant point for the ECU is executed similarly as the one before. Afterwards, Supplier 1 takes over the modeling of the logical component, which is the ECU. Therefore, they initially model the relevant information artifacts for the logical component and, in doing so, triggers the update of the RDF namespaces. From the RDF namespaces in the data integration layer, the information is transmitted to the OEM's and Supplier 2's MBSE IT tools. Now, in turn, the OEM and Supplier 2 have the duty to confirm or decline the modeling of the logical component. This data is transferred to Supplier 1. In the case of a consensus, the RDF namespaces for PDM will be updated and likewise the separate PDM systems of all three engineering partners. These process steps repeat themselves for the modeling of a configuration for hardware and software until a distinct variant of the ECU is modeled. Certainly, these process steps can further be evolved until a finer level of granularity, for instance for modeling versions, releases, or other changes.

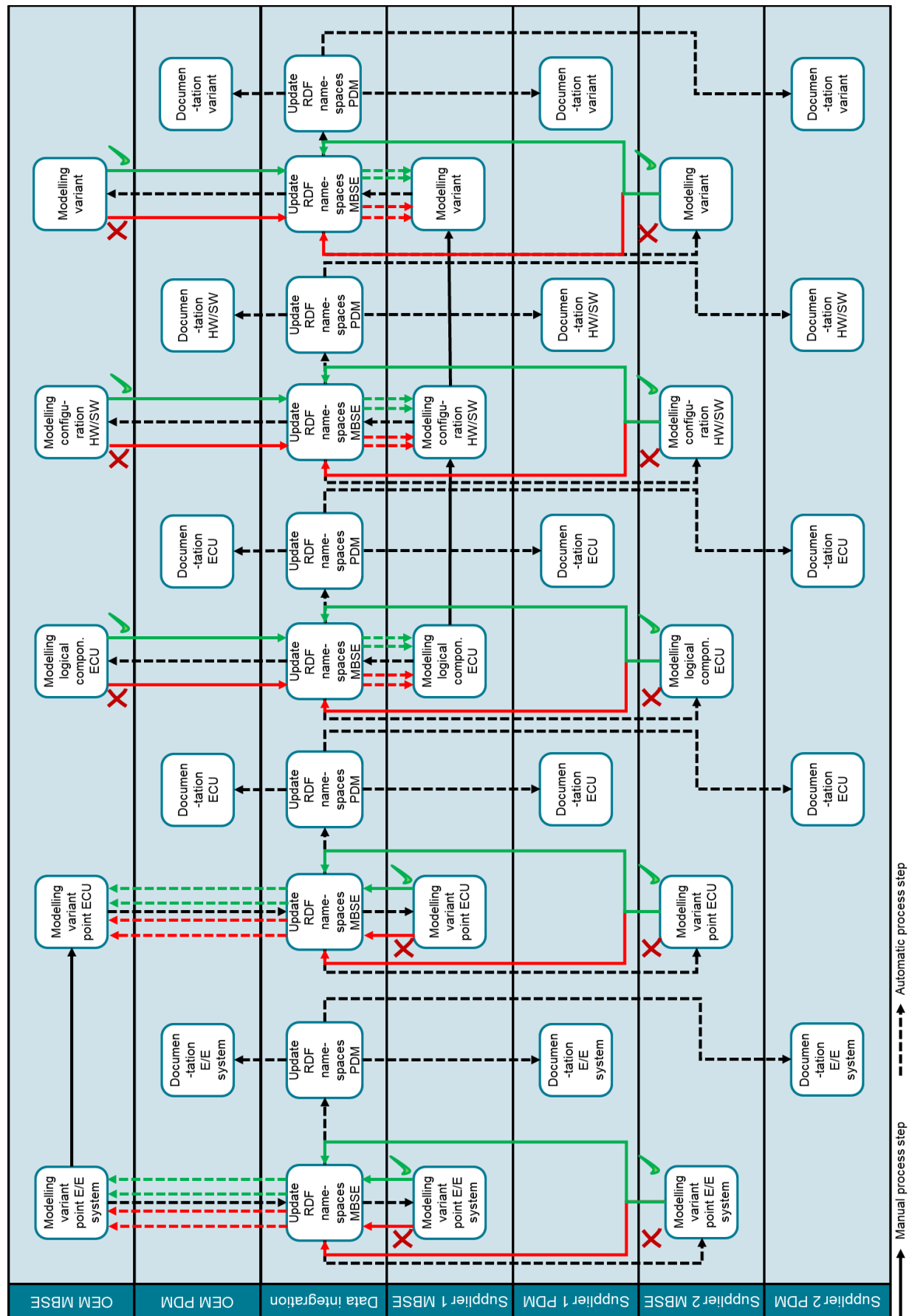


Figure 4-28: Generic process for the creation of a new configuration and variant with three engineering partners.

4.3 Definition of a technology

This framework to foster traceability of E/E information artifacts in an engineering collaboration during automotive development is fundamentally based upon a technology to connect IT tools and systems of the involved engineering partners. It was shown in Chapter 2.3 and Figure 2-12 that not only within one company the heterogeneous IT landscape is an impediment but becomes even more difficult to handle among multiple engineering partners (cf. Chapter 2.6 and Figure 2-22).

As objective 1. a. of this thesis is to foster internal traceability, particularly by means of the alignment of MBSE and PDM for E/E, the conceptual IT tool shall serve as an enabler for this internal traceability, according to the PLM backbone described in Chapter 2.3. Additionally, external traceability (objective 2.) shall be enabled by an IT solution. For that purpose, the proposed technology shall include a consensus mechanism (requirement 7) and an automatic change propagation (requirement 8). For legal and compliance reasons, an immutable product history shall be enabled by the IT solution (requirement 9) as well as the multi-directional synchronization among engineering partners (requirement 10). Data integrity (requirement 12) is crucial for sensible development data and also has to be granted in the case of the standardized inclusion of IT systems of other engineering partners or of other intra-company IT systems (requirement 15). System downtimes (availability) and the exit of engineering partners (robustness) shall be addressed by the conceptual IT tool and not harm the engineering network (requirement 16).

In Table 3-4, the merits of an unstructured P2P network with redundant data were already assessed and deemed to be the preferred form of data base. In the evaluation of the current state of science and technology, according to the objectives and requirements in Table 3-5, the Blockchain technology addressed many of the IT-related requirements. It was therefore chosen to serve as the basis for the conceptual IT tool in the context of distributed engineering collaborations in the automotive environment.

A prerequisite for the conception of a technological solution is the definition of a fundamental IT architecture. This will be described in Chapter 4.3.1. There, the above-introduced concepts for a linked data model, trace links, UUIDs, URLs, etc. are brought in relation to the IT architecture. The combination of the fundamental IT architecture in alignment to a generic IT architecture in automotive E/E development will be examined further. Here the focus lies on objective c., the alleviation of the connection of

engineering partners. Each new engineering partner must integrate the conceptual IT tool into their own IT architecture including all legacy IT systems. It will be shown how the presented technology can foster traceability not only within one company but also across companies. Additionally, it will be described how the consensus mechanism will be implemented, not only from a processual perspective (cf. Chapter 4.2) but also from an IT standpoint (Chapter 4.3.2).

4.3.1 Fundamental IT architecture of the IT solution

The fundamental IT architecture for the proposed technological solution to foster traceability is similar to the IT concept of a PLM backbone overstretching multiple IT systems and tools, as presented in Chapter 2.3 and Figure 2-12. The Blockchain technology serves as an equivalent for the PLM backbone in Figure 2-12 in order to connect different authoring tools and systems, as well as potential TDM systems of affiliated domains across the lifecycle¹¹³. It might also connect additional IT systems, such as ERP, SCM, PPS, etc. which are not within scope here. Figure 4-29 depicts the simplified generic IT architecture in which the so-called *PLM Blockchain backbone* enables intra- and inter-company traceability by linkage of data models, as presented in Chapter 4.1.

The PLM Blockchain backbone is connected to the proprietary, domain-specific IT systems via standardized interfaces. The native data, such as a simulation or functional model written in SysML, remains at each engineering partners MBSE authoring tool. The same applies to native data included in, for instance, a BOM in the PDM systems. Only the metadata of each system is translated into RDF namespaces, integrated in joint work routines, and transferred to the PLM Blockchain backbone (cf. Figure 4-19). In the interfaces between MBSE tools or PDM systems to the Blockchain, the translations between the different domain-specific data models, such as SysML, and the RDF namespaces, are implemented. Therefore, the PLM Blockchain backbone serves as the data integration layer.

¹¹³ For reasons of simplicity, the TDM layer is omitted here in the depiction. As the TDM systems often manage access rights as well as metadata, this will be relevant again for the prototypical implementation in Chapter 5.

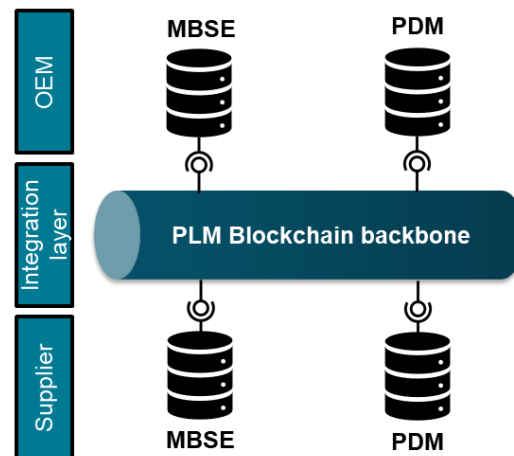


Figure 4-29: Generic IT architecture including a PLM Blockchain backbone simplified for one OEM and one supplier¹¹⁴.

According to the process defined in Chapter 4.2, information artifacts are then exchanged between the engineering partners, enabled by the IT architecture. Given the SPES development method, the *System under development* is modeled in the MBSE IT tool. The relevant metadata is transferred to the PLM Blockchain backbone where the RDF namespaces for MBSE will be updated. Corresponding with the data model (cf. Chapter 4.1.2), UUIDs and URIs (cf. Chapters 2.7.3 and 2.8.1) provide the basis for identification of information artifacts across IT systems. The header information includes all features for immutable, distributed, and transparent traceability within the Blockchain. For instance, the pointer to the previous block's header, time stamp, etc. (cf. Figure 2-23). These information artifacts are mirrored to the PDM system accordingly. Consecutively following the SPES process from the system, to function, to the logical component (ECU), its sub-components hardware and software, via modeling of interfaces to the low-level technical component, each additional information artifact's metadata is written in the PLM Blockchain backbone, concatenated with the previous block of information, and transferred to the attached IT systems of both the OEM and their suppliers. This is depicted in Figure 4-30¹¹⁵.

¹¹⁴ The symbol for the API (-@-) denotes both the required API (socket notation) as well as the offering API (ball notation) and can be considered here in this context as a bilateral API where both connected IT systems require and offer an API.

¹¹⁵ For the description of the *Channel ECU_A*, please see below in paragraph *Integration into generic IT architecture and channels*.

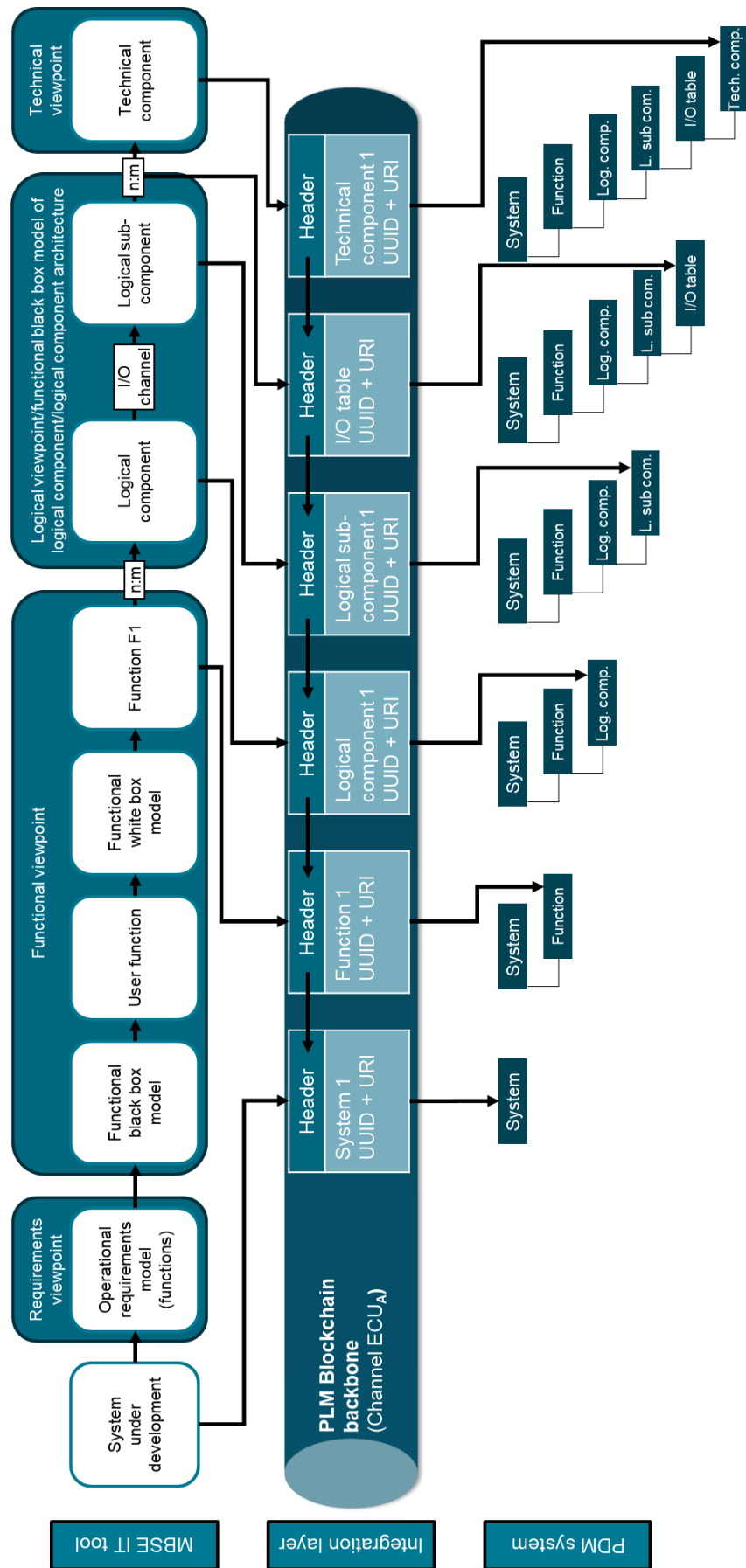


Figure 4-30: Generic IT architecture including the transfer of relevant information artifacts of the E/E development process according to the SPES methodology.

The above-mentioned generic IT architecture was simplified for the purpose of demonstration of its basic functionalities. However, in the case of multiple engineering partners, each partner possesses their own PLM Blockchain backbone. This is derived from the requirements for an immutable product history, multi-directional synchronization, data integrity, data availability and robustness, in the case of volatile participation of engineering partners who shall to be integrated easily into the engineering collaboration and the legacy IT architecture by means of standardized APIs (cf. requirements 9, 10, 12, 15, 16). Hence, there exist as many PLM Blockchain backbones in the engineering collaboration network as there are engineering partners. The nature of the Blockchain technology implies that all different PLM Blockchain backbones are connected to each other to form an unstructured P2P network with redundant data. This generic IT architecture for multiple engineering partners is depicted in Figure 4-31.

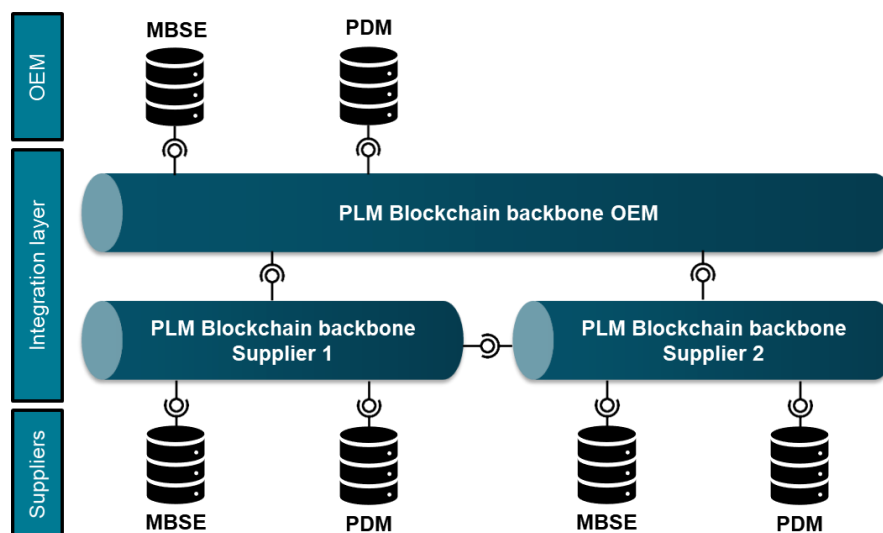


Figure 4-31: Generic IT architecture for multiple engineering partners with their own PLM Blockchain backbone.

APPLICATION PROGRAMMING INTERFACES

In order to connect all PLM Blockchain backbones within the engineering collaboration network, the unstructured P2P network builds the basic connection among nodes by means of the internet (HTTP, REST). On top of this, the P2P network enables communication among each linked node (BASHIR, 2018: pp. 49, 51). Therefore, the communication among the PLM Blockchain backbones, if implemented accordingly, is guaranteed. The inclusion of new engineering partners is facilitated by this standardized data integration layer and enables ad hoc participation in the development also for, e.g., start-ups without prior extensive adaptations of APIs.

The challenge for the integration of development activities in an existing IT architecture is the connection of many legacy IT systems to create, first, intra-company traceability, and further extend this traceability towards the other engineering partners. As depicted in Figure 4-19, the data integration is implemented using OSLC including all its components and standards (cf. Chapter 2.8.1). The data model, itself being used for enabling a traceability scheme between OEM and suppliers (requirement 11), is implemented as linked data using RDF in the distinct RDF namespaces and the joint collaborative work routine. In turn, this information can be written into the payload of the blocks within the Blockchain. As the semantic uses RDF, the APIs connecting the legacy IT systems, such as the MBSE tools and the PDM IT systems, are implemented according to the REST standard which is also part of OSLC. Here, the actual challenge is that many legacy IT systems do not support the REST standard and their APIs have to be first enabled in order to use it. As new IT systems are commonly already equipped with REST APIs, the connection of intra-company IT systems will become simpler over time and requirement 15 could be satisfied out of the box.

TYPE OF BLOCKCHAIN

As development data is sensitive, intellectual property requires a high degree of data protection and data integrity, the IT architecture has to reflect this requirement (requirement 12). This is especially the case for ad hoc engineering collaboration networks where partners can join just for the contribution of, for instance, one software component. Therefore, data has to be also robust, i.e., the exit of an engineering partner from the network shall not affect the availability of the relevant data. Also, data availability is important for distributed engineering collaboration (requirement 16).

The Blockchain technology does innately satisfy requirements 12 and 16. However, data shall not be publicly available entirely, only for the associated engineering partners within the collaboration network. For this purpose, the Blockchain must not be a public or unpermissioned. Contrarily, the Blockchain has to be a private, permissioned or a consortium Blockchain (cf. Chapter 2.7.3). In this case, the OEM is the owner of the Blockchain and grants access to suppliers that will contribute to the development of the automobile.

INTEGRATION INTO GENERIC IT ARCHITECTURE AND CHANNELS

As already depicted in Figure 4-30, some Blockchain technologies¹¹⁶ offer different so-called *channels*. By means of distinct channels, engineering partners can transfer confidential data in transactions within the same Blockchain network but using separate Blockchains. Channels offer a separate permission level for participants of the Blockchain network. Hence, only permissioned engineering partners can view data in channels that they are a member of. Other engineering partners within the same engineering collaboration using the same Blockchain network cannot access transactions in a channel they have not been granted access to (cf. BASHIR, 2018: p. 477). This IT architecture addresses the *need-to-know principle* and also corresponds to data integrity (requirement 12), data availability and robustness (requirement 16). Figure 4-32 depicts the integration of multiple PLM Blockchain backbones into a generic IT architecture with multiple channels in an engineering collaboration. There, the OEM outsourced the E/E system development to Supplier 1. Both share a channel for System_A which is implemented in each engineering partner's PLM Blockchain backbone. In turn, each PLM Blockchain backbone is connected to the respective MBSE tool and PDM system of the OEM and Supplier 1¹¹⁷. In this particular channel, all relevant blocks regarding System_A, documenting each transaction in case of any updates of metadata during development, are stored and only visible to the OEM and Supplier 1.

As Supplier 1 also contributes some component for ECU_A, for instance the hardware, there is a separate channel for this ECU together with Supplier 2 who might be developing software for ECU_A. Therefore, the OEM, Supplier 1 and 2 each have a distinct PLM Blockchain backbone building a sub-network exclusively for the development of ECU_A.

In case of ECU_B, there are also two other suppliers who contribute components to the development for the OEM. Again, distinct PLM Blockchain backbones with a *Channel ECU_B* are implemented for the purpose of compartmentation of data that is relevant only for contributing engineering partners.

¹¹⁶ The Blockchain technology *Hyperledger Fabric* features such channels. The channel ID is written in the transaction of each block in the Blockchain (BASHIR, 2018: p. 474). See Chapter 5 for more information on *Hyperledger Fabric*.

¹¹⁷ Supplier 1 and its MBSE tool and PDM system are depicted twice in Figure 4-32. This is done for representational reasons only. In order to have a single source of truth at each engineering partners' site, the MBSE and PDM systems shall exist only once in reality.

Separated channels for each component which is being developed with different engineering partners require distinct PLM Blockchain backbones. This is so that traceability is not hindered by such an IT architecture due to more fragmentation as before. Data has to be stored in a single source of truth, i.e., domain-specific IT systems that are the master for a dedicated type of data, such as a MBSE tool for MBSE data and PDM system for PDM data. Therefore, each engineering partner has to connect their own legacy IT systems with the respective PLM Blockchain backbones and their channels used for the development of a specific component where they participate.

As the PLM Blockchain backbone is merely a data integration layer enabling traceability across legacy IT systems and within engineering collaboration networks, these Blockchains do not serve as single sources of truth. However, all PLM Blockchain backbones form a joint network are linked to each other (cf. Figure 4-32). By this connection, the RDF namespaces dedicated to one channel and stored within the associated Blockchain can be synchronized and updated in case this is required and desired. This could be the case for the channels System_A and ECU_A due to both are used by the OEM and Supplier 1 for collaboration. Hence, there might be no necessity for separation of data due to IP. RDF namespaces and the combined work routine could be connected. In the case of separated channels, each engineering partner updates their RDF namespaces in the linked Blockchain out of their MBSE tool or PDM system and each channel has its own work routine, mirrored at each Blockchain, where RDF namespaces will be combined (cf. Chapter 4.1.2). Consequently, the PLM Blockchain backbones enable traceability of E/E artifacts within the engineering collaboration networks. The exchange of necessary information artifacts is limited to the involved engineering partners by means of the described, connected IT architecture. The Blockchain technology also supports the processual traceability through the immanent consensus mechanism which will be described in the following¹¹⁸.

The different aspects of the inclusion of the technological solution framework into an existing, generic, brown-field IT architecture, for instance APIs, data models and synchronization of the same across IT systems, have all been described above.

¹¹⁸ The separate depiction of the internet, the P2P network in contrast to the Blockchain network, and the different nodes has not been done for simplicity. In Figure 4-32, the internet is immanent as is the P2P network. The Blockchain network is depicted by the separate PLM Blockchain backbones. The users or nodes are displayed by the distinct MBSE tools and PDM systems of each engineering partner. For more information about the IT architecture of the Blockchain technology, please refer to BASHIR (2018).

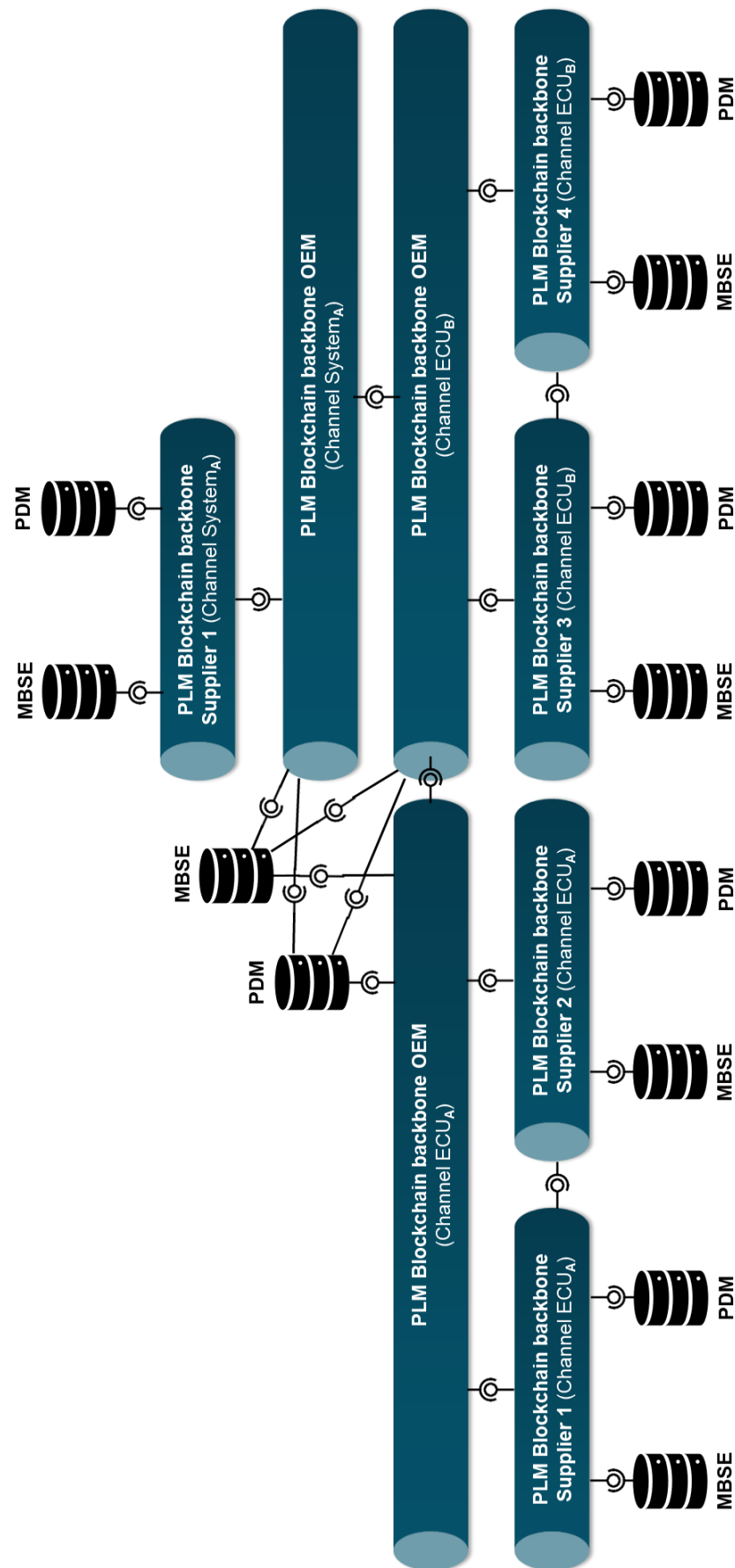


Figure 4-32: Integration into generic IT architecture with multiple channels implemented by different suppliers.

4.3.2 Consensus mechanism

The consensus mechanism implemented in the Blockchain technology facilitates reconciliation of engineering changes (objective 2.a., requirement 7 and 8). This is a bonus with respect to the above-addressed advantages of the Blockchain technology, satisfying the multiple requirements due to no further approval workflow that has to be included on this level of the IT architecture in the development process. As the OEM possesses the final developed product, it is in charge of the integration of components and legally responsible for placing the automobile on the market. Hence, the consensus mechanism is implemented such as that the OEM's rejection supersedes the approval of suppliers (cf. Chapter 4.2).

The most common consensus mechanism in the Blockchain technology, called *proof of work* (cf. Chapter 2.7.3), devours a lot of energy due to a mathematical problem that has to be solved by finding the nonce, to prove that sufficient computational resources have been mustered to avoid attacks on the data integrity of the Blockchain¹¹⁹. In contrast to a public Blockchain which has to be protected from hacking and attacks, the permissioned consortium Blockchain does not need such a security measure. Hence, the engaged nodes, i.e., the engineering partners, are not required to spend a high amount of computational power to reach a consensus about the validity of information artifacts shared. The prior admission of each supplier into the engineering network implies a scrutiny by the OEM and a contractual agreement and hence there should be no risk induced into the engineering network with respect to data integrity. Therefore, the consensus mechanism is being implemented simply by the click of a button for approval or rejection by the respective engineer who assesses the validity and correctness of data. Given certain metrics and rules, such an evaluation and approval process could be implemented in an automatic manner as a consensus mechanism. For that purpose, smart contracts could be used. However, due to the complexity of such an automated evaluation process for development data, smart contracts will not be addressed here (cf. Chapters 2.7.3 and 7.2).

In the following, the consensus mechanism is explained in more detail, as depicted in Figure 4-33. In this case, the particular scenario is described as that OEM, Supplier 1, 2, and 3 all develop together ECU_A and hence are all part of the channel ECU_A which is implemented in each of their distinct PLM Blockchain backbones. In the first step (①),

¹¹⁹ Please refer to BASHIR (2018: pp. 35 ff.) for more details about different consensus mechanisms.

the OEM updates information artifacts mostly relevant to the component of ECU_A that Supplier 1 develops, and writes this metadata into their PLM Blockchain backbone. In parallel, the RDF namespaces of the other suppliers will be updated preliminarily (②). Subsequently in step three (③), all suppliers confirm the validity and correctness of the received metadata. Their approval is depicted by a green check mark and is also transferred to all other engineering partners. The OEM stores this consensus knowing that they can proceed with development as planned. After this, the RDF namespaces will be updated permanently (cf. Chapter 4.2).

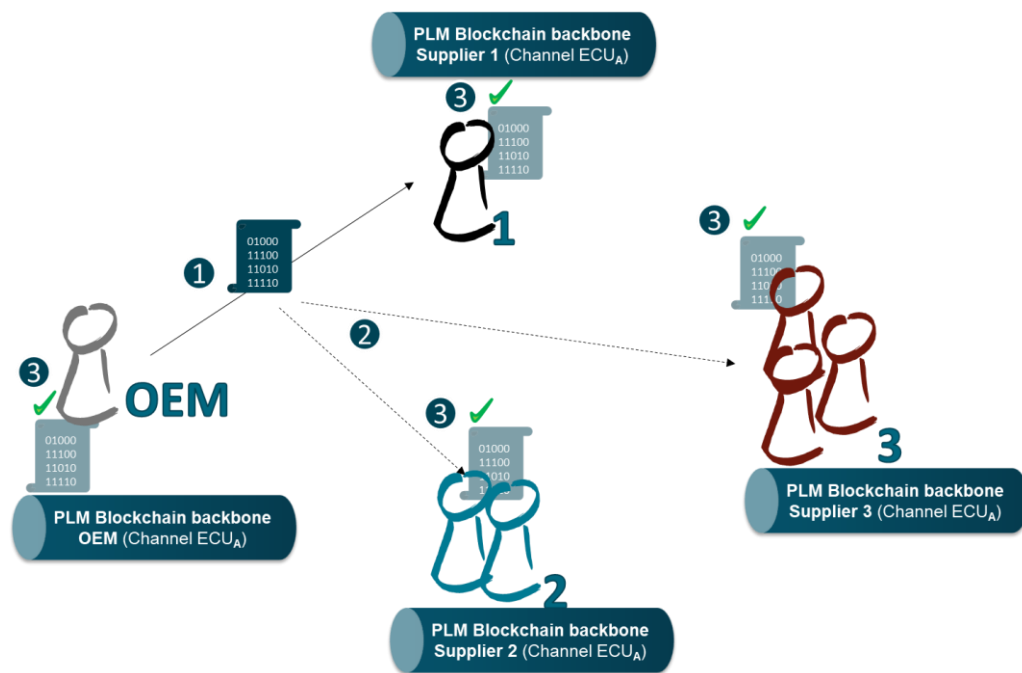


Figure 4-33: Consensus mechanism: Initial creation and distribution of data by the OEM and approval by the engineering partners.

Likewise, the consensus mechanism operates in the case that a supplier creates data, distributes it, and the other engineering partners will have to approve it. When Supplier 2 creates metadata which may be most relevant for Supplier 3 (④), for instance if the I/O table is changed and Supplier 3 has to adjust their interface, this data will also be synchronized to the others' PLM Blockchain backbones. The approval of the recently updated metadata occurs in step five (⑤). This is depicted in Figure 4-34.

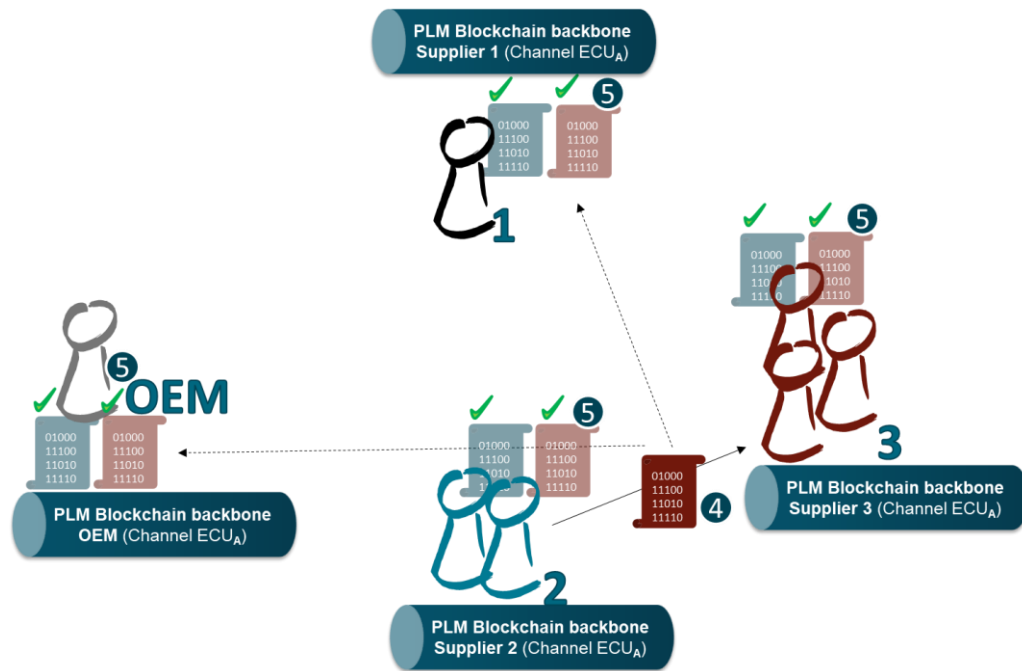


Figure 4-34: Consensus mechanism: Creation and distribution of data by a supplier and approval by the engineering partners.

There might also be the case where the synchronized metadata reveals discrepancies in the alignment of all the relevant components for ECU_A. This situation is depicted in Figure 4-35. Supplier 3 updates development data relevant for Supplier 1 and submits their metadata to the Blockchain network (⑥). However, Supplier 1 does not agree with the proposal for the update of the component of ECU_A. Supplier 1 rejects this update which will be immediately synchronized with the other engineering partners' PLM Blockchain backbones (⑦). Also, the initiator of this update, Supplier 3, receives the rejection and hence RDF namespaces will not be updated with a valid information artifact. As the rejection becomes apparent immediately to the entire engineering collaboration network, traceability regarding changes is increased and issues can be addressed in due course (requirement 7 and 8).

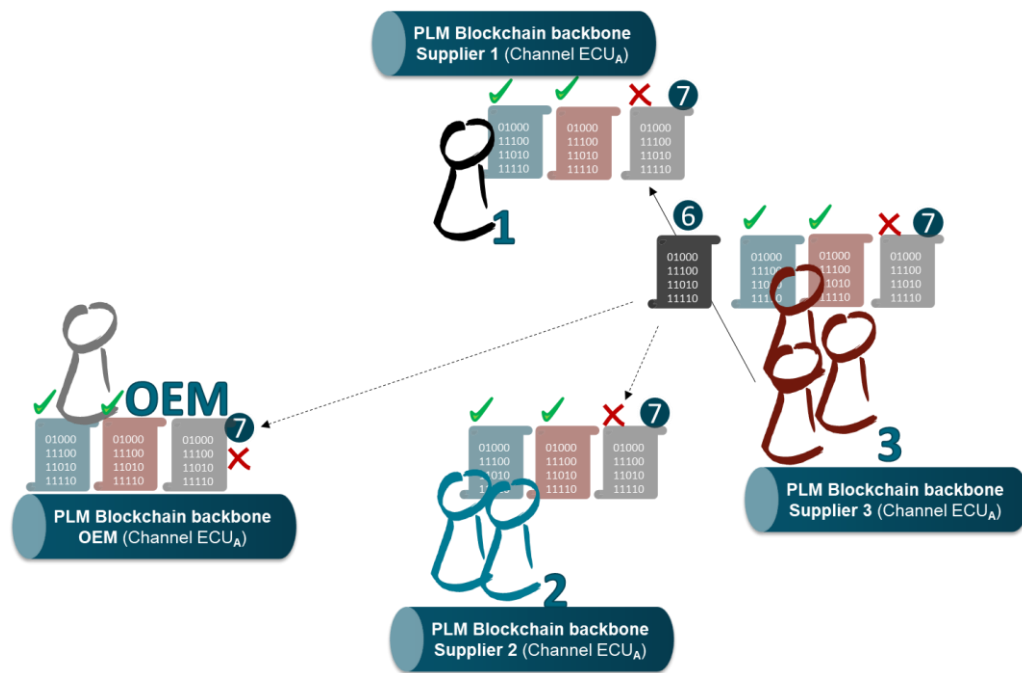


Figure 4-35: Consensus mechanism: Creation and distribution of data by a supplier and rejection by one engineering partner.

4.4 Solution framework and its satisfaction of requirements

As motivated in Chapter 1.3 and 1.4, internal and external traceability of E/E artifacts during automotive development in consideration of MBSE within distributed engineering collaboration has been assessed in Chapter 2. This was then evaluated in Chapter 3 using the three categories of enablers: data model, process model, and technology (cf. Figure 1-7). In Chapter 4, the synthesis of the solution approaches for each enabler has been conducted separately. Therefore, in this chapter the entire solution framework shall be described including how the framework satisfies the requirements.

In Figure 4-36, the entire framework is depicted. On top of this, the OEM's perspective is shown including the product lifecycle, highlighting the development process by a stylized *V-model*. Both IT tools and systems in scope, i.e., MBSE and PDM, are illustrated by database icons and the immanent peculiarities of both domains are presented by a schematical SysML model, as well as the effectivity of configurations over time. Below, the OEM's PLM Blockchain backbone with a dedicated channel for ECU_A serves as the data integration layer to enable internal and external traceability. On the bottom, the IT architecture of two suppliers is outlined. Both have their own PLM Blockchain backbone that are connected with the OEM's and each other's via the joint Blockchain network. Additionally, both suppliers have their own MBSE tool and PDM system which are, again, linked to their PLM Blockchain backbones.

The satisfaction of the requirements by the solution framework is depicted by circled numbers in Figure 4-36 which correlate with the numbered requirements in Chapters 3.2 and 3.3. Graphically, the circled numbers have been placed there where the requirements were addressed by either the data model, process model, or technology. Certainly, there may be more points where each requirement is addressed by the framework but the most important ones are depicted where they can be placed logically. In the following, it is described how and where in the solution framework the requirements have been implemented.

1. **Requirement ①:** The pins of an ECU are modeled explicitly in the MBSE SysML data model.
2. **Requirement ②:** NCDs, communication bus systems, signals, and interfaces are modeled explicitly in the MBSE SysML data model.
3. **Requirement ③:** The ECU's software versions including parametrization files are modeled explicitly in the MBSE SysML data model.
4. **Requirement ④:** The linked data model is implemented using UUIDs and URIs which can be used across all IT systems and are the pivotal identifier within work routines and RDF namespaces.
5. **Requirement ⑤:** Trace links are implemented using *http* schemes that can link from one IT system to another and are implemented in the RDF namespaces for referencing.
6. **Requirement ⑥:** Distributed engineering with the focus on MBSE and E/E is fostered by integration of data models that can be accessed by different engineering partners, as well as a joint IT architecture for data exchange such as the Blockchain technology.
7. **Requirement ⑦:** The consensus mechanism is inherent to the Blockchain technology and allows for approval or rejection of publicized engineering changes documented in separate information blocks.
8. **Requirement ⑧:** The automated change propagation is achieved by the P2P network of the Blockchain technology wherein changes in the form of new blocks are automatically distributed to all peers.
9. **Requirement ⑨:** The immutable product history is implemented by connected information blocks, creating hash values of the previous blocks, that are already immanent to the Blockchain technology.

-
- 10.Requirement ⑩:** Multi-directional synchronization of data is realized by the automatic change propagation within the Blockchain network and the updates of each engineering partner's RDF namespaces.
- 11.Requirement ⑪:** A traceability scheme for OEM and suppliers is implemented using the joint RDF work routines which integrate the distinct RDF namespaces of each engineering partner by means of trace links, UUIDs, and URIs.
- 12.Requirement ⑫:** Data integrity among multiple engineering partners is reached by means of separate PLM Blockchain backbones for each engineering partner.
- 13.Requirement ⑬:** The standardized data model for data exchange is implemented in the RDF work routine referencing each engineering partner's RDF namespaces.
- 14.Requirement ⑭:** The standardized development process is prescribed in alignment to the SPES method for MBSE and includes the consensus mechanism for each engineering partner.
- 15.Requirement ⑮:** The ad hoc integration of new engineering partners and their legacy IT systems as well as the OEM's is accomplished using the standardized REST API included in the OSLC framework.
- 16.Requirement ⑯:** Availability and robustness of data results from distinct PLM Blockchain backbones for each engineering partner, and hence data redundancy.

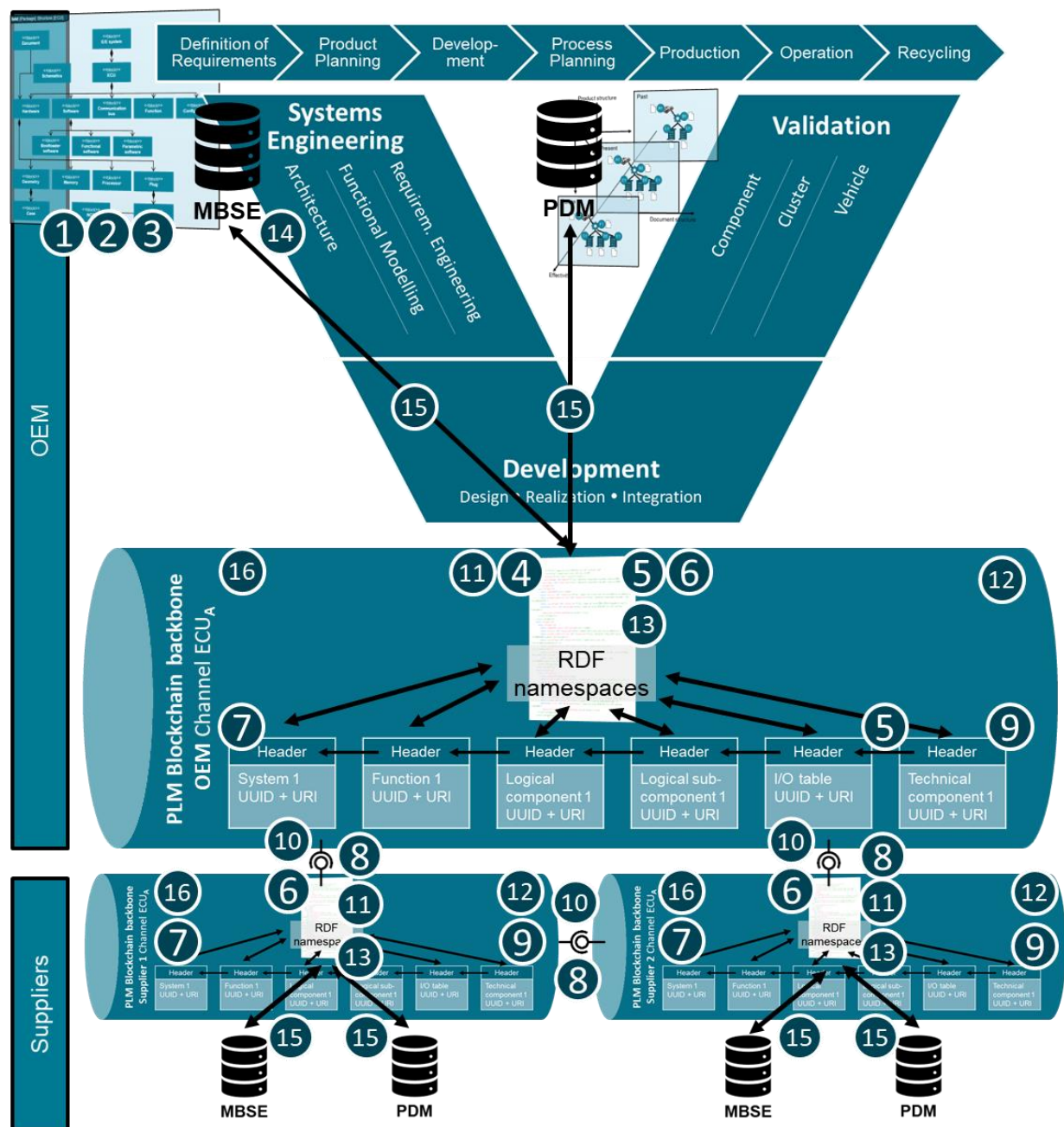


Figure 4-36: Solution framework for traceability of E/E artifacts during automotive development in consideration of MBSE within distributed engineering collaboration by means of the Blockchain and the satisfaction of requirements.

5 Prototypical implementation

The description of the different aspects under scrutiny, i.e., the enablers (cf. Chapters 1.4 and 3), follows the funnel approach coming from the broadest topic, the realization of the basic architecture (technology), via the process model to the data model. This approach was chosen as the Blockchain technology already determines processual and data aspects which later do not have to be repeated.

5.1 Goal and scope of the prototypical implementation

In the Chapters 4.1, 4.2, and 4.3 the entire framework to foster E/E traceability within distributed engineering collaboration during automotive development, with a focus on MBSE and PDM, was presented. The basic solution modules of the framework comprise a data model connecting MBSE and PDM intra- and inter-company wide, the definition of a process model for engineering processes in distributed development, and the conceptualization of an IT solution, in this case is the PLM Blockchain backbone network. The assessment of this solution framework and its applicability requires an integrated and extensive implementation of its solution modules.

Therefore, the goal of this implementation is the evaluation of the framework later in Chapter 6. Accordingly, the objectives of this thesis must be addressed, as presented in Chapter 1.3. The questions have to be answered whether this solution framework supports traceability of E/E artifacts within distributed engineering collaboration sufficiently? Hence, the implementation of a prototype shall address the objectives (cf. Chapter 3.5):

1. a. Internal traceability by means of the alignment of MBSE and PDM for E/E (requirements 1 to 4 in Chapter 3.2).
2. External traceability by means of
 - a. Reduction of reconciliation (requirements 5 to 8 in Chapter 3.3).
 - b. Transparent and safe product changes (requirements 9 to 12 in Chapter 3.3).
 - c. Alleviated connection of engineering partners (requirements 13 to 16 in Chapter 3.3).

As these objectives and thereof deduced requirements, according to the state of science and technology, serve as the leitmotif for this thesis, they also provide the basis for the prototypical implementation and the evaluation in Chapter 6.

The implementation of a prototypical IT architecture, the underlying framework, and tools stipulates how precisely a data and a process model can be implemented. Consequently, the decision of which programming language will be used and which information artifact will be created at which point in time in this case are aligned with the *Hyperledger Fabric* platform¹²⁰.

This prototypical implementation was executed together between one OEM and two engineering collaboration partners as all three partners face the same challenges regarding internal and external traceability. Thus, albeit being a specific implementation, it addresses deficiencies of multiple engineering partners and, in its generic form, the prototypical implementation can be applied for other similar engineering collaborations within the automotive industry or others (cf. Chapter 7).

The direct connection with legacy IT systems such as MBSE tools and PDM systems could not be implemented during this work due to the tremendous complexity this would have induced into the prototype development.

5.2 Implementation of a prototypical IT framework

In addition to Chapter 4.3, some more details regarding the technological realization of the prototype and its basic IT architecture have to be described.

Due to being a standard, modular platform used for the Blockchain technology and the convenient inclusion of distinct channels (cf. Chapter 4.3.1), the *Hyperledger Fabric* platform is used for the prototypical implementation (cf. Footnote 116 on p. 157). Moreover, *Hyperledger Fabric* focuses on permissioned Blockchains (BASHIR, 2018: p. 471), as this is the case here in this work for distributed engineering collaboration¹²¹.

¹²⁰ Commonly, the strategic level provides the basis for an operational level including, for instance, the product development process. Underneath, there is yet another operational level with sub-processes with, e.g., the design process. The lowest or PLM level constitutes of concrete IT solutions on a functional and system level (EIGNER and STELZER, 2009: pp. 23–24). Here in this work, the scope does not include strategic considerations of a company and its impact on the operational level. Therefore, due to the necessity to reduce complexity, it is assumed that the IT solution is also aligned with how the processes will be implemented.

¹²¹ For more information about the *Hyperledger Fabric* platform or framework, please refer to BASHIR (2018: pp. 461 ff.).

Figure 5-1 depicts the entire network deployment for a network with four engineering collaboration partners. Each engineering partner has the same set of databases that are part of the modular platform of *Fabric*¹²². The *PostgreSQL* or short *Postgres* data base (DB) is an opensource, object-relational DB (SCHICKER, 2017: p. 13) and is used here to store cryptographic credentials about the organizations and users. As the Blockchain comprises of blocks which, in turn, contain transactions in the payload (cf. Figure 2-23 and Chapter 2.7.3), these transactions can again be executed by the *chain code*¹²³ and update the so-called *world state* (cf. Chapters 2.7.3 and 4.3.1). The *world state* is a key-value DB and stores further information of transactions, sub-transactions and further automatisms, such as smart contracts. Each peer of the Blockchain network stores its *world state*, which in this case is on a *CouchDB*¹²⁴ (BASHIR, 2018: pp. 473, 477). A *CouchDB* is a document-based DB (SCHICKER, 2017: p. 16) and in the *Hyperledger Fabric* platform connected to the peers which can execute the above-mentioned updates of the *world state*¹²⁵. The actual ledger, i.e., the Blockchain with its concatenated blocks of transactions, is maintained on each peer's file system, denoted by "OS" (ordering service) underneath the green "THINAPP" that denotes the *Fabric client* of each peer's Blockchain node. For the purpose of the distribution of each transaction, each engineering partner's (peer) nodes are connected to the "common ordering service", depicted in the middle of Figure 5-1. The common ordering service receives endorsed transactions, orders them into a block, sorts them according the specific channel ID, and executes the transmission to all participating peers¹²⁶ (BASHIR, 2018: pp. 481–483).

Broadcasting of information, i.e., of the consensus mechanism as well as the in blocks combined transactions, is implemented using *google remote procedure calls* (gRPC), based on *HTTP/2*, including protocol buffers. Nodes in the Blockchain network within the *Hyperledger Fabric* platform exchange four main message types: i) discovery, ii) transaction, iii) synchronization, iv) consensus. The first message type is used to

¹²² This implementation of the basic IT architecture using the *Hyperledger Fabric* platform can vary according to its modular composition and hence the chosen databases also could be substituted by others that serve the same purpose.

¹²³ Chain code is synonym for smart contracts due to its code being executed on the Blockchain (BASHIR, 2018: 472).

¹²⁴ As described in Chapters 2.7.3 and 4.3.1, smart contracts are not in the scope of this work and hence the *world state* is just mentioned because it is a standard part of the *Hyperledger Fabric* platform. The prototype was already designed to include smart contracts for potential future use cases.

¹²⁵ Given the modular definition of the *Hyperledger Fabric* platform, instead of a *CouchDB* also other databases, such as *LevelDB*, can be implemented (BASHIR, 2018: 473).

¹²⁶ A more detailed description of each module of the *Hyperledger Fabric* platform is given in BASHIR (2018).

discover and identify new network nodes on launch of the network or in case of a new entry of a peer to it. Transaction messages include the handling of transactions in all their states. Updates of all the nodes and their synchronization is achieved via the corresponding message type. Likewise, consensus messages are defined (BASHIR, 2018: p. 474).

The “CA” in each peer’s network setup represents the company’s own certificate authority (CA) for the purpose of authorization operations and identity management including role assignment. Upon identification, a peer can join the network. By short-term certificates, peers can join for one-time transactions only (BASHIR, 2018: p. 472) which might be the case for a one-time contribution to a source code of an ECU by, for instance, a start-up.

The orange and green “APP”, or “THINAPP” respectively, denote the applications a user can interfere with. As the peers are the only clients within the Blockchain network, they only have thin applications (“THINAPP”) whereas via the common ordering service the entire network can be configured (“APP”). On top of each peer’s thin app, additionally there is a web app coded in *Java*. This is for enhanced user experience for the end user, e.g., the engineer, and will be described in more detail in Chapter 5.2.2.

The setup of each the *Hyperledger Fabric* node including all relevant data bases for each engineering collaboration partner was executed using *Docker* software for the creation of virtual container machines based on a Linux kernel. Each *Docker* image can be deployed on the premise of engineering partners and it contains all relevant information to set up the Blockchain network and participate in the development. The relevant code already is included in the *Docker* image and was coded in *Java*, *Python*, *JSON*, and *GO language*¹²⁷.

In Figure 5-1, each engineering partner has two peers. This stems from the fact that the *Hyperledger Fabric* platform stipulates the endorsement of transactions, i.e., the execution of transactions, is simulated before they are submitted (BASHIR, 2018: pp. 481–483). For the implementation of this prototype, the setup of two peers per engineering partner was chosen for this reason. This process of endorsement of transactions will be described in more detail in Chapter 5.3.

¹²⁷ Please refer to RAVAL (2016), DHILLON et al. (2017), PRUSTY (2017), and BASHIR (2018) for more information about the architecture of a Blockchain network, *Hyperledger Fabric*, and *Docker*.

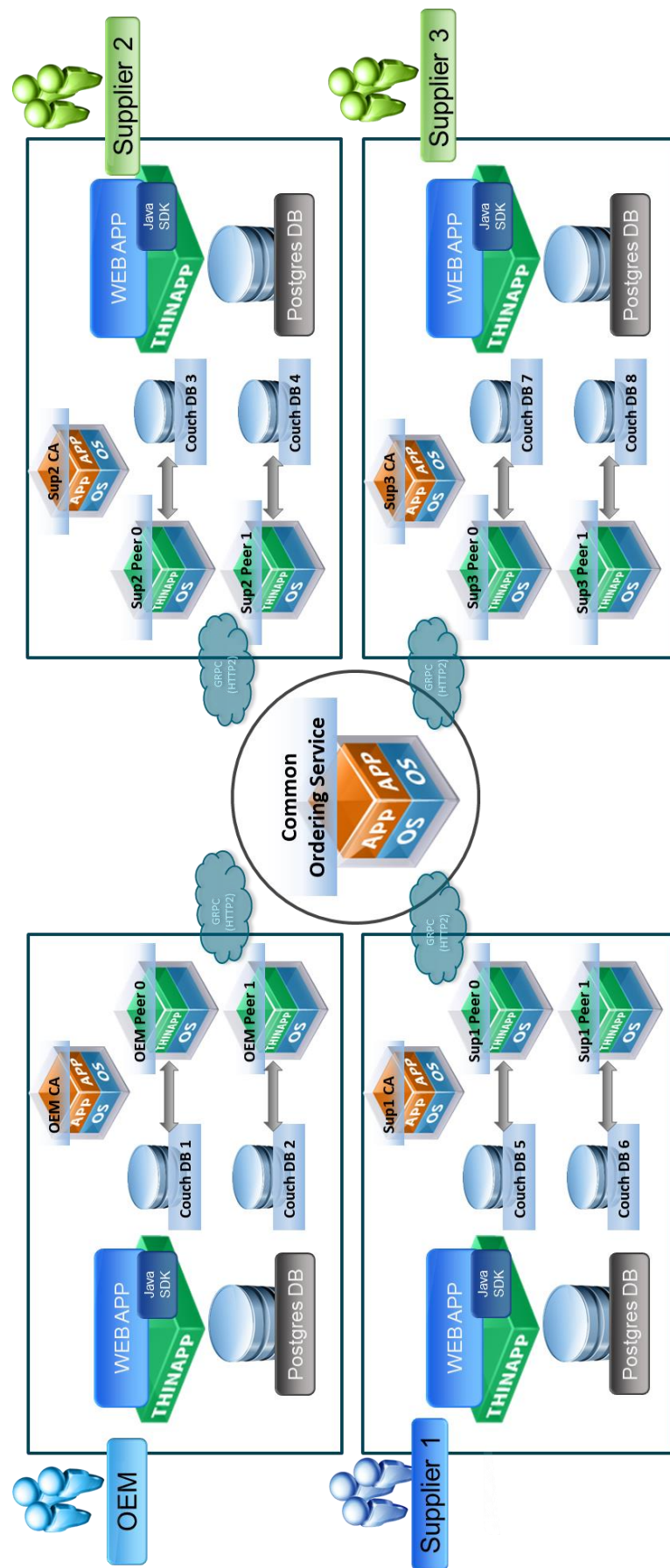


Figure 5-1: Network deployment of the prototypical implementation of the Blockchain network with multiple suppliers.

5.2.1 The structure of the prototype

The Blockchain prototype consists of 429 files and 296 folders. The overview of the entire structure of the prototype is depicted in Figure 5-2. On the highest structural level, the prototype consists of two main sections that are visible in Figure 5-2 on the left-hand side and the right-hand side, respectively, of the grey oval. In the following, each branch and its functionalities will be described in more detail.

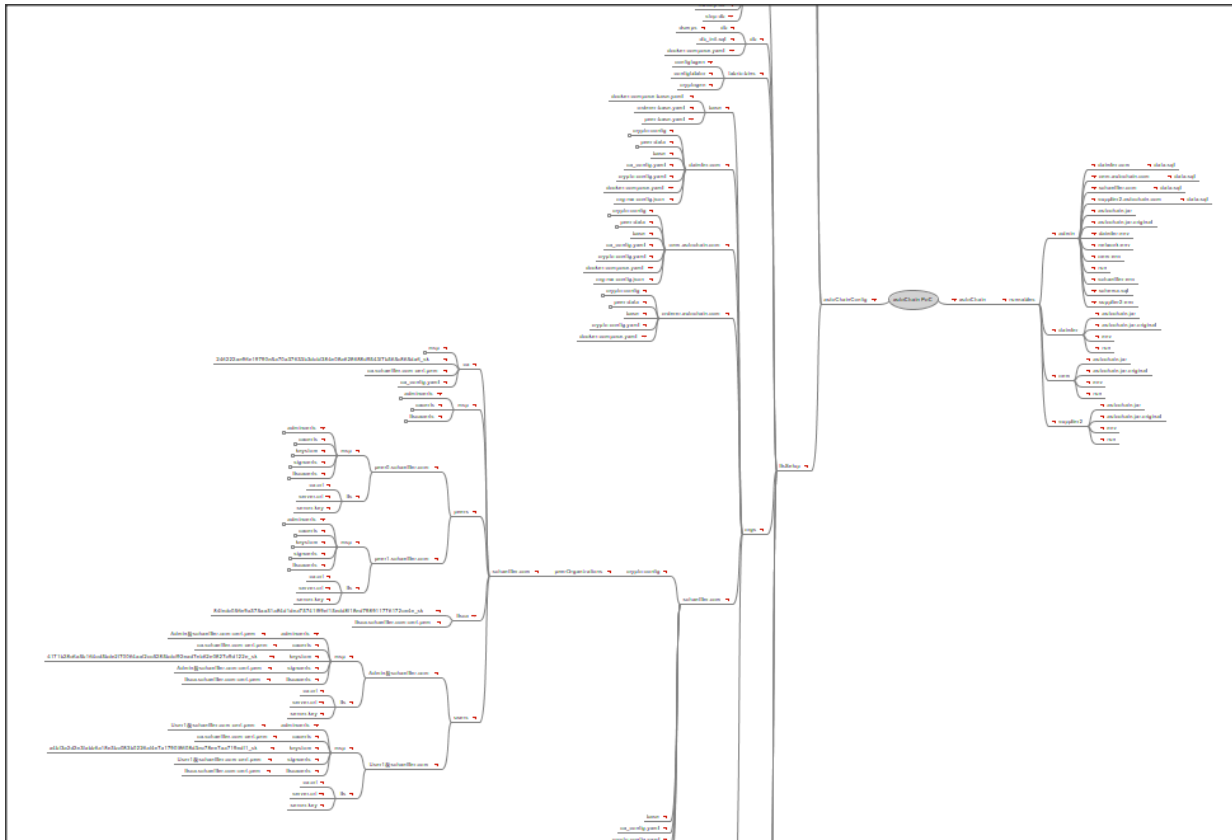


Figure 5-2: The structure of the prototype.

This top-level structure of the prototype, as depicted in Figure 5-3, contains on the left-hand side, the `autoChainConfig` node, all relevant files and scripts which bootstrap the Blockchain network. The `autoChain` node on the right-hand side comprises Java compiled runnables which provide the application interfaces, e.g., the “THINAPP” and “WEB APP” (cf. Chapter 5.2), as well as the possibility to interact with the Blockchain network for each engineering partner.

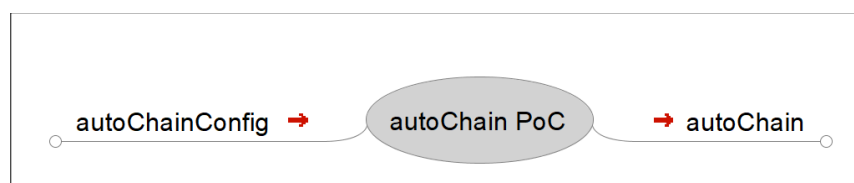


Figure 5-3: Top level structure of the prototype.

The `runnables` include sub-structures for each engineering partner¹²⁸. For each of them, different information is provided in further folders or archives such as the *Java archives* (`.jar`). Included are the environment settings for the application, ports definitions, DB names, and organization names. In Figure 5-4, the `runnables` for `daimler` are depicted which are the same as for `oem` and `supplier2`.

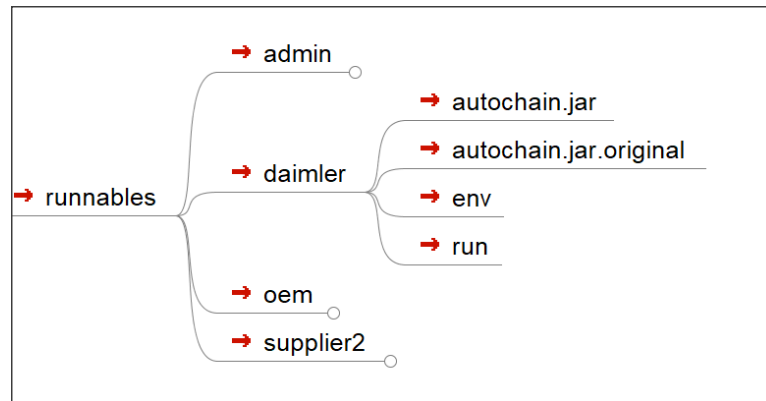


Figure 5-4: Runnables including different, organization-specific settings.

For the administrator of the Blockchain network, here in this case this would be superimposable with the OEM as defined above, there are other environments implemented and all organizations are combined to one network by the `admin`. For each engineering partner, further data is stored using `SQL`. This is depicted in Figure 5-5.

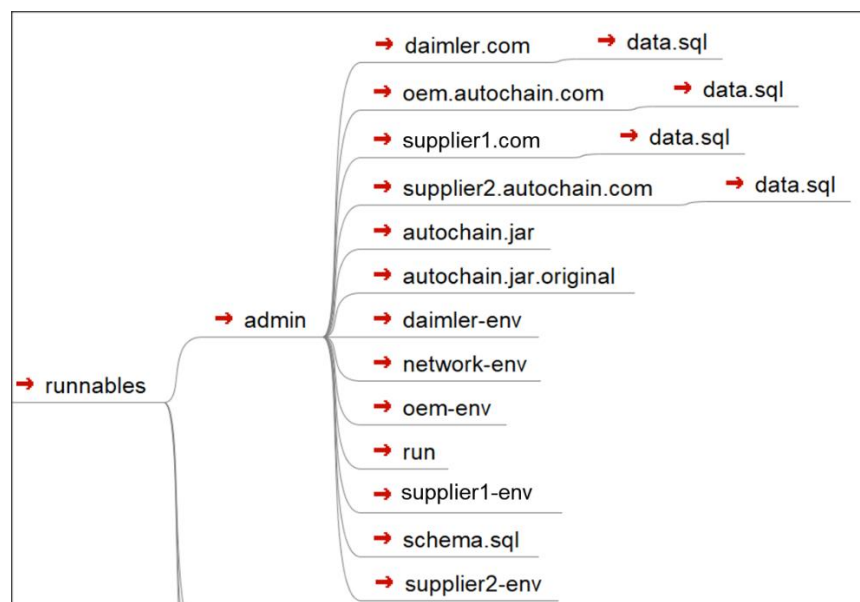


Figure 5-5: Runnables for the administrator.

¹²⁸ The nomenclature of the following images is not congruent with the previous one. Previously, Daimler was described as the OEM, being responsible for the final automobile, and the other engineering partners which were denoted as suppliers.

The left-hand side of the above-mentioned delineation of the structure of the Blockchain prototype consists of a Blockchain explorer application, files for documentation, setup scripts, and files for bootstrapping the Blockchain network. Figure 5-6 depicts the Blockchain network definition files.

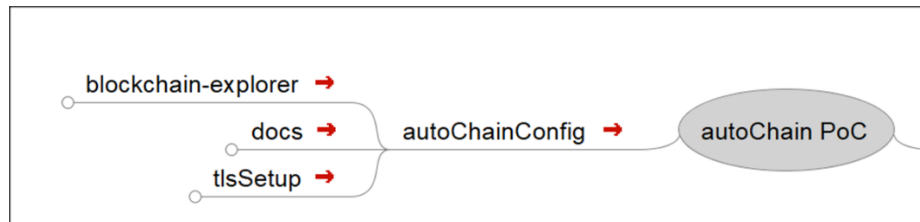


Figure 5-6: Blockchain network definition files.

Figure 5-7 depicts the Blockchain network explorer files in more detail, for instance the database, metrics, router, services, monitoring, initiation and termination of the network. The explorer app is implemented using `Node.js express`. Moreover, the `Fabric explorer` is used for the similar purpose as the *Blockchain explorer*. The file `Docker compose mysql` written in `YAML` is used to facilitate the network setup for new peers in the network, as described above, by cloning the repositories, bootstrapping all relevant files and repositories, and starting the joint Blockchain network.

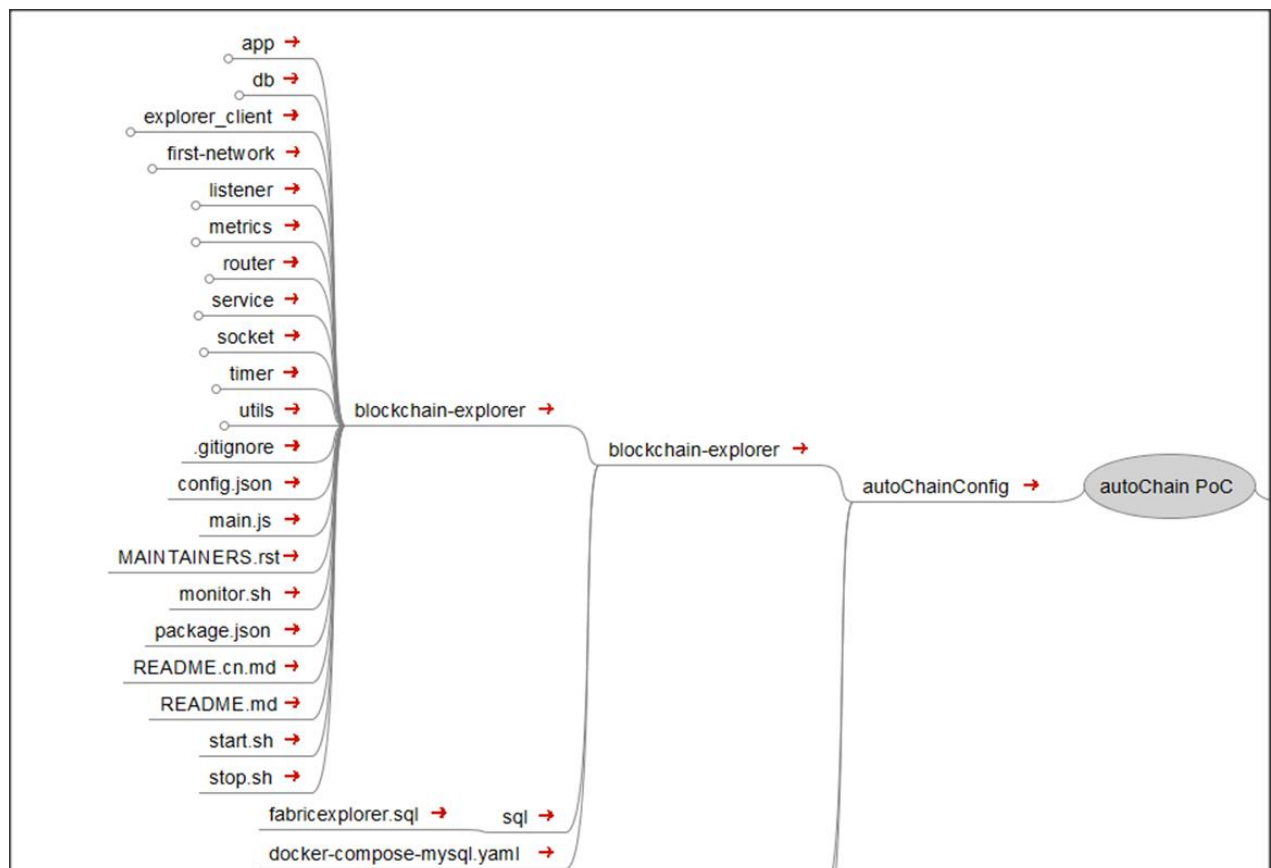


Figure 5-7: Blockchain network explorer files.

Figure 5-8 depicts the *transport layer security (TLS)* setup files. For the initiation of bootstrapping of the Blockchain network, these files are essential. Furthermore, these files include all configuration files which are necessary for using `Docker compose`. These files will be described in more detail in the following, due to their relevance and distinctiveness for the Blockchain network and the prototypical implementation.

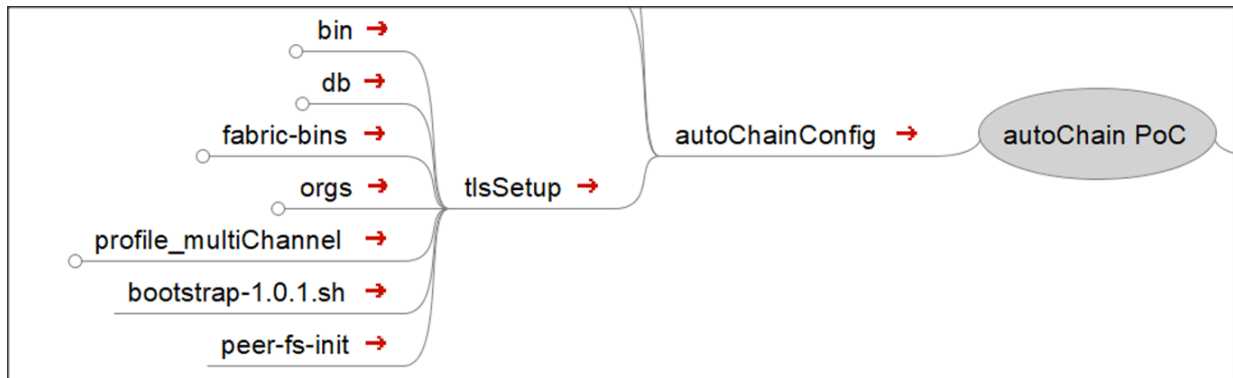


Figure 5-8: Blockchain network setup files.

The `TLS setup` folder further contains scripts to generate and manage the Blockchain network. In Figure 5-9 it is visible that there are folders including information for the generation of channels including the *genesis block* and providing of cryptographic items to the network's peers such as private keys and certificates. `init-db` denotes the local database and all folders with the prefix `nw` which either starts or stops the Blockchain network. Moreover, the local database can be stopped (`stop-db`).

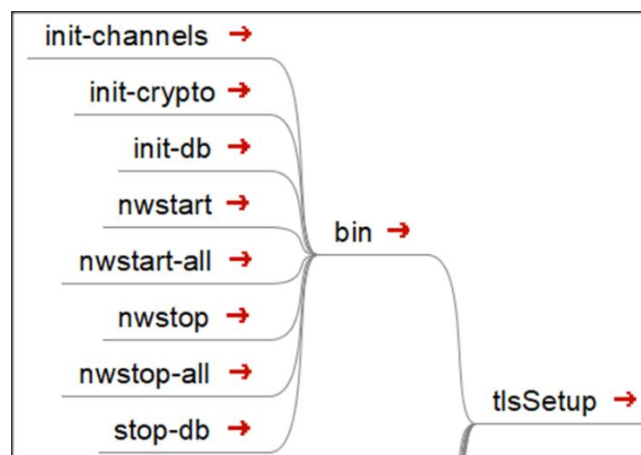


Figure 5-9: Blockchain network setup binaries.

The folders `db` and `fabric-bins` contain the relevant binaries for the storage of the configuration files for the local database and the *Hyperledger Fabric* platform, respectively (cf. Figure 5-10).

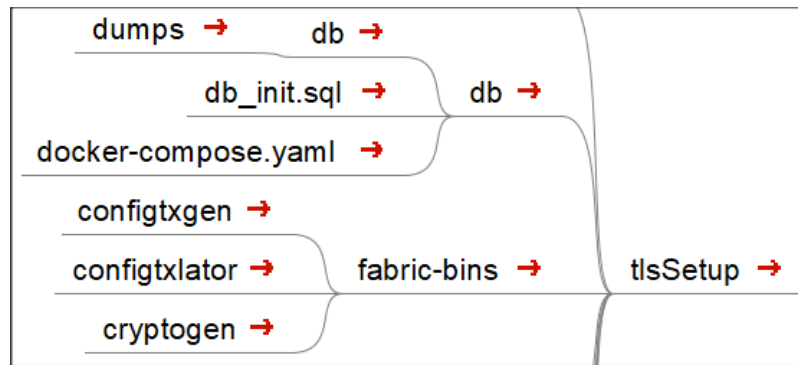


Figure 5-10: Local database and Fabric binaries.

The folder `orgs` in Figure 5-8 contains the individual setup files for the organizations within the network. Configuration files (`.yaml`) that are equal for all peers are stored within the folder `base` in Figure 5-11.

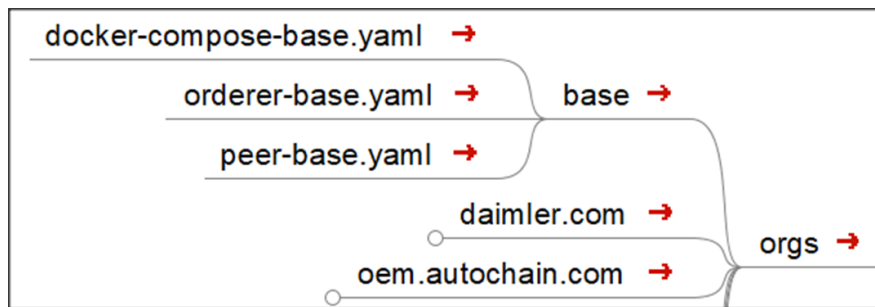


Figure 5-11: Base files for the organizations within the Blockchain network.

For each organization within the Blockchain network, configuration and the cryptographic files are stored within the `orgs` folder of the `tlssSetup` branch, as it is depicted exemplarily in Figure 5-12 for one peer.

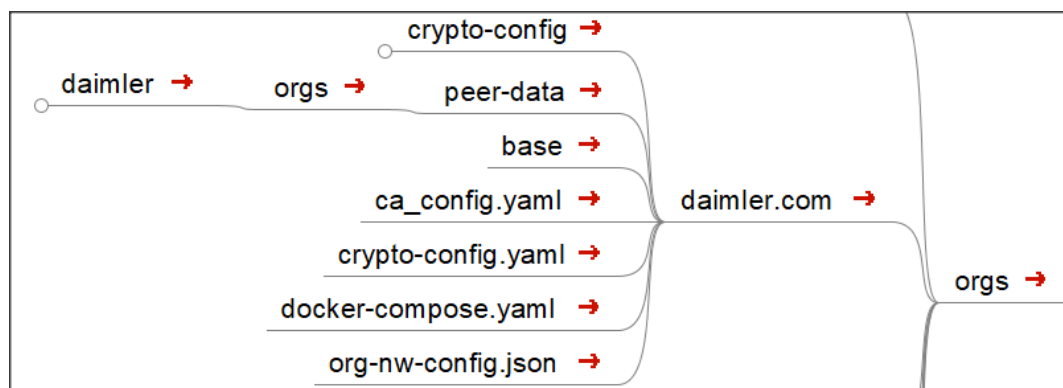


Figure 5-12: Cryptographic and configuration files for each organization.

Within each organization, there is again a folder called `crypto-config` within which the certificates (`ca`), the *membership service providers* (`msp`), and peer and user handling are located, among others. This is depicted in Figure 5-13 and Figure 5-14¹²⁹.

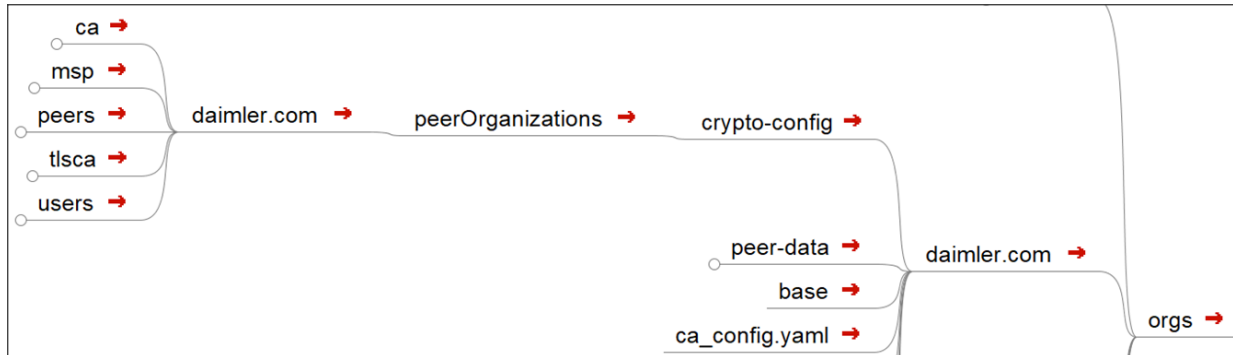


Figure 5-13: Structure of cryptographic files for peers of each organization.

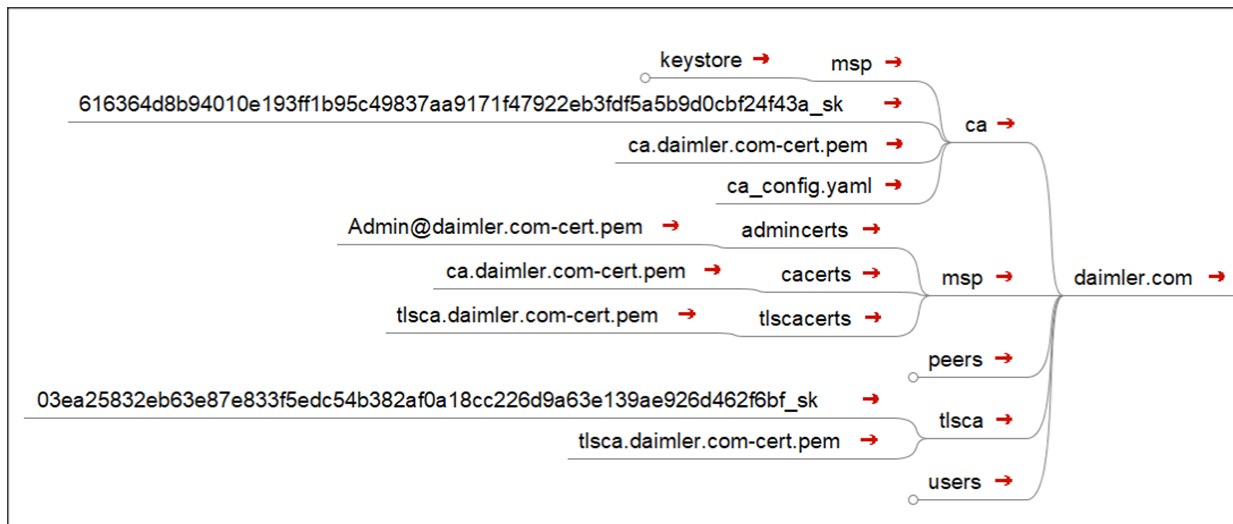


Figure 5-14: Structure of cryptographic files for one peer.

Maneuvering back to the `tlsSetup` node, the folder `profile_multiChannel` contains all relevant data for the implementation of multiple channels which are applied for the distinction of separate development of E/E systems with the potential for a different group of engineering partners for each channel. This is to satisfy the need-to-know-principle (cf. Chapter 1.2.2). It also comprises the chain code (`cc`), or smart contracts, which is sourced externally on *github*. This is depicted in Figure 5-15.

¹²⁹ Due to the same folder structure being used for all organizations and peers, the other organizations for the remaining engineering partners are omitted here.

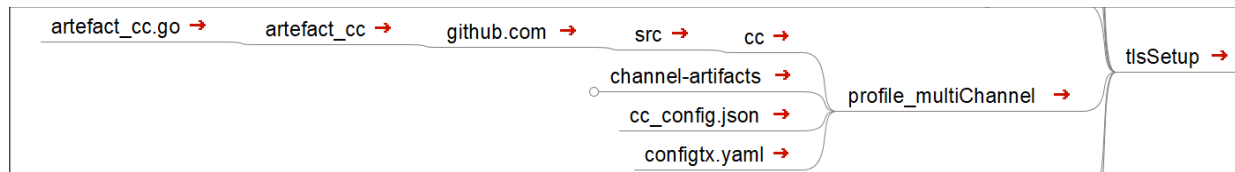


Figure 5-15: Structure for multi-channel setup including chain code.

In Figure 5-16, there are three generic E/E artifacts, `cu-com-module`, `cu-eng-ctrl-med`, and `cu-radar-sensor-long-driv-assist-sys`, implemented as distinct channels. The `tx` files contain the channel creation transactions, whereas the endorsement policy for each channel is described in the `yaml` files. The endorsement policy defines the consensus mechanism (cf. Chapter 2.7.3).

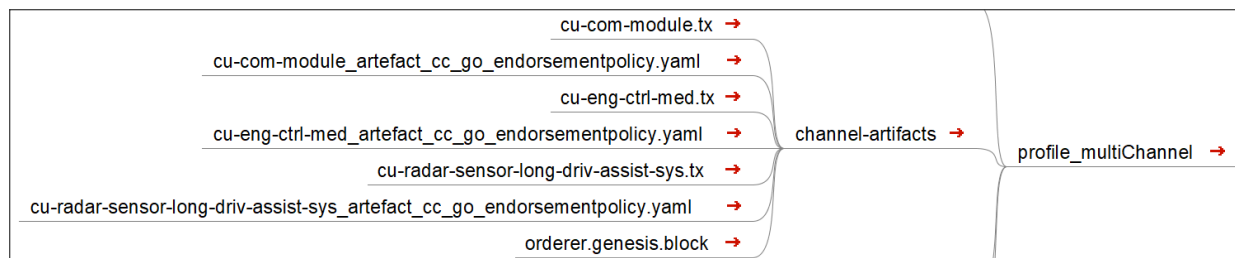


Figure 5-16: Channel artifacts.

5.2.2 GUI

The graphical user interface (GUI) was designed to be web-based. Hence, it is feasible to open up the GUI in which the engineer will document their changes on each device capable of using a browser. Each organization, e.g., the OEM and the separate suppliers, have access to their channels where they contribute to the development or to all, in case of the OEM. Moreover, individual users can be selected during login who will be documented as the creators of transactions. The user's name is displayed in the top right corner of the browser window. Additionally, in the menu on the left side under "Artifacts", all created artifacts are shown as a list including relevant details. The menu item "Blockchain" grants general information on channels and the Blockchain network.

The creation of an artifact by *Supplier1* is depicted in Figure 5-17. There, *Bob Supplier1* fills out the required attributes. A UUID is generated automatically. Bob provides the part number ("Sachnummer"), declares the artifact type as a "control_unit_software" in "version 1.1", and adds a description. The architecture of this software is *AUTOSAR* (cf. Chapter 3.4).

The screenshot shows a web browser window with the URL `localhost:3081/artefact/create?sachnummer=&description=&channel=`. The page title is "Supplier1.com" and the user is logged in as "Bob Supplier1". The left sidebar has "Artefacts" and "Blockchain" links. The main content area is titled "Create Artefact" and contains a form with the following fields:

- Artifact Information** (Section Header)
- UUID:** 9694ce41-63b1-498f-b249-fe1602374b58
- Sachnummer:** * (Input field with value "WS0001")
- Artifact Type:** * (Input field with value "control_unit_software")
- Version:** * (Input field with value "1.1")
- Issuer:** Supplier1.com
- Architecture:** * (Input field with value "AUTOSAR")
- Description:** * (Input field with value "Software für den Wankstabilisator WS0001")

At the bottom of the form are "Submit" and "Cancel" buttons.

Figure 5-17: Supplier1 creates an artifact.

After the artifact creation, *Supplier1* sees his created information artifacts within his list of artifacts, as depicted in Figure 5-18. Within this list he can search for a specific part number, a term within that description, as well as the release status which can be “in progress”, “rejected”, or “final”. The OEM also sees the transactions in their list in which artifacts have been created by the *Supplier1* and that their status is “in progress”, signifying that the OEM still has to vote for these artifacts. This is depicted in Figure 5-19.

The screenshot shows the "Artefacts" page in the Supplier1.com interface. The URL is `localhost:3081/listArtefacts`. The page title is "Supplier1.com" and the user is logged in as "Bob Supplier1". The left sidebar has "Artefacts" and "Blockchain" links. The main content area is titled "Artefacts" and contains a search bar and a list of artifacts.

Search Bar:

- Input fields for "sachnummer", "Description of the software", and "Release Status..."
- Buttons for "Search" and "Clear"

Artefact List:

Artefact with Sachnummer: FH0815 created successfully.	
Sachnummer: FH0815	Issuer: Supplier1.com
UUID: 79f09102-47ad-4816-b657-20ee5570c3cb	Release Status: In Progress
Artifact Type: control_unit_software	
Description: Software für das Fensterheber-Steuengerät	
Details Vote	

Sachnummer: WS0001	Issuer: Supplier1.com
UUID: 212fa0d5-a2a3-461b-9fb0-4302ea9c54b6	Release Status: In Progress
Artifact Type: control_unit_software	
Description: Software für den Wankstabilisator WS0001	
Details Vote	

Figure 5-18: List of artifacts of Supplier1 with pending voting answers.

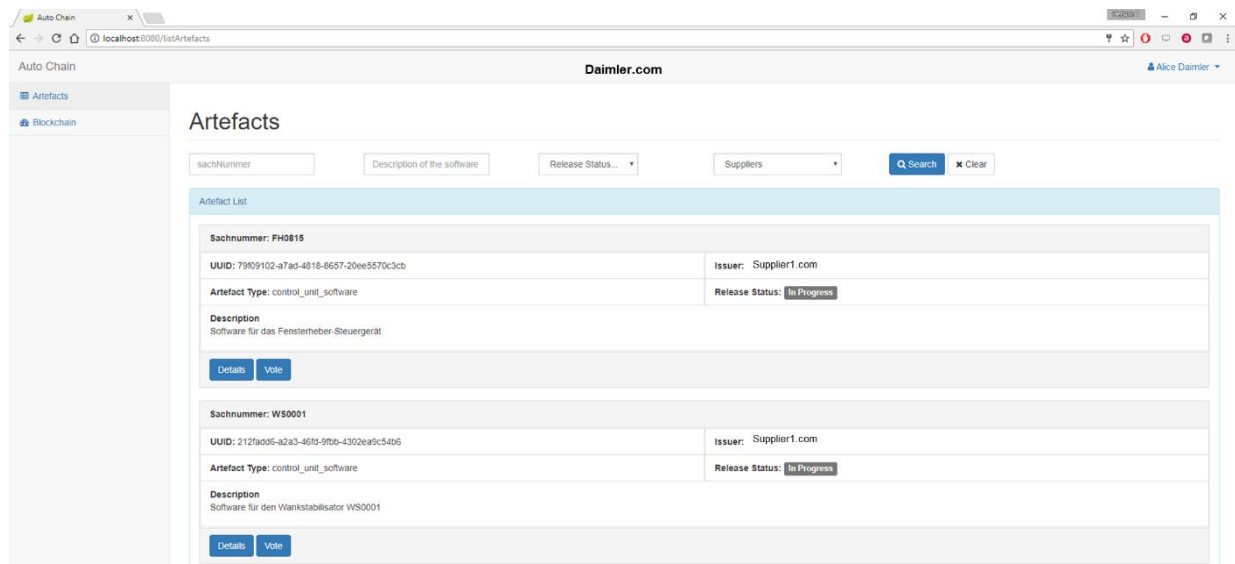


Figure 5-19: List of artifacts of OEM with pending voting actions.

Next, the OEM has to vote given the consensus mechanism. They can approve the engineering activity, i.e., the creation of a new information artifact as an initial development or as a change of an existing artifact, or reject it by clicking the respective buttons, as depicted in Figure 5-20¹³⁰.

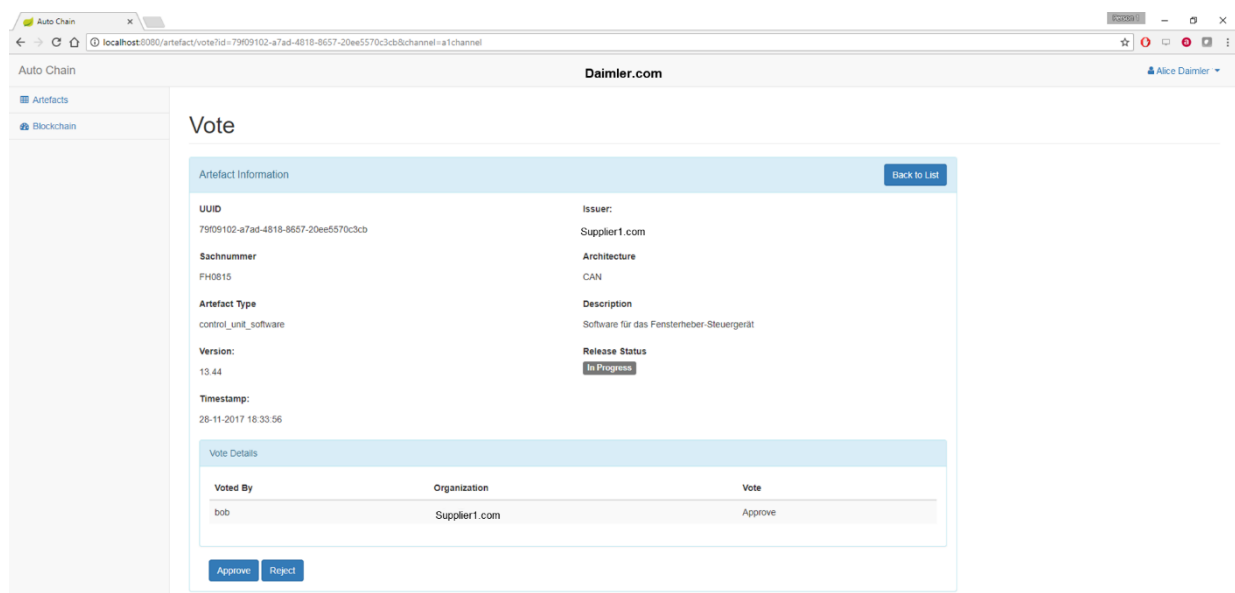


Figure 5-20: Voting by OEM.

¹³⁰ In the prototypical implementation as depicted here, the supplier still had to confirm his own creation of information artifacts. This is due to technical peculiarities given the design for multiple users of one organization who could vote independently. This scenario could be reasonable in the case of multiple parties within one company, such as different domains, shall also have to confirm or reject changes. However, this scenario is not in the scope here albeit it was designed initially according to this in the prototype.

After voting, the positive result of the OEM's action is visible to them as well as pending requests for voting, as it is depicted in Figure 5-21. Contrarily, in Figure 5-22 it is shown that the OEM also can reject any creation of a new information artifact.

Again, the *Supplier1* receives all results of the consensus mechanism by the OEM. These results are depicted in Figure 5-23 in the supplier's list of artifacts.

Details of artifacts can be displayed by clicking the corresponding button in the list of artifacts. Within the details, the transaction history for this particular information artifact is visible as well as, who has voted how. This is depicted in Figure 5-24.

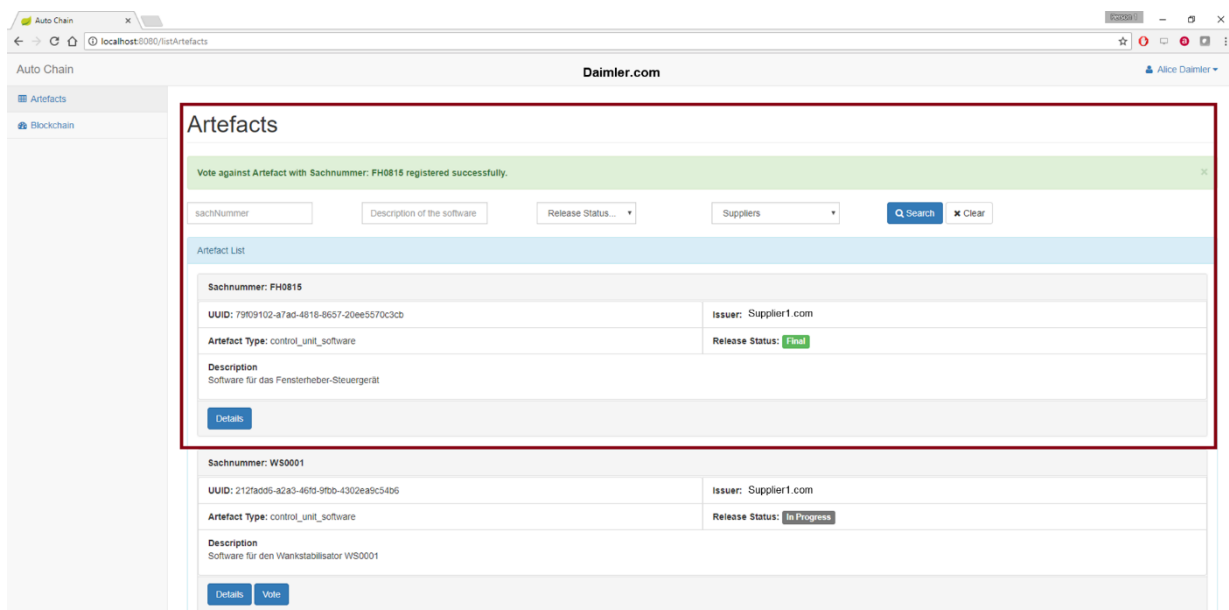


Figure 5-21: List of artifacts of OEM with different release status.

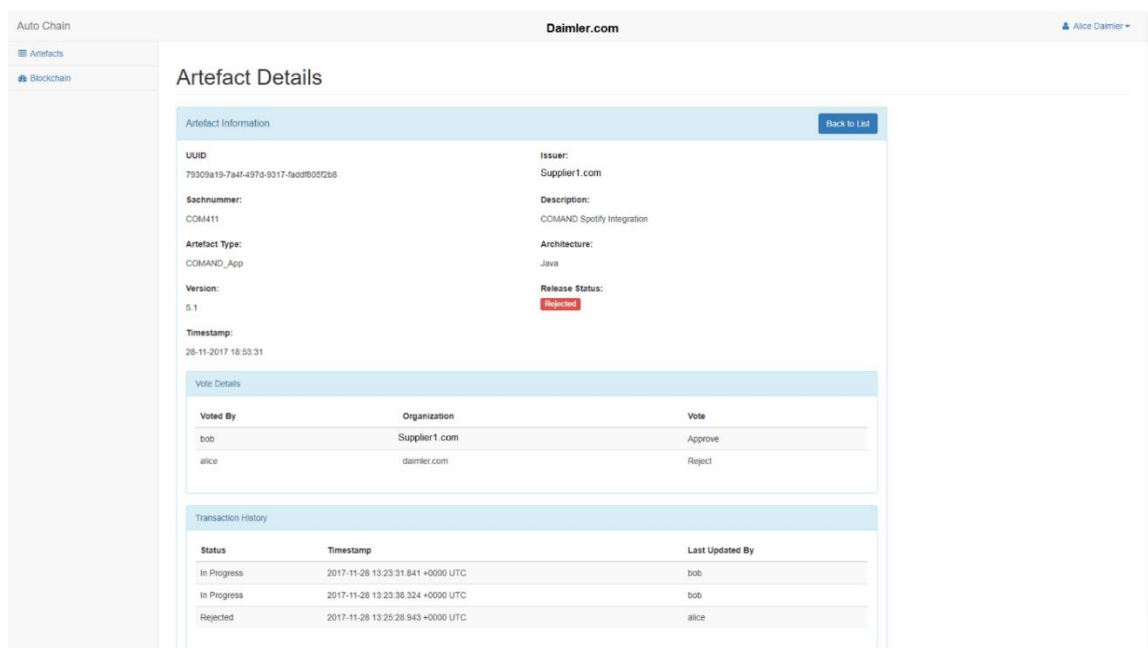


Figure 5-22: Details of rejection by the OEM of supplier's artifact creation.

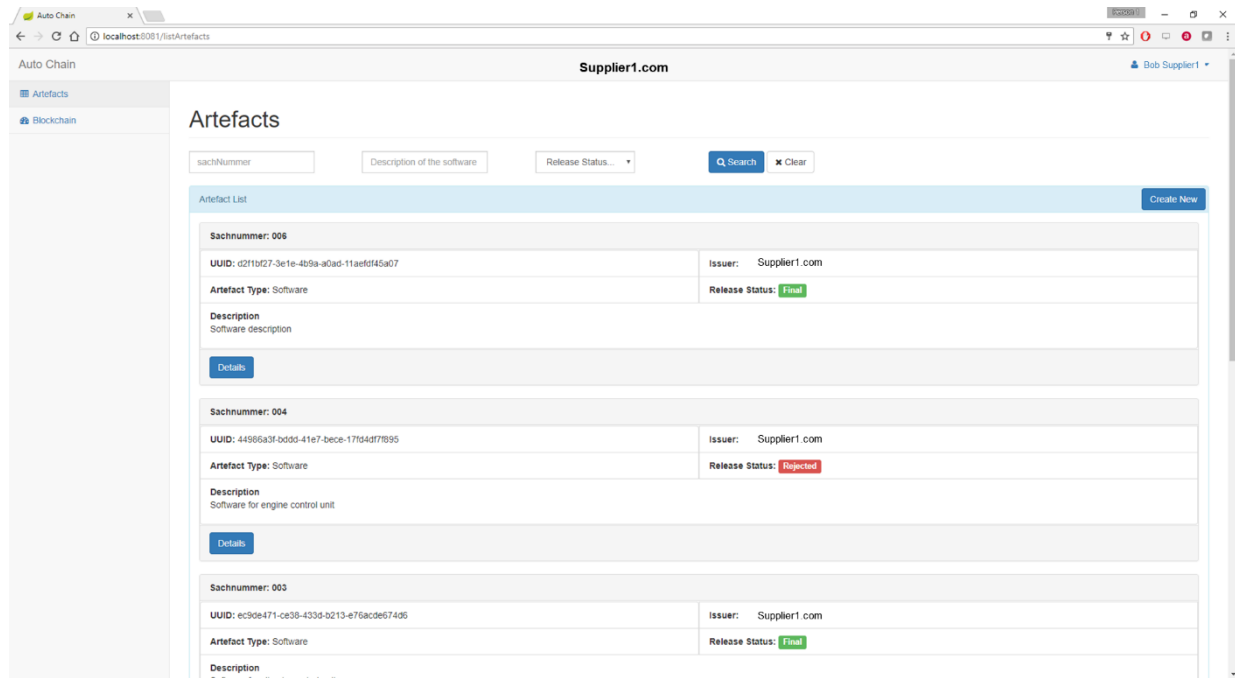


Figure 5-23: List of artifacts of Supplier1 after voting.

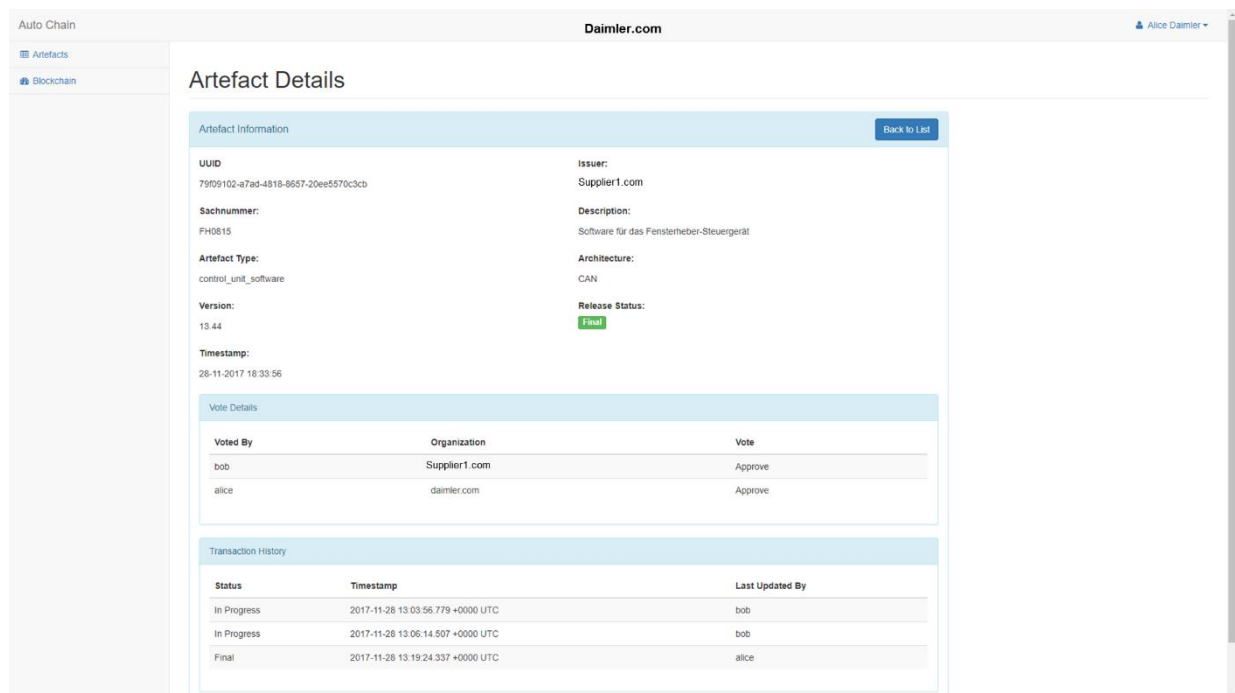


Figure 5-24: Artifact details including transaction history.

A repetitive voting by the OEM is rejected by the system, as depicted in Figure 5-25. Figure 5-26 depicts the responsive design of the web interface for an iPhone 6. This enables engineers to work on devices with different screen sizes which again, fosters mobile work and hence instantaneous responses to engineering changes. This might be particularly relevant in a testing scenario of a prototype where changes can be

confirmed in the field, contemporaneous updates of MBSE and PDM systems are triggered, and these updates could be flashed over the air for quick retried testing.

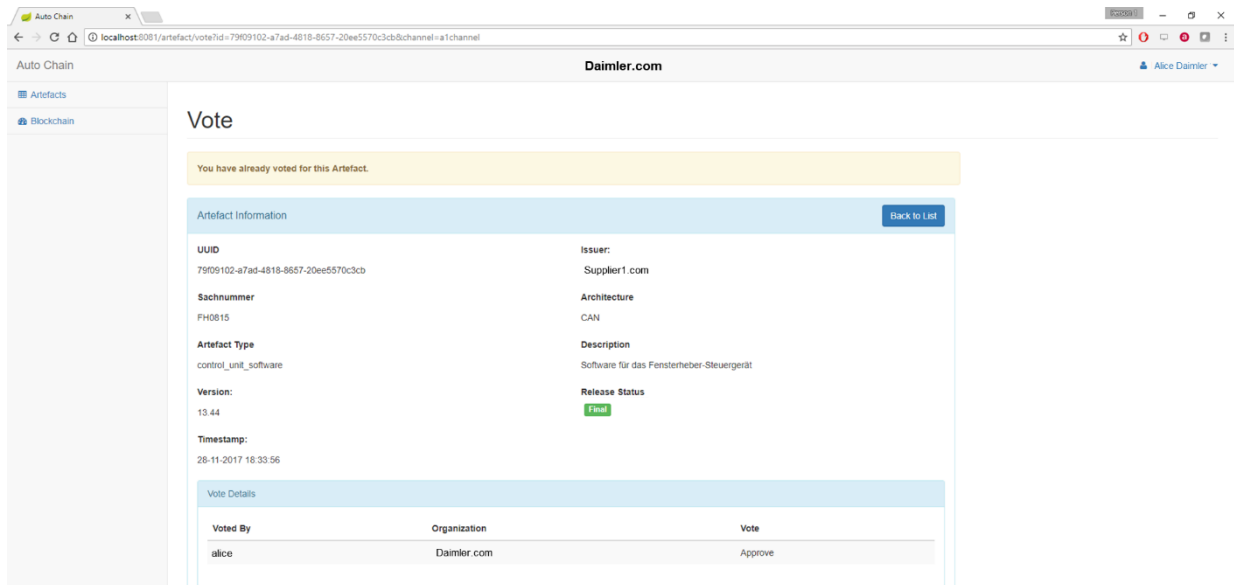


Figure 5-25: Voting retry.

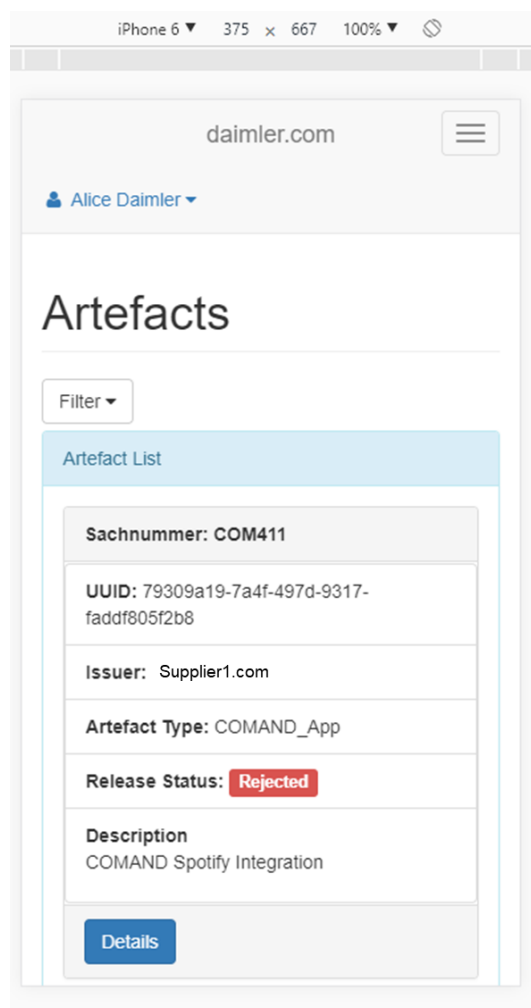


Figure 5-26: Responsive design of web interface.

5.2.3 Roles and permissions

The single files located in the above-mentioned folder structures contain the actual source code for all transactions executed within the Blockchain network written in GO language. The source code underlies the functionalities executed in the GUI as distinct operations or their concatenation.

For recording of votes, the variables `ArtefactId`, `VoteInput`, and `RejectReason` are created. Recording is only possible if these three variables are included in a transaction. Moreover, for the decision of which status the artifact will have when being recorded, the previous state has to be retrieved from the ledger beforehand. This is depicted in Source Code 5-1.

```

262 // Records a vote on the Artefact and decides the subsequent state
263 func (t *Artefact) vote(stub shim.ChaincodeStubInterface, args []string) pb.Response {
264     fmt.Println("##### artefact_cc vote #####")
265
266     var ArtefactId string // Identifier
267     var VoteInput string // Vote Input
268     var RejectReason string // Rejection Reason
269     var err error
270
271     if len(args) != 4 {
272         return shim.Error("Incorrect number of arguments. Expecting artefact id and the vote")
273     }
274
275     ArtefactId = args[1]
276     VoteInput = args[2]
277     RejectReason = args[3]
278
279     // Get the artefact state from the ledger
280     Avalbytes, err := stub.GetState(ArtefactId)
281     if err != nil {
282         jsonResp := "{\"Error\":\"Failed to read the artefact for id : " + ArtefactId + "\"}"
283         return shim.Error(jsonResp)
284     }
285
286     if Avalbytes == nil {
287         jsonResp := "{\"Error\":\"Artefact could not be found for id : " + ArtefactId + "\"}"
288         return shim.Error(jsonResp)
289     }
290
291     ArtefactJson := string(Avalbytes)
292     fmt.Println("ArtefactJson : ", ArtefactJson)

```

Source Code 5-1: Recording of votes.

In a scenario of a big company, it is reasonable that many engineers document their latest development information artifacts within their IT tools and systems. Therefore, the prototypical implementation of the Blockchain network contains different users per organization, i.e., each engineering partner has multiple users who can document their progress separately in the Blockchain. There are three hard-coded roles of organizations authorized to create artifacts and participate in voting: i) `customer`; ii)

supplier, iii) oem. Additionally, it is distinguished between the organization, i.e., the engineering partner, and different roles within this organization¹³¹. This is depicted in Source Code 5-2.

```
26 var AUTHORIZED_CREATORS = [3]string{"customer", "supplier", "oem"}
27 var AUTHORIZED_VOTERS = [3]string{"customer", "supplier", "oem"}
28
29 type PermittedVoter struct {
30     Org      string `json:"org"`
31     OrgRole  string `json:"orgUnit"`
32 }
33
34 type PermittedVoters []PermittedVoter
```

Source Code 5-2: Authorized roles.

Source Code 5-3 shows the initialization of configuration state upon the initial deployment of the Blockchain network. Permitted voters can vote as many as want to, even from the same organization (cf. Source Code 5-4). This might be the case if an engineer from another department within the same company also wants to give their statement.

The *Hyperledger Fabric* platform allows the adding of new organizations, such as a new engineering partner, to the already existing and running Blockchain network. Hereby, the organizations' peers can be added to the relevant channels where they want to contribute to the transactions (PEREPA and YELICK, 2017; HYPERLEDGER, 2020), i.e., information artifacts to development activities. This addresses requirement 15 where the ad hoc inclusion of engineering partners and hence their legacy IT systems is required.

¹³¹ For details regarding the implemented roles, please see Chapter 5.2.1.

```

65 // Init initializes the configuration state
66 func (t *Artefact) Init(stub shim.ChaincodeStubInterface) pb.Response {
67     fmt.Println("##### artefact_cc Init #####")
68
69     _, args := stub.GetFunctionAndParameters()
70
71     var ConfigJson string
72     var err error
73
74     if len(args) != 1 {
75         return shim.Error("Incorrect number of arguments. Expecting Permitted Voters Configuration json : " + string(len(args)))
76     }
77
78     ConfigJson = args[0]
79     fmt.Printf("ConfigJson = %d\n", ConfigJson)
80
81     permittedVoters := PermittedVoters{}
82     err = json.Unmarshal([]byte(ConfigJson), &permittedVoters)
83
84     if err != nil {
85         fmt.Println("error:", err)
86         jsonResp := "{\"Error\":\"Failed to unmarshal the configuration, permitted voters json could not parsed \"}"
87         return shim.Error(jsonResp)
88     }
89
90     // Write the config state to the ledger
91     err = stub.PutState(CONFIG_ID_PERMITTED_VOTERS, []byte(ConfigJson))
92     if err != nil {
93         return shim.Error(err.Error())
94     }
95
96     // Default transient map usage
97     if transientMap, err := stub.GetTransient(); err == nil {
98         if transientData, ok := transientMap["result"]; ok {
99             return shim.Success(transientData)
100         }
101     }
102     return shim.Success(nil)
103 }

```

Source Code 5-3: Initialization of configuration state.

```

342 // read the list of permitted voters
343 ConfigJson, err := stub.GetState(CONFIG_ID_PERMITTED_VOTERS)
344 if err != nil || ConfigJson == nil {
345     jsonResp := "{\"Error\":\"Failed to read the permitted voters configuration (initialization error).\"}"
346     return shim.Error(jsonResp)
347 }
348
349 permittedVoters := PermittedVoters{}
350 err = json.Unmarshal([]byte(ConfigJson), &permittedVoters)
351
352 if err != nil {
353     fmt.Println("error:", err)
354     jsonResp := "{\"Error\":\"Failed to unmarshal the initial configuration, permitted voters json could not parsed \"}"
355     return shim.Error(jsonResp)
356 }
357
358 // count approve votes
359 var approveCount int = 0
360 for _, voter := range permittedVoters {
361     for _, voteEl := range artefact.Votes {
362         if voter.Org == voteEl.VoterOrg && strings.EqualFold(ACCEPTED_VOTE, voteEl.Vote) {
363             approveCount++
364         }
365     }
366 }

```

Source Code 5-4: Permitted voters.

5.3 Implementation of process model

The Blockchain network requires a dedicated sequence of actions in order to ensure its advantages for traceability in distributed environments. In addition, the *Hyperledger Fabric* framework implements certain workflows given its IT architecture and distribution of data among multiple peers. In Figure 5-27, the implemented process model is depicted for an exemplary user of the supplier. In step 1, the user initiates the creation of a new information artifact by providing the artifact's attributes and documenting them

within the web app. Subsequently in step 2a, the “Thin-web-app”, which is a *Fabric* client, creates an endorsement proposal for the created artifact and sends it to its endorsing peers within its own organization. After receiving the endorsement, the endorsing peers simulate the transaction on the current ledger state and sign the endorsed transaction. Then in step 2b, all transaction endorsed messages will be collected into a valid endorsement which satisfies the endorsement policy. The synchronization of the valid endorsement will be transmitted from the supplier’s network to the OEM’s network via the common ordering service in step 3. In doing so, the transaction endorsement is broadcasted to all committing and endorsing peers. Step 4 depicts the reception of the endorsement. For this, the endorsement will be verified and set to “read” on the ledger. If the verification is positive, it will be set to “write” (cf. BASHIR, 2018: p. 481).

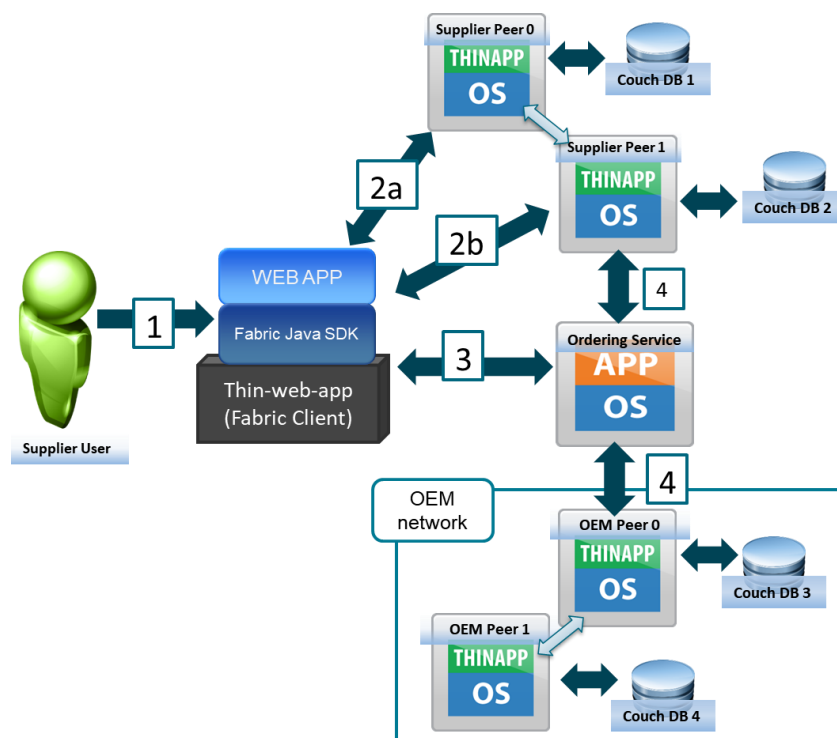


Figure 5-27: Process model of the Hyperledger Fabric framework.

The basic operations for a process model are depicted in Source Code 5-5. Functions are called via `if` arguments. The `create` function creates a new artifact on the ledger and voting for an artifact can be done with the `vote` function. The retrieval of an artifact can be executed using the `query` function. The connected *CouchDB*, storing the world state, can be searched for a specific artifact with the `search` function. Using `artledger`, the entire history for one artifact will be returned.

```

109     function, args := stub.GetFunctionAndParameters()
110
111     if function != "invoke" {
112         return shim.Error("Unknown function call")
113     }
114
115     if len(args) < 2 {
116         return shim.Error("Incorrect number of arguments. Expecting at least 2")
117     }
118
119     if args[0] == "create" {
120         // creates the Artefact
121         return t.create(stub, args)
122     }
123
124     if args[0] == "vote" {
125         // Records a vote on the Artefact
126         return t.vote(stub, args)
127     }
128
129     if args[0] == "query" {
130         // query an entity state Artefact
131         return t.query(stub, args)
132     }
133
134     if args[0] == "search" {
135         // search the artefacts
136         return t.search(stub, args)
137     }
138
139     if args[0] == "artledger" {
140         // artefacts transactions
141         return t.getHistoryForArtefact(stub, args)
142     }

```

Source Code 5-5: Basic operations.

The `create` artifact function uses all artifact attributes (cf. Chapter 5.4) as an input and creates an ID which is depicted in Source Code 5-6.

```

186 // Creates the Artefact state ledger. Throws error if artefact already exists.
187 func (t *Artefact) create(stub shim.ChaincodeStubInterface, args []string) pb.Response {
188     fmt.Println("##### artefact_cc create #####")
189
190     var ArtefactId string // Identifier
191     var ArtefactJson string // JSON representation
192     var err error
193
194     if len(args) != 3 {
195         return shim.Error("Incorrect number of arguments. Expecting artefact id and details as json string")
196     }
197
198     ArtefactId = args[1]
199     ArtefactJson = args[2]
200
201     // Get the artefact state from the ledger
202     Avalbytes, err := stub.GetState(ArtefactId)
203     if err != nil {
204         jsonResp := "{\"Error\":\"Failed to check if artefact already exists for id : " + ArtefactId + "\"}"
205         return shim.Error(jsonResp)
206     }

```

Source Code 5-6: Create artifact operation.

During the consensus mechanism, information artifacts have a specific status which indicates to the engineer in the GUI whether voting is still in progress, who accepted or declined, and what the overall combined status is. This is depicted in Source Code 5-7.

```
17 ▼ const (
18     CONFIG_ID_PERMITTED_VOTERS = "PV01"
19     ACCEPTED_VOTE               = "Accepted"
20     NOT_ACCEPTED_VOTE           = "Not Accepted"
21     STATUS_INPROGRESS           = "In Progress"
22     STATUS_ACCEPTED             = "Accepted"
23     STATUS_NOT_ACCEPTED         = "Not Accepted"
24 )
```

Source Code 5-7: Artifact states.

The initial status after creation of the information artifact is `In Progress` (cf. Source Code 5-8). It is `Accepted` in the case of *all* permitted voters have approved the new artifact (cf. Source Code 5-9). If *any* permitted voter rejected the newly created artifact, the state is `Not Accepted` (cf. Source Code 5-10). The latter logic is helpful to identify any inconsistencies in the development process.

```
220 // Set the creator name
221 var Org string
222 var OrgRole string
223 artefact.ReleaseStatus = STATUS_INPROGRESS
224 artefact.CreatedBy, Org, OrgRole, _ = t.getCreatorName(stub)
225 fmt.Println("Org Role :", OrgRole)
```

Source Code 5-8: Vote status "in progress".

```
367 // check if sufficient votes have been cast
368 if approveCount == len(permittedVoters) {
369     artefact.ReleaseStatus = STATUS_ACCEPTED
370 } else {
371     fmt.Println("Approve Vote added, but count insufficient for finalization")
372 }
```

Source Code 5-9: Vote status "accepted".

```
337 // process new vote, compute the new state
338 if strings.EqualFold(NOT_ACCEPTED_VOTE, VoteInput) {
339     // Set the artefact status as rejected since the vote is to reject
340     artefact.ReleaseStatus = STATUS_NOT_ACCEPTED
341 } else if strings.EqualFold(ACCEPTED_VOTE, VoteInput) {
```

Source Code 5-10: Vote status "not accepted".

Chapter 4.1, such as `EE_system`, are integrated as attributes within the artifact structure in JSON format, as depicted in Source Code 5-11. This data model also serves as a fundament for the RDF namespaces of each engineering partner as it contains all relevant information artifacts as well as their relationships among each other. Due to this data model is implemented in the Blockchain itself, it is multiplied for each newly joining peer and enables traceability with legacy IT systems as well as within the Blockchain network (cf. Figure 4-36).

```

1 // Artefact structure to marshal/unmarshal json string
2 type Artefact struct {
3     Uuid                string `json:"uuid"`
4     SachNumber          string `json:"sachNumber"`
5     ExternalPartNo      string `json:"externalPartNo"`
6     ArtefactType        string `json:"artefactType"`
7     Description         string `json:"description"`
8     Version             string `json:"version"`
9     Issuer              string `json:"issuer"`
10    Timestamp           string `json:"timestamp"`
11    Architecture        string `json:"architecture"`
12    ReleaseStatus       string `json:"releaseStatus"`
13    CreatedBy           string `json:"createdBy,omitEmpty"`
14    LastUpdatedBy       string `json:"lastUpdatedBy,omitEmpty"`
15    Votes               []Vote `json:"votes,omitEmpty"`
16    ArtefactCategory    string `json:"artefactCategory"`
17    DescriptionGerman    string `json:"descriptionGerman,omitEmpty"`
18    Flashable           string `json:"flashable,omitEmpty"`
19    SignaturKennzeichen string `json:"signaturKennzeichen,omitEmpty"`
20    Link_HTTPS          string `json:"link"`
21    EE_System           string `json:"eeSystem"`
22    ECU                 string `json:"ecu"`
23    Hardware            string `json:"hardware"`
24    Geometry            string `json:"geometry"`
25    Memory              string `json:"memory"`
26    Processor           string `json:"processor"`
27    Plug                string `json:"plug"`
28    Pin                 string `json:"pin"`
29    NCD                 string `json:"ncd"`
30    IO_channel          string `json:"ioChannel"`
31    Signal              string `json:"signal"`
32    Software            string `json:"software"`
33    Bootloader_SW       string `json:"bootloaderSW"`
34    Functional_SW        string `json:"functionalSW"`
35    Parametric_SW       string `json:"parametricSW"`
36    Communication_Bus   string `json:"communicationBus"`
37    SW_Function          string `json:"SWfunction"`
38    Configuration        string `json:"configuration"`
39    Variant_point       string `json:"variantPoint"`
40    Variant             string `json:"variant"`
41    Model_series        string `json:"modelSeries"`
42    Release_model        string `json:"releaseModel"`
43    Engineering_change   string `json:"engineeringChange"`
44    Deployment_date     string `json:"deploymentDate"`
45    Requirements_view    string `json:"requirementsView"`
46    Functional_view      string `json:"functionalView"`
47    Logical_view         string `json:"logicalView"`
48    Technical_view      string `json:"technicalView"`
49    IT_system           string `json:"itSystem"`
50    MBSE_tool           string `json:"mbseTool"`
51    PDM_system          string `json:"pdmSystem"`
52 }

```

Source Code 5-11: Generic information artifact structure.

METAMODEL

The metadata model includes information on the definition of information artifacts, their attributes and stats, as well as interface objects. It has been implemented using RDF and OWL. Source Code 5-12 depicts the generic metamodel structure.

```

1  @prefix : <http://www.daimler.engineering-mbc.com/mra2/datamodel-metadata/v1#> .
2  @prefix owl: <http://www.w3.org/2002/07/owl#> .
3  @prefix pdm: <http://www.daimler.engineering-mbc.com/mra2/datamodel/v1#> .
4  @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
5  @prefix xml: <http://www.w3.org/XML/1998/namespace> .
6  @prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
7  @prefix foaf: <http://xmlns.com/foaf/0.1/> .
8  @prefix pdma: <http://www.daimler.engineering-mbc.com/mra2/datamodel-metadata/v1#> .
9  @prefix pdmm: <http://www.daimler.engineering-mbc.com/mra2/ontology/v1#> .
10 @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
11 @prefix dcterms: <http://purl.org/dc/terms/> .
12 @base <http://www.daimler.engineering-mbc.com/mra2/datamodel-metadata/v1#> .
13
14 <http://www.daimler.engineering-mbc.com/mra2/datamodel-metadata/v1#> rdf:type owl:Ontology ;
15   owl:imports pdmm: ;
16   rdfs:label "Daimler engineering-mbc MRA2 Data Model Meta Data"@en ;
17   dcterms:created "2018-04-15"^^xsd:dateTime ;
18   owl:versionInfo "0.1.0"^^xsd:string .
19
20 ### http://xmlns.com/foaf/0.1/familyName
21 foaf:familyName rdf:type owl:AnnotationProperty .
22
23 ### http://xmlns.com/foaf/0.1/firstName
24 foaf:firstName rdf:type owl:AnnotationProperty .
25
26 #####
27 # META DATA
28
29 ### http://www.daimler.engineering-mbc.com/mra2/datamodel-metadata/v1#InterfaceObject
30 pdma:InterfaceObject rdf:type owl:NamedIndividual ,
31   pdmm:BackendSystemInterfaceObject ;
32   pdmm:containsElement pdma:csArtifact_Description ,
33   pdma:csArtifact_Number ,
34   pdma:csArtifact_Project ,
35   pdma:csArtifact_State ,
36   pdma:csPart_ChangeNumber ,
37   pdma:csPart_DeepLink ,
38   pdma:csPart_IsAssembly ,
39   pdma:csPart_IsLatestVersion ,
40   pdma:csPart_Version ,
41   pdma:csPart_Weight ,
42   pdma:csPart_ZGS .
43
44 ### http://www.daimler.engineering-mbc.com/mra2/datamodel-metadata/v1#InterfaceParam_PartNumber
45 pdma:InterfaceParam_PartNumber rdf:type owl:NamedIndividual ,
46   pdmm:BackendSystemInterfaceParameter ;
47   pdmm:hasParameterName "partNumber"^^xsd:string ;
48   pdmm:hasParameterType "httpPathParam"^^xsd:string ;
49   pdmm:hasParameterizationType "term"^^xsd:string ;
50   pdmm:isParameterFor pdm:partNumber ;
51   rdfs:label "Interface Parameter for the partNumber"@en .
52
53 ### http://www.daimler.engineering-mbc.com/mra2/datamodel-metadata/v1#Artifact_Project
54 pdma:Artifact_Project rdf:type owl:NamedIndividual ,
55   pdmm:BackendSystemInterfaceField ;
56   pdmm:hasElementName "Artifact_Project"^^xsd:string ;
57   pdmm:hasFieldType xsd:string ;
58   pdmm:isDataSourceFor pdm:projectName .
59
60 ### http://www.daimler.engineering-mbc.com/mra2/datamodel-metadata/v1#Artifact_State
61 pdma:Artifact_State rdf:type owl:NamedIndividual ,
62   pdmm:BackendSystemInterfaceField ;
63   pdmm:hasElementName "Artifact_State"^^xsd:string ;
64   pdmm:hasFieldType xsd:string ;
65   pdmm:isDataSourceFor pdm:partState .
66
67 ### Generated by the OWL API (version 4.2.8.20170104-2310) https://github.com/owlcs/owlapi

```

Source Code 5-12: Generic metamodel structure.

ONTOLOGY

The ontology defines the object and data properties as well as classes. It has also been implemented with RDF and OWL. The generic description of the ontology is depicted in Source Code 5-13 where all relevant taxonomies are preloaded and annotations are defined. Source Code 5-14 depicts the generic object properties for, e.g., interface elements, business objects, or an email address.

```

1  @prefix : <http://www.daimler.engineering-mbc.com/mra2/ontology/v1#> .
2  @prefix org: <http://www.w3.org/ns/org#> .
3  @prefix owl: <http://www.w3.org/2002/07/owl#> .
4  @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
5  @prefix xml: <http://www.w3.org/XML/1998/namespace> .
6  @prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
7  @prefix foaf: <http://xmlns.com/foaf/0.1/> .
8  @prefix pdmm: <http://www.daimler.engineering-mbc.com/mra2/ontology/v1#> .
9  @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
10 @prefix dcterms: <http://purl.org/dc/terms/> .
11 @base <http://www.daimler.engineering-mbc.com/mra2/ontology/v1#> .
12
13 <http://www.daimler.engineering-mbc.com/mra2/ontology/v1#> rdf:type owl:Ontology ;
14   rdfs:comment "The ontology defines classes and properties
15   used to describe meta data." ;
16   rdfs:label "Daimler Data Model Ontology"@en ;
17   owl:versionInfo "0.1.0"^^xsd:string ;
18   dcterms:created "2018-05-02"^^xsd:dateTime .
19
20 #####
21 #   Annotation properties
22
23 ### http://www.daimler.engineering-mbc.com/mra2/ontology/v1#hasAbbreviation
24 pdmm:hasAbbreviation rdf:type owl:AnnotationProperty ;
25   rdfs:label "Abkürzung"@de ,
26   "abbreviation"@en ;
27   rdfs:range xsd:string ;
28   rdfs:domain rdfs:Class .
29
30 ### http://www.daimler.engineering-mbc.com/mra2/ontology/v1#longDescription
31 pdmm:longDescription rdf:type owl:AnnotationProperty ;
32   rdfs:label "lange Beschreibung"@de ,
33   "long description"@en ;
34   rdfs:subPropertyOf rdfs:comment ;
35   rdfs:range rdf:PlainLiteral .
36
37 ### http://www.daimler.engineering-mbc.com/mra2/ontology/v1#shortDescription
38 pdmm:shortDescription rdf:type owl:AnnotationProperty ;
39   rdfs:label "kurze Beschreibung"@de ,
40   "short description"@en ;
41   rdfs:subPropertyOf rdfs:comment ;
42   rdfs:range rdf:PlainLiteral .

```

Source Code 5-13: Generic ontology description.

Data properties, such as an element name, interface link URI, or parameter name, are depicted in Source Code 5-15. Different classes are implemented for the categorization of backend system interface elements which helps alignment of data. This is depicted in Source Code 5-16.

```

44 #####
45 #   Object Properties
46
47 ### http://www.daimler.engineering-mbc.com/mra2/ontology/v1#containsElement
48 pdmm:containsElement rdf:type owl:ObjectProperty ;
49                       rdfs:domain pdmm:BackendSystemInterfaceObject ;
50                       rdfs:range pdmm:BackendSystemInterfaceElement ;
51                       rdfs:label "contains element"@en ,
52                                "enthält Element"@de .
53
54 ### http://www.daimler.engineering-mbc.com/mra2/ontology/v1#hasBackendComponentContact
55 pdmm:hasBackendComponentContact rdf:type owl:ObjectProperty ;
56                                rdfs:domain pdmm:BackendSystemComponent ;
57                                rdfs:range pdmm:Person ;
58                                rdfs:label "has backend component contact"@en ,
59                                           "hat Kontakt für Backendkomponente"@de .
60
61 ### http://www.daimler.engineering-mbc.com/mra2/ontology/v1#hasBackendInterface
62 pdmm:hasBackendInterface rdf:type owl:ObjectProperty ;
63                           rdfs:domain pdmm:BackendSystem ;
64                           rdfs:range pdmm:BackendSystemInterface ;
65                           rdfs:label "Backendschnittstelle"@de ,
66                                    "backend interface"@en .
67
68 ### http://www.daimler.engineering-mbc.com/mra2/ontology/v1#hasBusinessControlOf
69 pdmm:hasBusinessControlOf rdf:type owl:ObjectProperty ;
70                           rdfs:subPropertyOf pdmm:isInterestedIn ;
71                           rdfs:label "geschäftliche Kontrolle"@de ,
72                                    "has business control of"@en .
73
74 ### http://www.daimler.engineering-mbc.com/mra2/ontology/v1#hasElementListType
75 pdmm:hasElementListType rdf:type owl:ObjectProperty ;
76                           rdfs:domain pdmm:BackendSystemInterfaceElementList ;
77                           rdfs:range pdmm:BackendSystemInterfaceElement ;
78                           rdfs:label "Typ der Elementliste"@de ,
79                                    "has element list type"@en .
80
81 ### http://www.daimler.engineering-mbc.com/mra2/ontology/v1#hasEmailAddress
82 pdmm:hasEmailAddress rdf:type owl:ObjectProperty ;
83                       rdfs:subPropertyOf foaf:mbox ;
84                       rdfs:domain pdmm:Person ;
85                       rdfs:label "E-Mail Adresse"@de ,
86                                "email address"@en .

```

Source Code 5-14: Generic object properties of the ontology.

```

155 #####
156 #   Data properties
157
158 ### http://www.daimler.engineering-mbc.com/mra2/ontology/v1#hasDataExchangeProtocol
159 pdmm:hasDataExchangeProtocol rdf:type owl:DatatypeProperty ;
160     rdfs:domain pdmm:BackendSystemInterface ;
161     rdfs:range xsd:string ;
162     rdfs:comment ""The data exchange protocol of the interface refers to the transport
163     protocol. This can be \"http\" or \"soap\" or \"sql\", etc.""@en ;
164     rdfs:label "Datenaustauschformat"@de ,
165     "data exchange protocol"@en .
166
167 ### http://www.daimler.engineering-mbc.com/mra2/ontology/v1#hasElementName
168 pdmm:hasElementName rdf:type owl:DatatypeProperty ;
169     rdfs:domain pdmm:BackendSystemInterfaceElement ;
170     rdfs:range xsd:string ;
171     rdfs:comment "The name of the element in the backend interface in XML or a json."@en ;
172     rdfs:label "Elementname"@de ,
173     "element name"@en .
174
175 ### http://www.daimler.engineering-mbc.com/mra2/ontology/v1#hasInterfaceLink
176 pdmm:hasInterfaceLink rdf:type owl:DatatypeProperty ;
177     rdfs:domain pdmm:BackendSystemInterface ;
178     rdfs:range xsd:string ;
179     rdfs:comment ""The interface URI""@en ;
180     rdfs:label "Schnittstellenadresse"@de ,
181     "interface link"@en .
182
183 ### http://www.daimler.engineering-mbc.com/mra2/ontology/v1#hasInterfaceResultFormat
184 pdmm:hasInterfaceResultFormat rdf:type owl:DatatypeProperty ;
185     rdfs:domain pdmm:BackendSystemInterface ;
186     rdfs:range xsd:string ;
187     rdfs:comment "The result mime-type the interface uses."@en ;
188     rdfs:label "Schnittstellenergebnisformat"@de ,
189     "interface result format"@en .
190
191 ### http://www.daimler.engineering-mbc.com/mra2/ontology/v1#hasParameterName
192 pdmm:hasParameterName rdf:type owl:DatatypeProperty ;
193     rdfs:domain pdmm:BackendSystemInterfaceParameter ;
194     rdfs:range xsd:string ;
195     rdfs:label "Parametername"@de ,
196     "parameter name"@en .

```

Source Code 5-15: Generic data properties of the ontology.

```

229 #####
230 #   Classes
231
232 ### http://www.daimler.engineering-mbc.com/mra2/ontology/v1#BackendSystem
233 pdmm:BackendSystem rdf:type owl:Class ;
234                      rdfs:subClassOf pdmm:BackendSystemComponent ;
235                      rdfs:label "Backendsystem"@de ,
236                      "backend system"@en .
237
238 ### http://www.daimler.engineering-mbc.com/mra2/ontology/v1#BackendSystemComponent
239 pdmm:BackendSystemComponent rdf:type owl:Class ;
240                             rdfs:label "Backendsystemkomponente"@de ,
241                             "backend system component"@en .
242
243 ### http://www.daimler.engineering-mbc.com/mra2/ontology/v1#BackendSystemInterface
244 pdmm:BackendSystemInterface rdf:type owl:Class ;
245                             rdfs:subClassOf pdmm:BackendSystemComponent ;
246                             rdfs:label "Backendsystemschnittstelle"@de ,
247                             "backend system interface"@en .
248
249 ### http://www.daimler.engineering-mbc.com/mra2/ontology/v1#BackendSystemInterfaceElement
250 pdmm:BackendSystemInterfaceElement rdf:type owl:Class ;
251                                   rdfs:label "Backendsystemschnittstellenelement"@de ,
252                                   "backend system interface element"@en .
253
254 ### http://www.daimler.engineering-mbc.com/mra2/ontology/v1#BackendSystemInterfaceElementList
255 pdmm:BackendSystemInterfaceElementList rdf:type owl:Class ;
256                                       rdfs:subClassOf pdmm:BackendSystemInterfaceElement ;
257                                       rdfs:label "Backendsystemschnittstellenelementliste"@de ,
258                                       "backend system interface element list"@en .
259
260 ### http://www.daimler.engineering-mbc.com/mra2/ontology/v1#BackendSystemInterfaceField
261 pdmm:BackendSystemInterfaceField rdf:type owl:Class ;
262                                  rdfs:subClassOf pdmm:BackendSystemInterfaceElement ;
263                                  rdfs:label "Backendsystemschnittstellenfeld"@de ,
264                                  "backend system interface field"@en .
265
266 ### http://www.daimler.engineering-mbc.com/mra2/ontology/v1#BackendSystemInterfaceObject
267 pdmm:BackendSystemInterfaceObject rdf:type owl:Class ;
268                                  rdfs:subClassOf pdmm:BackendSystemInterfaceElement ;
269                                  rdfs:label "Backendsystemschnittstellenobjekt"@de ,
270                                  "backend system interface object"@en .

```

Source Code 5-16: Generic classes of the ontology.

INFORMATION ARTIFACTS

The relations and information artifacts depicted in Figure 5-28 are transferred, again in OWL and RDF, to the implemented data model. In that, relations are described by bundles and the association with other domains, as it is depicted generically in Source Code 5-17.

Additional information is provided by data properties, such as the type version of an automobile, the model series, an aggregated process status, or the current state. This is depicted in Source Code 5-18.

```
52  ### http://www.daimler.engineering-mbc.com/mra2/datamodel/v1#bundleContainsPart
53  pdm:bundleContainsPart rdf:type owl:ObjectProperty ;
54                          rdfs:domain pdm:Bundle ;
55                          rdfs:range pdm:PartVersion ;
56                          rdfs:label "Bauteileliste"@de ,
57                                  "Bundle Contains Part"@en .
58
59  ### http://www.daimler.engineering-mbc.com/mra2/datamodel/v1#bundleDeploymentDate
60  pdm:bundleDeploymentDate rdf:type owl:ObjectProperty ;
61                          rdfs:domain pdm:Bundle ;
62                          rdfs:range pdm:DeploymentDate ;
63                          rdfs:label "Deployment Date"@en ,
64                                  "Planeinsatztermin"@de .
65
66  ### http://www.daimler.engineering-mbc.com/mra2/datamodel/v1#bundleModuleComponent
67  pdm:bundleModuleComponent rdf:type owl:ObjectProperty ;
68                          rdfs:domain pdm:Bundle ;
69                          rdfs:range pdm:ModuleComponent ;
70                          rdfs:label "Module Component"@en ,
71                                  "Modulkomponente"@de .
72
73  ### http://www.daimler.engineering-mbc.com/mra2/datamodel/v1#constructionMainGroup
74  pdm:constructionMainGroup rdf:type owl:ObjectProperty ;
75                          rdfs:domain pdm:Part ;
76                          rdfs:range pdm:ConstructionGroup ;
77                          rdfs:label "Construction Main Group"@en ,
78                                  "Konstruktionshauptgruppe"@de .
79
80  ### http://www.daimler.engineering-mbc.com/mra2/datamodel/v1#constructionSubGroup
81  pdm:constructionSubGroup rdf:type owl:ObjectProperty ;
82                          rdfs:domain pdm:Part ;
83                          rdfs:range pdm:ConstructionGroup ;
84                          rdfs:label "Construction Sub Group"@en ,
85                                  "Konstruktionsuntergruppe"@de .
86
87  ### http://www.daimler.engineering-mbc.com/mra2/datamodel/v1#containsBundle
88  pdm:containsBundle rdf:type owl:ObjectProperty ;
89                  rdfs:domain pdm:EngineeringChange ;
90                  rdfs:range pdm:Bundle ;
91                  rdfs:label "Contained Bundles"@en ,
92                          "enthaltene Bündel"@de .
```

Source Code 5-17: Generic object properties of the information artifacts.


```

272 #####
273 #   Data properties
274
275 ### http://www.daimler.engineering-mbc.com/mra2/datamodel/v1#AA
276 ▼ pdm:AA rdfs:type owl:DatatypeProperty ;
277     rdfs:domain pdm:DeploymentDate ;
278     rdfs:range xsd:string ;
279     pdmm:longDescription "Ausführungsart (Karosserieform, Motor-Hubraum)."@de ,
280                          "Type version (body variant, engine displacement)."@en ;
281     pdmm:shortDescription "Ausführungsart"@de ,
282                          "Type version"@en ;
283     rdfs:label "AA"@de ,
284               "AA"@en .
285
286 ### http://www.daimler.engineering-mbc.com/mra2/datamodel/v1#BR
287 ▼ pdm:BR rdfs:type owl:DatatypeProperty ;
288     rdfs:domain pdm:DeploymentDate ;
289     rdfs:range xsd:string ;
290     pdmm:longDescription "Fahrzeug-Baureihe oder Aggregat-Baureihe."@de ,
291                          "Model series, e.g. a car or an aggregate."@en ;
292     pdmm:shortDescription "Baureihe"@de ,
293                          "Model series"@en ;
294     rdfs:label "BR"@de ,
295               "BR"@en .
296
297 ### http://www.daimler.engineering-mbc.com/mra2/datamodel/v1#aggregatedProcessStatus
298 ▼ pdm:aggregatedProcessStatus rdfs:type owl:DatatypeProperty ;
299     rdfs:domain pdm:EngineeringChange ;
300     rdfs:range xsd:string ;
301     rdfs:label "Aggregated process status"@en ,
302               "Aggregierter Prozessstatus"@de .
303
304 ### http://www.daimler.engineering-mbc.com/mra2/datamodel/v1#bundleCurrentState
305 ▼ pdm:bundleCurrentState rdfs:type owl:DatatypeProperty ;
306     rdfs:domain pdm:Bundle ;
307     rdfs:range xsd:string ;
308     rdfs:label "Current State"@en ,
309               "Istzustand"@de .

```

Source Code 5-18: Generic data properties of the information artifacts.

5.5 Alignment with legacy IT architecture

The *Hyperledger Fabric* platform is integrated in the existing IT architecture landscape of each engineering partner as the PLM Blockchain backbone, as presented above. Hereby, each node of each engineering partner represents the PLM Blockchain backbone for their own internal IT architecture. This is depicted in Figure 5-29¹³³.

Different REST APIs enable the quick and easy connection of the in the *Hyperledger Fabric* platform to legacy systems (BASHIR, 2018: p. 476) and hence is in alignment with requirement 15 for standardized APIs to alleviate the connection of engineering partners as well as legacy IT systems.

¹³³ The fourth layer for ERP etc., as depicted in Figure 2-12, was omitted due to it not being in the scope of this work.

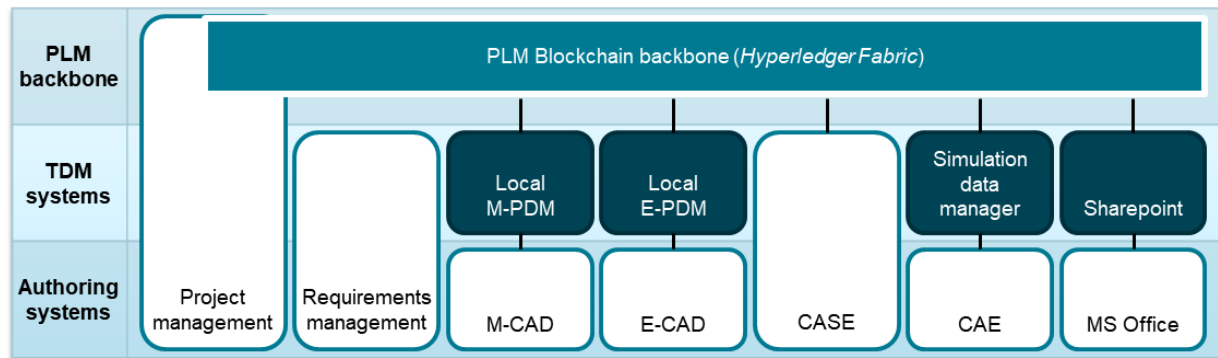


Figure 5-29: Generic positioning of the prototypical implementation of a PLM Blockchain backbone within a legacy IT architecture. For abbreviations, please refer to the description of Figure 2-12.

As presented in Chapter 5.2.2, the GUI is web-based and allows for IT system-independent documentation of relevant metadata. Due to it could not have been implemented directly into an existing IT system architecture, based on limitations in time and a huge complexity (cf. Chapter 5.2), the interims step of separate documentation of metadata in the web browser has to be implemented. This implies the conjecture that, after the engineer modeled objects according the SPES method in the MBSE tool, would have to switch tools and document the relevant metadata manually in the web browser. This would, in turn, trigger an update of RDF namespaces for MBSE within the Blockchain framework (cf. Chapter 4.2).

The implementation scenario for a first step including manual documentation within the web browser presented above would consist of stand-alone IT tools and systems. This means that after modeling MBSE SysML models, the manual documentation has to be started by also manually opening the Blockchain framework's GUI. Afterwards, the engineering collaboration partner also has to open their web browser in order to access the changes made by the counterpart. In the next step, these alterations have to be documented manually again in the respective MBSE tool. The updating of the RDF namespaces after confirmation will be triggered automatically within the Blockchain framework. In a further development stage, there might be a button within the MBSE tool directly linking the Blockchain framework's web browser GUI via OSLC where the MBSE tool serves as consumer and the Blockchain framework as provider for the delegated UI (cf. Chapter 2.4.3). This would already increase usability slightly. The rest of the steps would still be executed manually.

In the ultimate development stage, metadata will be exchanged automatically between the MBSE tools and the *Hyperledger Fabric* platform to allow for maximum efficiency and minimum error-proneness.

6 Evaluation of the solution framework

This chapter conducts the evaluation of the solution framework according to the four objectives of this research. For this purpose, in Chapter 6.1 the evaluation approach will be presented. The following Chapters 6.2 and 6.3 describe use cases for the evaluation of different aspects of the framework. Chapter 6.4 summarizes the overall evaluation of the solution framework based on the research objectives and Chapter 6.5 discusses the results as well as ramifications.

6.1 Evaluation approach

The terms validation and verification are used differently in engineering and literature (SEEPERSAD et al., 2006: p. 303). Here, verification is defined as the evaluation that specifications have been fulfilled, i.e., internal view and consistency. Validation is defined as the evaluation of if the product fulfills the intended business use required by the customer or stakeholder, i.e., external view and consistency¹³⁴. In this work, the focus will mainly lie on the verification of the defined framework according to the deduced requirements. Hence, the internal view will be applied, and internal consistency will be evaluated accordingly. Additionally, validation was done within the project with the three engineering partners with whom the prototype was developed. Given the complexity of the defined solution framework, the empirical validation with end-users was not in scope. This is often the case due to the difficulty in the assessment of methodologies and approaches. The design process, also in automotive industry, is often longer than a research project and hence the effects might not be visible directly¹³⁵ (BLESSING and CHAKRABARTI, 2009: p. 183). Moreover, “the synthetic nature of design is incompatible with the controlled experiments useful for theory testing” (GAVER, 2012: p. 940). For these reasons, a qualitative evaluation approach will be chosen.

In alignment with BLESSING and CHAKRABARTI (2009), the *application evaluation* targets on the assessment of the desired values with respect to the applicability of the support for this dedicated task. Usability and applicability are considered to be focal (BLESSING and CHAKRABARTI, 2009: p. 37). Here in this context, the desired values resemble the

¹³⁴ Also, please refer to Footnote 48 on p. 54 for the distinction of verification and validation and the respective sources as well as INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS (1983: p. 37).

¹³⁵ Please also refer to DEUTSCHES INSTITUT FÜR NORMUNG E. V. (2018b, 2018a, 2018c) for an assessment of specific criteria of the openness of IT products in the realm of PLM which is deemed to be too generic for the evaluation of this work.

requirements, and the support is equal to the solution framework¹³⁶. It shall be clarified whether:

1. the framework can be used,
2. the framework does indeed address the requirements it is meant to,
3. the requirements are affected as expected (BLESSING and CHAKRABARTI, 2009: p. 184).

Commonly, the actual support of the implemented, synthesized solution framework does not address all requirements of the intended support (BLESSING and CHAKRABARTI, 2009: p. 183). Therefore, different use cases will be described in the following chapters which highlight different aspects of the framework. The use cases have been chosen as generically as possible to alleviate the transfer to other automotive engineering collaborations as well as to engineering collaborations in other industries. Use case 1, the door control module (DCM), evaluates a very common ECU and hence allows for a transfer to all automotive OEMs and their engineering partners and, partially, to other industries where such ECUs are developed jointly. This use case was developed, implemented, and tested with three major companies in the automotive sector. Use case 2, the centralized ECU, is an approach the entire automotive industry follows more or less and hence does not limit the scope to one OEM. Similar approaches might be possible also for other industries where there are multiple ECUs whose functionalities could be combined into one.

Here, the explicitly formal, i.e., mathematical, evaluation of MBSE and its data models and other implemented data models is not within scope due to it only being one part of the entire solution framework. Moreover, the evaluation of data models commonly assesses the quality of data models by distinct metrics¹³⁷. However, here in this work the focus is on generating traceability by means of an integrated data model, inter alia, which is not designed to achieve the highest performance in modeling or data exchange¹³⁸.

Table 6-1 depicts according to which exemplary use cases the requirements are addressed.

¹³⁶ As described in Chapter 1.4, the DRM only serves as a blueprint where suitable and is not followed stoically.

¹³⁷ For more information about formal verification techniques, please refer to DEBBABI et al. (2010).

¹³⁸ For more information regarding methods for evaluation, please refer to BORTZ and DÖRING (2006) and FELDHUSEN and GEBHARDT (2008: 189 ff.) for the quantitative evaluation of PDM IT systems.

Table 6-1: Evaluation scheme for use cases.

		<div> <div>○</div> not addressed </div> <div> <div>◐</div> partially addressed </div> <div> <div>●</div> addressed </div> <div> <div>■</div> not applicable </div>		Addressing	
				Use case 1: DCM	Use case 2: Central ECU
Objectives	Requirements				
1. Internal traceability	a. Alignment of MBSE & PDM for E/E	1. Include ECU pins			
		2. Include NCD			
		3. Include ECU SW			
		4. Linked data model			
2. External traceability	a. Reduction of reconciliation	5. Trace links available			
		6. Foster distributed engineering			
		7. Consensus mechanism			
		8. Automatic change propagation			
	b. Transparent & safe product changes	9. Immutable product history			
		10. Multi-directional synchronization			
		11. Traceability scheme OEM-supplier			
		12. Data integrity			
	c. Alleviated connection of engineering partners	13. Standard data model for exchange			
		14. Stand developm. processes			
		15. Standard APIs/ integration in legacy IT			
		16. Availability of data & robustness			

6.2 Use case 1: Door control module

6.2.1 Technical problem statement

Figure 1-5 depicts the generic system development process in automotive E/E development with multiple engineering collaboration partners. This generic process is the basis for use case 1 which focuses on fostering traceability up- and downstream of the process within the OEM's company (internal) and in alignment with suppliers (external). For this purpose, the development process of the DCM will be evaluated.

OBJECT OF INVESTIGATION

The functionalities implemented in the DCM have increased in the last years due to an increase in functionalities in an automobile's door itself. This yields in a greater variety of variants and hence a higher complexity which has to be bundled within a DCM. Commonly, a DCM comprises of the control of the electric window lifter, mirrors, functional illumination, locks, signal lights in the mirrors, seat adjustment, etc. Due to these functionalities being connected with each other within the automobile, the DCM must be interconnected with other ECUs. Depending on the amount and complexity of functions implemented in the DCM, it is connected via a CAN, or in the case of less complexity, a LIN bus is used (REIF, 2014: p. 245). Figure 6-1 depicts a DCM from an E/E perspective (in alignment to REIF, 2014: p. 246). The respective software, bootloader, functional, and parametric, is added in darker shades. The parametric software is stored within the EEPROM. In principle, a DCM can be divided into the input area, the processing unit, and the output area (from left to right in Figure 6-1) (REIF, 2014: p. 246). Particularly, the I/O structure, for instance pins, plugs, signals, etc. are relevant here (cf. Figure 4-15). Single E/E sub-parts, sensors, and actors within or attached to the ECU are not in scope and subsumed under "ECU".

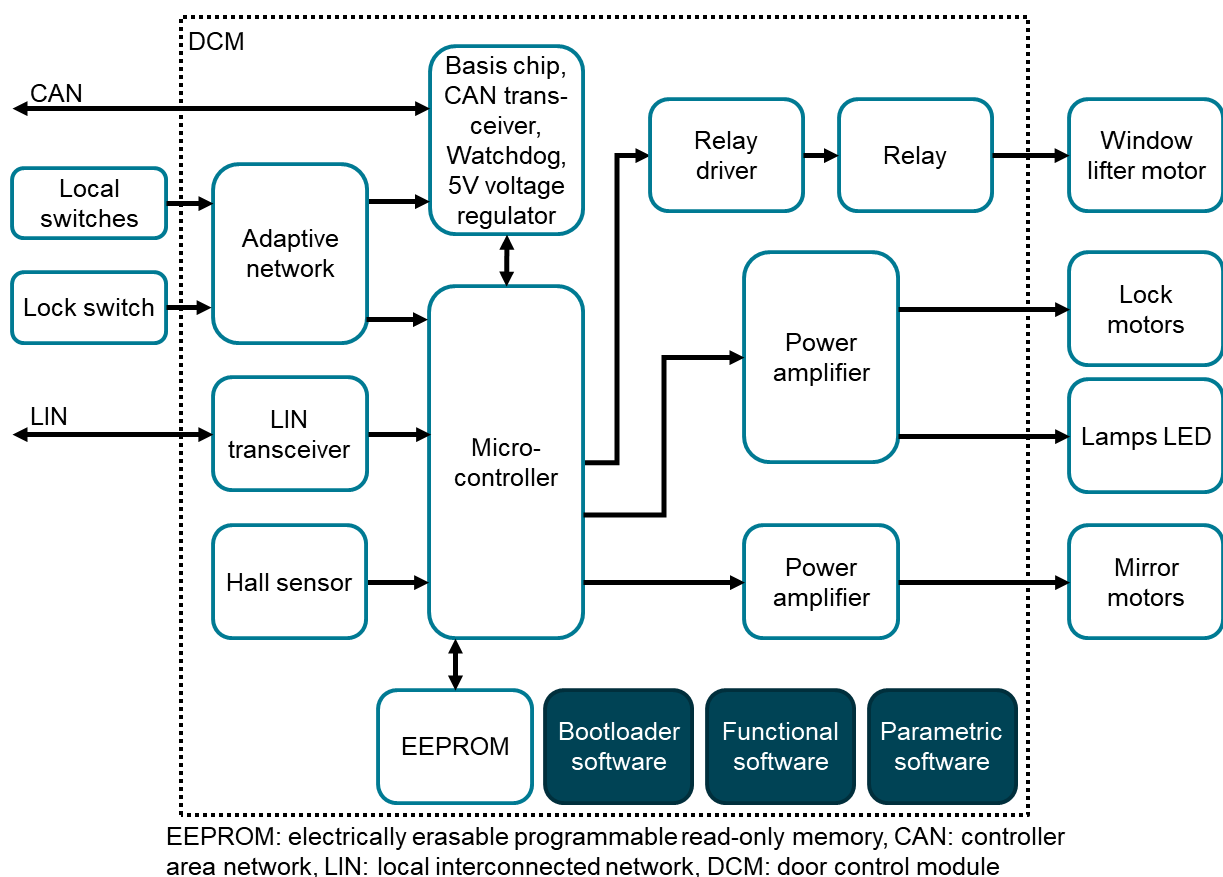


Figure 6-1: Door control module (in alignment to REIF, 2014: p. 246).

6.2.2 Use case description

The DCM can be considered as one of the least complex ECUs in an automobile and hence serves here as an entry level example for the joint development in an engineering collaboration. It is assumed that the entire DCM hardware, i.e., the physical ECU, is developed by one supplier including the bootloader and functional software for the main car variant (cf. Figure 2-14 and Figure 2-16). The parametric software usually is coded by the OEM. The DCM is embedded in the automobile's E/E system "comfort system" and interacts there with other ECUs within this system as well as across other E/E systems. For instance, in case of an emergency, windows close automatically and this signal by another ECU has to be processed within the DCM. Therefore, changes in other ECUs within the same or from another E/E system have impacts on the DCM itself and vice versa. These dependencies have to be considered early during the development.

Commonly, the OEM starts by defining the E/E architecture and within this dedicated E/E systems, such as the "comfort system" where the DCM is included. For the search of a supplier, the OEM writes a specification sheet for the DCM including all relevant technical parameters that have to be considered during development to ensure proper operability. After the confirmation of the order, the supplier starts with the engineering process. Partially in parallel, many other engineering partners develop other ECUs and E/E systems as well as the OEM starts coding the parametric software. Given the above-mentioned multiple dependencies, engineering changes affecting other parts or information artifacts of different engineering partners have to be detected as early as possible (cf. Chapter 1.2). For that purpose, the actual implementation of the use case within the solution framework has to be described in the following.

6.2.3 Exemplary implementation of use case 1

The relevant information artifacts for use case 1 were addressed in Chapter 4.1.1 where the synthesis of the data model for the solution framework was presented. Figure 4-6 depicts the generic structure of the ECU, also including the DCM's relevant information artifacts. The implementation of these information artifacts was shown in Chapter 5.4 and depicted in Figure 5-28 and Source Code 5-11.

Objective 1.a., foster internal traceability by means of the alignment of MBSE and PDM for E/E, requires the explicit inclusion of ECU pins (*requirement 1*), the NCD (*requirement 2*), and the ECU software (*requirement 3*) as crucial information artifacts specifically for E/E. Therefore, these information artifacts must be modeled within the

data model. This is depicted in Source Code 6-1. Although the relevant information artifacts to foster traceability were defined and implemented, the holistic integration of this data model with these of MBSE and PDM in their respective IT systems were not implemented (cf. *requirement 4*). However, *requirement 5* was addressed, including `https` links in the data model which serves as part of the basis for *requirement 11*. Yet, the traceability scheme for OEM and supplier is only partially addressed. This is due to OSLC including RDF, `https` trace links, and REST APIs are implemented but not connected across all IT systems. The implemented standard data model alleviates data exchange. Due to there is no automatic data exchange within the MBSE tool and PDM system implemented, *requirement 13* is only partially addressed.

```

1  // DCM
2  {
3      "Uuid"           : "923b8f25-b8e9-4348-9366-fdc99f76bd33",
4      "SachNummer"     : "A223123456789",
5      "ExternalPartNo" : "258-6AC-8732",
6      "ArtefactType"   : "ECU",
7      "Description"    : "Door Control Module",
8      "Version"        : "V07.20",
9      "Issuer"         : "Alice",
10     "Timestamp"       : "19-07-2020",
11     "Architecture"    : "EtherStar",
12     "ReleaseStatus"   : "Released",
13     "CreatedBy"       : "Alice Daimler",
14     "LastUpdatedBy"   : "Bob Supplier",
15     "ArtefactCategory": "E/E",
16     "DescriptionGerman": "Tuersteuergeraet",
17     "Flashable"       : true,
18     "SignaturKennzeichen": "DCM_V07.20",
19     "Link_HTTPS"      : "<https://daimler.engineering-mbc.com/mra2/223/etherstar/ee-system/DCM>",
20     "EE_System"       : "Interior_comfort",
21     "ECU"             : "DCM",
22     "Hardware"        : "Case_V07.20",
23     "Geometry"        : "drawing_A223123456789",
24     "Memory"          : 512,
25     "Processor"       : "ASIC438",
26     "Plug"            : "M17/1*2CK6-B",
27     "Pin"             : "1/12",
28     "NCD"             : "NCD_DCM_V07.20",
29     "IO_channel"      : "LIN1",
30     "Signal"          : "Signal_DCM",
31     "Software"        : "SW_DCM_V07.20",
32     "Bootloader_SW"   : "SW_boot_DCM_V02.19",
33     "Functional_SW"   : "SW_funct_DCM_V07.20",
34     "Parametric_SW"   : "SW_param_DCM_V07.20",
35     "Communication_Bus": "LIN",
36     "SW_Function"     : "DCM_V07.20_funct",
37     "Configuration"   : "DCM_high-end",
38     "Variant_point"   : "ECU_DCM_high-end",
39     "Variant"         : "DCM_high-end_BR223",
40     "Model_series"    : "BR223",
41     "Release_model"   : "BR223_MY07.20",
42     "Engineering_change": "ECM_3907158_07.20",
43     "Deployment_date"  : "19-07-2020",
44     "Requirements_view": "requirementsView",
45     "Functional_view"  : "functionalView",
46     "Logical_view"    : "logicalView",
47     "Technical_view"  : "technicalView",
48     "IT_system"       : "itSystem",
49     "MBSE_tool"       : "Windchill_modeler",
50     "PDM_system"      : "E/E_PDM",
51 }

```

Source Code 6-1: Data model for door control module.

When starting with the process, the engineer models the E/E system in *Windchill modeler*, the MBSE tool. According to the SPES method and the RFLP logic, information artifacts are created for each engineering partner within the authoring MBSE tool, transferred to the RDF namespaces by means of the standard data model, eventually updating the documentation in the PDM system for E/E, and satisfying *requirement 14*. The standard development process is relevant for internal traceability and even more so for synchronized work with engineering partners fostering external traceability. This is depicted schematically in Figure 6-2.

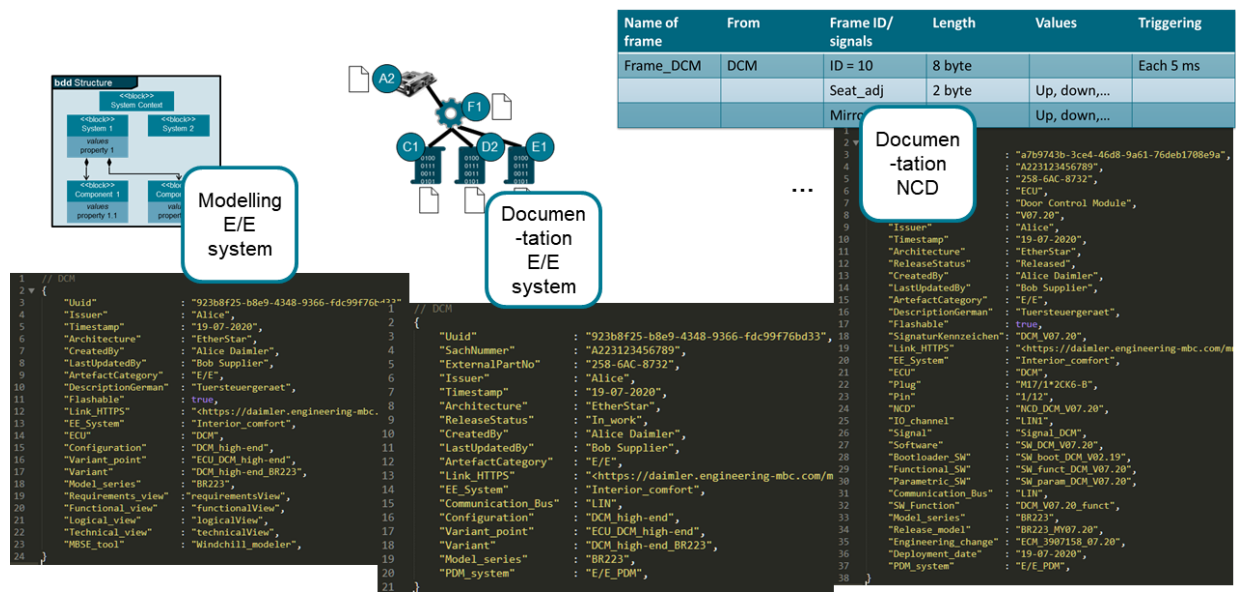


Figure 6-2: Development process for door control module.

Moreover, external traceability is strongly enabled by the Blockchain technology serving as the PLM backbone for each OEM and engineering partner which fosters distributed engineering. Given that the OSLC key components are implemented (cf. Chapter 5.4), however not across all IT systems, *requirement 6* is only partially addressed. Through the consensus mechanism (*requirement 7*) and the automatic change propagation (*requirement 8*), engineering changes for the DCM become directly apparent, which can then be assessed, and the confirmation or veto can be communicated. Thus, the reduction of reconciliation is fostered during the development of the DCM. Engineers modeling and documenting their information artifacts for the DCM can rely on an immutable product history (*requirement 9*) when exchanging data with engineering partners and suppliers. The multi-directional synchronization (*requirement 10*) is not completely addressed due to the above-mentioned reasons of no implementation in the MBSE tool and PDM system. The same applies to *requirement 15*, although the *Hyperledger* framework provides REST APIs. Data integrity for safe product changes

(*requirement 12*) which is ensured for the DCM development through the Blockchain technology, as well as the availability of data and its robustness (*requirement 16*). The technological aspects of the use case for DCM development are depicted in Figure 6-3.

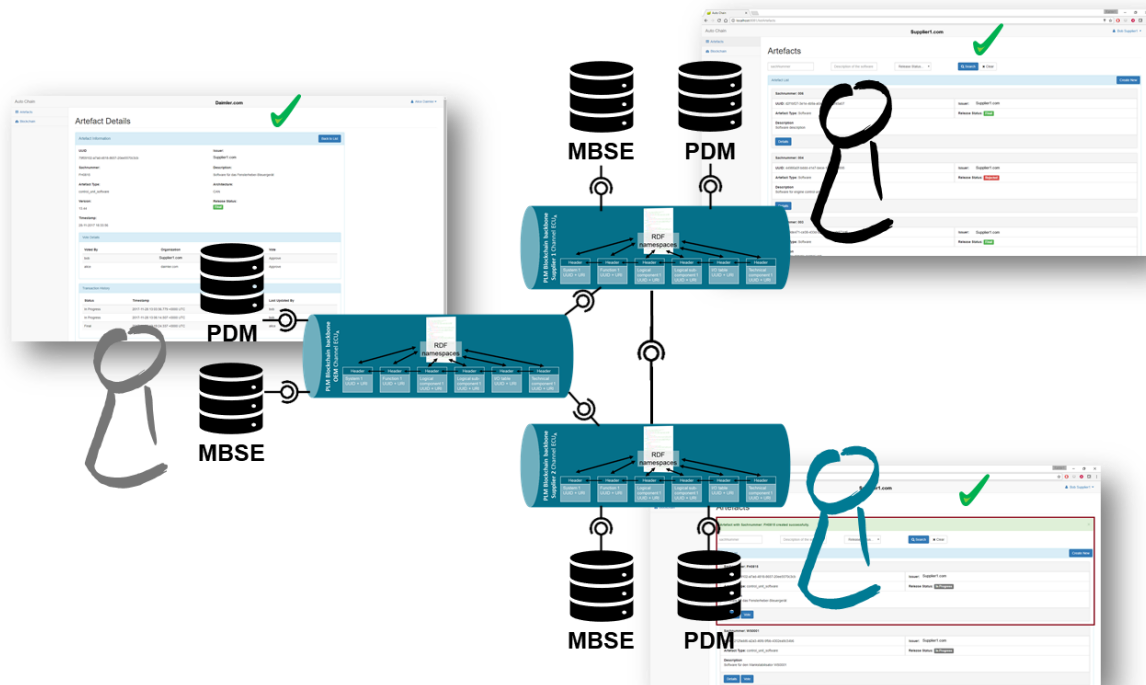


Figure 6-3: Technology aspects for door control module.

6.3 Use case 2: Centralized, server-oriented E/E architecture

Due to the maturity of this use case during the definition and implementation of the prototype and elaboration of this work, the following chapters describe a conceptual state. Nevertheless, the developed solution framework for traceability suits very well the future server-oriented automotive E/E architecture with a centralized control unit and a server-oriented E/E architecture, for which reasons this use case is deemed to be evaluated, albeit given some assumptions.

6.3.1 Technical problem statement

Similar to use case 1, the engineering collaboration with multiple partners is assumed as well as the current impediments. The motivated, synthesized solution framework (cf. Chapter 4) is deduced from the impediments and used to address these with the compiled, implemented framework (cf. Chapter 5).

OBJECT OF INVESTIGATION

In order to counter the increasing complexity of the automotive E/E architecture (cf. Chapter 1.2), OEMs and suppliers develop and already offer a centralized, server-oriented approach where a centralized ECU with more processing power and more memory substitutes multiple single, less powerful ECUs. This does not only reduce complexity of the wiring harness, communication busses, and packaging but also enables future software and feature updates which might require more powerful ECUs than today's used ones. The centralized ECU consists of distinct hardware, middleware/OS, and multiple software. Sometimes, several virtual ECUs are emulated within the centralized ECU to provide different software environments for different applications as well as for security reasons. The centralized ECU is connected to other ECUs controlling distinct zones of the automobile's E/E architecture, for instance the multimedia system. Additionally, gateways and body control modules can be connected (CONTINENTAL AG, 2020, 2021; DAIMLER AG, 2020). The structure of the centralized ECU is depicted in Figure 6-4.

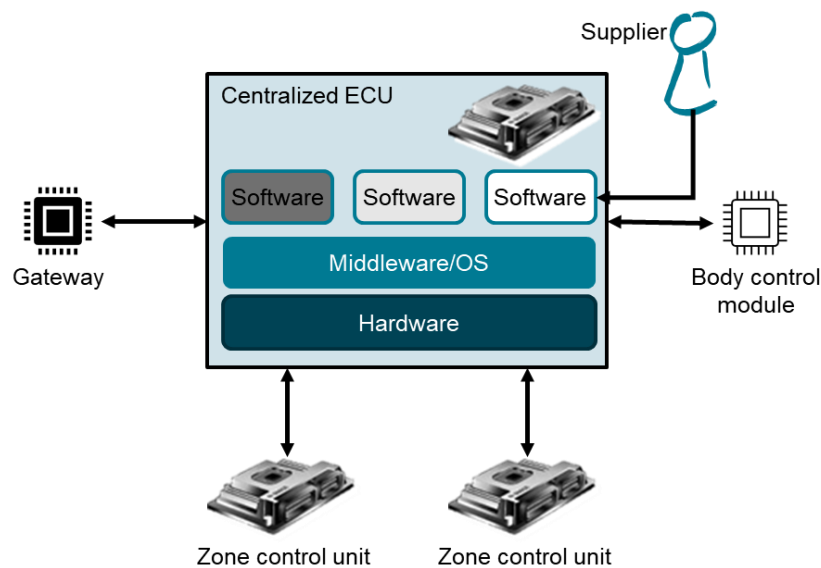


Figure 6-4: Centralized ECU (in alignment to CONTINENTAL AG, 2021).

The multiple software can be separated in sandboxes, again for security reasons. This enables the access of third parties, such as a supplier, to this dedicated software sandbox on the centralized ECU via the automobile's antenna. Consequently, the supplier can update existing, or add new features and applications throughout the automobile's lifecycle.

6.3.2 Use case description

Already today, automobiles receive updates over the air during their operations phase with the customer. These can be minor bug fixes, updates of, e.g., navigation data and enabling of features for which the hardware already was integrated in the automobile but the feature has not been purchased initially. Such updates during the operation phase reflect the processes during the initial development including the development *V-model* where the OEM serves as a final integrator of all the contributions of the suppliers and their own. In contrast, the centralized, server-oriented E/E architecture with sandboxes for dedicated software applications on the centralized ECU enables the direct transfer of software updates and features by the suppliers to the centralized ECU. Therefore, the OEM does not have to serve as a carrier for specific software from suppliers into all vehicles anymore. This fosters the easier integration of new software suppliers in order to decrease time to market for new features. Still, all engineering partners write their changes of information artifacts into their Blockchain PLM backbone from where these information artifacts are synchronized within the engineering collaboration's Blockchains and the respective MBSE tools and PDM systems. After the consensus process and positive responses, the supplier can directly flash the new software onto all vehicles in scope without the need for a detour via the OEM's IT infrastructure. This use case is depicted in Figure 6-5. Of course, granting access to the transmission infrastructure of the OEM still is necessary for a supplier to execute such a remote update.

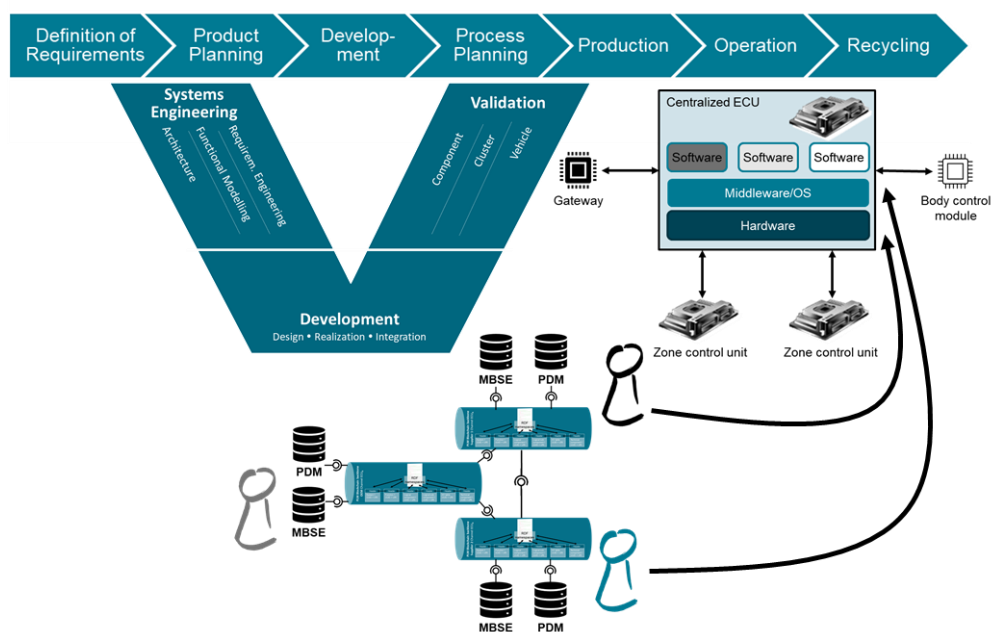


Figure 6-5: Use case centralized ECU.

6.3.3 Exemplary implementation of use case 2

Again, the exemplary implementation of the centralized, server-oriented E/E architecture with a central ECU highlights the peculiarities of this scenario. Further details have already been given above.

Requirements 1 to 3 have been addressed, although not all pins, NCDs, and software have been included in the data model which would probably exist in an integrated, powerful centralized ECU. The data model is depicted in Source Code 6-2.

```

1  // Cental ECU
2  {
3      "Uuid"           : "435ff21c-0df0-4368-a257-429559f59509",
4      "SachNummer"     : "A223111222333",
5      "ExternalPartNo" : "223-ECU-3914",
6      "ArtefactType"   : "ECU",
7      "Description"    : "Central ECU",
8      "Version"        : "V07.20",
9      "Issuer"         : "Alice",
10     "Timestamp"       : "19-07-2020",
11     "Architecture"    : "EtherStar",
12     "ReleaseStatus"   : "Released",
13     "CreatedBy"       : "Alice Daimler",
14     "LastUpdatedBy"   : "Bob Supplier",
15     "ArtefactCategory" : "E/E",
16     "DescriptionGerman" : "Zentrales Steuergeraet",
17     "Flashable"       : true,
18     "SignaturKennzeichen" : "CECU_V07.20",
19     "Link_HTTPS"      : "<https://daimler.engineering-mbc.com/mra2/223/etherstar/ee-system/CECU>",
20     "EE_System"       : "Central",
21     "ECU"             : "CECU",
22     "Hardware"        : "Case_V07.20",
23     "Geometry"        : "drawing_A223111222333",
24     "Memory"          : 3200,
25     "Processor"       : "AMD5678",
26     "Plug"            : "M17/1*2CK6-B",
27     "Pin"             : "1/12",
28     "NCD"             : "NCD_CECU_V07.20",
29     "IO_channel"      : "ETHERNET_MAIN",
30     "Signal"          : "Signal_CECU",
31     "Software"        : "SW_CECU_V07.20",
32     "Bootloader_SW"   : "SW_boot_CECU_V02.19",
33     "Functional_SW"    : "SW_funct_CECU_V07.20",
34     "Parametric_SW"   : "SW_param_CECU_V07.20",
35     "Communication_Bus" : "ETHERNET",
36     "SW_Function"     : "CECU_V07.20_funct",
37     "Configuration"   : "CECU_high-end",
38     "Variant_point"   : "CECU_high-end",
39     "Variant"         : "CECU_high-end_BR223",
40     "Model_series"    : "BR223",
41     "Release_model"   : "BR223_MY07.20",
42     "Engineering_change" : "ECM_7937158_07.20",
43     "Deployment_date"  : "19-07-2020",
44     "Requirements_view" : "requirementsView",
45     "Functional_view"  : "functionalView",
46     "Logical_view"    : "logicalView",
47     "Technical_view"  : "technicalView",
48     "IT_system"       : "itSystem",
49     "MBSE_tool"       : "Windchill_modeler",
50     "PDM_system"      : "E/E_PDM",
51 }

```

Source Code 6-2: Data model for centralized ECU.

A linked data model is enabled given the `https` links and UUIDs. However, the complete integration with the MBSE tool and PDM systems is not implemented despite REST compatibility of all systems (*requirement 4*). Trace links are implemented for each information artifact (*requirement 5*). *Requirement 6* is addressed only partially by the implementation of OSLC, yet not including all IT systems. *Requirements 7, 8, 9, and 12* are addressed again through the usage of the Blockchain technology. Due to changes for the central ECU also first have to be shared with the engineering partners through the individual PLM Blockchain backbones, a multi-directional synchronization is ensured within the Blockchain network (*requirement 10*). A traceability scheme is enabled by OSLC and its components, but it was not completely implemented across all IT systems (*requirement 11*). The standard development process is implemented (*requirement 14*) and the standard data model for exchange (*requirement 13*) is addressed partially due to limitations in the connection of IT systems. The alleviated connection of engineering partners is enabled by means of standardized APIs; however, the integration of legacy IT systems is not implemented completely (*requirement 15*). Data availability and its robustness is addressed by the Blockchain (*requirement 16*). These different aspects are depicted in Figure 6-6.

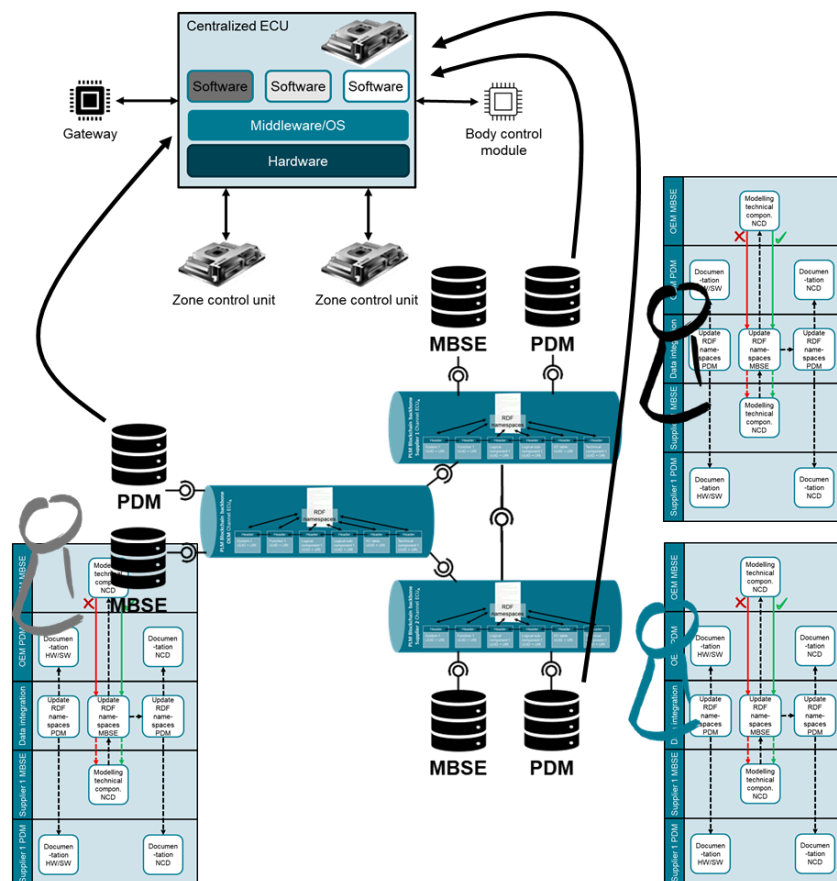


Figure 6-6: Development process and technology aspects for centralized ECU.

6.4 Evaluation of research objectives

The evaluation of the research objectives (cf. Chapter 1.3) and the resulting requirements (cf. Chapter 3) is done by means of the three questions for application evaluation (cf. Chapter 6.1).

THE SOLUTION FRAMEWORK CAN BE USED

The scope is to assess whether the support presented here, which is the solution framework, is actually usable and applicable to the situation which shall be improved. This means, if the support is able to address the key factors or objectives and requirements in this case. BLESSING and CHAKRABARTI (2009) give the example that, albeit a support for an issue is implemented, it might be too complicated for designers to use. Hence, usability would not be given in this case, and proper solution approaches to increase usability must be considered in order to not discard the intended support (BLESSING and CHAKRABARTI, 2009: pp. 167, 187–188).

Transferred to the framework at hand, the usability and applicability of it must be evaluated. This has to be looked from the perspective of two different angles. First, it shall be deduced if the solution framework addresses the research objectives and resulting requirements (*applicability*). Secondly, as mentioned by BLESSING and CHAKRABARTI (2009), it shall be considered whether the solution framework supports engineers in their work (*usability*).

The applicability of the solution framework is inherent, given the deduction of it from the initial problem statement via all the steps in between. The problem statement is the basis for the research objectives, which have been categorized to address different kinds of traceability and subitems within them. The state of science and technology was assessed given the three dimensions of enablers, data model, process model, and technology, that are relevant for the solution framework, as motivated in Chapter 1.4. Based upon the research objectives and the state of science and technology, the requirements have been derived from. Deduced from these previous assessments, the solution framework was synthesized. Ergo, the applicability of the solution framework to address the problem statement is in alignment with the nature of deduction of the entire research approach. The deductive approach is depicted in Figure 1-8.

The usability of the solution framework was analyzed above by means of use cases. Here, the perception was made that the engineer has to document less in multiple tools, reduces effort for reconciliation and searching for the most recent product updates, and

is supported by the solution framework in their engineering tasks within the engineering collaboration. In general, the usability of the framework is given. Limitations will be addressed further below in the next paragraphs and in Chapter 6.5.

THE SOLUTION FRAMEWORK ADDRESSES THE REQUIREMENTS

As stated above, the requirements have been derived from the research objectives after the evaluation of the state of science and technology. Each objective was further subclassified and within these subitems the requirements were formulated. For the synthesis of the solution framework, each requirement was addressed individually by one of the three enablers. This means, that for instance, an information artifact was modeled explicitly, a processual step was defined, or a given logic was implemented into the IT tools to satisfy these specific requirements. Therefore, the solution framework addresses all established requirements.

THE REQUIREMENTS ARE AFFECTED AS EXPECTED

The expectation is that objectives and their requirements are addressed completely by the solution framework. The solution framework does indeed address all requirements generically and conceptually. However, complexity impeded the holistic implementation of all requirements (cf. Chapters 6.2 and 6.3). Hence, the presented use cases only partially address some requirements and conclusively also only partially address the defined objectives.

As presented in Table 6-2, both use cases display the same level of generic implementation and thus also the same extent of requirements which are addressed completely or only partially. As presented above, the challenges are to fully implement the defined linked data model with existing MBSE and PDM data. For this reason, objective 1.a. is only addressed partially. Likewise, objective 2.a. is confined regarding the complete implementation of OSLC across all IT systems. The same reason applies to objective 2.b., where the implementation with MBSE and PDM systems is not fully achieved. This also prevents a complete automatic data exchange and the integration of legacy IT systems. Hence, objective 2.c. is only addressed partially.

In conclusion, it can be stated that the developed solution framework addresses all objectives and requirements, albeit not all requirements could be implemented to their fullest extend in each use case due to the reasons mentioned above (cf. Table 6-2).

multiple engineering partners. A major obstacle is the connection with and inclusion of legacy IT systems and their existing data models with their own semantics which is implemented here as a generic joint metadata model.

PROCESS MODEL

The definition of a joint development process model that fits the domains of MBSE, PDM, and has a technological fit with the Blockchain technology, is achieved by extending the SPES method for MBSE. The SPES method defines which information artifact is created in which scenario, but not at which point in time. The alignment with the consensus mechanism of the Blockchain technology is the basis for the reduction of reconciliation and transparent changes. The major PDM operations, for instance, creation of a configuration and variant, change management, and inactivation, are described for a limited number of engineering partners. However, there are many more relevant PDM processes, such as release or audit management (cf. Chapter 2.3.2) which are not addressed here in this work.

TECHNOLOGY

As the Blockchain technology has already existed for some years in the financial world, it is slowly finding its applications in other domains. Therefore, what was defined initially for finances has to be adapted and extended to fit the engineering domain. Here in this work, a first step towards this application of the Blockchain technology is done. The connection with tools and IT systems from engineering is very challenging and only is implemented partially in the presented use cases. Although the underlying technology for the interfaces between the Blockchain and MBSE and PDM systems, i.e., REST API, is the same, a complete definition of these interfaces is not feasible during this work and still requires a lot of standardization. Moreover, advantages of the Blockchain technology, e.g., anonymity, ad hoc participation, transparency, etc. do not always apply in an industrial setting. Although the *Hyperledger fabric* framework addressed some of these adaptations, not all issues for engineering collaboration are resolved.

The *Hyperledger fabric* framework, used for the implementation of the Blockchain, does not support the addition of further information artifacts once the Blockchain network was installed among all peers. This is particularly cumbersome in agile development where not all scope is specified at the beginning but rather during engineering. The latest versions of *Hyperledger fabric* might address this issue by implementation of a chain code lifecycle (SORNIOTTI and YELICK, 2018).

7 Summary and outlook

7.1 Summary

The research work at hand assesses challenges that are arising during the automotive development process in distributed engineering collaborations. Particularly, traceability of E/E information artifacts in early development within a company and among multiple engineering partners is in scope. The state of science and technology so far has not established sufficient concepts to address these issues in detail.

This work constitutes the creation of a new solution framework by composition and assembly of existing approaches, similar to the Blockchain technology itself, and new solution elements applied in a complex and heterogeneous context with multiple stakeholders, different IT standards, and emerging technologies. Such a problem space in engineering can be denominated as a *wicked* problem and the research thereof as *science of the artificial*. The envisioned solution framework composing and assembling different solutions even could yield a surplus benefit, such as traceability reduces error proneness, increases quality, reduces efforts, etc., what is called *emergence*. Therefore, this work follows a design research approach.

The objectives of this work are to foster internal as well as external traceability by means of a solution framework consisting of a data model, process model, and technology as enablers. The objective to foster internal traceability is further refined by the connection of the early systems development (MBSE) with product data management (PDM). External traceability is subdivided further into: cultivating the reduction of reconciliation among engineering partners, to encourage transparent and safe product changes, and to foster alleviated connection of engineering partners. Given the conceptualization of the enablers, the solution framework offers a sufficient satisfaction of the requirements to foster traceability of E/E information artifacts in engineering collaborations and addresses all previously defined objectives. During the implementation of the solution framework, not all aspects of the objectives and requirements are addressed by the enablers due to the underlying complexity of the connection of the solution framework with legacy IT systems.

DATA MODEL

The presented metadata model incorporates aspects of E/E, MBSE, and PDM in automotive development. System modeling aspects shape the hierarchy of the data model, which is strongly aligned with existing industry standards, and extending the data model further to include crucial missing aspects of E/E and PDM. For E/E in particular, the E/E systems all the way down to an ECU's pin as well as a NCD are modeled. Configurations, variants, and versions are included to reflect the PDM aspects of a product lifecycle and for the collaborative exchange of data with engineering partners.

For the purposes of availability of data among engineering partners, association of data is achieved by including UUIDs and links into the metadata. Data integration is accomplished by RDF namespaces which comprise the joint metadata models of each domain and can be included into each engineering partner's own IT systems using the defined links and UUIDs.

The definition of a metadata model, stored in standard form, its integration in the engineering collaboration, and the linkage to each partner's legacy IT systems provide the fundament for traceability.

PROCESS MODEL

In alignment with the chosen systems development approach and the selected technology underneath, the elaborated process model defines at which point in time which information artifact is created in which IT system of which engineering partner. It has been considered that the systems development is aligned with the PDM.

Moreover, the process model includes, provided by the Blockchain technology, instantaneous change propagation and the underlying consensus mechanism regarding these changes. Additionally, the exchange of data for the purpose of synchronization of all involved IT systems is modeled in the process models. Both aspects are transferred to engineering collaborations.

Combining the systems development with PDM aspects, as well as the reduction of reconciliation combined with transparent product changes, fosters traceability during the early automotive development phase with multiple engineering partners.

TECHNOLOGY

As for the integrational technology for collaborative data exchange, a decentral peer-to-peer data base is chosen, namely the Blockchain technology and specifically the *Hyperledger fabric* framework. By the use of the *Hyperledger fabric* framework, some

objectives of this work are addressed inherently. The Blockchain technology fosters traceability through the immutable documentation of transactions among many peers, offers a consensus mechanism which can be applied to enterprise necessities, spreads changes in form of new transactions instantaneously, and many more. This work applies the Blockchain technology to the realm of engineering IT. Data and process models from engineering are adapted to fit the solution framework. The application of this solution framework in the early systems development in the automotive industry is a novelty. Moreover, ad hoc participation of engineering partners for the contribution of engineering content is alleviated by means of the standardized Blockchain technology.

As the Blockchain technology in the financial domain is classically not connected to further, extensive data bases, except those storing information of the transactions, users, or their assets, the connection to legacy IT tools and systems is tremendously challenging. Implementing standard APIs, joint data model namespaces, and the alignment of information artifacts across IT systems, enable this connection.

The Blockchain technology is a very powerful solution which helps fostering traceability in the systems development within an engineering collaboration, as it is shown in this elaboration. Yet, further research must be conducted to enable a standardized industrial application of it.

7.2 Ramifications and outlook for automotive engineering IT

RAMIFICATIONS

From engineering and engineering IT perspective the introduction of a new, disruptive technology is often associated with many challenges. The Blockchain technology can be considered disruptive due to its decentralized setup, versus centralized, more traditional approaches. Such challenges arise in the technical area, as this work describes abundantly, as well as in the organizational area.

Enabling and training people to work with new technologies, such as presented here, is an important task to foster acceptance and remove barriers. As the effort for searching, alignment, reworking, etc. diminish due to a holistic framework for traceability by means of the Blockchain technology, efficiency of engineers will increase. This in turn means that for the same amount of work, less people are necessary. Generally speaking, this could lead to a reduction of the workforce. For companies, efficiency gains are crucial

to remaining competitive. On the other hand, increased efficiency and the potential reduction of the workforce might have negative effects for employees.

The organizational ramifications of the introduction of the Blockchain technology on organizational aspects, not only for the engineering IT but also in the wider social context, will be addressed in more detail in Chapter 7.4.

OUTLOOK

A further standardization of data models in automotive development across the entire industry would further foster traceability. At least each domain involved in the development phase up to the involvement in the entire lifecycle of an automobile should adapt the same business objects on a high level in order to achieve traceability across many IT systems and tools as well as among engineering partners. Such standardization would further contribute to the concept of a *Digital Twin*, which will be touched briefly further below.

Additionally, the associations or relations between information artifacts could be enriched further with relevant knowledge for development or production. This concept has already been presented by GROLL (2008) for PDM in the automotive industry. Important knowledge for MBSE, E/E, engineering collaboration, specific processes, etc. could be added to the interconnections of information artifacts, creating a model-based process description. For this purpose, the *association block* in *UML* or *SysML* could be used and adapted (cf. WEILKIENS, 2008: pp. 154 ff.).

As mentioned in Chapters 2.7.3 and 4.3.2, smart contracts¹³⁹ are not in scope of this work due to their complexity in themselves, as well as for PDM in general. Smart contracts could however be implemented for the purpose of containing and executing Boolean expressions for the combinatory, which is required in the automotive development (ZENGLER and KÜCHLIN, 2013). Smart contracts could execute the *if-then-relations* to combine and exclude technical dependencies, such as `if 'gas engine' then NOT 'diesel engine'`, or map sales-oriented combinations, for instance the upgraded navigation system requires the entire business package. On a lower level, in the sense of technical product granularity, smart contracts could also represent dependencies within data models. Thereby, the incompatibility of information artifacts could be modeled and checked automatically. This automatism could be implemented early in the development phase and could partially replace manual process steps of

¹³⁹ For more information about smart contacts, please refer to BASHIR, 2018: pp. 261 ff.

agreement or decline during the consensus mechanism (cf. Chapter 4.2). A higher automatized and more efficient, error-robust development process would be the result.

The Blockchain technology and one framework, the *Hyperledger Fabric* platform, are implemented here exemplarily due to *Hyperledger Fabric's* focus on consortia solutions including permissioned Blockchains. However, there are plenty of other Blockchain or *distributed ledger technologies* (DLT) or frameworks as well as *decentralized autonomous organizations* (DAO). Each of this framework has its peculiarities, different foci, and research in this field as well as implementation evolve quickly. Hence, there could be, in the meanwhile, a superior Blockchain technology available for dedicated purposes. Furthermore, different Blockchains can be pegged together, allowing for the transfer of assets between them (BACK et al., 2014). This approach could further foster data sovereignty of each engineering partner, not having to rely on the OEM's channels. Therefore, future research also could focus on the specific definition of a Blockchain platform, tailor-made for the automotive industry¹⁴⁰.

The *Digital Twin* is an extension of the PDM and PLM concepts with individual, product-specific scope. The term *Digital Twin* can be understood in general as the holistic, physical (described by laws of physics) and functional description of a component, a product or a system which has a physical (real, haptic) instance and a digital copy ("twin") in IT backbone systems. Included in this is all relevant information which could be necessary for current or future lifecycle phases (BOSCHERT and ROSEN, 2016: p. 59; HEBER et al., 2018: p. 9; HEBER and GROLL, 2018c: p. 324). The concept of the *Digital Twin* is depicted in Figure 8-1¹⁴¹. Changes applied to the *Digital Twin* could be traced by means of the Blockchain technology (KIRKPATRICK and KAUL, 2019: p. 3). By this, the Blockchain technology could foster traceability on the level of individual products.

As described in Chapter 5.2, *Docker containers* were used to install the prototype on each peer's computer. Beyond that, *Docker* images could be applied to transfer the complete initial Blockchain network, data model for MBSE and PDM, as well as process model information to all new engineering partners who want to contribute to the development of the dedicated automotive E/E system. The image of the *Docker container* could be ready for download once granting access to it. After installation, the

¹⁴⁰ For more information about different Blockchain technologies, DAOs, and DLTs, please refer to FROST & SULLIVAN (2017), FRØYSTAD and HOLM (2015), HILEMAN and RAUCHS (2017), DHILLON et al. (2017), UK GOVERNMENT CHIEF SCIENTIFIC ADVISER (2016).

¹⁴¹ For more information about the concept of the *Digital Twin*, please refer to HEBER et al. (2018), HEBER and GROLL (2018c), BOSCHERT and ROSEN (2016), EIGNER et al. (2017).

Blockchain network would be initialized for the new peer, APIs configured, metadata model implemented, and information regarding the development process shared. This would further alleviate the connection of engineering partners and foster external traceability.

In addition to the presented use case 2 (centralized, server-oriented E/E architecture), the automobile also could serve as a distinct node of the Blockchain network. Not only would engineering partners directly write their software into each car, but the car would also document these changes transparently, immutably, and decentralized. This would mitigate the possibilities of manipulation of software particularly during the after sales phase and could contribute to the above-mentioned concept of the *Digital Twin*. Whether smart contracts would automatically agree or decline changes of the software, or the PLM Blockchain within the car just serves as a documentation of results without participation in the consensus mechanism, still has to be defined. This outlook is depicted in Figure 7-1.

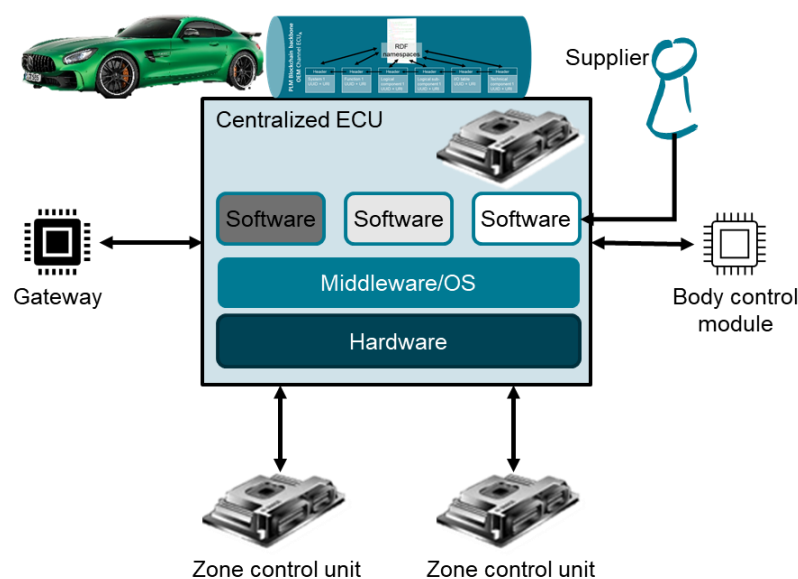


Figure 7-1: Automobile as distinct node in the Blockchain network.

7.3 Ramifications and outlook for further industries

The Blockchain technology has already revolutionized parts of the financial industry by the introduction of crypto currencies. Besides this, the application of the Blockchain technology has evolved into many other industries where traceability combined with immutability, transparency, and decentralization is required.

“Insurance, healthcare, transportation, real estates, manufacturing, networking, [and] IoT, and energy management” (FROST & SULLIVAN, 2017: pp. 25 ff., 51) are supposed to be industries which will be disrupted by 2025 by the Blockchain technology where a wide array of applications of the Blockchain technology considers the documentation of ownership of physical and digital assets (cf. MORABITO, 2017: pp. 28 ff.; ANTONOPOULOS, 2017: p. 278). Likewise, the financial sector will be impacted (FROST & SULLIVAN, 2017: p. 51). Supply chain and verification of provenance might be one of the most popular application of the Blockchain technology outside the financial area (LU and XU, 2017: p. 22). Confirmation of the reliability of scientific studies and their results, or medical studies probably rank among the rather exotic use cases (MORABITO, 2017: p. 30). For governmental affairs and the public sector, for instance for notaries, land charge registers, or auditing, traceability with the amenities of the Blockchain technology are possible applications (VOSHMGIR, 2016: pp. 21–22; SCHLATT et al., 2016: p. 30).

In the automotive industry, not only could the engineering IT be affected by the Blockchain technology, but also connected cars as well as autonomous vehicles, supply chain and logistics, and leasing, to name a few (FROST & SULLIVAN, 2017: p. 58).

7.4 Ramifications and outlook for the wider social context

The Blockchain technology automizes process steps and tasks which should otherwise have to be done by engineers. Examples for this are redundant documentation, controlling multiple systems for data synchrony, manual alignments with other engineering partners, inter alia. If process steps and tasks are automatized, people have more time for duties which are not fully automated. However, it also could be the case that parts of the workforce can or must be laid-off due to a higher degree of automation¹⁴² (ABELE and REINHART, 2011 according to SPATH, 2013: p. 46).

Prof. Dr. Hans-Jörg Bullinger also highlights that digitalization and interconnectedness in production will generate advantages in efficiency. This might yield to a reduction of employment in industry¹⁴³. Simultaneously, new jobs will emerge from the construction of automatization technologies. However, this cannot be achieved with the workforce released from industry and factories. A potential solution for this issue might be, according to Prof. Dr. Bullinger, extensive qualification of people towards systemic

¹⁴² Depending on the degree of automation, labor costs, and other key factors, jobs could also increase in the manufacturing industry (ABELE and REINHART, 2011 according to SPATH, 2013: p. 46)

¹⁴³ It is not yet clear whether substitution or complementary effects of digitalization on employment will predominate (BULLINGER et al., 2017: p. 112).

knowledge of electronics, mechanics, and IT¹⁴⁴ (BULLINGER et al., 2017: pp. 112–114; GORGS, 2016).

Transferred to the elaboration at hand, for those cases where automatization due to the Blockchain technology might result in the dismissal of engineers, qualifications to these people shall be offered. This training could include software development and coding, particularly for the Blockchain. Thus, engineers could maintain and improve their Blockchain installation, adapt smart contracts, and integrate legacy IT systems.

Hence, the Blockchain technology could not only foster traceability, in some cases, it might also increase efficiency. If so, these employees relieved of inefficient, manual work could obtain further qualifications towards future technologies, such as the Blockchain technology.

¹⁴⁴ For more information about the connection of digitalization and qualification, please refer to DEUTSCHE AKADEMIE DER TECHNIKWISSENSCHAFTEN (2014), SPATH et al. (2017), BULLINGER et al. (2009).

8 Appendix

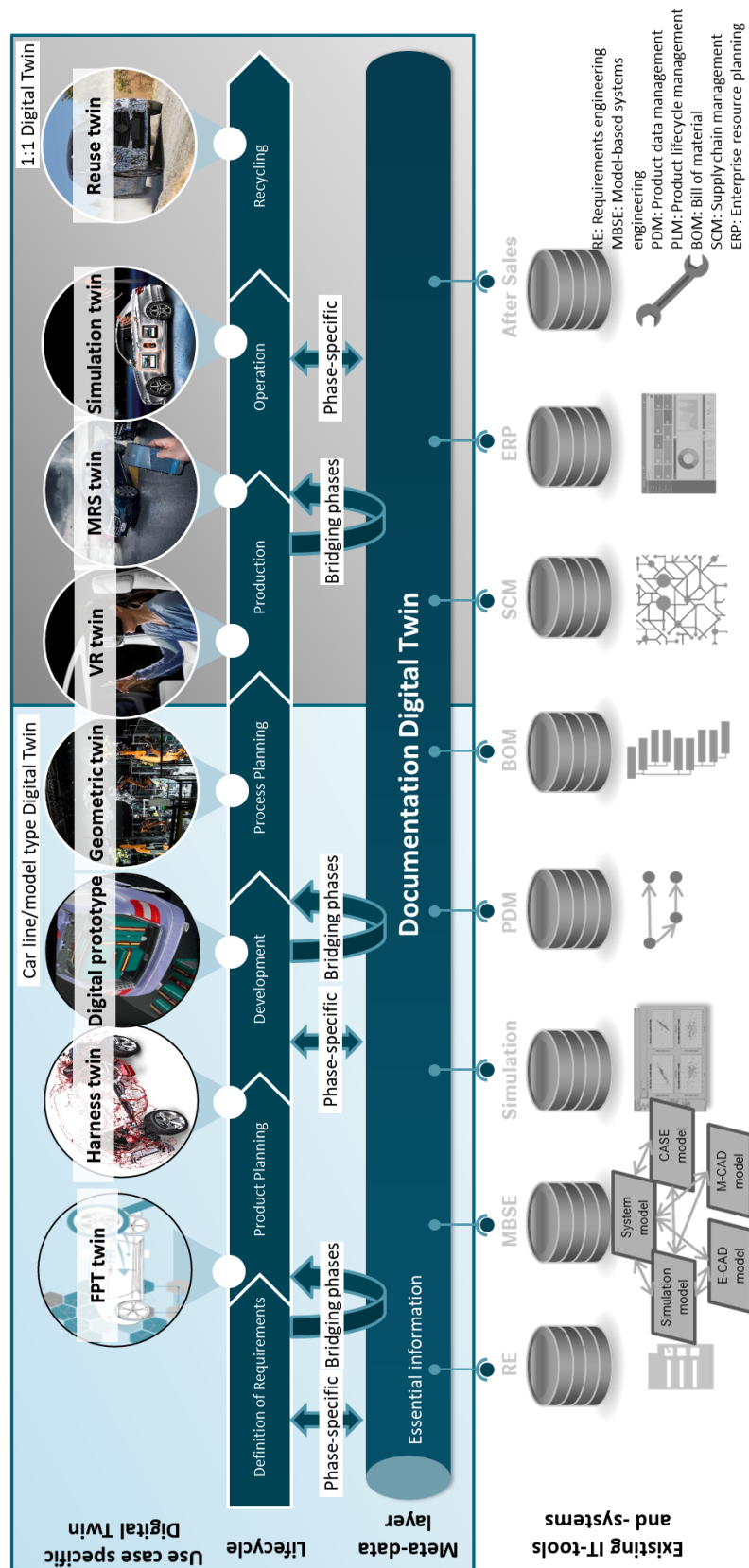


Figure 8-1: Digital Twin for the automotive lifecycle (HEBER et al., 2018).

9 References

- ABELE, E. and G. REINHART (2011): Zukunft der Produktion. Herausforderungen, Forschungsfelder, Chancen. Hanser, München.
- ALBERS, A. and C. ZINGEL (2013): Challenges of Model-Based Systems Engineering: A Study towards Unified Term Understanding and the State of Usage of SysML. In: Abramovici, M. and R. Stark (Eds.): Smart Product Engineering. Proceedings of the 23rd CIRP Design Conference, Bochum, Germany, March 11th - 13th, 2013. Lecture Notes in Production Engineering: pp. 82–92.
- ALT, O. (2012): Modell-basierte Systementwicklung mit SysML. Hanser, München.
- ALVAREZ-RODRÍGUEZ, J. M., J. LLORENS, M. ALEJANDRES and J. FUENTES (2014): Why avoiding how when defining what? Towards an OSLC-based approach to support Model-Driven Requirements Engineering. In: INCOSE International Symposium 24 (1): pp. 990–1005.
- ANTONOPOULOS, A. M. (2015): Mastering Bitcoin. Unlocking digital crypto-currencies. O'Reilly, Sebastopol Calif. u.a.
- ANTONOPOULOS, A. M. (2017): Mastering Bitcoin. Programming the open Blockchain. O'Reilly Media, Sebastopol, CA.
- ARKLEY, P. (2007): Benefits of Traceability in Software Development. Dissertation. School of Computing Science, Newcastle University, Newcastle.
- ATLASSIAN (2020a): DVCS workflows for Bitbucket. In: <https://confluence.atlassian.com/get-started-with-bitbucket/dvcs-workflows-for-bitbucket-860009652.html>. Call: 13.4.2020.
- ATLASSIAN (2020b): Types of version control. In: <https://confluence.atlassian.com/get-started-with-bitbucket/types-of-version-control-856845192.html>. Call: 13.4.2020.
- ATLASSIAN (2020c): Version control software for professional teams. Bitbucket supports Version Control Solutions for Git and Mercurial. In: <https://bitbucket.org/product/version-control-software>. Call: 13.4.2020.
- AUTOSAR (2019): Adaptive platform. In: <https://www.autosar.org/standards/adaptive-platform/>. Call: 30.3.2019.
- AVAK, B. (2006): Variant management of modular product families in the market phase. Dissertation. Zurich, ETH Zurich.
- AXE, D. (2012): Marines' first frontline stealth fighter lacks vital gear. In: <https://www.wired.com/2012/11/marines-jsf/>. Call: 18.9.2018.
- BACHELOR, G. (2011): OSLC PLM Reference Model Release. In: <https://archive.open-services.net/bin/view/Main/PlmHome.html>. Call: 1.8.2020.
- BACK, A., M. CORALLO, L. DASHJR, M. FRIEDENBACH, G. MAXWELL, A. MILLER, A. POELSTRA, J. TIMÓN and P. WUILLE (2014): Enabling Blockchain Innovations with Pegged Sidechains.
- BARAN, P. (1964): On Distributed Communications Networks. In: IEEE Transactions on Communications 12 (1): pp. 1–9.

- BASHIR, I. (2018): Mastering blockchain. Distributed ledger technology, decentralization, and smart contracts explained. Expert insight. Packt Publishing Ltd, Birmingham.
- BECK, K., M. BEEDLE, A. VAN BENNEKUM, A. COCKBURN, W. CUNNINGHAM, M. FOWLER, J. GRENNING, J. HIGHSMITH, A. HUNT, R. JEFFRIES, J. KERN, B. MARICK, R. C. MARTIN, S. MELLOR, K. SCHWABER, J. SUTHERLAND and D. THOMAS (2001): Manifesto for Agile Software Development. Twelve Principles of Agile Software. In: <http://agilemanifesto.org/>. Call: 8.3.2019.
- BECK, T., C. REICHMANN and J. SCHÄUFFELE (2016): E/E-Entwicklung für zukünftige Fahrzeuginnovationen: Ein integrierter, modellbasierter Ansatz. Vector Informatik GmbH. In: https://assets.vector.com/cms/content/know-how/_technical-articles/PREEvision/PREEvision_EE_Development_ATZelektronik_201612_PressArticle_DE.pdf.
- BEECK, M. von der (2007): Development of logical and technical architectures for automotive systems. In: *Software & Systems Modeling* 6 (2): pp. 205–219.
- BEIER, G. (2014): Verwendung von Traceability-Modellen zur Unterstützung der Entwicklung technischer Systeme. Berichte aus dem Produktionstechnischen Zentrum Berlin. Fraunhofer-Verlag, Stuttgart.
- BEIHOFF, B., C. OSTER, S. FRIEDENTHAL and J. WADE (2014): A World in Motion – Systems Engineering Vision 2025. Systems Engineering Vision 2025. INCOSE - International Council on Systems Engineering, San Diego.
- BENDER, K. (Ed.) (2005): Embedded Systems. Qualitätsorientierte Entwicklung. Springer, Berlin.
- BERNERS-LEE, T. (2006): Linked data-design issues. In: <https://www.w3.org/DesignIssues/LinkedData.html>. Call: 30.5.2019.
- BERTSCHE, B., P. GÖHNER, U. JENSEN, W. SCHINKÖTHE and H.-J. WUNDERLICH (2009): Zuverlässigkeit mechatronischer Systeme. Grundlagen und Bewertung in frühen Entwicklungsphasen. VDI-Buch. Springer, Berlin, Heidelberg.
- BEUTNER, E., H. NEUKIRCHNER and G. MAAS (Eds.) (2013): Virtuelle Produktentwicklung. Vogel, Würzburg.
- BIAHMOU, A. (2015a): Sustainable Mobility. In: Stjepandić, J., N. Wognum and W. J. C. Verhagen (Eds.): Concurrent engineering in the 21st century. Foundations, developments and challenges. Springer, Cham: pp. 779–803.
- BIAHMOU, A. (2015b): Systems Engineering. In: Stjepandić, J., N. Wognum and W. J. C. Verhagen (Eds.): Concurrent engineering in the 21st century. Foundations, developments and challenges. Springer, Cham: pp. 221–254.
- BITZER, M., M. EIGNER, K.-G. FAISST, C. MUGGIO and T. EICKHOFF (2018): Framework of the evolution in virtual product modelling and model management towards digitized engineering. In: Maier, A. et al. (Eds.): Product, services and systems design. DS, 87, 3. Curran Associates Inc, Red Hook, NY: pp. 345–354.
- BLESSING, L. T.M. and A. CHAKRABARTI (2009): DRM, a Design Research Methodology. Springer London, London.
- BOEHM, B. W. (1979): Guidelines for Verifying and Validating Software Requirements and Design Specifications. In: Samet, P. A. (Ed.): Euro IFIP 79. North-Holland Publishing Company, Amsterdam.

- BOEHM, B. W. (1988): A Spiral Model of Software Development and Enhancement. In: Computer: pp. 61–72.
- BOHN, J. (2007): User-centric dependability concepts for ubiquitous computing. Zugl.: Zürich, Techn. Hochsch., Diss., 2006. Dissertation.de, Issue 1314. dissertation.de, Berlin.
- BORGEEST, K. (2014): Elektronik in der Fahrzeugtechnik. Springer Fachmedien Wiesbaden, Wiesbaden.
- BORSATO, M. and M. PERUZZINI (2015): Collaborative Engineering. In: Stjepandić, J., N. Wognum and W. J. C. Verhagen (Eds.): Concurrent engineering in the 21st century. Foundations, developments and challenges. Springer, Cham: pp. 165–196.
- BORTZ, J. and N. DÖRING (2006): Forschungsmethoden und Evaluation. Für Human- und Sozialwissenschaftler. Springer-Lehrbuch Bachelor, Master. Springer-Medizin-Verl., Heidelberg.
- BOSCHERT, S. and R. ROSEN (2016): Digital Twin - The Simulation Aspect. In: Hehenberger, P. and D. Bradley (Eds.): Mechatronic Futures. Springer International Publishing, Cham: pp. 59–74.
- BRAUN, A. (2013): Modellbasierte Unterstützung der Produktentwicklung - Potentiale der Modellierung von Produktentstehungsprozessen am Beispiel des integrierten Produktentstehungsmodells (iPeM). Model Based Support of Product Development - Potentials of Modelling Product Engineering Processes using the example of the Integrated Product Engineering Model (iPeM). Forschungsberichte: IPEK, Issue 72. KIT-Bibliothek, Karlsruhe.
- BRETZ, L., C. TSCHIRNER and R. DUMITRESCU (2016): A concept for managing information in early stages of product engineering by integrating MBSE and workflow management systems. In: Institute of Electrical and Electronics Engineers (Ed.): ISSE 2016, Edinburgh, Scotland. 2016 International Symposium on Systems Engineering: George Hotel, October 3-5, 2016: proceedings papers. IEEE, Piscataway, NJ: pp. 1–8.
- BRICOGNE, M., L. RIVEST, N. TROUSSIER and B. EYNARD (2012): Towards PLM for Mechatronics System Design Using Concurrent Software Versioning Principles. In: Rivest, L., A. Bouras and B. Louhichi (Eds.): Product Lifecycle Management. Towards Knowledge-Rich Enterprises. IFIP WG 5.1 International Conference, PLM 2012, Montreal, QC, Canada, July 9-11, 2012, Revised Selected Papers. IFIP Advances in Information and Communication Technology, Issue 388. Springer, Berlin, Heidelberg: pp. 339–348.
- BROODNEY, H., U. SHANI and A. SELA (2013): Model Integration - Extracting Value from MBSE. In: INCOSE International Symposium 23 (1): pp. 1174–1186.
- BROWN, A. W. (2000): Large-scale, component-based development. Object and component technology series. Prentice Hall PTR, Upper Saddle River, NJ.
- BROY, M. (2013): Modellbasiertes Software und Systems Engineering als Element eines durchgängigen Systems Lifecycle Managements (SysLM). In: Sendler, U. (Ed.): Industrie 4.0. Beherrschung der industriellen Komplexität mit SysLM. Xpert.press. Springer Berlin Heidelberg, Berlin, Heidelberg, s.l.: pp. 73–90.
- BRUEGGE, B. and A. H. DUTOIT (2010): Object-oriented software engineering. Using UML, patterns, and Java. Prentice Hall, Boston.

- BRUSA, E., A. CALÀ and D. FERRETTO (2018): Systems engineering and its application to industrial product development. Studies in Systems, Decision and Control, Issue 134. Springer International Publishing, Cham.
- BUCHANAN, R. (1992): Wicked Problems in Design Thinking. In: Design Issues 8 (2): pp. 5–21.
- BUEDE, D. M. (2009): The engineering design of systems. Models and methods. Wiley series in systems engineering and management. John Wiley & Sons, Hoboken, N.J.
- BUHL, H. U., A. HUTHER and B. REITWIESNER (2001): Information Age Economy. 5. Internationale Tagung Wirtschaftsinformatik 2001. Physica-Verlag HD, Heidelberg.
- BULLINGER, H.-J., W. GANZ and J. NEUHÜTTLER (2017): Smart Services – Chancen und Herausforderungen digitalisierter Dienstleistungssysteme für Unternehmen. In: Bruhn, M. and K. Hadwich (Eds.): Dienstleistungen 4.0. Springer Fachmedien Wiesbaden, Wiesbaden: pp. 97–120.
- BULLINGER, H.-J., D. SPATH, H.-J. WARNECKE and E. WESTKÄMPER (2009): Handbuch Unternehmensorganisation. Strategien, Planung, Umsetzung. VDI-Buch. Springer-Verlag, Berlin, Heidelberg.
- BURMANN, C. and R. KOTHES (2014): Variantenvielfalt und Intramarkenimagekonfusion. LiM-Arbeitspapiere No. 54. Universität Bremen, Lehrstuhl für innovatives Markenmanagement, Bremen. In: http://www.lim.uni-bremen.de/files/burmann/publikationen/AP%2054_Variantenvielfalt%20und%20Intramarkenimagekonfusion_1.pdf. Call: 21.8.2018.
- BURR, H. (2008): Informationsmanagement an der Schnittstelle zwischen Entwicklung und Produktionsplanung im Karosserierohbau. Dissertation. Lehrstuhl für Konstruktionstechnik/CAD, Universität des Saarlandes, Saarbrücken.
- CADET, M. and H. MEISSNER (2017): Cybertronische Systeme. In: Eigner, M., W. Koch and C. Muggeo (Eds.): Modellbasierter Entwicklungsprozess cybertronischer Systeme. Der PLM-unterstützte Referenzentwicklungsprozess für Produkte und Produktionssysteme. Springer Vieweg, Berlin: pp. 19–22.
- CHACON, S. and B. STRAUB (2014): Pro Git. Everything you need to know about Git. Books for professionals by professionals. Apress/Springer, New York, NY.
- CHADZYNSKI, P. Z. (2022a): Incorporating MBSE Across a PLM Managed Digital Thread - Three Real Life Use Cases. In: ISCIE (The Institute of Systems, Control and Information Engineers) 66 (8): pp. 309–314.
- CHADZYNSKI, P. Z. (2022b): Is Variant Management Your Achilles Heel? It's All About Systems Thinking. In: <https://community.aras.com/b/english/posts/is-variant-management-your-achilles-heel-it-s-all-about-systems-thinking>. Call: 6.11.2022.
- CHURCHMAN, C. W. (1967): Guest Editorial: Wicked Problems. In: Management Science 14 (4): B141–42.
- COMERFORD, R. (1994): Mecha...what? [mechatronics]. In: IEEE Spectrum 31 (8): pp. 46–49.
- CONTINENTAL AG (2020): Start in die Elektro-Ära: Neues E-Modell VW ID.3 fährt mit Technologien von Continental. In: <https://www.continental.com/de/presse/pressemitteilungen/volkswagen-id3>. Call: 21.7.2021.

- CONTINENTAL AG (2021): More Intelligence with Server-based E/E Architectures. Continental's Body High Performance Computer. In: <https://www.continental-automotive.com/en-gl/Passenger-Cars/Vehicle-Networking/Body-High-Performance-Computer>. Call: 21.7.2021.
- CROSS, N. (2006): *Designerly Ways of Knowing*. Springer, London.
- CROSSLEY, N. (2019): OSLC Configuration Management 1.0 Part 4: RDF Vocabulary. Working Draft 01. In: <https://oslc-op.github.io/oslc-specs/specs/config/config-vocab.html>.
- DAENZER, W. F. and F. HUBER (Eds.) (2002): *Systems Engineering. Methodik und Praxis*. Verl. Industrielle Organisation, Zürich.
- DAIMLER AG (2017): *Annual Report 2017*. Daimler AG, Stuttgart.
- DAIMLER AG (2018): Future mobility. Bosch and Daimler join forces to work on fully automated, driverless system. In: <https://www.daimler.com/innovation/case/autonomous/bosch-cooperation.html>. Call: 21.4.2019.
- DAIMLER AG (2020): Mercedes-Benz and NVIDIA to Build Software-Defined Computing Architecture for Automated Driving Across Future Fleet. In: <https://media.daimler.com/marsMediaSite/en/instance/ko/Mercedes-Benz-and-NVIDIA-to-Build-Software-Defined-Computing-Architecture-for-Automated-Driving-Across-Future-Fleet.xhtml?oid=46665504>. Call: 21.7.2021.
- DALGARNO, M. and D. BEUCHE (2007): *Variant Management*. 3rd British Computer Society Configuration Management Specialist Group Conference, Oxford.
- DASSAULT SYSTÈMES (2014): Dassault Systèmes' 3DEXPERIENCE Platform Based on V6 Architecture Empowers Leading Companies Worldwide. In: <https://www.3ds.com/press-releases/single/dassault-systemes-3dexperience-platform-based-on-v6-architecture-empowers-leading-companies-wo/>. Call: 7.4.2020.
- DASSAULT SYSTÈMES (2018a): 3DEXPERIENCE for software lifecycle management (SWLM). A federated platform for software driven innovation. High-Tech.
- DASSAULT SYSTÈMES (2018b): 3DEXPERIENCE platform for intelligent connected systems. Imagine, engineer and experience smart products and systems.
- DASSAULT SYSTÈMES (2020): 3DEXPERIENCE platform. Creating sustainable innovation. In: <https://www.3ds.com/products-services/3dexperience/>. Call: 7.4.2020.
- DATE, C. J. (1990): What is a Distributed Database. In: Warden, A. and C. J. Date (Eds.): *Relational database. Writings 1985-1989*. Addison-Wesley, Reading.
- DAUN, M., P. BOHN, J. BRINGS and T. WEYER (2016): *Structured Model-Based Engineering of Long-living Embedded Systems: The SPES Methodological Building Blocks Framework*. Software Engineering 2016 - Software engineering für smart cities, Wien.
- DEBBABI, M., F. HASSAÏNE, Y. JARRAYA, A. SOEANU and L. ALAWNEH (2010): *Verification and Validation in Systems Engineering*. Springer Berlin Heidelberg, Berlin, Heidelberg.

- DEUTSCHE AKADEMIE DER TECHNIKWISSENSCHAFTEN (2014): Smart service Welt. Umsetzungsempfehlungen für das Zukunftsprojekt Internetbasierte Dienste für die Wirtschaft. acatech, Berlin.
- DEUTSCHES INSTITUT FÜR NORMUNG E. V. (2018a): Code of PLM Openness (CPO) - Certification Handbook ICS 03.120.20; 35.240.50 (DIN SPEC 91372-2). Beuth Verlag GmbH, Berlin.
- DEUTSCHES INSTITUT FÜR NORMUNG E. V. (2018b): Code of PLM Openness (CPO) - IT Openness Criteria ICS 35.240.50 (DIN SPEC 91372-1). Beuth Verlag GmbH, Berlin.
- DEUTSCHES INSTITUT FÜR NORMUNG E. V. (2018c): Code of PLM Openness (CPO) - Mapping of CPO criteria to assessment criteria ICS 35.240.50 (DIN SPEC 91372-3). Beuth Verlag GmbH, Berlin.
- DEUTSCHES INSTITUT FÜR NORMUNG E. V. (2018d): Terminology for blockchains (DIN SPEC 16597) ICS 01.040.35; 35.240.99 (DIN SPEC 16597). Berlin.
- DHILLON, V., D. METCALF and M. HOOPER (2017): Blockchain Enabled Applications. Understand the Blockchain Ecosystem and How to Make It Work for You. Apress L. P, Berkeley, CA.
- DICKOPF, T., T. SCHULTE and M. SCHNEIDER (2017): Analyse existierender SysML-basierter Ansätze aus Industrie und Forschung. In: Eigner, M., W. Koch and C. Muggeo (Eds.): Modellbasierter Entwicklungsprozess cybertronischer Systeme. Der PLM-unterstützte Referenzentwicklungsprozess für Produkte und Produktionssysteme. Springer Vieweg, Berlin: pp. 65–72.
- DORI, D. and E. CRAWLEY (2016): Model-based systems engineering with OPM and SysML. Springer, New York.
- DÖRN, S. (2018): Programmieren für Ingenieure und Naturwissenschaftler. Intelligente Algorithmen und digitale Technologien. eXamen.press. Springer Vieweg, Berlin.
- DORSCHER, J. (2015): Praxishandbuch Big Data. Springer Fachmedien Wiesbaden, Wiesbaden.
- DUMITRESCU, C., P. TESSIER, C. SALINESI, S. GÉRARD, A. DAURON and R. MAZO (2014): Capturing Variability in Model Based Systems Engineering. In: Aiguier, M. et al. (Eds.): Complex Systems Design & Management. Springer International Publishing, Cham: pp. 125–139.
- EBELING, R. and M. EIGNER (2018): OSLC based approach for product appearance structuring. In: Maier, A. et al. (Eds.): Product, services and systems design. DS, 87, 3. Curran Associates Inc, Red Hook, NY: pp. 259–266.
- ECKERT, C. M., P. J. CLARKSON and M. K. STACEY (2003): The spiral of applied research: A methodological view on integrated design research. In: Folkesson, A. et al. (Eds.): Research for practice - innovation in products, processes and organisations. ICED 03, 14th International Conference on Engineering Design; 19 - 21 August 2003, The Royal Institute of Technology, Stockholm. DS / Design Society, Issue 31. Design Society, Glasgow: pp. 245–257.
- ECLIPSE FOUNDATION, I. (2020): Equivalences and differences between SysML and Arcadia/Capella. In: https://www.eclipse.org/capella/arcadia_capella_sysml_tool.html. Call: 8.4.2020.

- EHRENSPIEL, K., A. KIEWERT, U. LINDEMANN and M. MÖRTL (2014): Kostengünstig Entwickeln und Konstruieren. Kostenmanagement bei der integrierten Produktentwicklung. VDI-Buch. Springer Vieweg, Berlin.
- EHRENSPIEL, K. and H. MEERKAMM (2017): Integrierte Produktentwicklung. Denkabläufe, Methodeneinsatz, Zusammenarbeit. Carl Hanser Verlag, München.
- EIGNER, M. (2014a): Einleitung - Modellbasierte virtuelle Produktentwicklung. In: Eigner, M., D. Roubanov and R. Zafirov (Eds.): Modellbasierte virtuelle Produktentwicklung. Springer Vieweg, Berlin: pp. 1–13.
- EIGNER, M. (2014b): Product Lifecycle Management (PLM). In: Eigner, M., D. Roubanov and R. Zafirov (Eds.): Modellbasierte virtuelle Produktentwicklung. Springer Vieweg, Berlin: pp. 267–300.
- EIGNER, M. (2014c): Technische Organisation des Produktentwicklungsprozesses. In: Eigner, M., D. Roubanov and R. Zafirov (Eds.): Modellbasierte virtuelle Produktentwicklung. Springer Vieweg, Berlin: pp. 227–266.
- EIGNER, M. (2014d): Überblick Disziplin-spezifische und -übergreifende Vorgehensmodelle. In: Eigner, M., D. Roubanov and R. Zafirov (Eds.): Modellbasierte virtuelle Produktentwicklung. Springer Vieweg, Berlin: pp. 15–52.
- EIGNER, M., T. DICKOPF, M. SCHNEIDER and T. SCHULTE (2018): MECPRO² - A holistic concept for the model-based development of cybertronic systems. In: Maier, A. et al. (Eds.): Product, services and systems design. DS, 87, 3. Curran Associates Inc, Red Hook, NY: pp. 379–388.
- EIGNER, M., T. DICKOPF, T. SCHULTE and M. SCHNEIDER (2016a): mecPro² - Entwurf einer Beschreibungssystematik zur Entwicklung cybertronischer Systeme mit SysML. In: Schulze, S.-O., C. Muggeo and J. Abulawi (Eds.): Tag des Systems Engineering. Ulm, 11.-13. November 2015. Hanser, München: pp. 163–172.
- EIGNER, M., K.-G. FAISST, T. EICKHOFF, A. EIDEN and C. MUGGEO (2016b): Der Engineering Backbone für ein interdisziplinäres Product Development und Life Cycle Management. System Lifecycle Management. In: ProduktDatenJournal 23 (2): pp. 56–61.
- EIGNER, M., F. GERHARDT, T. GILZ and F. MOGO NEM (Eds.) (2012a): Informationstechnologie für Ingenieure. SpringerLink Bücher. Springer Vieweg, Berlin, Heidelberg.
- EIGNER, M., T. GILZ and R. ZAFIROV (2012b): Proposal for functional product description as part of a PLM solution in interdisciplinary product development. In: Marjanović, D. (Ed.): DESIGN 2012. Proceedings of the 12th International Design Conference, May 21 - 24, 2012, Dubrovnik, Croatia. DS, Issue 70. Fac. of Mechanical Engineering and Naval Architecture, Zagreb: pp. 1667–1676.
- EIGNER, M., W. KOCH and C. MUGGEO (Eds.) (2017): Modellbasierter Entwicklungsprozess cybertronischer Systeme. Der PLM-unterstützte Referenzentwicklungsprozess für Produkte und Produktionssysteme. Springer Vieweg, Berlin.
- EIGNER, M., D. ROUBANOV and R. ZAFIROV (Eds.) (2014): Modellbasierte virtuelle Produktentwicklung. Springer Vieweg, Berlin.
- EIGNER, M. and R. STELZER (2009): Product Lifecycle Management. Ein Leitfaden für Product Development und Life Cycle Management. VDI. Springer, Dordrecht.

- ELMARAGHY, H., G. SCHUH, W. ELMARAGHY, F. PILLER, P. SCHÖNSLEBEN, M. TSENG and A. BERNARD (2013): Product variety management. In: CIRP Annals 62 (2): pp. 629–652.
- ERCIYES, K. (2013): Distributed Graph Algorithms for Computer Networks. Computer Communications and Networks. Springer London, London.
- ESTEFAN, J. A. (2008): Survey of Model-Based Systems Engineering (MBSE) Methodologies. INCOSE-TD-2007-003-01. International Council on Systems Engineering, Seattle, WA.
- EVERSHEIM, W. and G. SCHUH (Eds.) (2005): Integrierte Produkt- und Prozessgestaltung. VDI. Springer, Berlin.
- FELDHUSEN, J. and B. GEBHARDT (2008): Product Lifecycle Management für die Praxis. Ein Leitfaden zur modularen Einführung, Umsetzung und Anwendung. Springer-Verlag Berlin Heidelberg, Berlin, Heidelberg.
- FELDHUSEN, J. and K.-H. GROTE (Eds.) (2013): Pahl/Beitz Konstruktionslehre. Methoden und Anwendung erfolgreicher Produktentwicklung.
- FELDHUSEN, J., K.-H. GROTE and J. THON (2013): Grundsätzliche Überlegungen zur Rationalisierung. Der agile Entwicklungs- und Konstruktionsprozess. In: Feldhusen, J. and K.-H. Grote (Eds.): Pahl/Beitz Konstruktionslehre. Methoden und Anwendung erfolgreicher Produktentwicklung: pp. 773–815.
- FERREIRA, F., J. FARIA, A. AZEVEDO and A. L. MARQUES (2017): Product lifecycle management in knowledge intensive collaborative environments. An application to automotive industry. In: International Journal of Information Management 37 (1): pp. 1474–1487.
- FIGGE, A. (2014): Effiziente Erfassung und Pflege von Traceability-Modellen zur Entwicklung technischer Systeme. Berichte aus dem Produktionstechnischen Zentrum Berlin. Fraunhofer Verlag, Stuttgart.
- FISHER, A., M. NOLAN, S. FRIEDENTHAL, M. LOEFFLER, M. SAMPSON, M. BAJAJ, L. VANZANDT, K. HOVEY, J. PALMER and L. HART (2014): Model Lifecycle Management for MBSE. In: INCOSE International Symposium 24 (1): pp. 207–229.
- FRANCO, P. (2015): Understanding Bitcoin. Cryptography, Engineering and Economics. The Wiley Finance Series. Wiley, Chichester, West Sussex.
- FRIEDENTHAL, S., A. MOORE and R. STEINER (2012): A practical guide to SysML. The systems modeling language. Morgan Kaufmann, Amsterdam.
- FROST & SULLIVAN (2017): Blockchain Technology Revolutionizing Automotive Industry. Automotive Ecosystem Participants to Spend ~0.6% of their Total IT Spend on Blockchain by 2025. K13A-18.
- FROST & SULLIVAN (2018): Automotive ECUs for ADAS and Autonomous Driving Systems, North America and Europe, 2017. E/E Architecture Will Transform from Multiple ECUs to 5 Domain Controllers. MD81-18.
- FRØYSTAD, P. and J. HOLM (2015): Blockchain: Powering the Internet of Value. Evry, Fornebu. In: <https://www.evry.com/globalassets/insight/bank2020/bank-2020---blockchain-powering-the-internet-of-value---whitepaper.pdf>. Call: 20.8.2018.
- GAUSEMEIER, J., A. TRÄCHTLER, W. SCHÄFER and H. ANACKER (2014): Semantische Technologien im Entwurf mechatronischer Systeme. Effektiver Austausch von Lösungswissen in Branchenwertschöpfungsketten. Hanser, München.

- GAVER, W. (2012): What should we expect from research through design? In: Association for Computing Machinery, New York, NY, United States (Ed.): CHI 2012, it's the experience. The 30th ACM Conference on Human Factors in Computing Systems; Austin, Texas, USA, May 5 - 10, 2012. ACM, New York, NY: pp. 937–946.
- GERICKE, K., A. J. QURESHI and L. BLESSING (2013): Analyzing Transdisciplinary Design Processes in Industry. An Overview. In: ASME (Ed.): 25th International Conference on Design Theory and Methodology; ASME 2013 Power Transmission and Gearing Conference. ASME: V005T06A031.
- GIFT, N. and A. SHAND (2009): Introduction to distributed version control systems. Learn about and compare how to use Bazaar, Mercurial, and Git. IBM.
- GILZ, T. (2014): PLM-Integrated Interdisciplinary System Models in the Conceptual Design Phase Based on Model-Based Systems Engineering. Dissertation. Maschinenbau und Verfahrenstechnik, Technische Universität Kaiserslautern, Kaiserslautern.
- GIT (2020a): About - Small and Fast. In: <https://git-scm.com/about/small-and-fast>. Call: 13.4.2020.
- GIT (2020b): git - everything is local. In: <https://git-scm.com/>. Call: 13.4.2020.
- GLASER, F. (2017): Pervasive Decentralisation of Digital Infrastructures: A Framework for Blockchain enabled System and Use Case Analysis. In: Hawaii International Conference on System Sciences 2017 (HICSS-50).
- GLASNER, J. (2018): Automakers Pump Record Sums Into Startups. In: <https://news.crunchbase.com/news/automakers-pump-record-sums-into-startups/>. Call: 22.4.2019.
- GORGS, C. (2016): Digitalisierung gibt den Menschen mehr Freiheiten - Interview mit Prof. Dr. Hans-Jörg Bullinger. In: <https://www.manager-magazin.de/unternehmen/artikel/hans-joerg-bullinger-ueber-chancen-der-digitalisierung-a-1118356.html>. Call: 29.8.2021.
- GOTEL, O., J. CLELAND-HUANG, J. H. HAYES, A. ZISMAN, A. EGYED, P. GRÜNBACHER, A. DEKHTYAR, G. ANTONIOL, J. MALETIC and P. MÄDER (2012): Traceability Fundamentals. In: Cleland-Huang, J., O. Gotel and A. Zisman (Eds.): Software and Systems Traceability. Springer London, London: pp. 3–22.
- GOVERNMENT OFFICE FOR SCIENCE (2016): Distributed Ledger Technology: beyond block chain. UK Government Chief Scientific Adviser, London.
- GRANDE, M. (2013): 100 Minuten für Konfigurationsmanagement. Kompaktes Wissen nicht nur für Projektleiter und Entwickler.
- GRIEVES, M. W. (2012): Virtually Indistinguishable. Systems Engineering and PLM. In: Rivest, L., A. Bouras and B. Louhichi (Eds.): Product Lifecycle Management. Towards Knowledge-Rich Enterprises. IFIP WG 5.1 International Conference, PLM 2012, Montreal, QC, Canada, July 9-11, 2012, Revised Selected Papers. IFIP Advances in Information and Communication Technology, Issue 388. Springer, Berlin, Heidelberg: pp. 226–242.
- GROLL, M. W. (2008): Interconnection Based Product and Process Documentation. Dissertation. University of Twente, Twente.
- GROLL, M. W. and D. T. HEBER (2016): E/E-Product Data Management in Consideration of Model-Based Systems Engineering. In: Borsato, M. et al. (Eds.): Transdisciplinary

- engineering. Crossing boundaries: proceedings of the 23rd ISPE Inc. International Conference on Transdisciplinary Engineering, October 3-7, 2016. Advances in transdisciplinary engineering, Volume 4. IOS Press, Amsterdam: pp. 289–298.
- GRUNDEL, M., J. ABULAWI, G. MOESER, T. WEILKIENS, A. SCHEITHAUER, S. KLEINER, C. KRAMER, M. NEUBERT, S. KÜMPEL and A. ALBERS (2014): FAS4M – No more: “Please mind the gap!”. In: Maurer, M., S.-O. Schulze and J. Abulawi (Eds.): Tag des Systems Engineering 2014. Bremen, 12. - 14. November 2014. Hanser, München: pp. 63–74.
- HABERFELLNER, R. (Ed.) (2012): Systems Engineering. Grundlagen und Anwendung. Orell Füssli, Zürich.
- HALVORSON, B. (2016): Software Now To Blame For 15 Percent Of Car Recalls. In: <https://www.popsci.com/software-rising-cause-car-recalls>. Call: 18.09.2018.
- HARASHIMA, F., M. TOMIZUKA and T. FUKUDA (1996): Mechatronics - "What Is It, Why, and How?" An editorial. In: IEEE/ASME Transactions on Mechatronics 1 (1): pp. 1–4.
- HARMS, E. (2009): Änderungs- und Konfigurationsmanagement unter Berücksichtigung von Verwendungsinstanzen. Arbeitsmethoden für integrierte Produktmodelle im Rahmen des Produkt-Lebenszyklus-Managements der Automobilindustrie. Dissertation. Fakultät für Maschinenbau, Universität Karlsruhe, Karlsruhe.
- HASS, A. M. J. (2003): Configuration management principles and practice. The Agile software development series. Addison-Wesley, Boston, MA.
- HAUSMANN, K. (2010): Perimeter: Performanzmessung in der Produktentwicklung. Performanzmessung in der Produktentwicklung auf Basis semantisch integrierter Produktmodelle. Suedwestdeutscher Verlag fuer Hochschulschriften, Saarbrücken.
- HEBER, D. T. and M. W. GROLL (2018a): A Meta-Model to Connect Model-based Systems Engineering with Product Data Management by Dint of the Blockchain. In: IEEE TEMS (Ed.): 2018 International Conference on Intelligent Systems (IS). IEEE: pp. 280–287.
- HEBER, D. T. and M. W. GROLL (2018b): How the Blockchain fosters E/E traceability for MBSE and PLM in distributed engineering collaboration. In: IEEE TEMS (Ed.): 2018 IEEE International Conference on Technology Management, Operations and Decisions (ICTMOD). IEEE: pp. 125–130.
- HEBER, D. T. and M. W. GROLL (2018c): Towards a digital twin: How the blockchain can foster E/E-traceability in consideration of model-based systems engineering. In: Maier, A. et al. (Eds.): Product, services and systems design. DS, 87, 3. Curran Associates Inc, Red Hook, NY: pp. 321–330.
- HEBER, D. T., F. MICHELBACH, F. S. MORELLI and M. W. GROLL (2018): Digital Twin-Konzeption in der Automobilindustrie: Einsatzpotenziale der Blockchain-Technologie. In: Anwendungen und Konzepte der Wirtschaftsinformatik (8): pp. 7–19.
- HECKMANN, O., R. STEINMETZ, N. LIEBAU, A. BUCHMANN, C. ECKERT, J. KANGASHARJU, M. MÜHLHÄUSER and A. SCHÜRR (2006): Qualitätsmerkmale von Peer-to-Peer Systemen. Technical Report. Technische Universität Darmstadt, Darmstadt. In: <ftp://ftp.kom.e-technik.tu-darmstadt.de/papers/HSL+06-1-paper.pdf>.
- HEIHOFF-SCHWEDE, J., C. BREMER, M. RABE and C. TSCHIRNER (2017): Werkzeuge für den Mittelstand – MBSE leicht. In: Schulze, S.-O., C. Tschirner and R. Kaffenberger

- (Eds.): Tag des Systems Engineering. Herzogenaurach, 25.-27. Oktober 2016. Hanser, München: pp. 35–44.
- HENSELER, J. (2015): Is the whole more than the sum of its parts? On the interplay of marketing and design research. Inaugural lecture. University of Twente, Twente.
- HENSELER, J. (2017): Bridging Design and Behavioral Research With Variance-Based Structural Equation Modeling. In: *Journal of Advertising* 46 (1): pp. 178–192.
- HENSELER, J. (2021): Composite-based structural equation modeling. Analyzing latent and emergent variables. *Methodology in the social sciences*. The Guilford Press, a division of Guilford Publications, Inc, New York, NY.
- HERMAN, I., G. MELANCON and M. S. MARSHALL (2000): Graph visualization and navigation in information visualization. A survey. In: *IEEE Transactions on Visualization and Computer Graphics* 6 (1): pp. 24–43.
- HEVNER, A. and S. CHATTERJEE (2010): *Design Research in Information Systems*, Issue 22. Springer US, Boston, MA.
- HEYN, M. (1999): *Methodik zur schnittstellenorientierten Gestaltung von Entwicklungskooperationen*. Shaker Verlag, Aachen.
- HILEMAN, G. and M. RAUCHS (2017): *Global Blockchain benchmarking study*. Cambridge Centre for Alternative Finance, Cambridge, UK.
- HITZLER, P. (2008): *Semantic Web. Grundlagen*. eXamen.press. Springer-Verlag Berlin Heidelberg, Berlin, Heidelberg.
- HOFFMANN, H.-P. (2011): *Systems Engineering Best Practices with the Rational Solution for Systems and Software Engineering - Deskbook*. Model-Based Systems Engineering with Rational Rhapsody and rational Harmony for Systems Engineering. IBM Corporation.
- HOLLAND-LETZ, D., M. KÄSSER, B. KLOSS and T. MÜLLER (2019): Start me up: Where mobility investments are going. Automotive & Assembly. In: <https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/start-me-up-where-mobility-investments-are-going>. Call: 22.4.2019.
- HOLT, J. and S. PERRY (2008): *SysML for systems engineering*. Professional applications of computing series. Institution of Engineering and Technology, London.
- HOOSMAND, Y. (2015): *Transparenzerhöhung bei der Entwicklung von individualisierten Produkten in der Einzelfertigung*. Ingenieurwissenschaften. Dr. Hut, München.
- HOOSMAND, Y., D. ADAMENKO, S. KUNNEN and P. KÖHLER (2018): An approach for holistic model-based engineering of industrial plants. In: Maier, A. et al. (Eds.): *Product, services and systems design*. DS, 87, 3. Curran Associates Inc, Red Hook, NY: pp. 101–110.
- HOOSMAND, Y., M. HÖNER, S. DANJOU and P. KÖHLER (2016): Ein integriertes Gesamtsystemmodell für die modellbasierte Entwicklung. In: Krause, D., K. Paetzold and S. Wartzack (Eds.): *Design for X - Beiträge zum 27. DfX-Symposium Oktober 2016*. TuTech Verlag TuTech Innovation GmbH, Hamburg: pp. 243–254.
- HORVATH, L. (2017): New method for enhanced driving of entity generation in RFLP structured product model. In: Institute of Electrical and Electronics Engineers (Ed.): *Proceedings of the 2017 12th IEEE Conference on Industrial Electronics and Applications (ICIEA)*. 18-20 June 2017, Siem Reap, Cambodia. IEEE, Piscataway, NJ: pp. 541–546.

- HORVATH, L., I. J. RUDAS and M. TAKACS (2015): Content representation structure for product system modeling. In: Szakál, A. (Ed.): INES 2015. IEEE 19th International Conference on Intelligent Engineering Systems: September 3-5, 2015, Bratislava, Slovakia: proceedings. IEEE, Piscataway, NJ: pp. 85–90.
- HORVÁTH, L. and I. J. RUDAS (2015): Intelligent Content for Product Definition in RFLP Structure. In: Fujita, H. and Selamat A. (Eds.): Intelligent software methodologies, tools and techniques. 13th international conference, SoMeT 2014, Langkawi, Malaysia, September 22-24, 2014; revised selected papers. Communications in Computer and Information Science, Issue 513. Springer, Cham: pp. 55–70.
- HYPERLEDGER (2020): Adding an Org to a Channel. In: https://hyperledger-fabric.readthedocs.io/en/release-2.2/channel_update_tutorial.html. Call: 19.4.2021.
- IBM CORPORATION (2020a): IBM Engineering Systems Design Rhapsody. In: <https://www.ibm.com/us-en/marketplace/systems-design-rhapsody>. Call: 9.4.2020.
- IBM CORPORATION (2020b): IBM Engineering Systems Design Rhapsody - Model Manager. In: <https://jazz.net/products/rhapsody-model-manager/>. Call: 9.4.2020.
- IBM KNOWLEDGE CENTER (2020): IBM Maximo Asset Management Multitenancy 7.6. Specification of OSLC resources. In: https://www.ibm.com/support/knowledgecenter/de/SSLKT6_7.6.0/com.ibm.mt.doc/gp_intfrmwk/oslc/c_oslc_res_spec.html. Call: 5.12.2020.
- IEEE COMPUTER SOCIETY (2007): Systems engineering - Application and management of the systems engineering process (ISO/IEC 26702, IEEE 1220-2005). IEEE Computer Society Press, New York, NY.
- INSTITUTE OF CONFIGURATION MANAGEMENT AND CMII RESEARCH INSTITUTE (2014): CMII Standard for Enterprise-Wide Configuration Management and Integrated Process Excellence (CMII-100H).
- INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS (1983): IEEE Standard Glossary of Software Engineering Terminology (729). IEEE, Piscataway, NJ, USA. In: <https://standards.ieee.org/standard/729-1983.html>.
- INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS (2012): IEEE Standard for Configuration Management in Systems and Software Engineering (828-2012). IEEE, Piscataway, NJ, USA.
- INTERNATIONAL ORGANIZATION FOR STANDARDIZATION (2003): Information technology - Reference Model of Data Management (ISO/IEC TR 10032:2003). Geneva. In: <https://www.iso.org/standard/38607.html>.
- INTERNATIONAL ORGANIZATION FOR STANDARDIZATION (2011a): Industrial automation systems and integration - Product data representation and exchange - Part 1251: Application module: Interface (ISO/TS 10303-1251:2011). Geneva. In: <https://www.iso.org/standard/60631.html>.
- INTERNATIONAL ORGANIZATION FOR STANDARDIZATION (2011b): Road vehicles - Functional safety - Part 1: Vocabulary (ISO 26262-1:2011). Geneva. In: <https://www.iso.org/standard/43464.html>.
- INTERNATIONAL ORGANIZATION FOR STANDARDIZATION (2011c): Road vehicles - Functional safety - Part 5: Product development at the hardware level (ISO 26262-5:2011). Geneva. In: <https://www.iso.org/standard/43464.html>.

- INTERNATIONAL ORGANIZATION FOR STANDARDIZATION (2011d): Road vehicles - Functional safety - Part 6: Product development at the software level (ISO 26262-6:2011). Geneva. In: <https://www.iso.org/standard/43464.html>.
- INTERNATIONAL ORGANIZATION FOR STANDARDIZATION (2011e): Road vehicles - Functional safety - Part 8: Supporting processes (ISO 26262-8:2011). Geneva. In: <https://www.iso.org/standard/43464.html>.
- INTERNATIONAL ORGANIZATION FOR STANDARDIZATION (2011f): Systems and software engineering - Architecture description (ISO/IEC/IEEE 42010:2011). Geneva. In: <https://www.iso.org/standard/50508.html>.
- INTERNATIONAL ORGANIZATION FOR STANDARDIZATION (2012a): Industrial automation systems and integration — Product data representation and exchange Part 233: Application protocol: Systems engineering (ISO 10303-233:2012(E)). Geneva.
- INTERNATIONAL ORGANIZATION FOR STANDARDIZATION (2012b): Technical product documentation – Vocabulary - Terms relating to technical drawings, product definition and related documentation (ISO 10209:2012). Geneva. In: <https://www.iso.org/standard/51183.html>.
- INTERNATIONAL ORGANIZATION FOR STANDARDIZATION (2014): Industrial automation systems and integration — Product data representation and exchange Part 242: Application protocol: Managed model-based 3D engineering (ISO 10303-242:2014(E)). Geneva.
- INTERNATIONAL ORGANIZATION FOR STANDARDIZATION (2015a): Quality management systems - Requirements (DIN EN ISO 9001:2015). Geneva. In: <https://www.iso.org/standard/62085.html>.
- INTERNATIONAL ORGANIZATION FOR STANDARDIZATION (2015b): Software and systems engineering - Reference model for product line engineering and management (ISO/IEC 26550). Geneva. In: <https://www.iso.org/standard/69529.html>.
- INTERNATIONAL ORGANIZATION FOR STANDARDIZATION (2015c): Systems and software engineering - System life cycle processes (ISO/IEC/IEEE 15288:2015). Geneva. In: <https://www.iso.org/standard/63711.html>.
- INTERNATIONAL ORGANIZATION FOR STANDARDIZATION (2017a): Quality management - Guidelines for configuration management (ISO 10007:2017). Geneva. In: <https://www.iso.org/standard/70400.html>.
- INTERNATIONAL ORGANIZATION FOR STANDARDIZATION (2017b): Systems and software engineering - Vocabulary (ISO/IEC/IEEE 24765:2017(E)). Geneva.
- IWANEK, P., L. KAISER, R. DUMITRESCU and A. NYBEN (2013): Fachdisziplinübergreifende Systemmodellierung mechatronischer Systeme mit SysML und CONSENS. In: Maurer, M. and S.-O. Schulze (Eds.): Tag des Systems Engineering. Carl Hanser Verlag GmbH & Co. KG, München: pp. 337–346.
- JOHNSIRANI, B. and M. NATARAJAN (2015): An Overview of Distributed Database Management System. In: International Journal of Trend in Research and Development (IJTRD) 2 (5): pp. 118–121.
- JOHNSON, D. and S. SPEICHER (2013): Open Services for Lifecycle Collaboration Core Specification Version 2.0. In: <https://archive.open-services.net/bin/view/Main/OslcCoreSpecification.html>. Call: 3.8.2020.

- KAAS, H.-W., D. MOHR, P. GAO, N. MÜLLER, D. WEE, R. HENSLEY, M. GUAN, T. MÖLLER, G. ECKHARD, G. BRAY, S. BEIKER, A. BROTSCHI and D. KOHLER (2016): Automotive revolution – perspective towards 2030. How the convergence of disruptive technology-driven trends could transform the auto industry. Advanced Industries. In: <https://www.mckinsey.com/~media/mckinsey/industries/high%20tech/our%20insights/disruptive%20trends%20that%20will%20transform%20the%20auto%20industry/auto%202030%20report%20jan%202016.ashx>. Call: 22.4.2019.
- KÄSSER, M., T. MÜLLER and A. TSCHIESNER (2017): Analyzing start-up and investment trends in the mobility ecosystem. How can companies identify and source the technologies that will be critical for crafting a strategy to keep up in the shifting automotive landscape? In: <https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/analyzing-start-up-and-investment-trends-in-the-mobility-ecosystem>. Call: 22.4.2019.
- KATZENBACH, A. (2015a): Automotive. In: Stjepandić, J., N. Wognum and W. J. C. Verhagen (Eds.): Concurrent engineering in the 21st century. Foundations, developments and challenges. Springer, Cham: pp. 607–638.
- KATZENBACH, A. (2015b): Informationstechnik und Wissensverarbeitung in der Produktentwicklung (Lecture notes). Institut für Konstruktionstechnik und Technisches Design, 2015, Stuttgart.
- KATZENBACH, A., S. HANDSCHUH, R. DOTZAUER and A. FRÖHLICH (2015): Product Lifecycle Visualization. In: Stjepandić, J., N. Wognum and W. J. C. Verhagen (Eds.): Concurrent engineering in the 21st century. Foundations, developments and challenges. Springer, Cham: pp. 287–318.
- KAUFMANN, U. and R. SCHULER (2017): Systems Re-Engineering – ein Beitrag zur Integration von MBSE und PLM. In: Schulze, S.-O., C. Tschirner and R. Kaffenberger (Eds.): Tag des Systems Engineering. Herzogenaurach, 25.-27. Oktober 2016. Hanser, München: pp. 343–353.
- KEYDEL, C. and O. MEDING (2008): Installing and Using a Version Control System. In: Ganssle, J. et al. (Eds.): Embedded systems. World class designs. Newnes, Oxford: pp. 225–246.
- KIRKPATRICK, K. and A. KAUL (2019): Digital Twins. Global Market Demand Across Manufacturing, Aerospace, Connected Vehicles, Smart Cities, Retail, Healthcare, Industrial IoT, and Other Industries. Tractica, Boulder, CO.
- KIRSCH, L., C. MUGGEO, M. SCHNEIDER, T. SCHULTE and C. DETTMERS (2017a): Funktionen im PDM / PLM. In: Eigner, M., W. Koch and C. Muggeo (Eds.): Modellbasierter Entwicklungsprozess cybertronischer Systeme. Der PLM-unterstützte Referenzentwicklungsprozess für Produkte und Produktionssysteme. Springer Vieweg, Berlin: pp. 155–160.
- KIRSCH, L., C. MUGGEO, T. SCHULTE and M. SCHNEIDER (2017b): Verwaltung von Systemmodellen. In: Eigner, M., W. Koch and C. Muggeo (Eds.): Modellbasierter Entwicklungsprozess cybertronischer Systeme. Der PLM-unterstützte Referenzentwicklungsprozess für Produkte und Produktionssysteme. Springer Vieweg, Berlin: pp. 161–167.
- KIRSCH, L., C. MUGGEO, T. SCHULTE, M. SCHNEIDER and P. MÜLLER (2017c): PLM-Funktionen im Kontext von Systemmodellen. In: Eigner, M., W. Koch and C. Muggeo (Eds.): Modellbasierter Entwicklungsprozess cybertronischer Systeme. Der PLM-

- unterstützte Referenzentwicklungsprozess für Produkte und Produktionssysteme. Springer Vieweg, Berlin: pp. 169–176.
- KIRSCH, L., P. MÜLLER, M. EIGNER and C. MUGGEO (2017d): SysML-Modellverwaltung im PDM/PLM-Umfeld. In: Schulze, S.-O., C. Tschirner and R. Kaffenberger (Eds.): Tag des Systems Engineering. Herzogenaurach, 25.-27. Oktober 2016. Hanser, München: pp. 333–342.
- KLEINER, S. and C. KRAMER (2013): Model Based Design with Systems Engineering Based on RFLP Using V6. In: Abramovici, M. and R. Stark (Eds.): Smart Product Engineering. Proceedings of the 23rd CIRP Design Conference, Bochum, Germany, March 11th - 13th, 2013. Lecture Notes in Production Engineering: pp. 93–102.
- KÖNIGS, S. F. (2013): Konzeption und Realisierung einer Methode zur templategestützten Systementwicklung.
- KÖNIGS, S. F., G. BEIER, A. FIGGE and R. STARK (2012): Traceability in Systems Engineering – Review of industrial practices, state-of-the-art technologies and new research solutions. In: Advanced Engineering Informatics 26 (4): pp. 924–940.
- KORDON, F. (Ed.) (2013): Embedded systems. Analysis and modeling with SysML, UML and AADL. Electronics engineering series. ISTE, London.
- KRAUSE, F.-L., H.-J. FRANKE and J. GAUSEMEIER (Eds.) (2007): Innovationspotenziale in der Produktentwicklung. Hanser, München.
- KRIEG, A., A. RAJKO and F. BOUCHÉ (2018): Agil und attraktiv - Wie Entwicklungsdienstleister die Zukunft mitgestalten. In: ATZextra 23 (S2): pp. 14–17.
- KRUIJFF, J. de and H. WEIGAND (2017): Understanding the Blockchain Using Enterprise Ontology. In: Dubois, E. and K. Pohl (Eds.): Advanced information systems engineering. 29th International Conference, CAiSE 2017, Essen, Germany, June 12-26, 2017: proceedings. Lecture Notes in Computer Science, Issue 10253. Springer, Cham: pp. 29–43.
- LAMM, J. G. and T. WEILKIENS (2010): Funktionale Architekturen in SysML. In: Maurer, M. and S.-O. Schulze (Eds.): Tag des Systems Engineering 2010. Carl Hanser Verlag, München: pp. 109–118.
- LÄMMER, L. and M. THEISS (2015): Product Lifecycle Management. In: Stjepandić, J., N. Wognum and W. J. C. Verhagen (Eds.): Concurrent engineering in the 21st century. Foundations, developments and challenges. Springer, Cham: pp. 455–490.
- LANKENAU, A. and D. T. HEBER (2017): The Importance of E/E in the Context of the Digital Twin, EDM CAE Forum 2017, Daimler AG, 19.7.2017, Stuttgart.
- LEACH, P., M. MEALLING and R. SALZ (2005): A Universally Unique Identifier (UUID) URN Namespace. Request for Comments No. 4122. In: <https://tools.ietf.org/html/rfc4122>. Call: 19.7.2020.
- LIESE, H., S. RULHOFF and J. STJEPANDIĆ (2013): Enhancing Product Innovation by Implementing Intellectual Property Protection into the Virtual Product Creation. In: Stjepandić, J., G. Rock and C. Bil (Eds.): Concurrent Engineering Approaches for Sustainable Product Development in a Multi-Disciplinary Environment. Proceedings of the 19th ISPE International Conference on Concurrent Engineering. Springer, London: pp. 267–278.
- LINDEMANN, G. and M. KRASTEL (2017): Die Implementierung des Informationsmodells in PLM-Systemen. In: Eigner, M., W. Koch and C. Muggeo (Eds.): Modellbasierter

- Entwicklungsprozess cybertronischer Systeme. Der PLM-unterstützte Referenzentwicklungsprozess für Produkte und Produktionssysteme. Springer Vieweg, Berlin: pp. 149–151.
- LINDEMANN, U. (2009): Methodische Entwicklung technischer Produkte. Springer, Berlin, Heidelberg.
- LINDEMANN, U., M. MAURER and T. BRAUN (2009): Structural complexity management. An approach for the field of product design. Springer, Berlin, Heidelberg.
- LOPER, M. L. (Ed.) (2015): Modeling and simulation in the systems engineering life cycle. Core concepts and accompanying lectures. Springer, London.
- LOTAR INTERNATIONAL: LONG TERM ARCHIVING AND RETRIEVAL. In: <http://www.lotar-international.org>. Call: 3.5.2019.
- LU, Q. and X. XU (2017): Adaptable Blockchain-Based Systems. A Case Study for Product Traceability. In: IEEE Software 34 (6): pp. 21–27.
- MARTIN, J. N. (1996): Systems Engineering Guidebook: A Process for Developing Systems and Products. CRC Press, Boca Raton, FL.
- MECPRO² ABSCHLUSSBERICHT (2016a): Arbeitspaket 2.1 Referenzprozesse (Prozessanalyse) CTP und CTPS. Modellbasierter Entwicklungsprozess Cybertronischer Produkte und Produktionssysteme (mecPro²).
- MECPRO² ABSCHLUSSBERICHT (2016b): Arbeitspaket 2.3 Integrierte Beschreibungssystematik. Modellbasierter Entwicklungsprozess Cybertronischer Produkte und Produktionssysteme (mecPro²).
- MECPRO² ABSCHLUSSBERICHT (2016c): Arbeitspaket 6.5 - Normen, Standards und Formate. Modellbasierter Entwicklungsprozess Cybertronischer Produkte und Produktionssysteme (mecPro²).
- MECPRO² ABSCHLUSSBERICHT (2016d): Arbeitspaket 6.7 PLM-Funktionen. Modellbasierter Entwicklungsprozess Cybertronischer Produkte und Produktionssysteme (mecPro²).
- MORABITO, V. (2017): Business Innovation Through Blockchain. The B³ Perspective. Springer, Cham.
- MÜLLER, P. and A. HAßE (2017a): Fokus der Demonstratoren. In: Eigner, M., W. Koch and C. Muggeo (Eds.): Modellbasierter Entwicklungsprozess cybertronischer Systeme. Der PLM-unterstützte Referenzentwicklungsprozess für Produkte und Produktionssysteme. Springer Vieweg, Berlin: pp. 183–187.
- MÜLLER, P. and A. HAßE (2017b): Gemeinsame Erkenntnislage. In: Eigner, M., W. Koch and C. Muggeo (Eds.): Modellbasierter Entwicklungsprozess cybertronischer Systeme. Der PLM-unterstützte Referenzentwicklungsprozess für Produkte und Produktionssysteme. Springer Vieweg, Berlin: pp. 227–229.
- MÜLLER, P. and L. KIRSCH (2017): Vernetzung von Entwicklungsdaten. In: Eigner, M., W. Koch and C. Muggeo (Eds.): Modellbasierter Entwicklungsprozess cybertronischer Systeme. Der PLM-unterstützte Referenzentwicklungsprozess für Produkte und Produktionssysteme. Springer Vieweg, Berlin: pp. 177–180.
- MÜLLER, P., L. KIRSCH, M. SCHNEIDER and A. HERZMANN (2017): Demonstrator 1 – Modellbasierte Entwicklung cybertronischer Produkte. In: Eigner, M., W. Koch and C. Muggeo (Eds.): Modellbasierter Entwicklungsprozess cybertronischer Systeme.

- Der PLM-unterstützte Referenzentwicklungsprozess für Produkte und Produktionssysteme. Springer Vieweg, Berlin: pp. 189–208.
- MÜLLER, P., M. MUSCHIOL and R. STARK (2012): PLM-Based Service Data Management in Steam Turbine Business. In: Rivest, L., A. Bouras and B. Louhichi (Eds.): Product Lifecycle Management. Towards Knowledge-Rich Enterprises. IFIP WG 5.1 International Conference, PLM 2012, Montreal, QC, Canada, July 9-11, 2012, Revised Selected Papers. IFIP Advances in Information and Communication Technology, Issue 388. Springer, Berlin, Heidelberg: pp. 170–181.
- NAKAMOTO, S. (2008): Bitcoin: A Peer-to-Peer Electronic Cash System. www.bitcoin.org. In: <https://bitcoin.org/bitcoin.pdf>.
- NARAYANAN, A., J. BONNEAU, E. FELTEN, A. MILLER and S. GOLDFEDER (2016): Bitcoin and cryptocurrency technologies. A comprehensive introduction. Princeton University Press, Princeton.
- NARAYANAN, A. and J. CLARK (2017): Bitcoin's academic pedigree. The concept of cryptocurrencies is built from forgotten ideas in research literature. In: Communications of the ACM 15 (4): pp. 1–30.
- NASA (2007): Systems Engineering Handbook. NASA/SP-2007-6105 Rev1. National Aeronautics and Space Administration, Washington, D.C.
- NELSON, H. G. and E. STOLTERMAN (2003): The design way. Intentional change in an unpredictable world; foundations and fundamentals of design competence. Educational Technology Publications, Englewood Cliffs, N.J.
- NEUMEYER, S., P. LÜNNEMANN, R. WOLL, H. HAYKA and R. STARK (2017): Systems Engineering im Kontext der unternehmensübergreifenden Produktentwicklung. In: Schulze, S.-O., C. Tschirner and R. Kaffenberger (Eds.): Tag des Systems Engineering. Herzogenaurach, 25.-27. Oktober 2016. Hanser, München: pp. 23–32.
- NEWCOMB, D. (2012): The next big OS war is in your dashboard. In: <https://www.wired.com/2012/12/automotive-os-war/>. Call: 18.9.2018.
- NO MAGIC, I. (2015): Cameo DataHub. User Guide 18.1. Allen, TX. In: <https://www.nomagic.com/files/manuals/CameoDataHubUserGuide.pdf>. Call: 12.4.2020.
- NO MAGIC, I. (2020a): Cameo Systems Modeler - Features. In: <https://www.nomagic.com/products/cameo-systems-modeler#features>. Call: 12.4.2020.
- NO MAGIC, I. (2020b): User Guide - Defining hyperlinks. In: <https://docs.nomagic.com/display/MD190SP2/Defining+hyperlinks>. Call: 12.4.2020.
- NORFOLK, D. (2015): PTC Integrity Modeler... a standards-based tool for systems and software engineering. InDetail. Bloor. In: <https://www.ptc.com/-/media/Files/PDFs/ALM/Integrity/PTC-Integrity-Modeler-Bloor-InDetail.pdf?la=en&hash=D41860FAF851610B3C45180221FB398A>.
- NOY, N. F. and D. L. MCGUINNESS (2001): Ontology Development 101: A Guide to Creating Your First Ontology. Stanford Knowledge Systems Laboratory Technical Report KSL-01-05. Stanford University, Stanford, CA. In: <https://protegewiki.stanford.edu/wiki/Ontology101>.

- OASIS (2019): OSLC Core Version 3.0. Part 1: Overview - Project Specification Draft 04. Pennsylvania. In: <http://docs.oasis-open.org/oslc-core/oslc-core/v3.0/csprd03/part1-overview/oslc-core-v3.0-csprd03-part1-overview.html>.
- OLLERTON, P. (2016): Integrity modeler 8.3 Windchill integration - overview and use cases. PTC Inc. In: https://community.ptc.com/sejnu66972/attachments/sejnu66972/Integrity-Tips/16/1/PTC_IntegrityModeler_8.3_Windchill_Integration-Overview_and_UseCases.pdf.
- OMG (2015): OMG Systems Modeling Language (OMG SysML™). Object Management Group. In: <https://www.omg.org/spec/SysML/1.4/PDF>.
- ÖZSU, M. T. and P. VALDURIEZ (2011): Principles of Distributed Database Systems, Third Edition. Springer New York, New York, NY.
- PAVALKIS, S. (2016): Towards Industrial Integration of MBSE into PLM for Mission-Critical Systems. In: INCOSE International Symposium 26 (1): pp. 2462–2477.
- PEARCE, P. and M. HAUSE (2012): ISO-15288, OOSEM and Model-Based Submarine Design. SETE APCOSE 2012.
- PEREPA, B. and J. YELICK (2017): Add an organization to your existing Hyperledger Fabric blockchain network using an easy tool. Use configtxlator to customize the Hyperledger Fabric first-network sample. In: <https://developer.ibm.com/technologies/blockchain/tutorials/cl-add-an-organization-to-your-hyperledger-fabric-blockchain/>. Call: 19.4.2021.
- PETERSON, L. L. and B. S. DAVIE (2012): Computer networks. A systems approach. Morgan Kaufmann, Burlington, Mass.
- PFENNING, M. (2017): Durchgängiges Engineering durch die Integration von PLM und MBSE. Schriftenreihe VPE.
- PFENNING, M. (2020): Breaking Down the Silos. How Systems Architecture Enables Interdisciplinary Collaboration. Aras, Gröbenzell.
- PFLEEGER, S. L. and S. A. BOHNER (1990): A framework for software maintenance metrics. In: Institute of Electrical and Electronics Engineers (Ed.): November 26 - 29, 1990, San Diego, Ca. IEEE Comput. Soc. Press: pp. 320–327.
- PIMMLER, T. U., EPPINGER, S. D. (1994): Integration analysis of product decompositions. In: Hight, T. K. and F. Mistree (Eds.): Design theory and methodology, DTM '94. Presented at the 1994 ASME Design Technical Conferences, 6th International Conference on Design Theory and Methodology, Minneapolis, Minnesota, September 11-14, 1994. DE, vol. 68. American Society of Mechanical Engineers, New York: pp. 343–351.
- POHL, K., G. BÖCKLE and F. LINDEN (2005): Software Product Line Engineering. Foundations, Principles, and Techniques. Springer-Verlag, Berlin, Heidelberg.
- POHL, K., H. HÖNNINGER, R. ACHATZ and M. BROY (Eds.) (2012): Model-Based Engineering of Embedded Systems. The SPES 2020 Methodology. Springer, Berlin, Heidelberg.
- POMBERGER, G. and W. PREE (2004): Software Engineering. Architektur-Design und Prozessorientierung.

- PONN, J. and U. LINDEMANN (2011): Konzeptentwicklung und Gestaltung technischer Produkte. Systematisch von Anforderungen zu Konzepten und Gestaltlösungen. VDI-Buch. Springer-Verlag Berlin Heidelberg, Berlin, Heidelberg.
- PROBST, K. E. (2010): automotiveDAY. Die Business-IT der Zukunft. BMW Group IT.
- PROSTEP INC.: Partner Integrations. OSLC Integration. In: <https://prostep.us/home/solutions/partner-integrations/oslc-integration/>. Call: 8.11.2022.
- PROSTEP IVIP E.V. (2020a): OpenPDM integrate. In: <https://www.prostep.com/en/products-and-solutions/openpdm/openpdm-integrate.html>. Call: 26.4.2020.
- PROSTEP IVIP E.V. (2020b): OSLC Integration. OpenPDM Open Services for Lifecycle Collaboration (OSLC) Adapter. In: <https://prostep.us/home/solutions/partner-integrations/oslc-integration/>. Call: 26.4.2020.
- PRUSTY, N. (2017): Building Blockchain projects. Building decentralized Blockchain applications with Ethereum and Solidity. Packt, Birmingham, Mumbai.
- PTC INC. (2019): Windchill Modeler data sheet. In: <https://www.ptc.com/-/media/Files/PDFs/ALM/Integrity/PTC-Integrity-Modeler-Data-Sheet.pdf?la=en&hash=E36AC88752AA3C1ED6F08BF6FA0ABF76>.
- PTC INC. (2020a): Einführung in Windchill 11. In: <https://ptc-solutions.de/produkte/windchill/windchill-11>. Call: 2.5.2020.
- PTC INC. (2020b): PTC Integrity Modeler - What's New! In: <https://www.ptc.com/en/products/plm/plm-products/integrity-modeler-what-is-new>. Call: 12.4.2020.
- PTC INC. (2020c): Windchill PDMLink for Digital Product Traceability. In: https://support.ptc.com/help/oslc/dpt/en/index.html#page/oslc_dpt/WCPDMLink/RM_windchill_ALMIntegration.html. Call: 2.5.2020.
- RAMESH, B. and M. JARKE (2001): Toward reference models for requirements traceability. In: IEEE Trans. Software Eng. (IEEE Transactions on Software Engineering) 27 (1): pp. 58–93.
- RAVAL, S. (2016): Decentralized applications. Harnessing Bitcoin's Blockchain technology. O'Reilly Media, Sebastopol, CA.
- REARDON, K. (2016): Aras to Present on MBSE at Zuken Innovation World 2016. Presentation entitled “Beyond ECAD Connectors” Addresses the Model-Based Systems Engineering Challenges Faced in IoT Design Processes. In: <https://www.aras.com/en/news/press-releases/2016/04/aras-to-present-on-mbse-at-zuken-innovation-world-2016>. Call: 6.11.2022.
- REIF, K. (2014): Automobilelektronik. Springer Fachmedien Wiesbaden, Wiesbaden.
- REIF, K. (2016): Sensoren im Kraftfahrzeug. Springer Fachmedien Wiesbaden, Wiesbaden.
- RITTBERG, S. (2014): Jahresbericht 2013/2014. Produktsicherheit - Rückrufe. Kraftfahrt-Bundesamt, Flensburg.
- RITTEL, H. W. J. and M. M. WEBBER (1973): Dilemmas in a General Theory of Planning. In: Policy Sciences 4 (2): pp. 155–169.

- ROBERT BOSCH GMBH (2014): Bosch Automotive Electrics and Automotive Electronics. Systems and Components, Networking and Hybrid Drive. Springer Fachmedien Wiesbaden, Wiesbaden.
- ROPOHL, G. (2009): Allgemeine Technologie. Eine Systemtheorie der Technik. KIT Scientific Publishing, s.l.
- RYMAN, A. (2013): Linked Data Interfaces. Define REST API contracts for RDF resource representation. IBM Corporation.
- SAKR, S., M. WYLOT, R. MUTHARAJU, D. LE PHUOC and I. FUNDULAKI (2018): Linked Data. Storing, Querying, and Reasoning. Springer International Publishing, Cham.
- SCHÄUFFELE, J. and T. ZURAWKA (2016): Automotive Software Engineering. Grundlagen, Prozesse, Methoden und Werkzeuge effizient einsetzen. ATZ / MTZ-Fachbuch. Springer Vieweg, Wiesbaden.
- SCHICKER, E. (2017): Datenbanken und SQL. Eine praxisorientierte Einführung mit Anwendungen in Oracle, SQL Server und MySQL. Informatik & Praxis.
- SCHLATT, V., A. SCHWEIZER, N. URBACH and G. FRIDGEN (2016): Blockchain: Grundlagen, Anwendungen und Potentiale. Projektgruppe Wirtschaftsinformatik des Fraunhofer-Institut für Angewandte Informationstechnik FIT.
- SCHLOTT, S. (2005): Wahnsinn mit Methode. In: Automobil-Produktion (1): pp. 38–42.
- SCHUH, G. (2005): Produktkomplexität managen. Strategien - Methoden - Tools. Hanser, München, Wien.
- SCHULTE, T., T. DICKOPF and A. STANDKE (2017a): Integration & CTP-Spezialisierung. In: Eigner, M., W. Koch and C. Muggeo (Eds.): Modellbasierter Entwicklungsprozess cybertronischer Systeme. Der PLM-unterstützte Referenzentwicklungsprozess für Produkte und Produktionssysteme. Springer Vieweg, Berlin.
- SCHULTE, T., S. GROß, S. LANGER and L. KIRSCH (2017b): ConfigML – Erste prototypische Realisierung einer Verwaltung von Modellen mit Modellen im PLM. In: Schulze, S.-O. et al. (Eds.): Tag des Systems Engineering. Paderborn, 8. -10. November 2017. Carl Hanser Verlag GmbH & Co. KG, München: pp. 177–186.
- SCHULTE, T., M. SCHNEIDER, T. DICKOPF and L. MAYERHOFER (2017c): Erweiterung des integrierten Konzeptes aus Prozessrahmenwerk und Beschreibungssystematik von mecPro2 um ein modellbasiertes Variantenmanagement. In: Schulze, S.-O., C. Tschirner and R. Kaffenberger (Eds.): Tag des Systems Engineering. Herzogenaurach, 25.-27. Oktober 2016. Hanser, München: pp. 257–268.
- SCHULTE, T., M. SCHNEIDER, U. JUDASCHKE and D. BATZ (2017d): Systemmodelle verwalten mit ConfigML - Motive, Grundlagen und erste Konzepte einer Sprache für das modellbasierte Konfigurationsmanagement. In: Schulze, S.-O., C. Tschirner and R. Kaffenberger (Eds.): Tag des Systems Engineering. Herzogenaurach, 25.-27. Oktober 2016. Hanser, München: pp. 321–332.
- SCHÜRR, A. (1995): Specification of graph translators with triple graph grammars. In: Goos, G. et al. (Eds.): Graph-Theoretic Concepts in Computer Science. Lecture Notes in Computer Science. Springer Berlin Heidelberg, Berlin, Heidelberg: pp. 151–163.
- SCHWARZ, H., J. EBERT and A. WINTER (2010): Graph-based traceability. A comprehensive approach. In: Software & Systems Modeling 9 (4): pp. 473–492.

- SEEPERSAD, C. C., K. PEDERSEN, J. EMBLEMSVÅG, R. BAILEY, J. ALLEN and F. MISTREE (2006): The Validation Square. How Does One Verify and Validate a Design Method? In: Lewis, K. E., W. Chen and L. C. Schmidt (Eds.): Decision Making in Engineering Design. ASME Press: pp. 303–313.
- SELLGREN, U. (2009): The journey towards PLM managed and interface-driven design. In: Malmqvist, J. and G. Gustafsson (Eds.): Proceedings of the 2nd Nordic Conference on Product Lifecycle Management - NordPLM'09, Göteborg, January 28-29, 2009: pp. 1–12.
- ŞENALTUN, G. and C. CANGELIR (2012): Software Management in Product Structure. In: Rivest, L., A. Bouras and B. Louhichi (Eds.): Product Lifecycle Management. Towards Knowledge-Rich Enterprises. IFIP WG 5.1 International Conference, PLM 2012, Montreal, QC, Canada, July 9-11, 2012, Revised Selected Papers. IFIP Advances in Information and Communication Technology, Issue 388. Springer, Berlin, Heidelberg: pp. 369–378.
- SENDER, U. (2009): Das PLM-Kompendium. Referenzbuch des Produkt-Lebenszyklus-Managements. Xpert.press. Springer, Berlin, Heidelberg.
- SIEMENS INDUSTRY SOFTWARE INC. (2019): Software Design Management and PLM-ALM Integration in Teamcenter 11.3. In: <https://community.sw.siemens.com/s/article/software-design-management-and-plm-alm-integration-in-teamcenter-11-3>. Call: 26.4.2020.
- SIEMENS INDUSTRY SOFTWARE INC. (2020a): Teamcenter. In: <https://www.plm.automation.siemens.com/global/de/products/teamcenter/>. Call: 7.4.2020.
- SIEMENS INDUSTRY SOFTWARE INC. (2020b): Teamcenter provides strategic pillar for aerospace firm as it seeks to grow its business. Using Siemens Digital Industries Software solutions enables ITP to improve traceability and efficiency while enhancing quality. In: <https://www.plm.automation.siemens.com/global/de/our-story/customers/industria-de-turbo-propulsores/66227/>. Call: 7.4.2020.
- SIMON, H. A. and J. E. LAIRD (2019): The sciences of the artificial. The MIT Press, Cambridge, Massachusetts, London, England.
- SINDERMANN, S. (2014): Schnittstellen und Datenaustauschformate. In: Eigner, M., D. Roubanov and R. Zafirov (Eds.): Modellbasierte virtuelle Produktentwicklung. Springer Vieweg, Berlin: pp. 327–347.
- SIXT, E. (2017): Bitcoins und andere dezentrale Transaktionssysteme. Blockchains als Basis einer Kryptoökonomie. Springer Gabler, Wiesbaden.
- SODIUS CORP. (2020): OSLC Connect for Windchill. In: <https://www.sodiuswillert.com/en/products/oslc-connect-for-windchill>. Call: 2.5.2020.
- SORNIOTTI, A. and J. YELICK (2018): Chaincode lifecycle - 2.0 improvements. In: <https://jira.hyperledger.org/browse/FAB-11237>. Call: 13.8.2021.
- SPARX SYSTEMS PTY LTD. (2020a): Enterprise Architect - Compare Editions. In: <https://sparxsystems.com/products/ea/compare-editions.html>. Call: 12.4.2020.
- SPARX SYSTEMS PTY LTD. (2020b): OSLC Architecture Management v2.0. In: https://sparxsystems.com/enterprise_architect_user_guide/14.0/model_repository/oslc_am_top.html. Call: 12.4.2020.

- SPATH, D. (Ed.) (2013): Produktionsarbeit der Zukunft - Industrie 4.0. [Studie]. Fraunhofer-Verl., Stuttgart.
- SPATH, D., E. WESTKÄMPER, H.-J. BULLINGER and H.-J. WARNECKE (2017): Neue Entwicklungen in der Unternehmensorganisation. VDI-Buch Ser. Vieweg, Berlin, Heidelberg.
- STAREPRAVO, I. (2019): How Automotive OEMs Partner with Startups to Create New Opportunities. The Startup. In: <https://medium.com/swlh/how-automotive-oems-partner-with-startups-to-create-new-opportunities-7d271bf650a7>. Call: 22.4.2019.
- STARK, J. (2015): Product Lifecycle Management (Volume 1). 21st Century Paradigm for Product Realisation. Springer International Publishing, Cham.
- STARK, J. (2016): Product Lifecycle Management (Volume 2). The Devil is in the Details. Decision Engineering. Springer International Publishing AG Switzerland, Cham.
- STELZER, R. (Ed.) (2014): Entwerfen Entwickeln Erleben 2014. Beiträge zur virtuellen Produktentwicklung und Konstruktionstechnik, Dresden, 26.-27. Juni 2014. TUDpress; Saechsische Landesbibliothek- Staats- und Universitaetsbibliothek Dresden, Dresden, Dresden.
- STEPHAN, N. K. (2013): Vorgehensmodell zur Unterstützung der interdisziplinären und förderierten Zusammenarbeit in der frühen Phase der Produktentstehung. Am Beispiel der Nutzfahrzeugindustrie. Kaiserslautern, Techn. Univ., Diss., 2013. KIMA-Schriftenreihe, Issue 9. Technische Universität, Kaiserslautern.
- STIEFEL, P. (2011): Eine dezentrale Informations- und Kollaborationsarchitektur für die unternehmensübergreifende Produktentwicklung. Vieweg+Teubner Verlag, Wiesbaden.
- STIGLITZ, J. E. (2010): Freefall. Free markets and the sinking of the global economy. Penguin Books, London.
- STJEPANDIĆ, J., H. LIESE and A. J. C. TRAPPEY (2015a): Intellectual property protection. In: Stjepandić, J., N. Wognum and W. J. C. Verhagen (Eds.): Concurrent engineering in the 21st century. Foundations, developments and challenges. Springer, Cham: pp. 521–551.
- STJEPANDIĆ, J., N. WOGNUM and W. J. C. VERHAGEN (Eds.) (2015b): Concurrent engineering in the 21st century. Foundations, developments and challenges. Springer, Cham.
- STÖCKERT, H. (2011): Fehlervermeidung an Schnittstellen-Prozessen der verteilten Produktentwicklung. Dissertation. Technische Universität Berlin, Berlin.
- STRINGHAM, G. (2010): Hardware/firmware interface design. Best practices for improving embedded systems development. Newnes, Amsterdam.
- SUTINEN, K., L. ALMEFELT and J. MALMQVIST (2000): Implementation of Requirements Traceability in Systems Engineering Tools.
- SUTINEN, K., L. ALMEFELT and J. MALMQVIST (2002): Supporting Concept Development Using Quantitative Requirements Traceability.
- TECHNICAL OPERATIONS INTERNATIONAL COUNCIL ON SYSTEMS ENGINEERING (2007): Systems Engineering Vision 2020. INCOSE-TP-2004-004-02. San Diego.
- TOMIZUKA, M. (2000): Mechatronics. From the 20th to 21st Century. In: IFAC Proceedings Volumes 33 (26): pp. 1–10.

- TORKAR, R., T. GORSCHKE, R. FELDT, M. SVAHNBERG, U. A. RAJA and K. KAMRAN (2012): Requirements Traceability. A systematic review and industry case study. In: International Journal of Software Engineering and Knowledge Engineering 22 (03): pp. 385–433.
- TRIPPNER, D., S. RUDE and A. SCHREIBER (2015): Challenges to Digital Product and Process Development Systems at BMW. In: Stjepandić, J., N. Wognum and W. J. C. Verhagen (Eds.): Concurrent engineering in the 21st century. Foundations, developments and challenges. Springer, Cham: pp. 555–569.
- ULRICH, H. and G. J. B. PROBST (1988): Anleitung zum ganzheitlichen Denken und Handeln. Ein Brevier für Führungskräfte. Haupt, Bern u.a.
- VACHER, A., D. BRISSAUD and S. TICHKIEWITCH (2007): Towards a framework for managing conceptual knowledge in distributed and collaborative R&D projects. In: Krause, F.-L. (Ed.): The Future of Product Development. Proceedings of the 17th CIRP Design Conference. Springer-Verlag, Berlin, Heidelberg: pp. 311–318.
- VAJNA, S. (2009): CAx für Ingenieure. Eine praxisbezogene Einführung. Springer, Berlin u.a.
- VAN RANDEN, H. J., C. BERCKER and J. FIEML (2016): Einführung in UML. Analyse und Entwurf von Software. Springer Vieweg, Wiesbaden.
- VDA QUALITY MANAGEMENT CENTER (2017): Automotive SPICE - Process reference model, process assessment model, version 3.1.
- VEREIN DEUTSCHER INGENIEURE (1987): VDI 2235 - Wirtschaftliche Entscheidungen beim Konstruieren - Methoden und Hilfen. VDI-Gesellschaft Produkt- und Prozessgestaltung, Düsseldorf.
- VEREIN DEUTSCHER INGENIEURE (1993): Systematic approach to the development and design of technical systems and products (VDI 2221). VDI-Gesellschaft Entwicklung Konstruktion Vertrieb, Düsseldorf.
- VEREIN DEUTSCHER INGENIEURE (2004a): Systematic embodiment design of technical products (VDI 2223). VDI-Gesellschaft Entwicklung Konstruktion Vertrieb, Düsseldorf.
- VEREIN DEUTSCHER INGENIEURE (2004b): VDI 2206 - Design methodology for mechatronic systems. VDI-Gesellschaft Produkt- und Prozessgestaltung. Call: 27.7.2017.
- VEREIN DEUTSCHER INGENIEURE (2014): VDI 2219 - Information technology in product development – Introduction and usage of PDM systems (VDI 2219). VDI-Gesellschaft Produkt- und Prozessgestaltung, Düsseldorf.
- VOSGIEN, T., T. NGUYEN VAN, M. JANKOVIC, B. EYNARD and J.-C. BOCQUET (2012): Towards Model-Based System Engineering for Simulation-Based Design in Product Data Management Systems. In: Rivest, L., A. Bouras and B. Louhichi (Eds.): Product Lifecycle Management. Towards Knowledge-Rich Enterprises. IFIP WG 5.1 International Conference, PLM 2012, Montreal, QC, Canada, July 9-11, 2012, Revised Selected Papers. IFIP Advances in Information and Communication Technology, Issue 388. Springer, Berlin, Heidelberg: pp. 612–622.
- VOSHMIGIR, S. (2016): Blockchains, Smart Contracts und das Dezentrale Web. Technologiestiftung Berlin, Berlin.

- WADE, J., R. ADCOCK, T. McDERMOT and L. STRAWSER (2018): Future Systems Engineering Research Directions. In: Madni, A. M. et al. (Eds.): *Disciplinary Convergence in Systems Engineering Research*. Springer International Publishing, Cham: pp. 1165–1179.
- WALDEN, D. D., G. J. ROEDLER, K. FORSBERG, R. D. HAMELIN and T. M. SHORTELL (Eds.) (2015): *Systems engineering handbook. A guide for system life cycle processes and activities*; INCOSE-TP-2003-002-04, 2015. Wiley, Hoboken, NJ.
- WALLMÜLLER, E. (2011): *Software quality engineering. Ein Leitfaden für bessere Software-Qualität*. Hanser Verlag, München.
- WATTS, F. B. (2011): *Engineering documentation control handbook. Configuration management and product lifecycle management*.
- WEBER, R., P. REINKEMEIER, E. THADEN and A. BAUMGART (2012): *Specification of an Architecture Meta-Model*. OFFIS Technical Report. OFFIS, Oldenburg.
- WEHN, N. (2013): *System Modelling HW/SW Co-Design Optimization*. Lecture notes. Technische Universität Kaiserslautern, Lehrstuhl Entwurf Mikro-elektronischer Systeme.
- WEILKIENS, T. (2008): *Systems Engineering with SysML/UML. Modeling, Analysis, Design*. The MK / OMG Press. Elsevier professional, s.l.
- WEILKIENS, T., A. SCHEITHAUER, M. DI MAIO and N. KLUSMANN (2016): Evaluating and comparing MBSE methodologies for practitioners. In: Institute of Electrical and Electronics Engineers (Ed.): *2016 IEEE International Symposium on Systems Engineering (ISSE)*. IEEE: pp. 1–8.
- WIEHMEIER, M. (2017): *Start-ups im Automobilsektor: Erhöhter finanzieller Einsatz für Gründer*. In: <https://www.oliverwyman.de/media-center/2017/Erhoehter-finanzieller-Einsatz-fuer-Grunder.html>. Call: 22.4.2019.
- WIERINGA, R. J. (1995): *An Introduction to Requirements Traceability*. IR-389. Free University, Faculty of Mathematics and Computer Science, Amsterdam.
- WINKLER, S. and J. von PILGRIM (2009): A survey of traceability in requirements engineering and model-driven development. In: *Software & Systems Modeling* 9 (4): pp. 529–565.
- WINNER, H., S. HAKULI, F. LOTZ and C. SINGER (Eds.) (2015): *Handbuch Fahrerassistenzsysteme. Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort*. ATZ/MTZ-Fachbuch. Springer Vieweg, Wiesbaden.
- WINZER, P. (2016): *Generic Systems Engineering. Ein methodischer Ansatz zur Komplexitätsbewältigung*. Springer Vieweg, Berlin, Heidelberg.
- WOGNUM, N. and R. CURRAN (2013): Current Concurrency in Practice. In: Stjepandić, J., G. Rock and C. Bil (Eds.): *Concurrent Engineering Approaches for Sustainable Product Development in a Multi-Disciplinary Environment*. Proceedings of the 19th ISPE International Conference on Concurrent Engineering. Springer, London: pp. 3–14.
- WOSS, W. (1997): A rule-driven generator for variant parts and variant bills of material. In: IEEE Computer Society (Ed.): *Database and Expert Systems Applications*. 8th International Conference, DEXA '97. Proceedings. IEEE Computer Society Press: pp. 556–561.

- ZAFIROV, R. (2014): Modellbildung und Spezifikation. In: Eigner, M., D. Roubanov and R. Zafirov (Eds.): Modellbasierte virtuelle Produktentwicklung. Springer Vieweg, Berlin: pp. 77–96.
- ZAFIROV, R. (2017): Model-based systems engineering methods for integrated product design, process planning, and production systems design. Dissertation. Technische Universität Kaiserslautern.
- ZAFIROV, R. and D. ROUBANOV (2014): Elektrik und Elektronik (E-CAD). In: Eigner, M., D. Roubanov and R. Zafirov (Eds.): Modellbasierte virtuelle Produktentwicklung. Springer Vieweg, Berlin: pp. 137–159.
- ZAWIŚLAK, S. and J. RYSIŃSKI (Eds.) (2017): Graph-based modelling in engineering. Mechanisms and machine science, Volume 42. Springer, Cham.
- ZENGLER, C. and W. KÜCHLIN (2013): Boolean Quantifier Elimination for Automotive Configuration – A Case Study. In: Dierkes, M. and C. Pecheur (Eds.): Formal Methods for Industrial Critical Systems. 18th International Workshop, FMICS 2013, Madrid, Spain, September 23-24, 2013, Proceedings. Lecture Notes in Computer Science / Programming and Software Engineering, v.8187. Springer Berlin Heidelberg, Berlin/Heidelberg: pp. 48–62.
- ZIMMERMANN, W. and R. SCHMIDGALL (2014): Bussysteme in der Fahrzeugtechnik. Protokolle, Standards und Softwarearchitektur. ATZ / MTZ-Fachbuch. Springer Vieweg, Wiesbaden.