Contract-based methods and activities in the validation of interfaces for System of Systems

Moritz Wäschle*, Matthias Behrendt*, Kangning Xing*, Hanwen Shi*, Albert Albers*
*Karlsruher Institut für Technologie (KIT), IPEK – Institut für Produktentwicklung, Kaiserstraße 10, 76131 Karlsruhe,
{Moritz.Waeschle, Matthias.Behrendt, Albert.Albers}@kit.edu, {kangning.xing, shi.hanwen221}@gmail.com

***Abstract*— One of the key challenges in System of Systems Engineering is the validation of System of Systems (SoS). In a SoS the interfaces between the systems need to be defined as early as possible in the product development process. Contracts may help to define and specify interfaces in an early phase of the product development and support the validation of interfaces. This contribution applies existing methods, analysis and synthesis activities for the contract-based validation of interfaces between vehicle and infrastructure as an exemplary SoS. Hereby, Model-Based Systems Engineering supports the use of contract-based validation with specific methods and activities. Three different virtual and mixed physical-virtual validation environments - each for the Vehicle-to-Everything use case of an intersection scenario - enable the verification of the applied methods and activities. Two of these are virtual containing MATLAB and MATLAB-Network Simulator 3 Co-Simulation, respectively. The third is a mixed physical-virtual validation environment, including a driving robot. It realizes the exchange of information between a BMW i3 on a roller test bench and the vehicle simulation environment in CarMaker.**

Keywords—System of Systems, interface, validation, Vehicle-to-Everything (V2X), contract-based design

I. INTRODUCTION

The National Platform Future of Mobility in Germany describes the need for standardization, specification and standard tests for interfaces [1]. Regarding mobility, consisting of systems like the vehicle and the traffic light, as a System of Systems (SoS), the question arises of how to cope with interfaces of these systems within the SoS. This independent existence of constituent systems (CS), originally designed for different contexts as well as other specific characteristics of the SoS, impact the interfaces [2, 3]. Moreover, the validation, as one of the key challenges of SoS Engineering, needs to be taken into account. Based on already existing methods and activities, the main focus of this paper lies on contracts in SoS interfaces and the validation of interfaces in three specific validation environments.

Section II describes the main contributions in the areas of focus. It leads to the research gap of using contracts with the support of Model-Based Systems Engineering (MBSE) in the context of validation of interfaces and results in two research questions.

II. FUNDAMENTALS

According to the INCOSE Systems Engineering Vision 2025, in the system design for SoS, engineers need to develop and validate even more interconnected systems [4]. They have diverse stakeholders with increasing demands for services and information [4].
In the following sections, the three parts: (A) “Model-Based Systems Engineering”, (B) “Validation of interfaces” and (C) “Contract-based design” build the basis of this contribution and are therefore introduced.

A. Model-Based Systems Engineering for interfaces in SoS

Systems Engineering is an "interdisciplinary approach governing the total technical and managerial effort required to transform a set of customer needs, expectations, and constraints into a solution and to support that solution throughout its life" [5]. The formalized use of models to improve traceability and consistency can be achieved in MBSE. The three pillars of MBSE, language, method and tool, need to be considered in every usage [6]. The following approach is realized in the tool Cameo Systems Modeler in the Systems Modeling Language using different methods. Focusing on the three diagrams Block Definition (BDD), Internal Block Definition (IBD) and Views and Viewpoints Diagram, the modeled requirements and contracts are linked to different hierarchical system layers.

Various publications cope with the handling of interfaces in SoS with MBSE. The Unified Architecture Framework is based on the Object Management Group and provides different views to understand the relationship between organizations and systems. It is based on the Department of Defense (DoDAF) and British Ministry Department Architecture Framework (MODAF). It uses the views: metadata, strategic, operational, services, personnel, resource, security, project, standards and resources. [7]
Eichmann et al. focus on the specific aspect of stakeholders and requirements of SoS in the understanding of MBSE. This results in a methodology for stakeholder needs and requirements. [8]

However, existent approaches in the understanding of MBSE are not focusing on the validation of interfaces for SoS by using validation environments.

B. Validation of SoS interfaces

Honour analyzes the issues with verification and validation of SoS. He points out in validation of SoS the need of "Interface certification testing" [9]. Different approaches focus on activities like using matrices and rules for the validation of SoS.
Luna et al. propose the sequencing of Design Structure Matrices (DSM) and the use of graph theory. These optimize the information flows and path identifications. In addition, they introduce interface layers to support the integration of systems to analyze emergence effects within a SoS. [10]
The application of validation rules are introduced in Silingas & Butleris for software requirements in UML diagrams [11]. They describe the reuse of validation rules in the Object Constraint Language (OCL) to validate requirement models [11]. Companies may include their specific validation rules to make sure the interface complies with it. This is especially beneficial to SoS with multiple companies involved.

Other contributions develop frameworks, sets and methods to apply in the context of interfaces and validation of SoS. The general need for relations in SoS Engineering was identified in an online survey with 113 participants [12].
An ontology framework and process set is derived in the research project COMPASS - Comprehensive Modelling for Advanced Systems of Systems [13]. A validation view includes i.a. a constraint validation view. The ontology contains rules, which can be set for requirement types, goals, capabilities and requirements. [14, 15]
Bilal et al. propose an interface and interaction model for SoS to manage the interaction between subsystems. The interfaces transport a flow of data, energy or material and the models include effects, constraints and rules. [16]
Lollini et al. describe different viewpoints of a SoS's structure and behaviors. The authors focused exclusively on SysML stereotypes and suggested eleven concepts, each with ten to over 45 stereotypes. [17]

In order to plan validation activities as early as possible [18] and consider involved systems, the IPEK-X-in-the-Loop (IPEK-XiL) approach [19] can be applied. The `X' hereby represents the System-under-Investigation, which is in focus of a validation activity [20]. A validation environment is a specification of an operating system for validation with methods and resource systems [20]. It contains at least one combination of product and validation objective in a certain point in time of the product life cycle phase [20]. The validation environment of a SoS shows significant amounts of evolutionary development, operational independence and managerial independence [21], which are key characteristics of a SoS summarized by Honour [9].

Few contributions consider contracts in validating interfaces. Furthermore, our research applies concrete test cases in different validation environments.

C. Contract-based design

Contract-based design is a well-known Software Engineering methodology, which is widely used in design of component-based, safety critical real-time embedded and software systems [22].
A contract consists of "formalizations of the conditions for correctness of element integration […], and for lower level of abstraction to be consistent with the higher ones […]" [23].
Dragomir et al. define a contract for a component by the pairing of an assumption with a guarantee [24]. The assumption models an abstraction of the component's environment behavior and the guarantee models an abstraction of the component's behavior given that the environment behaves according to the assumption [24]. In the following contribution, the component refers to an interface in SoS.
Benveniste et al. state the need of contracts to handle complex systems, complex OEM supplier chains, addressing certification and the management of requirements and risks [25]. Böhm et al. describe the challenges in the development and evaluation of collaborative systems using simulation [26]. They differentiate between design and run time, as well as different levels of abstractions from business level to contract level [26].

Bryans et al. propose contract patterns for System of Systems [27]. Faldik et al. apply it for modeling interface contract behavior [28]. The authors use interface automata to analyze the compatibility between different contracts [29, 28].

In most cases, existing approaches tend to omit contracts or precise details of how to include them in the validation of interfaces for SoS. Hence, this contribution applies contract-based methods and activities with the support of MBSE to validate Vehicle-to-Everything (V2X) use cases.

III. Research Questions

The goal of this contribution is to structure the validation of interfaces in SoS with the support of MBSE and to apply and validate methods and activities. The following research questions are answered:

I. How can contracts modeled in SysML be integrated in the validation of interfaces between vehicle and infrastructure as an exemplary SoS?
II. How can existing contract-based methods, analysis- and synthesis activities be applied in the validation of interfaces for specific V2I scenarios?

IV. Methods and Activities for Validation of Interfaces in SoS

Section A describes an approach to structure interfaces for the validation of SoS with contracts. Section B applies selected analysis and synthesis activities in the mentioned area.

A. Contract-based structure for interface validation

In order to integrate contracts in the validation of interfaces, interfaces between constituent systems (CS) are identified (see Fig. 1). Subsequently, certain requirements and contracts are derived for each identified interface. The structure is based on the connection of the three architectures functional (FA), logical (LA) and physical (PA) [30, 31]. The contracts impact certain functions in the functional architecture. These functions relate to hardware and software in a logical and physical architecture.

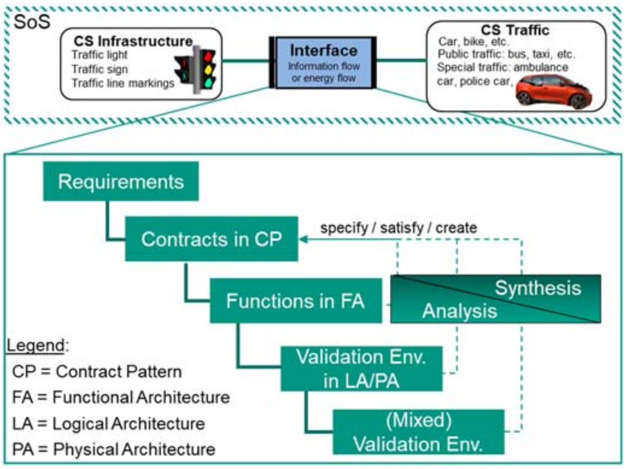

Fig. 1: Structure for the integration of contracts in the validation of interfaces for V2X

Deduced from the modelled architectures, the validation environment is built. Iterations enable the specification and evaluation of the interface with its architectures and validation environment(s). Using the pull principle of validation, the validation environment is not just built based on requirements and contracts. It also defines, identifies and adjusts certain requirements and contracts. The pull principle describes the development of the product with its validation as the starting point, by "pulling" validation elements into all the stages of the product design. Furthermore, the structure considers previous generations of elements like contracts, validation environments or products. This is described in the PGE - Product Generation Engineering (cf. [32]).

Faldik et al. propose the use of contract patterns in the context of SoS [28]. Different views on contracts allow their specification depending on their purposes [28].
Considering the SoS characteristics, the contract pattern is composed of several viewpoints, which are shown in TABLE I. In it, five viewpoints are listed for a contract between the CS Vehicle, which contains a BMWi3 and the CS Driver. The CS Driver contains a driving robot, an On-Board Unit (OBU) and an algorithm which transforms environment and vehicle information in set pedal positions. In the Contract Protocol Viewpoint, contract rules state the restrictions of every function, related to the contract of the interface in focus.

TABLE I. EXAMPLES OF CONTRACT PATTERN VIEWPOINTS

Name	Purpose of View
Contractual SoS Definition Viewpoint (CSDV)	Contract CS Vehicle-CS Driver
Contract Conformance Viewpoint (CCV)	This contract constrains the interface between CS Vehicle and CS Driver. The contract is fulfilled, only when both interface functions are satisfied.
Contract Connections Viewpoint (CConnV)	Interface function 1 is achieved by Driver OBU and Vehicle OBU; Interface function 2 is achieved by driving robot and pedal system.
Contract Definition Viewpoint (CDV)	Speed control: BUS sends out actual vehicle velocity at a fixed frequency and Driver OBU controls the driving robot using the velocity to a new velocity. State variables: Frequency for sending and receiving State invariants: Velocity in every cycle
Contract Protocol Viewpoint (CPV)	Contract rules for Function 1: a) Vehicle should send out the actual velocity in BUS. b) Driver OBU receives and processes the velocity every 400ms. c) reading frequency > sending frequency

An exemplary contract can be based on the Service Level Requirement with environmental factors like the Service Level Latency of 100ms (cf. [33]). This is taken as the assumption for the interface between vehicle and Road Side Unit (RSU), for the use case of an intersection movement assist. The guarantee states a minimal communication range of 100m and a reliability of 90% (cf. [33]).
Interface oriented layer models like OSI layers [34] allow the specification of requirements and contracts for interfaces more deeply. The number of layers necessary for modelling an interface depends on the interface with its contracts. This affects the design or selection of a validation environment, e.g. an explicit parametrization of the datalink layer according to 802.11p is necessary to test package transport loss rates between the communications of two RSUs. The structure allows to adress contracts in different interface layers modeled in FA, LA and PA.

An exemplary modeling of the structure with CP, LA and a PA of a mixed physical-virtual validation environment is shown in Fig 2.

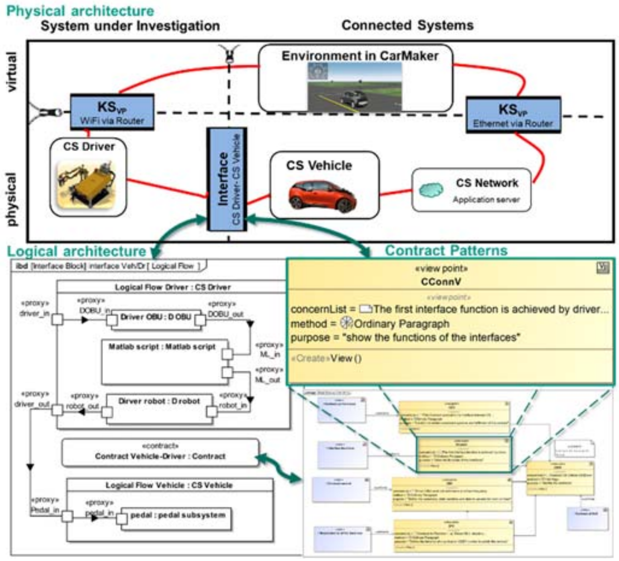

Fig. 2. Model of a XiL validation environment linked to an IBD model of the environment and a contract view pattern in a Views and Viewpoints Diagram

Fig. 2 visualizes involved systems in the validation. At the top is a model of the mixed physical-virtual validation environment of an IPEK-XiL-Architecture. The System-under-Investigation of the CS Driver and the interface to the CS Vehicle is connected to physical and virtual systems. The interface between CS Driver and CS Vehicle is modelled in an IBD and shown in the LA. Hereby, the LA does not only link CSs but also the subsystems involved in the interface functionalities. A contract for the information and energy flow between CS Driver and CS Vehicle is defined. The contract for the interface is further specified in five contract viewpoints in Fig. 2 (yellow boxes). The Contract Connections Viewpoint is shown as an example and the detailed contract viewpoints are in TABLE I. The contract impacts and is impacted by the interface in the IPEK-XiL-Architecture.

Based on the structure described above, analysis and synthesis activities are subsequently applied.

B. Analysis and synthesis activities in the context of interface validation

Analysis is a systematic investigation of an initial situation or result [35]. In contrast, synthesis elaborates and depicts solution alternatives for goals, based on an analysis [35]. The authors apply the activities “usage of validation rules”, “usage of time-based DSM” and “analysis of interfaces for selecting and developing a suitable validation environment”.

Different modeling tools enable the use of validation rules to verify models. The rules can be described in languages like OCL 2.0 and are bundled in validation suits. In the example of interfaces for SoS, company specific validation rules may be set up to verify interface models. In Fig. 3, the rules are applied with constraints on classifiers, so every (sub)system complies with the frequency range according to the European Telecommunications Standards Institute (ETSI) standard EN 302663 [36]. The standard describes the frequency range for cooperative Intelligent Transport Systems within the European Union for the intended usage of road safety related applications [36]. In the example, the contract is violated due to the frequency of the RSU of 5.906 MHz, which is higher than the allowed range between 5.875 and 5.905 MHz (cf. [36]). Despite technical rules, validation rules can restrict the model in specific architectures. For instance, the need to model a contract can be set as a rule if two CSs are connected.

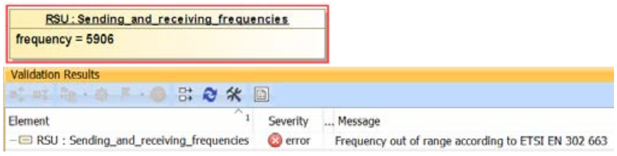

Fig. 3. Applied validation rule for frequency sending and receiving

The Design Structure Matrix (DSM) can show the interdependencies of elements like systems and interfaces within a SoS. It helps to analyze the impact between different interfaces. An interface-related DSM is generated with the following process:

1. Firstly, a DSM with functions according to FA or systems according to LA, PA are created.
2. Then the interdependencies of systems are analyzed (e.g. with a Sequence Diagram), considering information, energy and matter flow parameters like voltage or time.
3. The interdependencies of interfaces are further considered with parameter specific DSM based on the function involved. An exemplary time-based DSM, used for delay analysis of a speed control function, is used in the validation in section V-B TABLE II.

Different validation environments are suitable for different validation activities. The validation environment can be selected by using a criteria system, which is adapted from previous work in the area of distributed validation [29] and vehicle communication [37]. The criteria system consists of the five perspectives: technical, functional, organizational, user and economic. The criteria are defined based on test cases, e.g. in the V2X test case described in section V-A, 22 criteria are used. For IEEE 802.11p specific test cases, the Received Signal Strength Indicator is necessary to measure the physical layer and can be considered as performance criteria [38]. Moreover, the content of the contracts can be extracted as criteria in order to select a validation environment. For example, the data rate of the IEEE 802.11p standard and the latency between CSs can be part of the technical perspective in the criteria system.

Depending on the validation goal and the criteria, different environments are applicable.

V. VALIDATION OF METHODS AND ACTIVITIES

The validation is achieved with three validation environments in the context of IPEK-XiL validation.

The validation of an interface, either between two CSs or between their subsystems as Systems-under-Investigation (the "X" of XiL), considers a flexible modular kit development approach.

A. *Virtual validation environment*

The Vehicle-to-Infrastructure (V2I) scenario involves the interaction of the two constituent systems vehicle and infrastructure (including traffic lights with RSU). Contracts constraint and specify the behavior of their interfaces. In the scenario, the wireless communication must comply with certain contracts, which are based on standards like IEEE 802.11p or LTE-5G. In the performed V2I scenario, a vehicle is driving to an intersection and it receives controlled area messages of the traffic light state within a range of 100m. By using the tool in-build propagation delay model and propagation loss model, the MATLAB-NS3 Co-Simulation simulates the interface. Depending on the implemented model, different parameters like latency of the signal transmission and the reception power of the signal can be adjusted. For example, the frequency range of the signal transmission of controlled area messages is between 1 to 10Hz according to the ETSI standard EN 302 637-2 [38]. In ten test runs, the set frequencies of 1Hz and 2Hz in the log-distance propagation loss model lead to the different average distances between vehicle stopping point and traffic light of 8.2m and 5.2m, respectively. Therefore, contracts and contract-based activities, like using derived validation rules, may support the parametrization in the validation environment and are an integral part of the validation.

The criteria system, described in chapter III, helps to select a validation environment and hence certain models inside the environment. For instance, the detailed simulation of physical and datalink layer according to the OSI layer model lead to the decision to use the MATLAB-NS3 Co-Simulation.
Hereby, the validation environments address main SoS characteristics and needs like flexibility and modularity. Therefore, the authors use modular kit approaches including a library of V2X specific interface functions [39] and models in the environments.

In order to validate more realistically and include possible unpredictable events that cannot be represented in a model, physical hardware can be included in the validation environment.

B. *Mixed physical-virtual validation environment*

In the mixed physical-virtual validation environment, the V2I test scenario “Intersection Movement Assist with Vulnerable Road User” is applied. A stationary Host Vehicle drives a distance of 62 meters to an intersection. The validation goal of the Driver-Vehicle interface is to ensure the vehicle’s correct crossing of the intersection, avoiding any risk of collision. Contracts may state that the necessary operations are executed in the required sequence and that the data transmission is successful. The full realization of the interface is only achieved, when contracts for all the sub-functions (e.g. data transmission) in the FA of the respective interface are fulfilled.

The unidirectional time-based DSM in TABLE II shows the delay analysis of speed control function, referring to one test with the mixed physical-virtual validation environment.

TABLE II. Unidirectional time-based DSM for speed control in the V2I scenario with unexpected delay

Interfaces		1	2	3	4	5	6	7	8
Vehicle-Realtime System	1								
Realtime System-Router	2	<20ms							
Router-CarMaker	3		<20ms						
CarMaker-Router	4			<20ms					
Router-Driver OBU	5				<20ms				
Driver OBU-MATLAB	6					1000ms			
MATLAB-Driving Robot	7						<500ms		
Driving Robot-Vehicle	8					<1522ms		10ms	

The delay of the interface between Driver On-Board-Unit (OBU) and MATLAB is set to 400ms in the contract, which depends on the response time of the driving robot. When this delay exceeds the set value in the contract, the delay adds on to the following interfaces and the algorithm of the speed control is therefore disturbed.

Fig.4 shows the result of the same test in TABLE II with the mixed physical-virtual validation environment.

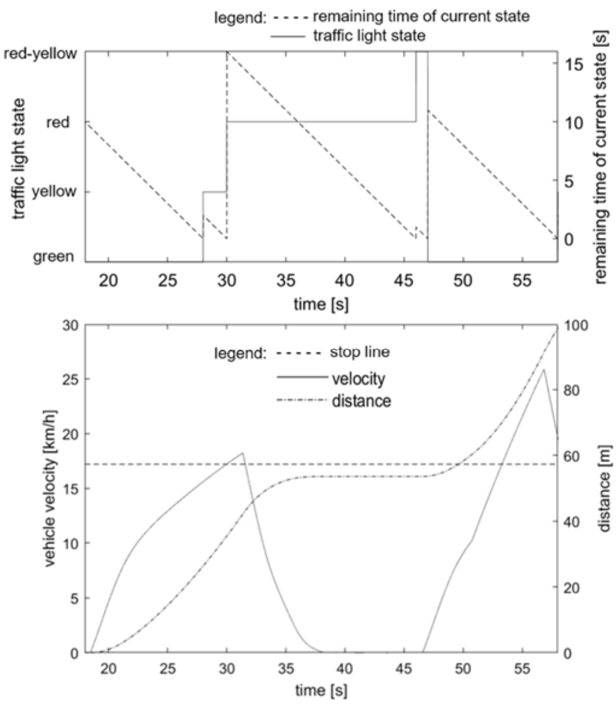

Fig. 4. Validation of the Driver-Vehicle interface showing a velocity, time and distance plot and traffic light states

In this test, the delay in the interface between Driver OBU and MATLAB is set to 1000ms, which exceeds the normal delay of 400ms in the contract. Previous work analyzed the delays of wireless hardware connections in a mixed physical-virtual validation environment [40]. In the test in Fig. 4, the vehicle stops after 53.6m, when the stop line is still 3.7m in front of the vehicle.

In Fig. 4 the higher latency leads to a disarrangement of the speed control algorithm, which results in a premature breaking. This breach of the contract is manifested by the no longer valid assumption that the information flow of the interface between CS Driver and CS Vehicle is within a restricted scope.

In contrast to previous work [21] with a human driver, using a driving robot results in a smoother graph. The integration of the driving robot in the validation environment was achieved with a remote control of the driving robot through MATLAB and the Wi-Fi module ESP8266.

VI. Discussion and Conclusion

This contribution integrated contracts in existing structures for the specific purpose of interface validation. This allows analyzing and synthesizing of (validation) systems with the consideration of contracts for interfaces. MBSE supports the integration of contracts by modeling different views, linking architectures with methods and including automated verification of models. Different activities like using parameter-tailored DSM or criteria systems for the selection of a validation environment are applied and validated for V2I specific use cases.

Contracts may help to cope with SoS-specific challenges like the different life-cycle phases, operational and managerial independence of CSs. With defined interface contracts, new system owners know from the very start of their development process, which restrictions to consider for a successful integration of their CS in an existing SoS. Contract-based activities can be tailored to support these challenges and the validation of interfaces.

The validation in this contribution is only applicable to specific test cases for V2I. Therefore, a generalization for all technical SoS is not achieved. The authors evaluate the validation environments based on the SoS characteristics (c f. [9]). Future work focuses on quantifiable requirements derived from SoS properties and characteristics. MBSE may support the realization and beneficial application of contracts. Using contracts for interfaces in SoS can be extended by considering more methods and activities. Further research should address safety and security within the CSs and interfaces which impact the whole SoS performance (c f. [41]). Additionally the validation environments should be part of an integrated, continuous and flexible validation of models. This leads to an efficient validation of even complex SoS.

[1] Nationale Plattform Mobilität der Zukunft, „SCHWERPUNKT-ROADMAP AUTOMATISIERTES UND VERNETZTES FAHREN“: Arbeitsgruppe 6 „Normung, Standardisierung, Zertifizierung und Typgenehmigung“, Ed., Berlin, May 2020, Bundesministerium für Verkehr und digitale Infrastruktur.

[2] J. Leite, F. Oquendo, and T. Batista, “SysADL: A SysML Profile for Software Architecture Description,” in Lecture notes in computer science, vol. 7957, Software Architecture: 7th European Conference, ECSA 2013, Montpellier, France, July 1-5, 2013. Proceedings, K. Drira, Ed , Berlin, Heidelberg: Springer Berlin Heidelberg; Imprint; Springer, 2013, pp. 106–113.

[3] T. Batista, “Challenges for SoS architecture description,” in SESoS 2013: Proceedings of the first ACM SIGSOFT/SIGPLAN International Workshop on Software Engineering for Systems-of-Systems : July 2, 2013, Montpeller, France, Montpellier, France, 2013, pp. 35–37.

[4] International Council on Systems Engineering (INCOSE), Ed., “INCOSE Systems Engineering Vision 2025”, 2014.

[5] ISO/IEC/IEEE24765, “ISO/IEC/IEEE24765:2017: Systems and software engineering — Vocabulary,” 2017, doi: 10.1109/IEEESTD.2010.5733835.

[6] L. Delligatti, “SysML distilled: A brief guide to the systems modeling language”. Upper Saddle River, NJ: Addison-Wesley, 2014. [Online]. Available: http://proquest.tech.safaribooksonline.de/9780133430356

[7] OMG Object Management Group, Ed., “UAF Unified Architecture Framework”.

[8] O. C. Eichmann, S. Melzer, F. Giertzsch, and R. God, “Stakeholder Needs and Requirements Definition During Service Development in a System of Systems,” in 2020 IEEE International Systems Conference (SysCon).

[9] E. Honour, “Verification and Validation Issues in Systems of Systems,” in Electron. Proc. Theor. Comput. Sci., vol. 133, no. 4, pp. 2–7, 2013, doi: 10.4204/EPTCS.133.1.

[10] S. Luna, A. Lopes, H. Y. S. Tao, F. Zapata, and R. Pineda, “Integration, Verification, Validation, Test, and Evaluation (IVVT&E) Framework for System of Systems (SoS),” in Procedia Computer Science, vol. 20, pp. 298–305, 2013, doi: 10.1016/j.procs.2013.09 276.

[11] D. Silingas and R. Butleris, “Towards implementing a framework for modeling software requirements in MagicDraw UML,” Information technology and control, vol. 38, 2009.

[12] A. Kurrle, „Durchgängige Dokumentation von verteilten Zielsystemen in der Produktentwicklung durch Verwendung semantischer Metainformationen am Beispiel Connected Car,“ 2017.

[13] J. Holt, S. Perry, M. Brownsword, D. Cancila, S. Hallerstede, and F. O. Hansen, “Model-based requirements engineering for system of systems,” in 2012 7th International Conference on System of Systems Engineering (SoSE), Genova, Jul. 2012 - Jul. 2012, pp. 561–566.

[14] P. Weinmeister, “Supporting Your Business with Validation Rules,” in Practical Salesforce Development Without Code: Apress, 2019, pp. 179–199.

[15] Europäische Kommission, “COMPASS Research Project,”. Accessed: Jan. 22 2021. [Online]. Available: https://cordis.europa.eu/project/id/287829/de

[16] M. Bilal, N. Daclin, and V. Chapurlat, “System of Systems Design Verification: Problematic, Trends and Opportunities,” in Enterprise Interoperability VI: Springer International Publishing, 2014, pp. 405–415.

[17] Lollini, Paolo; Mori, Marco; Babu, Arun; Bouchenak, Sara, “AMADEOS SysML Profile for SoS Conceptual Modeling,” in Cyber-Physical Systems of Systems: Springer International Publishing, 2016, pp. 97–127.

[18] C. Hood, S. Wiedemann, S. Fichtinger, and U. Pautz, “Requirements Management: The Interface Between Requirements Development and All Other Systems Engineering Processes”. Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg, 2008. [Online]. Available: http://site.ebrary.com/lib/alltitles/docDetail.action?docID=10217695

[19] A. Albers, T. Düser, O. Sander, C. Roth, and J. Henning, “X-in-the-Loop-Framework für Fahrzeuge, Steuergeräte und Kommunikationssysteme,” in ATZelektronik, vol. 5, no. 5, pp. 60–65, 2010, doi: 10 1007/BF03224034.

[20] A. Albers, M. Behrendt, S. Klingler, and K. Matros, “Verifikation und Validierung im Produktentstehungsprozess: Handbuch Produktentwicklung,” in Udo Lindemann, Ed., München: Carl Hanser Verlag, 2016, pp. 541–569.

[21] M. Wäschle, Y. Jiang, M. Behrendt, and A. Albers, “Development of an integrated validation environment for System of Systems in the context of V2X using the XiL-Approach,” in IEEE, 2020.

[22] S. S. Bauer et al , “Moving from Specifications to Contracts in Component-Based Design,” in Fundamental Approaches to Software Engineering: Springer Berlin Heidelberg, 2012, pp. 43–58.

[23] A. Sangiovanni-Vincentelli, W. Damm, and R. Passerone, “Taming Dr. Frankenstein: Contract-Based Design for Cyber-Physical Systems*,” in European Journal of Control, vol. 18, no. 3, pp. 217–238, 2012, doi: 10.3166/ejc.18.217-238.

[24] I. Dragomir, I. Ober, and C. Percebois, “Contract-based modeling and verification of timed safety requirements within SysML,” in Software & Systems Modeling, vol. 16, no. 2, pp. 587–624, 2017, doi: 10.1007/s10270-015-0481-1.

[25] Albert Benveniste, Benoit Caillaud, Dejan Nickovic, Roberto Passerone, Jean-Baptiste Raclet, et al.: “Contracts for System Design”. [Research Report] RR-8147, INRIA. 2012, pp.65., hal-00757488, RR-8147.

[26] W. Böhm, M. Broy, C. Klein, K. Pohl, B. Rumpe, and S. Schröck, “Model-Based Engineering of Collaborative Embedded Systems,” in Cham: Springer International Publishing, 2021.

[27] J. Bryans, J. Fitzgerald, R. Payne, A. Miyazawa, and K. Kristensen, “SysML contracts for systems of systems,” in 2014 9th International Conference on System of Systems Engineering (SOSE).

[28] O. Faldik, R. Payne, J. Fitzgerald, and B. Buhnova, “Modelling System of Systems Interface Contract Behaviour,” in Electron. Proc. Theor. Comput. Sci., vol. 245, no. 5, pp. 1–15, 2017, doi: 10.4204/EPTCS 245.1.

[29] L. de Alfaro and T. A. Henzinger, “Interface automata,” in Proceedings of the 8th European software engineering conference held jointly with 9th ACM SIGSOFT international symposium on Foundations of software engineering - ESEC/FSE-9, 2001.

[30] Jesko G. Lamm and Tim Weilkiens, “Functional Architectures in SysML,” in M. Maurer and S.-O. Schulze (eds.), Tag des Systems Engineering 2010, pp. 109–118. Carl Hanser Verlag, München, Germany, November 2010. English translation by J. Lamm.“,” in Tag des Systems Engineering (TdSE 2010), 2010. Accessed: Feb. 13 2021. [Online]. Available: https://www.oose.de/wp-content/uploads/2016/10/110427_TdSE2010_Lamm_Weilkiens_Functional_Architectures_English-1.pdf

[31] K. Pohl, H. Hönninger, R. Achatz, and M. Broy, “Model-Based Engineering of Embedded Systems: The SPES 2020 Methodology”. Berlin, Heidelberg: Springer, 2012. [Online]. Available: http://site.ebrary.com/lib/alltitles/docDetail.action?docID=10625072

[32] A. Albers, N. Bursac, and E. Wintergerst, ″ Produktgenerationsentwicklung - Bedeutung und Herausforderungen aus einer entwicklungsmethodischen Perspektive,″ in Stuttgarter Symposium für Produktentwicklung 2015 SSP 2015, 2015.

[33] 5GAA, “C-V2X Use Cases, Methodology, Examples and Service Level Requirements”. [Online]. Available: https://5gaa.org/wp-content/uploads/2019/07/5GAA_191906_WP_CV2X_UCs_v1-3-1.pdf

[34] ISO/IEC 7498-1:1994, “Information technology -- Open Systems Interconnection -- Basic Reference Model: The Basic Model,” ISO/IEC, 1994.

[35] VDI 2221 Blatt 1, “Entwicklung technischer Produkte und Systeme,“ 2019.

[36] ETSI EN 302 663: “Intelligent Transport Systems (ITS); ITS-G5 Access layer specification for Intelligent Transport Systems operating in the 5 GHz frequency band,” ETSI EN 302 663 V1.2.1, ETSI, May. 2019. [Online]. Available: https://www etsi.org/deliver/etsi_en/302600_302699/302663/01.02.01_30/en_302663v010201v.pdf

[37] R. Protzmann, I. Radusch, A. Festag, R. Fritzsche, and M. Rehme, “IV2X - Integrierte Betrachtung Fahrzeugkommunikation,” Berlin, 2018. Accessed: Jan. 23 2021. [Online]. Available: https://www.testfeld-berlin.de/pdfs/iV2X%20Dokumentation%201.5.pdf

[38] ETSI EN 302 637-2: “Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative; Awareness Basic Service,” ETSI EN 302 637 V1.3.1, ETSI, Sep. 2014. [Online]. Available: https://www.etsi.org/deliver/etsi_en/302600_302699/30263702/01.03.01_30/en_30263702v010301v.pdf

[39] S. Chen, “Modellbasierter Ansatz in der Entwicklung einer Validierungsumgebung für V2X,” unpublished Masterthesis, IPEK - Institut für Produktentwicklung, Karlsruher Institut für Technologie (KIT), Karlsruhe, 2020.

[40] L. Eberle, “Weiterentwicklung einer Validierungsumgebung für die Entwicklung von kamerabasierten Fahrerassistenzsystemen zur Anwendung an verteilten Standorten,“ IPEK-thesis; 3790, IPEK - Institut für Produktentwicklung, Karlsruher Institut für Technologie (KIT), Karlsruhe, 2019.

[41] Causevic, Aida, “A risk and threat assessment approaches overview in autonomous systems of systems,” in 2017 XXVI International Conference on Information, Communication and Automation Technologies (ICAT): IEEE.