Modeling of Requirement-Based Effect Chains of Mechatronic Systems in Conceptual Stage

Mohamad W. Chamas
BMW Group, Munich, Germany
Email: Mohamad.Chamas@bmw.de

Kristin Paetzold
Universität der Bundeswehr München, Munich, Germany
Email: kristin.paetzold@unibw.de

***Abstract*—As a result of new challenges such as E-mobility and autonomous driving in the automotive industry, the amount of functions in vehicles increases rapidly. Accordingly, the development of mechatronic systems needs new methods towards a function oriented way in order to reduce the complexity. Due to the typical bottom-up development in the automotive sector the aspect of vertical traceability in requirement engineering is incomplete. This paper presents a new methodical concept for modeling of effect chains to support the domain independent view for functional modeling in conceptual stage. Based on effect chains a systematic and structured approach is defined to derive functional requirements. In conclusion, this builds the basis for a more efficient development of functional requirements in conceptual stage.**

***Index Terms*—conceptual design, effect chains, functional requirements, model-based systems engineering**

I. Introduction

During the last decades the automotive sector belonged to a bottom-up development branch in which hardware and software grew up separately. Components and software were developed without having any information about cross-impacts regarding customer functions. Mostly, effects of failures were discovered after the integration process or in the worst case by the customer. A further reason for late detection of failures is the increasing connectivity in the vehicle due to new technologies and services. More and more Electronic Control Units, cameras and sensors are needed to receive a huge amount of data. A change of mindset must occur because the connectivity in the car through functions based on software has been increasing exponentially over the last 30 years and there is no end [1]. Another related aspect is that early stages of product development defines the clarification and definition task. Mostly, there is less and vague data at these stages and additionally there are no common methods with computational support for the conceptual stage. Most approaches suffer from a limited to a narrow point of view by the developers. This leads to higher costs and iteration loops in following development stages.

Manuscript received March 21, 2018; revised June 25, 2018.

In order to encounter these challenges, new methods has to be developed which are based on a function oriented way. Different aspects have to be addressed: enhancing the top-down development, increasing the transparency to support the system understanding and applying in an interdisciplinary workflow.

The paper presents an approach for developing mechatronic systems in early design steps to collect an adequate content of information for series development. The focus is to introduce a method of systematic modeling by consolidation of vertical traceability of functional requirements as well as the resulting increase in system understanding. Therefore modeling of effect chains helps to develop functional requirements in a conclusive and traceable way.

First, the paper starts with the literature review and problem definition. Second, an approach for modeling requirement-based effect chains is then presented. The approach is validated through the case study of a customer function of the powertrain called *supply wheel torque*. Concluding the details on the implementation are discussed.

II. Literature Review

Effects can be thought as an emerging phenomenon between behavior and design [2]. Moreover an effect can also be expressed by input and output relations [3]. Connecting single effects to a chain leads to an effect chain. Becker [4] defined the effect chains as a sequence of actions, which are executed to fulfill a required function.

Thereby the effect chains are subject to the principle of causality. Regarding the definitions it is hard to differ between the meaning of a function and an effect. Often, the term effect chains is also used in the section of sensors and actuators. Therefore, high effort in analyzing of effect chains is made, e.g. real-time consideration, timing delays of controllers or any signal flows [5].

The function oriented development has several advantages. The main ones are solution independent, modularization in functional groups for reuse in other concepts and an extended solution space. In order to

doi: 10.18178/ijeetc.7.3.127-134

support a function oriented development in early development stages, the effect chain architecture helps to join the view of hardware and software. Especially developing mechatronic systems profits by modeling of effect chains because a functional view can be built without concerning on the specific domains. Another issue is the abstraction of components which has no directly interaction to the function, e.g. microcontroller (software), casing (hardware) or any other non-functional aspects, e.g. colors.

Göpfert proposes a decomposition of the customer function until components can be associated and a reversal aggregation of the components leads to the product architecture [6]. Another way of description is the effect chain architecture. Alt describes the effect chain architecture as a chain of system components [7]. This approach does not support a solution independent way of developing. Components and their relations will be designed in a static view without having further information about any behavioral influences. Additionally building up functional requirements by following the effect chains is not discussed yet.

In this research a new approach is explained how effect chains in conceptual stage could be developed in a more solution independent context and accompanied by deriving new functional requirements respecting static and dynamic behavior. Traceability of requirements is the connection of all information over the whole lifecycle, e.g. test cases or maintenance [8]. However, traceability is here used for the connection between functional requirements with a logical cause-effect relationship.

III. Problem Definition

A. Conceptual Design Process

There are many approaches in the field of systematic product development. One of the most well-known methods with a general description in the beginning to more precisely and detailed methods for realizing the product was given by Pahl [9]. Conceptual design has an important role on other processes especially in terms of time and quality, VDI (Verein deutscher Ingenieure) determined the guideline 2221 "Systematic Approach to the Design of Technical Systems and Products" [10]. Due to the fact that the VDI 2221 shows a sequence procedure and does not point out a multilevel development, the VDI 2206 offers with its V-Model a suitable completion. In Fig. 1 the V-Model is illustrated in a slightly different form. An extension to the VDI 2221 is the consideration of the different abstraction levels as well as adding the integration aspect to support the development processes of mechatronic products [11].

In common, all product development processes have the development of requirements in the conceptual stage by the creation of requirement lists [9], [10] or breaking requirements down from system to subsystem level [11]. Thus the output of the conceptual stage in the design process are requirements. Typically, these requirements are document-based, so that developing complex systems is almost impossible. Due to the many product development processes established in academia and industry, the computer-aided support for conceptual design is still improvable.

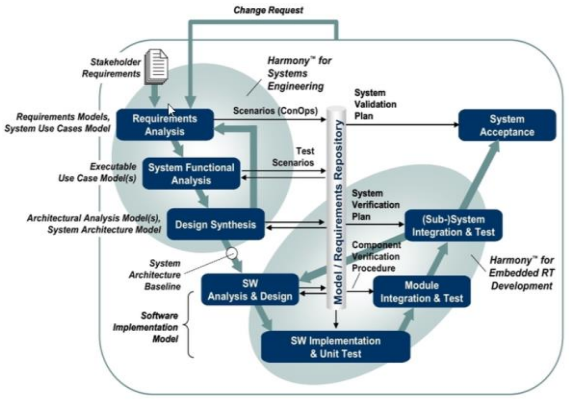

Figure 1. IBM-Harmony V-Model [13].

B. Model-Based Systems Engineering

Systems engineering focusses on interdisciplinary development of complex systems. To support the cross domain aspect, INCOSE (International Council on Systems Engineering) has introduced the model-based systems engineering (MBSE). INCOSE is an organization for research activities and technological progresses in the field of systems engineering. MBSE is defined by INCOSE as follows: "*The formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases.*" [12]

Regarding the task of developing system requirements in conceptual design phase, requirements are naturally text-based and stored in distributed places (e.g. databases, documents). Therefore no links between requirements are given, so that the traceability is lost. This makes the workflow between stakeholders more difficult. MBSE solves the problems of document based approaches by creating a single point of information of the system model, especially in early phases of product development. These model-based approaches are the basis for an interdisciplinary communication and cooperation considering further activities like testing and other technical management aspects, e.g. reliability analysis.

By reasons of many applications of MBSE and no strictly guidelines, there are evolved lots of different methods. Estefan [13] investigated amongst others the numerous methodologies. The almost closed approach and most important one for this research is IBM Telelogic Harmony-SE. Harmony-SE is only a part of a larger process called Harmony and mirrors the typical V-model of system design. The bubble in the upper left corner in Fig. 1 describes the conceptual stage and has following objectives [13]:

- Identify/derive required system functionality
- Identify associated system states and modes
- Allocate system functionality/modes to a physical architecture

First, the requirement analysis starts with input requirements by stakeholders or existing requirements

from earlier projects. All requirements will be collected and then categorized in functional and non-functional requirements.

Further, functional requirements will be considered while the non-functional requirements can be allocated later to architecture elements in design synthesis. Afterwards, system functional analysis explains the behavioral model for the main function in the system. In design synthesis, the behavior models can be decomposed into the identified subsystems. When the system architecture baseline is reached, then the model of the system engineer is committed to hardware or software development. The second bubble is not viewed because these steps belongs to the series development.

To enable model-based development as well as work in an interdisciplinary team a suitable tool has to be introduced which supports the conceptual design process and is accepted by the community.

C. System Modeling Language

System Modeling Language (SysML) is a visual modeling language developed by the Object Management Group together with INCOSE. SysML has the main goal to support MBSE and derives from Unified Modeling Language, which is commonly used in software development. [14]

SysML represents notation and semantics in nine different diagrams including categories requirement, behavior and structure. Fig. 2 illustrates the outlined diagrams used for this approach.

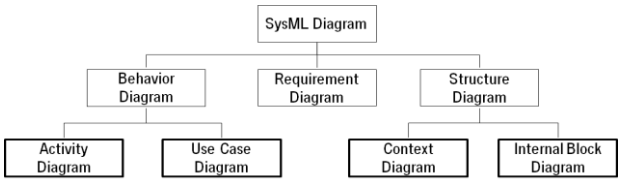

Figure 2. SysML diagram types [14].

A use case diagram in SysML belongs to the behavior diagrams and is used to represent the interaction between actors, which can be persons or even other systems. Representing the user perspective in combination with environmental and functional aspects is the main expression of these diagrams.

A context diagram is a specific block definition diagram that represents the system as a black box. All interfaces between the system and the external systems are listed to explain the inputs and outputs transformed through the customer function. The internal block diagram suggests a substructure for the high-level block. It contains the properties, interfaces as flow ports and connections through item-flow-connectors.

One of the most favorite behavior diagrams are activity diagrams. A sequence of actions allows to describe a function. Further, it is possible to make decisions by defining guard conditions. Therefore, a dynamic view onto a function could be made [7].

Following, SysML provides different relationships for modeling requirements:

- Derive requirement relationship signals that a derived requirement is generated from a source requirement
- Satisfy requirement relationship exists between requirements and design elements (e.g. hardware, software) so that a requirement is clearly concerned
- Refine requirement relationship is here used in context with use cases to increase the transparency
- Trace relationship shows a dependency between requirements

IV. Approach

Although there is no modeling standard or restrictions for usage this modeling language, SysML is accepted by most users [15]. Hence, there are no MBSE solutions that fits for every problem. Therefore a new guideline has to be draft to evolve the effect chains associated with its functional requirements to achieve a systematic development.

A. Framework

The fundamental procedure of building up effect chains is explained in this section and a scheme is illustrated in Fig. 3. Starting with customer requirements of different sources, i.e. driven by the market, strategy of the company, laws or competitors leads to a development project of a product or function. Afterwards, the customer function is designed in a multiperspective way to enable an entire view. The process contains three steps: functional, behavioral and structural. The Function-Behavior-Structure triplets together can be translated into an effect. By connecting the triplets accordingly to the logical process of the customer function emerges an effect chain. Hence, the effect chain carrying functional requirements (FR) causes a functional requirement-based effect chain. By repeating this procedure for different customer functions would generate more effect chains.

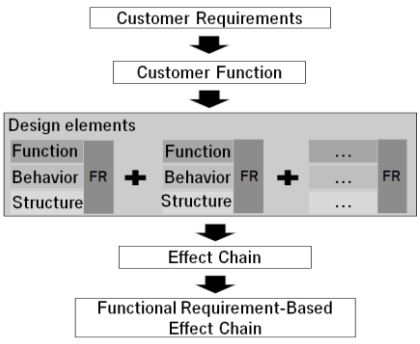

Figure 3. Procedure of modeling an effect chain for resulting a functional requirement-based effect chain.

B. From F-B-S Method to Effect Chains

Modeling of effect chains needs more than only a single view. In this respect effect chains are based on the comprehension of the Function-Behavior-Structure (F-B-S) by Gero [16]. The F-B-S method contains three

processes: the functional, behavioral and structural modeling.

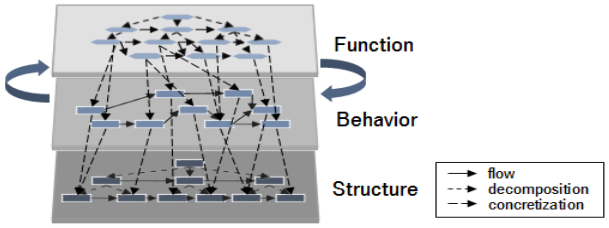

Figure 4. F-B-S layers – own representation based on [16].

Fig. 4 illustrates three single layers connected through different edges. The edges *concretization* is the transition of different levels. *Decomposition* is the division of a parent node into child nodes. The edge *flow* shows a sequence of the single nodes. Because there are several definitions of the terms function, behavior and structure found in literature, they are explained here in this context.

The function level describes the task, so that the designer's purpose is achieved. Often the top-down decomposition technique is used to detail the function in smaller and detailed low-level functions. Usually the function level is solution independent. Pahl [9] characterized functions by input and output relations with verb-noun pairs. Thus a function describes the transformation from input to output flows and is expressed as a verb. The noun describes the function data. Function data is always measurable and can hold a value with the appropriate unit, e.g. mass or temperature. The function level shows a hierarchy achieved through the decomposition of the main function into subfunctions.

The behavioral level is the connector between functions and structures. The causal relationship between the input and output flows is due to the behavior. These elementary blocks defines the working principles. However, a behavior is a dynamic description of the system in form of a sequence of working principles or definitions of states. A behavior can be associated after repeated decomposition of a customer function until no further division is possible and the physical effect occurs [17]. Terninko proposed a collection of behavioral effects for the fulfillment of functions, e.g. for moving an object a thermal expansion could be one possible physical effect [18].

The lowest level is the structural one, which is the most concrete stage. The derived behavioral elements needs to be fulfilled by the components to achieve the functions. Thus the product tree is finally generated if the embodied required behaviors are satisfied and therefore the required functionality as well. The structural level keeps fixed with a set of subsystems. The difference in the structure is characterized through the variable connections and information flows between the subsystems due to the viewed function.

The iterative process between function and behavior, shown in Fig. 5, prevents the extension of the dimensions in the behavioral level and helps to develop the function step by step respecting the exact abstraction level. A behavioral node can be concretized while the system function analysis step, but then it is transformed from a behavioral node to a subfunction, which appears in the function tree. When function analysis is done, design synthesis follows:

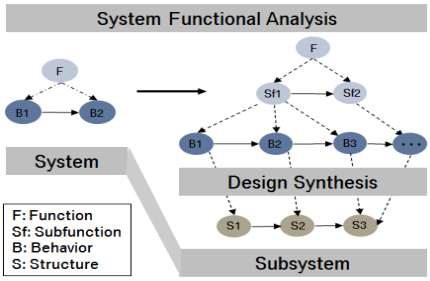

Figure 5. Building up the functional view by turning behavior nodes into subfunction and iterated decomposition in new behavioral sequence.

The aim of this three different abstraction levels focuses on the overall view. From functional across behavioral to structural perspective, there is an adequate rate of information to describe an effect chain. This chain of causality consists of the triplet F-B-S: the functional blocks and its relations, the logical sequence of behavioral elements and the port definitions between the structural elements. Fig. 6 shows the metamodel in which it is obvious that requirements plays a certain role and is throughout existing due to the vertical traceability. The metamodel is built with the aim to enhance the development of functional requirements. On the one hand requirements are refined through use cases and satisfied by influencing factors to clarify and define the task on customer level. On the other hand requirements are satisfied to functions and subfunctions as well as by the structure, which can be decomposed into system and subsystems. Additionally requirements traces to behavior nodes.

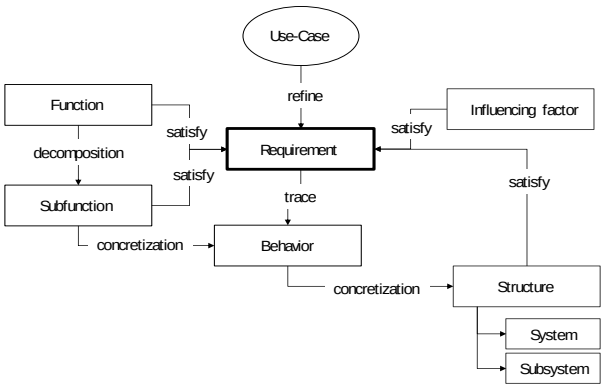

Figure 6. Metamodel.

C. From Effect Chains to Functional Requirements Chains

The next step is to evolve requirements to design elements and put them into relations for receiving the traceability. Functional requirements are used to document the operations that a system must be able to perform. Rupp specified a suitable template for expressing those requirements [19]. Fig. 7 illustrates the template for semantics and syntax of requirements. In this context, dark colored blocks were used to reach a common structure of the requirements in syntax and

semantics as well as defining hard ones. Hard requirements have no other solution space, but rather a forcing character to require something. The brighter blocks are used to express soft requirements. The first block is optional and can be used to define time or event-based dependencies regarding systems if any conditional clauses should be addressed. Next one is addressing the belonged system for whom the requirement is valid. The term shall declares hard requirements. According to Pahl [9], process verb and object are defined due to functions as verb-noun pairs. Here, object is comparable to the definition of function data. Putting the requirements in sequence as their referred design elements allows to treat them as requirement-based effect chains.

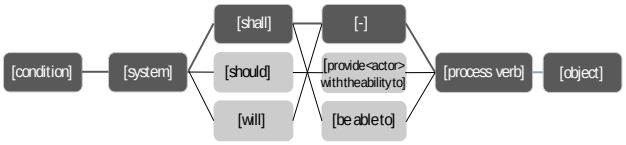

Figure 7. Template for developing requirements – own representation based on [19].

Using this template, different perspectives can be addressed. In the functional view instead of a system, a function or subfunction can be declared in the template. By using this pattern, all information are covered in the same way, independent neither from its abstraction, functional, behavioral or structural level.

V. Case Study

The example used for this investigation is a fundamental customer function of powertrain called *supply wheel torque*. The effect chain based on the customer function proceeds from drivers demand till conversion at the wheel is supplied by the powertrain. Further, the F-B-S triplet *detect driver command* cuts out the modeling steps regarding the different stages.

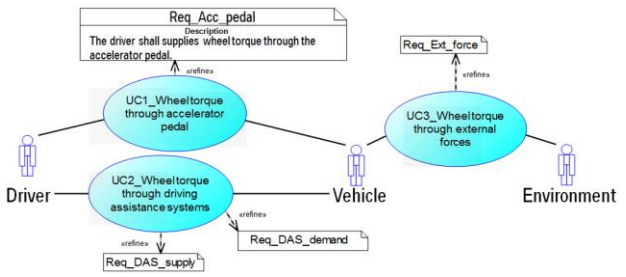

Figure 8. Use case diagram for *supply wheel torque.*

A. First Stage – Requirement Analysis

Assuming, that there were not any projects before, new requirements has to be developed without any preprocessing steps by refining or categorizing requirements.

First, a use case diagram is used to define the customer function in context with the actors driver, vehicle and environment. Fig. 8 illustrates three use cases: accelerator pedal, driving assistance systems, external forces. Every use case refines requirements in consideration of the requirement template by Rupp [19]. Following, use case (UC 1) is considered in detail: *The driver shall supplies wheel torque through the accelerator pedal (Req_Acc_pedal).*

Use cases guarantees the completeness of the requirements for the viewed customer function, so that every requirement in more concrete levels has the origin in the use cases. Contrary, if a requirement is developed in the bottom levels and no use case can be addressed, then it signals that a use case is missing or the requirement has to be reviewed.

After dealing with the user perspective it is required to specify the system boundaries and moreover to treat the system as a black box. Therefore the context diagram is used to identify the influence factors in the shape of information flows between the system and the external actors. Thus every input and output influences for the customer function is determined.

In Fig. 9, all information flows are listed, which has an influence on the UC 1 executed by the powertrain. The powertrain needs three information for transforming torque: accelerator value, gear value and the actual velocity. Again, the information flows are documented with requirements.

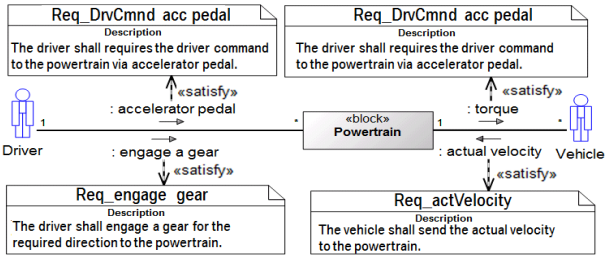

Figure 9. Context diagram for UC 1.

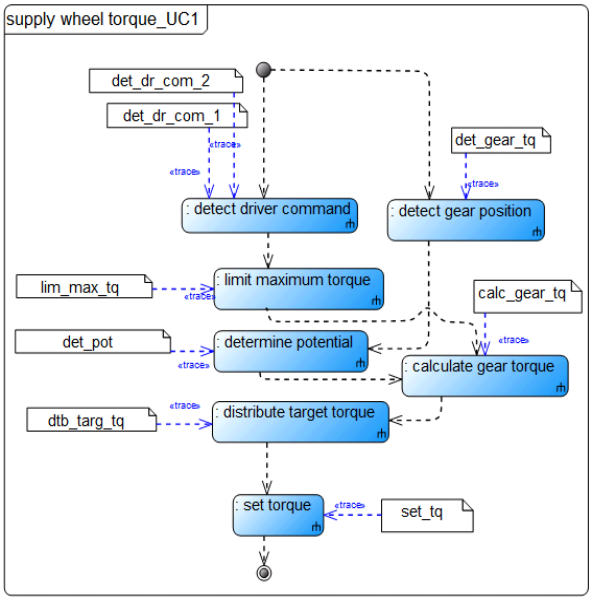

Figure 10. Activity diagram of the behavior model UC 1.

B. Second Stage – System Function Analysis

After the inputs and outputs of the powertrain have been identified, the behavioral model is built. In Fig. 10 UC 1 is broken down by a sequence of behavioral

elements. Every activity is traced by functional requirements. Beginning with the inputs *detect driver command* and *detect gear position* from the context diagram, a sequence of actions is described. Because of safety criteria, the requested torque by the driver must be limited due to conditions of the ground. The driver demand could be higher than it is possible for the system. Therefore an alignment step should be respected in which the potential of the system is recognized. Afterwards the calculation of the gear torque depends from the potential of the system as well as from the limited torque. In cases of an all-wheel powertrain, the target torque has to be distributed on both driving axles. Then, torque can be set at the wheels.

C. Third Stage – Design Synthesis

In the next step the behavioral nodes from the system function analysis are transformed into subfunctions. In Fig. 11 subfunctions of UC 1 are shown. For this purpose an internal block diagram is used.

This representation of the functional blocks has several benefits: clustering and reducing the complexity, well understanding representation and the needs of information of every functional block and their relations. The flow-ports satisfies requirements. For better visualization only three requirements are inserted. Transforming the behavior nodes into subfunctions represented in a static diagram supports the requirement definitions for the interfaces between the subfunctions. The internal functional requirements for the functional blocks are already described in the activity diagram from the system functional analysis.

Following, the functional block *detect driver command* is investigated again in a behavioral model for design synthesis and allocation to subsystems (Fig. 12). Elements in grey belongs to external functions addressing other stakeholder outside of the powertrain, e.g. stability control system (SCS) or driver assistance system (DAS). Additionally, guard conditions can be defined if any decisions has to be done. In this example, if DAS is used (DAS = 1), then the right path is chosen from the initial point. Since we refer to the UC 1, we expect that the accelerator pedal is used (accelerator = 1 & DAS = 0) and the control flow in the middle is activated. Therefore the requested torque by the driver has to be detected by determining the pedal value. Also for calculation the *driver command* the actual velocity is provided by SCS. Then, the requested torque by DAS and accelerator pedal is accepted and the maximum is taken.

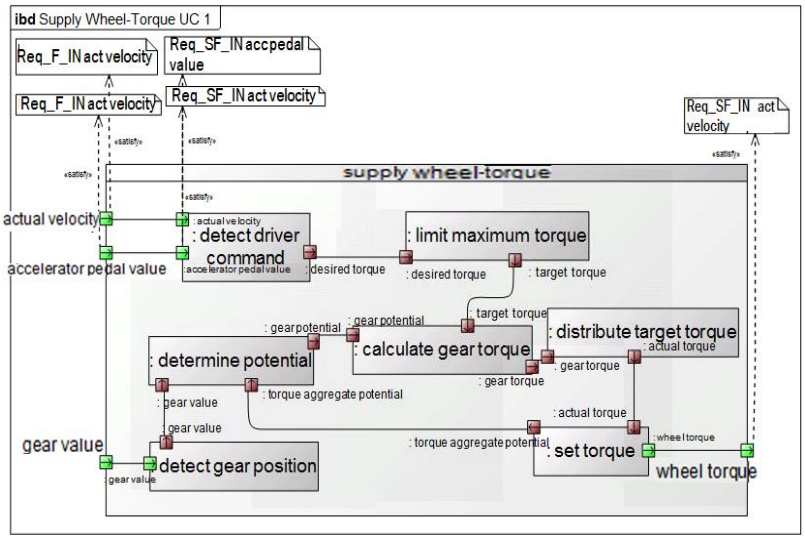

Figure 11. View of the functional blocks with port definitions of UC 1.

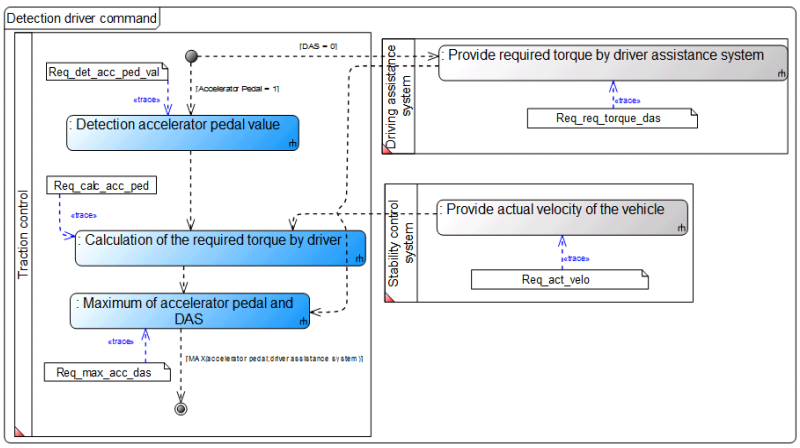

Figure 12. Concretization of the subfunction *detect driver command* onto behavior level and allocation to structure.

After the logical sequence of activities and their requirements is modeled, the actions can be grouped into several swimlanes and allocated into structural elements. The behavioral task is finished when the sequence of actions, the partitioning to structure and the relations to the requirements is done.

The activity *Provide actual velocity of the vehicle* heads to the traction control (TC) in which the activity *Calculation of required torque by driver* will be addressed. This control flow signals that an information exchange takes place between both, the TC and SCS. In Fig. 13 an internal block diagram is used to represent the structural level. All interfaces of the structural blocks are modeled by flow-ports. These flow-ports satisfies requirements about the information exchange between the structural blocks.

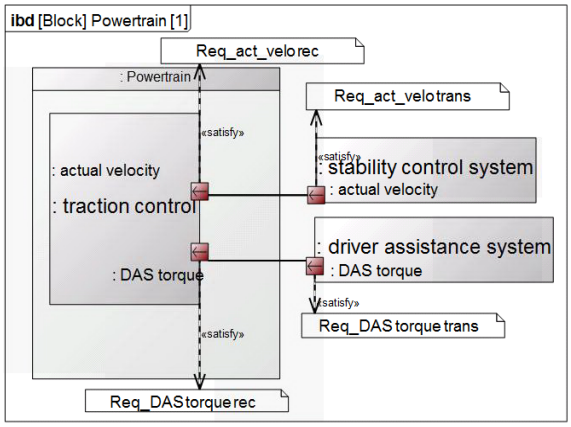

Figure 13. Concretization from the behavior onto subsystems with port definitions from the subfunction *detect driver command*.

VI. Discussion

In order to support the modeling processes of customer functions in early development stages, methods and tools are essential that are domain independent, general and supports the developing process of requirements. Such implementations are rare [17]. This papers approach explains the requirement-based effect chain modeling.

The advantage is that the systematically modeling of requirements in SysML based on the F-B-S method gives a conclusive way of traceability and offers a structured proceeding to build up requirement-based effect chains.

One of the great benefits given by MBSE is the domain independent and interdisciplinary work in a model. This represents that many stakeholder participates in this design step. On the other hand to get a higher connectivity between effect chains, it must be guaranteed that elements can be reused by different stakeholder. Only if elements are reused, the connection between customer functions can be generated in the database. Without this criteria every function would be an isolated development so that influences between customer functions cannot be identified. Knowledge bases for requirement-based effect chain modeling in the conceptual design is needed. Library development should be introduced in organizational or computational way to develop a common way in SysML and to increase the potential of reusable models.

In respect to the many definitions of behavior, a different point of view is here given through the behavioral level. The meaning of this behavior level corresponds to an auxiliary level, to combine the functional decomposition process with the abstraction levels from the V-Model as well as the organizational aspects driven by MBSE. Behavioral elements does not concern physical effects but more a sequence of elementary working principles which are in general functions. Since it is an interdisciplinary modeling step, there is a need in a systematically division of the abstraction levels and the responsible engineers. Therefore an introduction of an auxiliary level is important to distribute their respective tasks. Due to the different view of function and behavior, a system engineer on system level knows at which step to commit his model to system engineers at subsystem level.

From a system engineers perspective a global view on the entire system summing up every effect chains is still missing. Therefore additionally methods have to be introduced to aggregate effect chains. The aim should be, to get an effect net with many interactions between functions. In this way cross-impact analysis for a better system understanding as well as reliability analysis could be done. This issue has to be addressed to future work in order to wider the horizon of SysML under consideration of cross-impact analysis.

In terms of implementation, foundations on realizing requirement-based effect chain modeling based on F-B-S for developing customer functions have been laid down and the feasibility of the method has been proven.

VII. Conclusion

In this paper a method for systematic modeling of customer functions in conceptual stage with SysML was shown. Due to the complexity aspects driven by the increasing number of functions and their interactions, the conceptual stage becomes more and more complicated. Especially recent researches addresses the need for support of the conceptual stage.

This research responds to the need for modeling of effect chains in a solution independent and function oriented way. As a result of the model, functional requirements can be allocated to the design elements to achieve requirement-based effect chains with a more conclusive traceability.

Because of the increasing connectivity, a combination of several systems is necessary to avoid an oversizing in the product architecture. For this purpose the approach supports the partitioning process from functional idea to the mapping on an existing structure to handle with the structural complexity.

It can be concluded, that the shown concept gives a fundamental investigation of developing effect chains and the arising requirements considering top-down development, increasing transparency, interdisciplinary work and sustainability through reusability.

Future work includes building up knowledge databases from these models and creating functional requirement nets from the requirement-based effect chains by computational synthesis combined with smart data analysis based on pattern recognition algorithms. In this way the quality of the concepts concerning the reliability could be improved.

REFERENCES

[1] McKinsey & Company, "Connected Car, automotive value chain unbound," in Market Report of Advanced Industries, 2014.

[2] M. A. Orloff, *Toward the Modern TRIZ*, Springer International Publishing, 2016.

[3] D. Cavallucci, *TRIZ: The Theory of Inventive Problem Solving*, Spinger International Publishing, 2017.

[4] M. Becker, D. Dasari, S. Mubeen, M. Benham, and T. Nolte, "End-to-end timing analysis of cause-effect chains in automotive embedded systems," *Journal of Systems Architecture*, vol. 80, pp. 104-113, 2017.

[5] C. Pigorsch and B. Beyer, *Zeitanalyse in der E/E-Architektur-Entwicklung Praxisbericht von BMW*, ATZ Elektron, Springer Fachmedien, Wiesbaden, 2016.

[6] J. Göpfert, *Modulare Produktentwicklung – Zur gemeinsamen Gestaltung von Technik und Organisation*, Wiesbaden: Springer, 1998.

[7] O. Alt, *Modellbasierte System-Entwicklung mit SysML*, HANSER, München, 2012, pp. 40-60.

[8] S. Ishibashi, K. Hisazumi, T. Nakanishi, and A. Fukuda, "Model-Based methodology establishing traceability between requirements, design and operation information in lifecycle-oriented architecture," in *Trends in E-service and Smart Computing. Studies in Computational Intelligence*, T. Matsuo, T. Mine, S. Hirokawa, Ed., vol 742, pp. 47-63, Springer, Cham, 2018.

[9] G. Pahl, W. Beitz, J. Feldhusen, and K. Grote, *Engineering Design: a Systematic Approach*, 8th ed., Springer, Berlin, 2013.

[10] *Systematic Approach to the Design of Technical Systems and Products*, VDI 2221, Technical Report No. VDI, Düsseldorf, 2018.

[11] VDI Richtlinie 2206 – Entwicklungsmethodik für mechatronische Systeme, VDI-Richtlinienausschuss A127/VDI2206, Paderborn, 2004.

[12] *Systems Engineering Vision 2020*, INCOSE, INCOSE-TP-2004-004-02, September, 2007.

[13] J. A. Estefan, "Survey of Model-Based Systems Engineering (MBSE) methodologies," in *Proc. INCOSE MBSE Initiative*, 2008, pp. 15-18.

[14] *OMG Systems Modeling Language Specification*, Object Management Group, 2017.

[15] S. Friedenthal, A. Moore and R. Steiner, *A Practical Guide to SysML – The Systems Modeling Language*, 2nd. ed., Morgan Kaufmann Publishers, Waltham, 2011.

[16] J. W. T. Kan and J. S. Gero, "Theoretical framework," in *Quantitive Methods for Studying Design Protocols*, Springer, Dordrecht, 2017.

[17] M. S. Erden, H. Komoto, T. van Beek, V. D´Amelio, E. Echavarria, and T. Tomiyama, "A review of function modeling: approaches and applications," *Journal of Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 22, no. 2, pp. 147-169, 2008.

[18] VDI Richtlinie 4521 – Inventive Problem Solving with TRIZ. Fundamentals, Terms and Definitions, Berlin, 2016.

[19] C. Rupp, *Requirements-Engineering und –Management: Aus der Praxis von klassisch bis agil*, 6th. ed., München: Carl Hanser Verlag, Germany, 2014.

M. Chamas was born in Berlin in 1989 and completed a master´s degree in electrical and computer engineering at the Technische Universität München in Munich, Germany, in 2016. The major field of study focusses on the electric vehicle technology.
Since 2013 he works in the automotive section in different areas. Related to his electrical background he worked as a Calibration Engineer for electric vehicles and then as a Test Engineer for high-voltage batteries. Currently he graduates in product development at the Universität der Bundeswehr München. Further he accompanies his research aims as a System Engineer at BMW Group in Munich, Germany. His research interests are focused in the areas of computational support for requirements engineering, modeling and visualization of complex systems and smart data analytics through pattern recognition algorithms.

K. Paetzold was born in Thuringia in 1969 and graduated in mechanical engineering at the Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany, 2004. Since 2009 she is the Chair of the Technical Product Development Institute at the Universität der Bundeswehr München. Her research interests include Systems Engineering and process and workflow support for planning and controlling Engineering processes. Prof. Paetzold is member of the Advisory Board of the Design Society and product development in the VDI.