INTERNATIONAL DESIGN CONFERENCE – DESIGN 2020
https://doi.org/10.1017/dsd.2020.268

REVERSE-ARCHITECTING APPROACH FOR SYSTEM ARCHITECTURE MODELS

R. Pluhnau ✉, S. G. Kunnen, D. Adamenko and A. Nagarajah

University of Duisburg-Essen, Germany
✉ robin.pluhnau@uni-due.de

Abstract

For an efficient product family development an abstraction of concrete product variants is necessary in order to recognize and systematically describe characteristic properties of a variant. System architecture models represent a possibility for the systematic description of the product variety. The structure of architecture models for existing product families resembles a reverse engineering process, in which products have to be analyzed on their structures. This paper describes a reverse engineering approach for building up system architecture models as basis for developing product families.

Keywords: systems engineering (SE), systematic approach, product architecture

1. Introduction

Changing market conditions, new product requirements and the increasing amount of electronic components and software in products are today's companies challenges. In addition, there are trends such as the individualization of products, which customers can identify with. The classic processes and methods of domain-distributed and document-centric product development are reaching their limits due to the increasing organizational complexity. In order to meet the increasing complexity of products and to be able to differentiate themselves on the market with a sufficient variety of products, companies are confronted with the need to perform their product development more efficient. In terms of high competitiveness, it is essential for them to be able to economically develop and produce product families. Product development processes, which are characterized by a multitude of linked and exchangeable information from many domains, require a high level of data consistency. However, in product development each discipline uses its own development methods and often obtains its data manually. Compared to processes such as logistics, where the use of mature ERP systems ensures a high level of information consistency and data integrity, the product development is often characterized by a large number of isolated solutions. Model-based Systems Engineering (MBSE) is regarded as the development trend with which the increasing complexity of interdisciplinary product development processes is to remain manageable through a continuous and holistic model orientation in product development. Systems modeled by means of formal graphical modelling language should ensure that all domains involved in the development are able to build up a common understanding of the system to be developed in the future and to achieve a high degree of information consistency at the same time. Engineering activities from different disciplines are thus supported organizationally and system-technically throughout the entire product life cycle, from requirements recording to recycling.

For this purpose, the development processes and their logic have to be modelled in this system model. The basic structure of system models are system architectures - they are describing the structure and the interaction of functionalities and components of a system and thus represent the early phase of development, such as conception, and can serve as a basis for various authoring tools in which the product design is determined and validated step by step through simulation and analysis. With regard to product families, they offer an abstract and valuable reference for the development of generations and variants. The paper presents an approach for building system architectures for existing product families. The architecture models that can be generated by the help of this approach are supporting an efficient development of product variants and generations. (Eigner et al., 2017; Weilkiens et al., 2016)

1.1. Problem statement

Unusually product development starts on a white sheet of paper - companies that develop cars today probably will not develop coffee machines tomorrow. Albers et al. (2016) show that the majority of the activities of development departments are part of the development of product families (Albers et al., 2016). This refers to variants and generations of products, with variations in design and principle or functional integration. They are characterized by a high level of reuse of functions and components, however, the effort for creating such variants and generations increases rapidly with the number.
In context of the current research project with a plant manufacturer, the development process for variants of mechanical expander tools for steel pipes is to be investigated. The main challenges, the company has to face are due to the customer individuality of the variants. A customer order represents an area of application defined by the customer, which is usually unique and has to be implemented using different tools. It comprises different tube materials, with different tube dimensions and required plant cycle times, for which as few different tools as possible are required. The resulting demands on the tool range to be supplied are highly contradictory and require a circular problem-solving process to find the right tool configuration and its unique geometric characteristics. As an example of this, a sufficient tool strength at high material yield strength and small pipe diameters is to be mentioned, as this is where conflicts arise, especially due to space constraints. Similar conflicting requirements are represented by large tube diameter ranges at constantly high cycle times, which presuppose a tool stroke that cannot be realized. For an efficient product family development, an abstraction of concrete product variants is necessary in order to recognize and systematically describe the characteristic properties of a variant. System architecture models represent a possibility for a systematic description of the product variety in a useful abstraction. Within the system architecture, correlations between functional and logical structures can be mapped in a descriptive, qualitative model. Furthermore, the elements of an architecture, such as functions or components, can be specified by their most important properties. Architectural models thus are a useful reference for variant and further development, because both conceptual and physical elements can be inherited. The specific meaning of system architectures can be seen in the INCOSE Vision 2025, in which it is titled as an essential discipline for the successful application of systems engineering (INCOSE, 2014). In the context of MBSE, there are several approaches to support the creation of system architectures. Mostly, new developments are the focus of interest, which is why the development of functional architectures is particularly addressed. With reference to the statistics for development activities, it can be deduced that building architectures for existing products or product families should be focused. In the industrial practice, the building of system architectures should be understood as a reverse engineering process, in which physical products have to be analyzed on their structures. (Weilkiens et al., 2016)

1.2. Objective of the research activities

The research activities primarily aim at the creation and use of architectural models for the development of product families. The defined goal is to develop a systematic approach to analyze product families in order to be able to formally describe the systems architecture models. For this purpose, different approaches to analyze products and product families are taken up and extended with the philosophy of the Reverse-Engineering to define a Reverse-Architecting-Approach. This Reverse-Architecting-Approach describes necessary phases and activities to get from the real product family to a system architecture with defined abstraction levels. The architectural model must be able to describe

https://doi.org/10.1017/dsd.2020.268 Published online by Cambridge University Press

the complete composition of the product from a functional and physical perspective. Since the current project example represents a product family, different architectures are existing. Therefore, it is the explicit goal to create a reference architecture. It must be able to represent the totality of all possible variants in one architecture and allow changes for new variants at the same time. Levels of abstraction are intended to promote a downstream variation of the product in various phases of development. Thus, step-by-step abstraction can be used both in the design phase and in the conceptual phase of the development. Modelling the complete reference architecture enables the specification of architectural elements such as functions and components by means of certain properties. In this way, functional elements such as "increase force" or "reduce friction" can be specified by properties such as forces, transmission ratios, lubricant requirements or surface pressures. Components, on the other hand, can be specified by concrete design descriptive properties such as dimensions. These quantifiable properties are machine interpretable. For a model-based development of variants, the specification of architectural elements offers a number of advantages. Within the system model, the properties can be linked to requirements by decision routines and analytical relationships and enable a requirement-driven variant development. On the other hand, the transferability of architecture information to different domain models, in which the shape is determined and analyzed, is made possible. Thus, functional and physical structure and their properties can be transferred for the initial setup of a physical model to get a rough design of the components. Moreover, it can be used for the numerical determination of material stress in a structural-mechanical analysis if the architecture information is transferred to a FEM model. Similarly, it can be used for the verification of the tribological properties in flow simulations to determine the frictional power. The properties of a variant determined step by step in the domain models can be traced back to the architectural model and describe its specific characteristics. For this purpose, they are inherited from the reference architecture to a variant architecture to be available for creating the geometry in a CAD system. The starting point for configuration is the requirements specification, which is linked to the architecture within the system model. In this paper, an overall framework of the Reverse-Architecting-Approach is presented. The phases and activities of the approach are described as a procedure model. It serves as a basis for variant development and will be validated using the mechanical expander tool.

2. Literature overview

2.1. System architecture and architecture models

The origin of system architecture is due to the software industry, where it describes the structure of IT systems, consisting of software and hardware elements, properties of these elements and their relationship to one another (ISO/IEC, 2007). Systems Engineering, which represents system thinking and the holistic view of complex systems, reissues the term system architecture. System architecture thus describes *"the fundamental organization of a system, embodied in its components, their relationship to each other and the environment, and the principles governing its design and evolution"* (ISO/IEC, 2007). From the point of view of product development, this definition is not new - it is the well-known definition of the product architecture from Ulrich: *"In the informal sense, the architecture of the product is the schema according to which the functions of the product are mapped to physical components"* (Ulrich, 1995). He defines the product architecture more precisely as the *"arrangement of functional elements"*, the *"assignment of functional elements to physical elements"* and the *"specification of interfaces between interacting physical components"* (Ulrich, 1995). The multitude of existing definitions has the same core statement in common: The system architecture describes the functional structure and the physical structure of a system as well as the interactions of its elements in an abstract way (Figure 1). It defines how and why a system came to its form. In the context of MBSE, the definition of system architecture models was taken up again. Architecture models are formally described models that consist of architectural elements such as functions, function flows, interfaces, and logical and physical elements. They are arranged in architectural abstraction levels that allow different views of a system and describe properties and logical relationships to other elements that are consistent (Weilkiens et al., 2016). At this point it should be mentioned, that there is not a single type of architectural description. Depending on the centre of interest, service-oriented architectures, solution-oriented architectures, architectures for IT systems or for product lines are of importance. Within

https://doi.org/10.1017/dsd.2020.268 Published online by Cambridge University Press

the scope of research activities, architectures for product lines or product families and solution-oriented architectures are of particular interest.

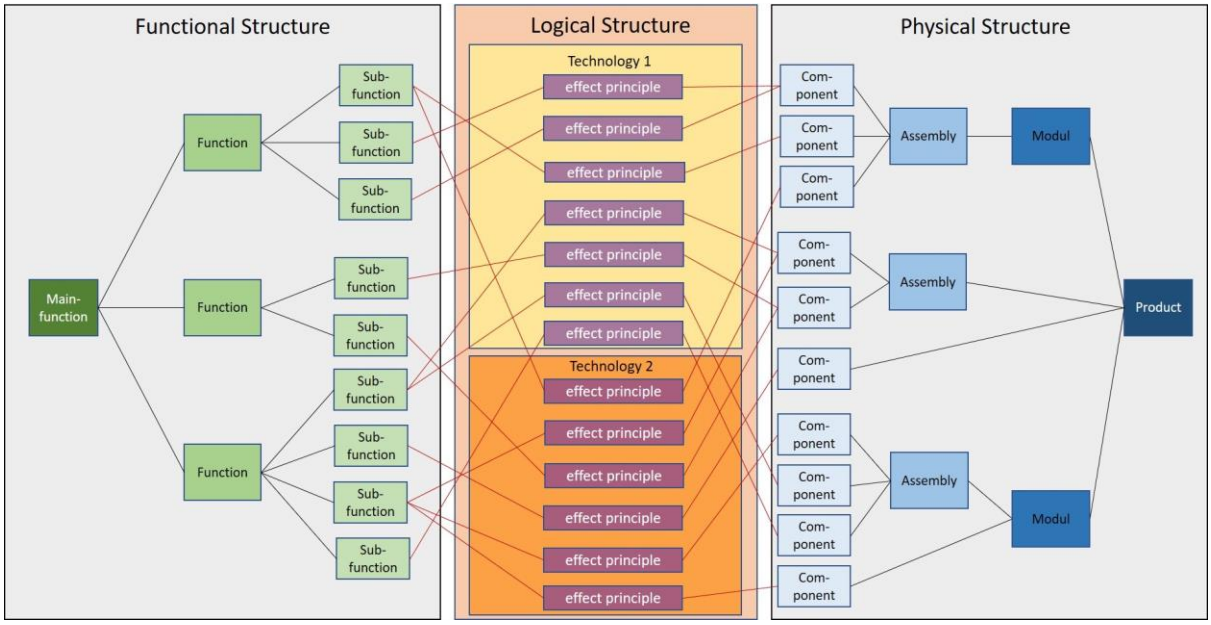

Figure 1. System architecture

Architectures for product families are strongly oriented to the physical structure. They describe the product variety in an abstracted way from the concrete design, which is why they can be applied for a certain degree of product variation and can serve dynamic stakeholder requirements. They are typically more concrete than solution-oriented architectures, as many elements and properties are usually inherited. They offer an efficient way to create variants. By definition, solution-oriented architectures correspond to the classic system architecture. Functions, principles as well as components are used for the description of the solution of a system for a special use. By these different abstraction levels they address both the conceptional, as well as the design phase of the development and possess a quite generic character. In this contribution an integrated approach is to be presented, which extends the procedure for building up solution-oriented architectures by aspects of the product families. This type of architecture is of particular interest when concepts of product families become increasingly obsolete and new concepts have to be considered. (Philips, 2018)

2.2. Architecture definition process

System architecture activities aim at a global system solution based on principles, concepts and properties that are logically and consistently linked. In principle, a distinction is made between architecture and design activities, which are based on different ways of mind. Architecture activities are abstract and strongly conceptually oriented. They describe the superordinate concepts, emergent properties and characterize elements of the system, e.g. functions, flows, interfaces and data of a system to fulfill the specified requirements. The resulting architectural models are shape-independent and very flexible. They describe the "what". Design activities are strongly technology oriented. They describe physical, structural properties of the system. Design activities take up the artifacts determined in the architecture activities to find solutions in order to examine compatibility and feasibility for the implementation. They analyze the architecture to determine possible technologies that fulfill the requirements and describe the "how". The architecture and design process is iterative and the physical architecture becomes more concrete as the number of design decisions increases. Most approaches and activities refer to the synthesis of systems and products. Since the building of system architectures does not begin on a white sheet of paper but with an existing physical product or product family, a suitable procedure for this reverse engineering process must be described to get from the actual shape of a product to its globally valid system architecture. (Walden et al., 2015)

https://doi.org/10.1017/dsd.2020.268 Published online by Cambridge University Press

2.3. Architecting and analyse approaches

The existing approaches are largely concerned with the new development of systems and describe the classical activities for the synthesis of architectures. The basic idea of the approaches can be traced back to standardized process models such as VDI2206 (development of mechatronic systems) or VDI2221 (development of technical systems and products). Requirements are the basis for all architectural processes. A constant alignment with the requirements has to take place in the subsequent architecting phases. The three common phases of an architecture are the functional design, logical design and physical design. Activities of functional design aim at the determination of a functional structure. An abstract description of the product is to be determined, which subdivides the product starting from its main function into subfunctions (Daniilidis, 2017; VDI, 2004). In principle, a distinction is made between hierarchical and flow-oriented consideration. According to Fixson, a rule-based method should answer the question: *"what are the functions of a product and how can they can be determined?"* (Fixson, 2005). He argues that functions of the same degree of abstraction must always be on the same hierarchy level and that the highest hierarchy level describes the entirety of the functions of all components. In addition, Göpfert and Tretow consider that three hierarchy levels are sufficient for the functional description (Feldhusen and Grote, 2013). SysML4Consens was developed as a specification language for the hierarchical modelling of function structures, which is used in tools for building architectures (Gausemeier, 2011). The flow-based consideration of functional structures is widely acknowledged as it strongly supports to find solutions by integrating system flows. Stone describes three heuristics used to identify basic system flows, simultaneous subfunctions, and the conversion and forwarding of flows (Stone, 2000). Albers et al. describe an Approach for the identification of functions and system flows by the consideration of effective surfaces and effective structure elements. In addition, they describe a possibility to cluster functions into functional units that are relevant for a particular discipline. For this purpose, functional units are extended with system flows. Like Weilkiens, they use SysML as description language for functional modelling (Albers et al., 2016; Weilkiens et al., 2016). In the logical design, the logical elements are mapped as binding part between functional and physical design. It is used to search for solutions and to develop concepts by determining effect principles or solution elements that fulfill the function (Daniilidis, 2017; VDI, 2004). Kleiner argues that technology decisions are also solution elements. In some approaches, the logical level is merely described as the linking relationship between function and component (Kleiner, 2012; Fixson, 2005). Gausemeier and Albers consider the logical level by integrating the effect structure model. It describes the elements required to fulfill functions and the system flows and interfaces and the merging of functional structures and effect elements (Gausemeier, 2011; Albers et al., 2016). The physical design describes the derivation of the product structure with its modules and components from the effect structure. Albers describes a possibility of deriving the physical structure using a matrix in which he correlates components and effect structure elements (Albers et al., 2016). Weilkiens furthermore describes an approach for building architectures for product lines using variation points for variants that extend the architecture (Weilkiens et al., 2016). The approaches do not describe an analysis procedure for existing products, but they contain valuable steps for use in a Reverse-Architecting Approach. Aspects of product and product family analysis must also be considered as the basis for the Reverse-Architecting-Approach. With regard to the analysis of product families, the reference product structure approach of Feldhusen and Grote is widely used. It aims at the determination of the structure of a theoretical product, which contains all possible characteristics. It describes the largest possible product structure that can be used for all product variants in product groups. The nodes of the structure are placeholders for different component variants or solutions. A multitude of variants can be described with a single structure using this approach. An method to systematically analyze products is given by van Wie. He refers to the steps taken by Ulrich and Eppinger to build system architectures: The simultaneous generation of schemata for functions and components, the clustering of elements, the creation of a geometric layout as well as the consideration of internal and external connections (Ulrich and Eppinger, 2000). Van Wie takes up the steps in his approach and describes 6 diagrams for the analysis of products. The Spatial Constraint Diagram is a product sketch with subdivisions of geometric units that define the systems boundary. In addition, material, energy and signal flows are assigned to the units. In the Functional Layout Diagram, product functions and their spatial boundaries are identified by looking at the system flows and the structure. Therefore function-related black boxes are created that are linked to system flows. In the Physical Solution Diagram, the actual physical

https://doi.org/10.1017/dsd.2020.268 Published online by Cambridge University Press

components of the component are identified, which are combined into modules in the following diagrams. (Feldhusen and Grote, 2013; van Wie, 2002)

3. Reverse Architecting Approach

3.1. Reverse Architecting Framework

The Reverse-Architecting-Framework describes a systematic procedure for building architectural models for products and product families. It is based on the well-known phases of product development and defines five superordinate states (Figure 2).

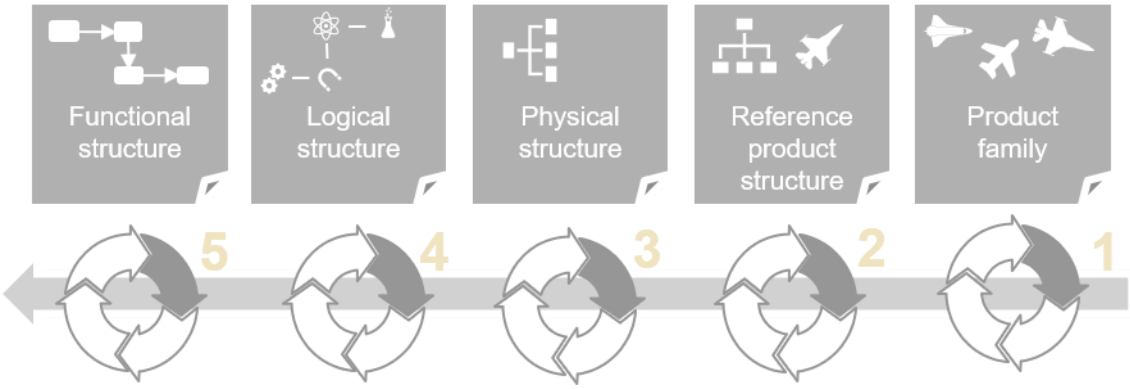

Figure 2. Reverse-Architecting-Framework

The starting point of the procedure is an existing physical product or an existing product family with variants and generations of a product. At this point, it should be noted that the level of abstraction has to be defined with caution. It has to be decided, how far a system can be abstracted, so that the abstract component is still sufficient close to all variants. Effort and benefit of system architectures are not in a good relation if a car and an aircraft are to be described abstractly by a system architecture of a transportation. At first a reference product structure is determined from the existing variants and generations of the product, which is able to describe the product family in a more abstract form. It contains all possible characteristics of the product in a single structure, which can have different options at defined variation points and is open to modifications. The reference product structure is the basis for creating the (reference) system architecture model. It contains the stages three, four and five, i.e. the classical levels of the system architecture (physical, logical and functional). The reference product is examined more closely in the physical structure. In addition to the composition of components and their variants at defined variation points, the relevant properties for describing the design characteristics such as dimensions are taken into account. They will be applied as value properties to specify the components in the physical reference structure. These architecture information, consisting of structure information and value properties, can be used in two cases. They can be linked to requirements of a system model using decision routines and analytic relationships to find the right configuration and can be transferred to domain-specific models like CAD or FEM models to define and analyze the shape of a variant. The architecture of a new variant is generated by inheriting the structure elements and their value properties out of the reference architecture. If the configuration of components is known and the geometric expression must be determined, this form of abstraction is sufficient for developing new variants. The more abstract levels of the reference product are described as the logical and functional structure. The logical structure describes the principles and technologies used to fulfill product functions within the product family. It describes the technical solution of the functionalities and the function-design context as a connecting element. The logical level does not have to be described necessarily - it is the result of the transformation relations between functional and physical level. Depending on the necessity, this transformation relationship can be examined in more detail. However, for a subsequent use of architecture models for developing product families, it is to be considered as it is an important element to enable a maximum variation of solutions, that should always be strived for.

https://doi.org/10.1017/dsd.2020.268 Published online by Cambridge University Press

The most abstract form, the functional structure, describes the product functions to be realized as well as their interaction through exchange and transformation of material, energy and signal. In context of product innovation and variation, it is an important basic framework into which new elements are to be integrated. Existing and missing functionalities and system flows, that are essential for implementing new solution, can be determined using it. As well as in the physical architecture, the specification of properties on the functional level promises great benefit for variant development. So that requirements and boundary conditions are already specified in that early concept phase, as well as system flows, the selection of the technical solutions can be supported by using rules or templates for the configuration. Hereinafter, the systematic procedure model for the implementation of the Reverse-Architecting-Approach is presented. According to Figure 3, the procedure is divided into 6 phases with various activities that have a distinct work result.

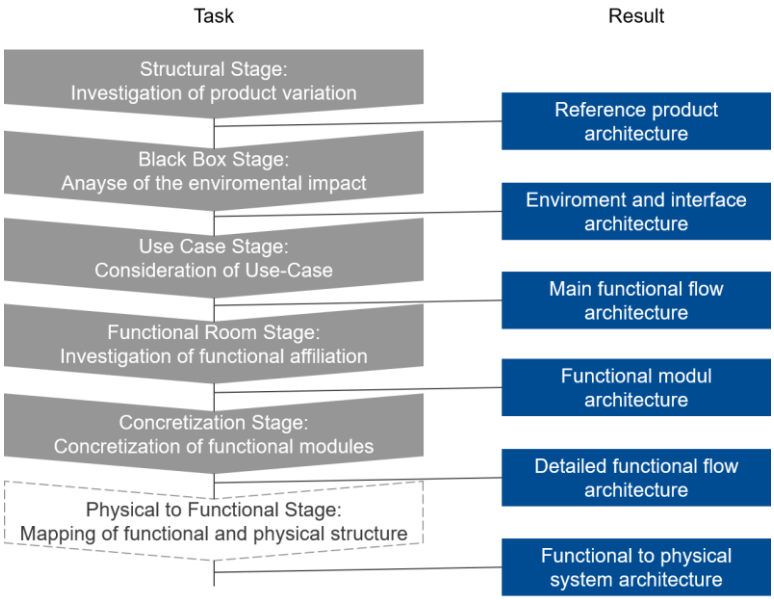

Figure 3. Tasks and results of Reverse-Architecting-Approach

The defined results are based on the phases of the Reverse-Architecting-Framework and thus represent the typical RFLP levels in modification. As mentioned in the previous paragraph, the logical structure is only considered by regarding the transformation relationship between the functional and physical levels. For an explicit attention, the method allows an extension of the physical to functional stage. The activities within the phases, which are listed in Figure 4, are described with regard to build up system architecture models. The structural stage is the initial phase of the method and deals with the physical structure of the product family to be analyzed. Within its framework, the product structures of different variants and generations are determined and compared. The comparison of the structures reveals points of variation, at which different characteristics of the product are generated by changing the principle or the shape. In this comparison, standard components as well as options of different types of components are identified. The mapping of the basic structure of standard components with different options at variation points leads to a reference product structure which is able to completely describe the variation of the product family. For the subsequent analysis of the shape-describing parameters, standard components only have to be considered once, which reduces the effort of the analysis. Options, on the other hand, must be examined with regard to different shapes and parameter characteristics in many product variants. The design parameters determined such as dimensions specify their components within the reference product architecture using value properties. The reference product architecture and their properties represent the work result of the first stage. It describes the physical structure and the important geometric information of the system. In the subsequent Black Box Stage the system environment, the input and output of the system as well as the basic interfaces with the system environment are analyzed and described. Black Box means, that no internal system elements and flows are considered. The first step is to identify the system boundary and the operating states of the system.

https://doi.org/10.1017/dsd.2020.268 Published online by Cambridge University Press

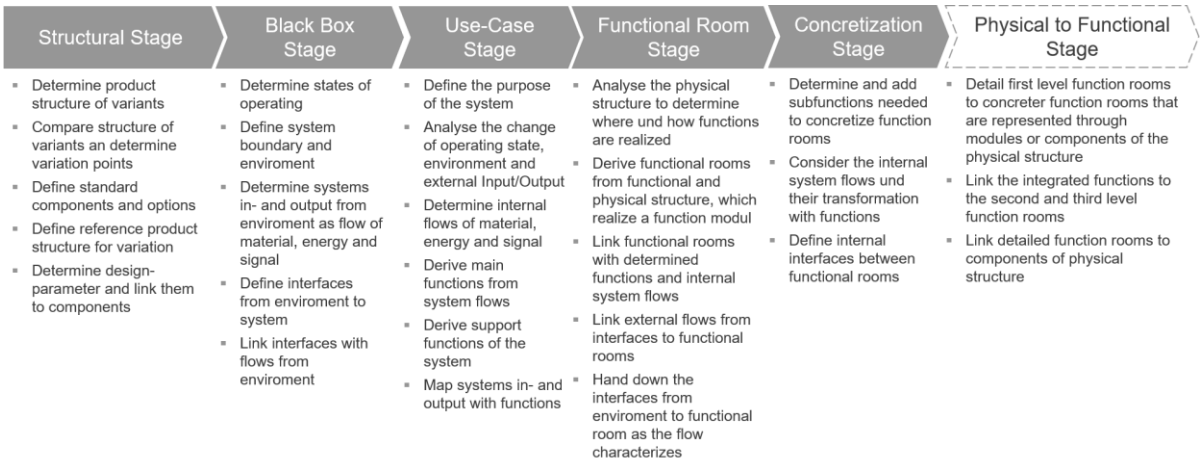

Figure 4. Reverse-Architecting Stages and activities

They describe whether fundamental changes of the overall system occur which influence the system environment and possible inputs and outputs. The consideration of the operating states provides system flows of materialistic, energetic or signalling nature that affect the system from his environment and that affect the environment from the system. It aims to identify interfaces of the system with its environment, which are located on the system boundary. They exchange system flows and properties that are inherited to their binding element of the system architecture and can be used for detecting technical solutions based on configuration rules (Figure 5, left). Since it is very difficult to represent architectures in their entirety, Figure 5 presents a layout for an architecture model, consisting of essential architectural elements and properties.

Interfaces in connection with system flows are already to be assigned for functions such as energy supply, force absorption or signal reception. At this point it should be noted that the actual definition of a black box is no longer correct, because the interfaces belong to the system. However, they should be considered separately. The result of the Black Box Stage is the environment- and interface-architecture. The use case stage is applied to examine the internal structure of the black box. It is dedicated to detect the purpose of the system and the associated main and auxiliary functions as well as the internal system flows. For this purpose, the transformations of the system flows taking place in the operating state of the system are considered across the system boundaries as well as inside the system and the main functions are derived from them. If a substance in a certain state is led into the system via the interface and led out of the system in another state via an interface, the function for that transformation is to be defined (Figure 5, middle). Auxiliary functions, whose necessity results from the operation, are to be supplemented to the architecture. The functions of the system are to be coupled with the system flows to be transferred from the interfaces. Additional internal system flows, which do not derive from the environment of the system, are to be considered and their connections to the system have to be defined. The result of the Use Case Stage is the main function architecture.

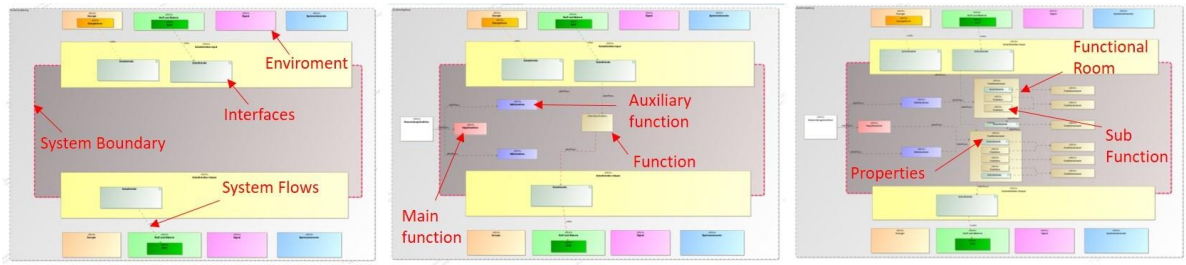

Figure 5. Layout Black Box Stage, Use Case Stage and Concretization Stage

In the following Functional Room Stage, the physical structure is examined with regard to how functions are realized. A combination of analysis of the form and the function is used to identify basic functional

https://doi.org/10.1017/dsd.2020.268 Published online by Cambridge University Press

spaces, i.e. areas of the product within which certain functions are fulfilled. At that point the user can feel free to define these spaces roughly or finely. Since the concretisation of these function spaces is to be done in the following phase, the first level function space should be chosen roughly. Function spaces are imaginary construction spaces in which main functions tend to be realised in the first level. This is why they have to be linked with the main functions or the corresponding functions. These main functions consist of a set of subfunctions. If interface-functions, e.g. force absorption, are realized within a functional space, they are inherited from the Black Box to the functional space. The result of this stage is a kind of modular functional architecture. At this point the properties of functional rooms regarding to system flows as well as to functions themselves have to be specified. As an example, this could be the kind of flow, like electrical energy, as well as value properties of the flow like the amount of voltage. They will be inherited to the physical component that fulfils the function and can be used for electrical or physical models in the following. The functional spaces are concretized in the Concretization Stage. A closer look at the first level functional spaces reveals a series of sub-functions that are necessary to fulfil the main function. They are to be defined within the functional space (Figure 5, right). The existing system flows from the Use Case Stage are to be linked with the functional spaces, or more precisely with the subfunctions contained therein. In addition, interfaces between the function spaces themselves must be defined. Subfunctions of a function space are able to define second level function spaces, if this level of detailing is necessary. They can consist of several subfunctions, which are fulfilled by a smaller function space. This stage results in detailed system flow architecture. The last level of detail to be reached for the function spaces is a single component of the physical structure. However, in many cases the assignment to modules is sufficient. They are reached in the final Physical to Functional Stage, where they are linked to the physical structure. As a result, they deliver a complete flow-describing system architecture that has a continuous link from functional to physical.

3.2. Validation

Within the project the approach is used for modelling the reference system architecture of a mechanical expander tools to generate new variants on the basis of specified requirements. Therefore, the architecture of all variants has been analyzed in order to identify and model a reference system architecture including all variations. To define and analyze the shape of variants in physical, fem and cad models, the relevant properties have been specified for their components. These properties have been coupled with requirements by parameter diagrams for analytic relationships and decision routines (Figure 6). The dependency on requirements and shape of the product variant described by relationships is used to automatically generate requirement-controlled variants, by triggering calculations based on properties. Current work in the project examines the validation of the procedure for the construction of the complete system architecture.

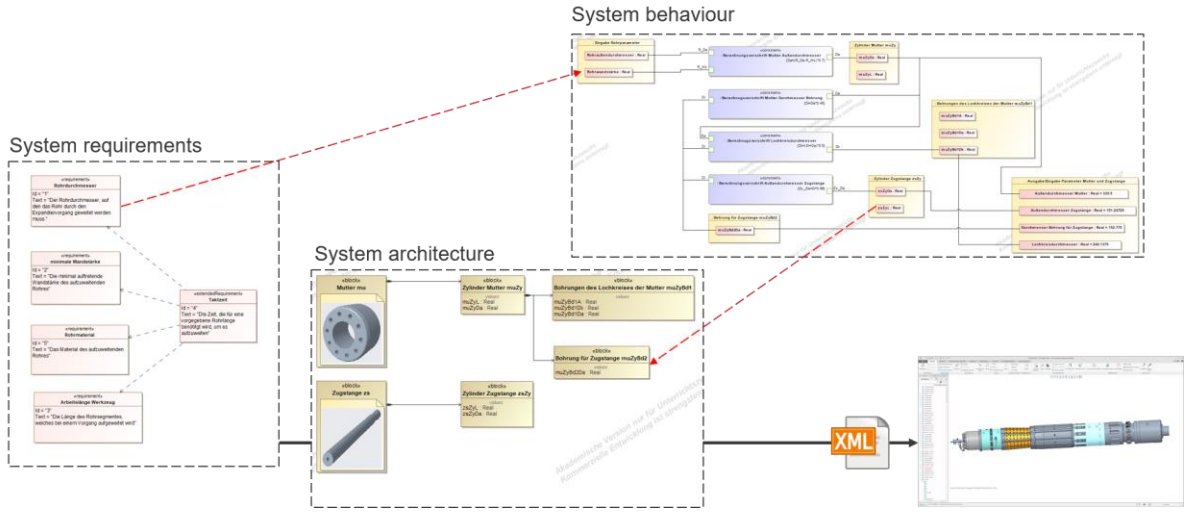

Figure 6. Physical (reference) architecture as a basis for creating product variation

https://doi.org/10.1017/dsd.2020.268 Published online by Cambridge University Press

4. Summary and conclusion

A systematic Reverse-Architecting-Approach for building up system architecture models that includes selected aspects of product analysis and architecture modelling has been fully described and successfully validated. The further activities are dealing with detailing the approach by investigation of different products from most diverse industries. The activities aim at establishing a standard for the development of system architectures for existing products. In the future, architectures created on the basis of this procedure should represent a useful basis for creating product innovation through integration of new functions into existing structures.

Acknowledgement

This work was supported by the German Federal Ministry for Economic Affairs and Energy within the Central Innovation Programme for SMEs.

References

Albers, A. et al. (2016), "iPeM – Integrated Product Engineering Model in Context of Product Generation Engineering", *Procedia CIRP*, Vol. 50, pp. 100-105.

Daniilidis, C. (2017), "Planungsleitfaden für die systematische Analyse- und Verbesserung von Produktarchitekturen, Planning guide for the systematic analysis and improvement of product architectures".

Eigner, M., Koch, W. and Muggeo, C. (2017), "Modellbasierter Entwicklungsprozess cybertronischer Systeme. Model-based development process of cybertronic systems", Springer Berlin Heidelberg, Berlin, Heidelberg.

Feldhusen, J. and Grote, K.-H. (2013), Pahl/Beitz Konstruktionslehre, Design Theory, Springer Berlin Heidelberg, Berlin, Heidelberg.

Fixson, S.K. (2005), "Product architecture assessment: a tool to link product, process, and supply chain design decisions", *Journal of Operations Management*, Vol. 23 No. 3-4, pp. 345-369.

Gausemeier, J.E.A. (2011), "Modellbasierte Konzipierung eines hybriden Energiespeichersystems für ein autonomes Schienenfahrzeug, Model-based design of a hybrid energy storage system for an autonomous rail vehicle", Tag des Systems Engineering.

INCOSE (2014), "A World in Motion - Systems Engineering Vision 2025".

ISO/IEC (2007), 42010:2007 Systems and software engineering - Recommended practice for architectural description of software-intense systems.

Kleiner, S. (2012), "Entwerfen und Entwickeln mit Systems Engineering auf Basis des RFLP-Ansatzes in V6, Design and development with Systems Engineering based on the RFLP approach in V6", Entwerfen, Entwickeln, Erleben.

Phillips, C. (2018), "The Big Happy Family of System Architecture Approaches".

Stone, R.B. (2000), "A heuristic method for identifying modules for product architectures", *Design Studies*, Vol. 21 No. 1, pp. 5-31.

Ulrich, K. (1995), "The role of product architecture in the manufacturing firm", *Research Policy*, Vol. 24 No. 3, pp. 419-440.

Ulrich, K.T. and Eppinger, S.D. (2000), *Product design and development*, 5th ed., McGraw-Hill Irwin, New York, NY.

van Wie, M.J. (2002), "Designing Product Architecture: A Systematic Method".

Verein Deutscher Ingenieure (VDI) (2004), Entwicklungsmethodik für mechatronische Systeme (VDI 2206): Design methodology for mechatronic systems, VDI.

Walden, D.D. et al. (2015), "INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities".

Weilkiens, T. et al. (2016), *Model-based system architecture, Wiley series in systems engineering and management*, Wiley, Hoboken, New Jersey.

https://doi.org/10.1017/dsd.2020.268 Published online by Cambridge University Press