Guidelines for systematic functional decomposition in model-based systems engineering

Jerome Kaspar
Model-Based Engineering Department
:em engineering methods AG
Darmstadt, Germany
jerome.kaspar@em.ag

Nicolae Cioroi
Model-Based Engineering Department
:em engineering methods AG
Darmstadt, Germany
nicolae.cioroi@em.ag

Martin Bauch
Model-Based Engineering Department
:em engineering methods AG
Darmstadt, Germany
martin.bauch@em.ag

Christian Zingel
Model-Based Engineering Department
:em engineering methods AG
Darmstadt, Germany
christian.zingel@em.ag

Sven Kleiner
Model-Based Engineering Department
:em engineering methods AG
Darmstadt, Germany
sven.kleiner@em.ag

***Abstract*—The increasing complexity in today's products require an ever more systematic product development process in which the use of model-based systems engineering (MBSE) plays an increasingly decisive role, especially against the background of cross-disciplinary collaboration. Having evaluated the actual state of the art, many companies still have true difficulties when modeling the functional architecture with the help of actual approaches. Thus, a new procedure with concrete guiding rules and control criteria to strategically define and decompose functions for the two perspectives a greenfield and brownfield environment is theoretically introduced and partly applied to a practical example.**

Keywords—model-based systems engineering, functional architecture, functional decomposition, systems engineering, guideline

I. Introduction and Motivation

Today, the networking of the physical world of machines, plants and devices with the virtual world of the internet (so-called cyberspace) is progressing ever faster [1]. Thus, modern advanced systems are more and more equipped with sensors and electronics, for example, to allow statements on predictive maintenance to improve operating times. Apart from the steadily increase of requirements on products, systems and processes (e.g., by customer expectations, legal restraints and corporate objectives) as well as the growing globalization and micro-segmentation of markets with its competitive complexity of the entire value chain of companies, the development of cyber-physical systems faces notable and hitherto unknown challenges. As a result, model-based systems engineering (MBSE) is becoming more and more established to satisfactorily manage complexity, facilitate scalability, assure traceability and maintain consistency during the early phase of system development.

Despite the numerous approaches regarding a systematic application of MBSE (e.g., OOSEM [2, 3], SYSMOD [4], FAS [5]) [6] and related concepts elaborating a functional reference architecture (e.g., via functional chain analysis [7, 8, 9]), the functional architecture – as an essential view of the system architecture [10] – has not yet been illuminated to the extent that a truly focused guide is available to define and decompose functions. Moreover, the approaches assume that either a pure new design in form of a greenfield approach is present or that reverse engineering is the focus, and thus mainly neglecting to address the most commonly used mixture of both, i.e. a given base architecture [11] within the philosophy of a product generation engineering [12]. So, when modeling the functional architecture of complex systems, many companies have true difficulties [13] especially in conducting a systematic functional decomposition while generally asking:

- How many subfunctions should a function be divided into? When is it too much? And up to which level is it needed at all?
- How to "cut" functions in general and which information can be used to do so? Is there a difference for greenfield or brownfield projects?
- To which system level does a particular function belong? And to which logical elements should the functions be allocated?

Accordingly, and mainly focusing on the second question, this contribution targets the general establishment of a guidance document leading to a stepwise elaboration of a functional architecture depending on possible conditions and specifications in the respective context. After presenting the actual state of the art (section 2) and introducing the basic procedure with systematic guidelines for the functional definition and decomposition (section 3), section 4 applies selected rules and advices in various parts of the introduced validation example. By giving a discussion and outlook in the end (section 5), the presented approach is critically being reviewed and further boundary aspects are listed that could slightly lead to an adjusted procedure of decomposing the functional architecture.

II. State of the Art – Functional Architecture in MBSE

Starting with the fundamentals of product design, almost all systematic development processes follow the four-stage procedure of task clarification, conceptual design, embodiment design and detail design [14]. Here, and given the fact that the notion of function is central to design but can be interpreted in multiple ways [15], the systematic modeling of the functional architecture can be assigned to the conceptual design and provides a functional understanding of a system without showing how the system is able to do this and reflecting the current functional state or its behavior being currently active. Thus, the functional architecture can be defined as the entity of functions of a system and its structured breakdown including described relationships/interactions

between functions. In addition, the functional architecture can also have a static and dynamic representation, where the first represents the functions and their interactions without information about the conditions and timing, and the second also represents the time sequence in which the functions are executed with a possible link to the behavioral model.

Divided into individual system functions identified from use cases and their scenarios, each system function is initially defined from a black-box perspective on the system, and thus states an abstract description of what the system shall do (i.e. defines a solution-neutral description of the relationship between input, output and state variables of a system with the objective of fulfilling a task [16]) before more concrete functions can be described in various, subdivided decomposition levels deepening the understanding of the system. Notwithstanding the pursued advocation of an as long as possible maintained solution neutrality [14], the hierarchical decomposition of the system functions into functions of subsystems and components is in practice often done in association with the elaboration of logical elements since subfunctions at least partly depend on taken embodiment decisions. As a result, and as already partly indicated above, the generation of a functional architecture consists of both a specific definition of a function and a decomposition strategy.

To do so, and compared to the preceding identification process to generally find and indicate capabilities or tasks of a system, different approaches define specific conventions to systematically name a function by using a "verb + object/flow". Starting with the more elementary and strictly scheme by Pahl/Beitz [14] or Hundal [17], Hirtz et al. [18] recommend a broader and nowadays well-known functional basis representation for functions and flows, see Fig. 1.

Class	**Basic**	**Class**	**Basic**	**Class**	**Basic**
Branch	Separate	Control Magnitude	Actuate	Signal	Sense
	Distribute		Regulate		Indicate
Channel	Import		Change		Process
	Export		Stop	Support	Stabilize
	Transfer	Convert	Convert		Secure
	Guide	Provision	Store		Position
Connect	Couple		Supply		
	Mix				

Fig. 1. Excerpt from the classification of functions by Hirtz et al. [18]

Having understood this systematic definition process, the basic idea of defining top level system functions and then decomposing them into their subfunctions is already partly explained in literature (e.g., [19, 20, 21]). In the context of MBSE, for example, the SPES 2020 framework [19] explains the functional decomposition with the black-box / white-box model, where all functions of the functional white-box model have to show the same behavior as specified by the user function of the functional black-box model.

To verify the conformity of the two models, a mapping of the individual inputs and outputs is required. As a rough indication regarding the amount of decomposition levels, SPES 2020 framework [19] only emphasizes that the chosen granularity depends on the choice and the skill of the developer. Here, Tang [22] provides an improved statement and couples the recursively refinement of lower-level technical functions identified from domain knowledge to the level where either the function specification is clear enough for allocation or realization without further decomposition or there are already identified products realizing it.

Looking at Lamm's and Weilkins' [5] heuristic approach to obtain a functional structure, their approach aims to group activities of a functional tree so that all composed functional groups are as independent as possible showing a low external complexity even though there is a high internal complexity. For this purpose, Lamm and Weilkins state that a functional group takes the functions that are related to system actors and grouping criteria of existing products can serve as a reference for current development. Thus, functions that share data can be grouped just like functions originating from abstract and secondary use cases provided that function calls imply a cohesion regarding their linked sub-functions.

Although these aforementioned approaches partly help to systematically elaborate a functional architecture, a real assistance with concrete guidelines to strategically define and decompose functions for both perspectives a greenfield (i.e., new innovation) and brownfield (i.e., product generation) development is still missing. Thus, this contribution aims to close this research gap and

- defines guiding rules that facilitate the decomposition of system functions into subsystem functions
- defines control criteria when a further functional decomposition is advisable
- provides a consistency check for the definition and decomposition of functions towards the logical viewpoint of the system architecture (e.g., by an optional "custom mapping" between functional and logical interfaces)

Demonstrating all these topics on a practical example to show a real-world applicability, the overall research question of "how should system functions be systematically defined and decomposed into its constituents to invariably link the logical architecture and thus unambiguously supports all further development steps in SE" will now be pursued in the following sections.

III. Systematic Guidelines of Function Definition and Decomposition

Based on the aforementioned deficits identified in the state of the art, the following section presents both a systematic procedure to define functions and generic guidelines focusing on specific decomposition strategies. Starting with the systematic procedure to define a function, the subsequent steps need to be successively carried out:

1) *Scope the function:*
   Look at your system-of-interest as a black box being the context of the function and make sure that the system border is clearly defined. Set a task or capability of your system as the scope of the function that you want to define. (note: those can exemparily be identified as a specific step in individual use cases being previously modeled)
2) *Name the function:*
   Name the function by its main nature formulated as a "verb + object". For verbs ideally use terms of a function classification system, such as the functional basis by [18].
3) *Specify inputs, outputs and impacts:*
   Find required inputs and generated outputs of the function that are externally visible. Define the inputs and

outputs by the type of flow (material, energy, information) across the system border. If the function affects the (internal) state of the system (e.g., charging a battery increases the energy stored in the system), describe the impact or change textually (especially important if the function has no outputs at all).

(note: states influence the behaviour of the system, but are not explicitly visible in the functional architecture and thus in functional chains of effects with flows (e.g. electric vehicles can only drive if there is sufficient stored energy in the system))

4) *<u>Validation check of the function:</u>*
   Check if the function provides an output at the system border (or affects the state of the system). If not, it indicates that the function may already be a subfunction and thus should better be integrated into another function of the system. However, there are cases, where such functions have to be defined as a system function (e.g., when a function detects the state of the system and forwards information about it to other system functions).

Once several functions are defined according to the prior procedure, a systematic decomposition of the functions is necessitated to further proceed within the early phases of systems engineering. Here, it is important to point out right at the beginning that there is a strict relation between the functional and logical decomposition, and thus functions may only be allocated to logical elements on the same system level [19]. Against this background, the following four basic guidelines (each formulated as a question) emerge to initially support the identification of subfunctions, and thus the functional decomposition at baseline:

1) *Based on the determined inputs and outputs of the function, which subfunction is required to process the input and convert it into the resulting output?*
2) *What requirements are already linked to the function and do these suggest or require a specific subfunction?*
3) *Are there any subfunctions to be defined based on general product and engineering knowledge, experience or company-specific pattern?*
4) *Can information from reference products (i.e., architectures of previous or similar generations) be considered and at least partly used for decomposing the actual system function?*

Taking into account these overarching guiding questions along with the general guidance to follow the "intended use" first (e.g., separate "spatial movement" first into "axial movement in x" and "axial movement in y" before further decomposing into "accelerating" and "braking" respectively) while listing as little as possible functions per hierarchy level (rule of thumb: 7 ± 2 elements per view [23]), however, a different procedure needs to be considered for a greenfield and brownfield development process, see Figure 2. While doing so, it should be supplementary noted that the systematic procedure for defining system functions can also be used to support the actual functional decomposition, predominantly for the systematic definition of subfunctions. Nevertheless, the precondition to proceed the guiding procedure is that the (super) function to be decomposed is already properly defined and the necessity for a further decomposition has been established.

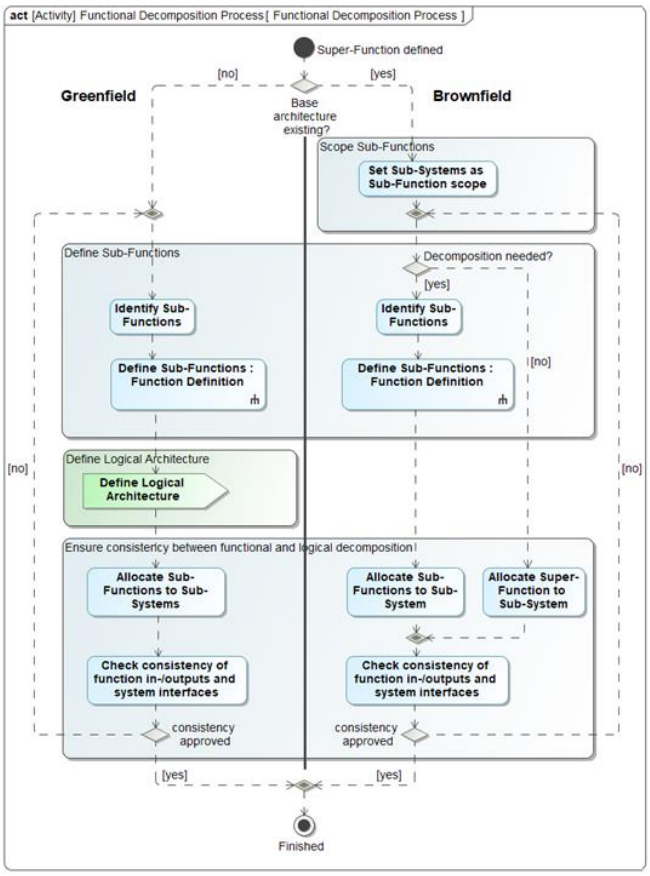


Fig. 2. Schematic guiding procedure for greenfield and brownfield scope

Here, and compared to various procedures of existing MBSE frameworks (e.g., FAS or SPES) solely providing an introductory guidance for a greenfield environment without any architectural restrictions, the guiding procedure depicted in Fig. 2 facilitates varying phases depending on the first choice regarding the availability of a base architecture, for example, from previous product generations or through customer constraints or a system-of-systems development. In contrast to the unimpeded development of a logical architecture based on impartially identified and defined functions (phase C), the latter case of existing basic architectural precepts recommends to additionally set the scope of the subfunctions (phase A) before identifying and defining them (phase B). The main reason for this is the almost directly desired consistency between the functional and logical architecture with fewer retrospective iterations as well as cases without any need for a decomposition since existing logical elements can already completely and independently fulfill specific functions. To ensure this consistency between the functional and logical decomposition, the comparatively novel extension of complying given control criteria (phase D) enables to answer when a decomposition is correct (i.e., no contradictions to the logical view exist, for example, whether all parameters can be delivered via an interface) and sufficiently detailed (i.e., with a clear allocation of one function to only one logical element).

As a result, the four-step procedure to define a function as well as the aforementioned guidelines to purposively decompose a function can both be assigned to phase B "identify subfunctions", which are now practically applied to an extensive application example.

IV. Exemplary Use of the Systematic Procedure and Guidelines

To demonstrate the validity of the systematic procedure and guidelines, the theoretical descriptions are practically applied to the example of a medical device to prepare blood samples for testing, see Fig. 3. This application example modeled in SysML and the findings published in this paper are based on the work from the BMBF-funded research project CyberTech, in which 6 partners from industry and research are pursuing the goal of developing socio-technical and digitalized processes, methods and tools being used to master the complexity of future development processes for smart machines or services in a human-centered way.

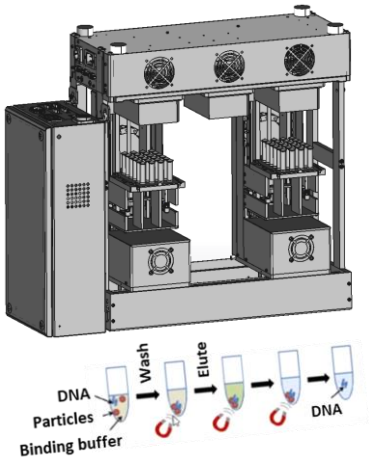


Fig. 3. CAD model and medical procedure to prepare blood samples

Apart from the more specific objective of obtaining a Digital Master to ensure a future model-based certification process, and thus offering a certain traceability of the conformity to standards already at an early stage, the following boundary conditions must be observed for the EXMO device:

- Strict relationship between functional and logical decomposition levels, whereas a specific function can only be allocated to one logical element on the same hierarchy level
- Already found or given system functions on top system level by technological expert knowledge

With that in mind, and starting with the function definition, first the system function(s) need to be methodologically defined.

Beginning with step 1 and considering the EXMO system as a black box, the scope is only on functions of the whole system, and thus does not focus tasks or capabilities of its subsystem. Accordingly, the machine-controlled extraction of the amino acids from blood samples can be formulated as "extract amino acids" corresponding to the "verb+object" pattern (step 2). As an exception, here, this basic system function can also be described as a use case of a specific stakeholder (i.e., the laboratory staff), which means that system functions can also be identified by describing particular use cases of the system. The further elaboration of the use case can subsequently help to understand the purpose of the system to be developed and to detect visible functional inputs and outputs (step 3). Based on the actions of the laboratory staff and the neighbor systems, the function to "extract amino acids" exhibits the inputs "electric current", "control command signal", "DWP [1] with washing agent", "DWP with samples", "empty DWP + tip plate" from which the "heat" and "amino acids" (which contain metal and washing agent particles) can be derived as output, see Fig. 4.

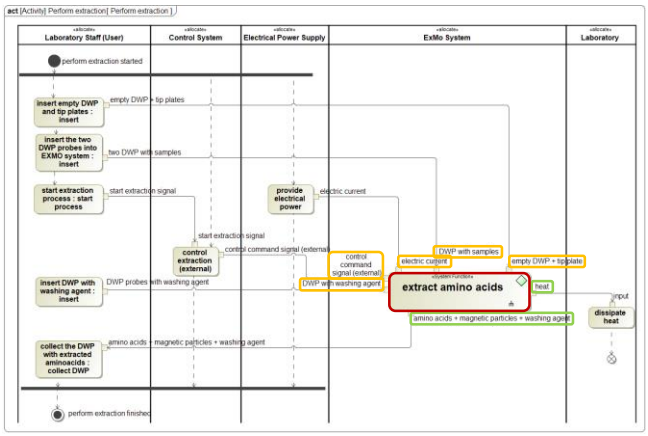


Fig. 4. Systematic detection of inputs and outputs of the system function by a detailed description of the use case

When checking the validity of the function in step 4, it can be seen that all of the inputs and outputs of the system function cross the EXMO system border (viz. swimlane), which indicates that it is indeed a function of the highest system level and not a subsystem function.

After defining the basic system function, and furtherly decomposing the system architecture and specifying the more specific system functions in a first instance with the help of a more detailed use case scenario description as well as process-specific or rather technological expert knowledge along with reference product insights (see Fig. 5), the previously proposed guidelines can be illustrated at various points to successively decompose these system functions.

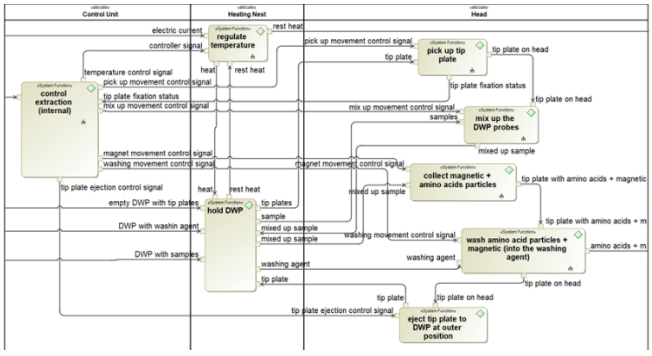


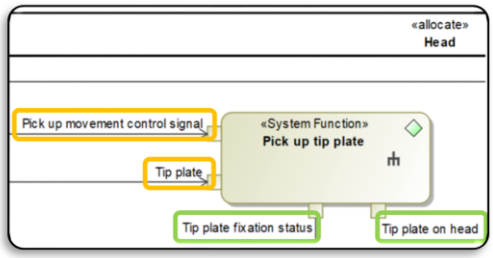
Fig. 5. Systematic definition of the main system functions of EXMO

Starting with the system function "pick up tip plate" being allocated to the logical element of the "head system" on hierarchy level 1 (a tip plate is a single use medical add-on, usually made of plastic and used to mix samples), the systematic view to required subfunctions addressing the successive transformation of a functional input into its output (Guideline 1) leads to a further functional decomposition.

[1] DWP or Deepweel Plate is a standard medical container for storing samples or process agents

Guideline 1 – functional inputs & outputs

Question: "Which subfunctions are needed to process the input and transform it into its output?"

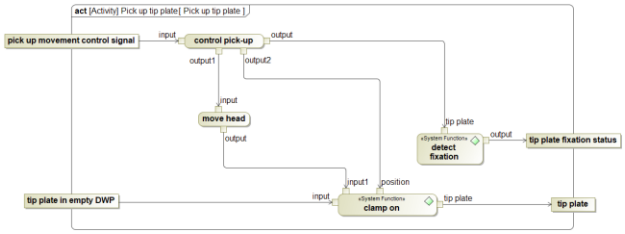


Input 1: "tip plate"
Input 2: "pick up movement control signal"

Output 1: "tip plate on head"
Output 2: "tip plate fixation status"

Explanation/Description:

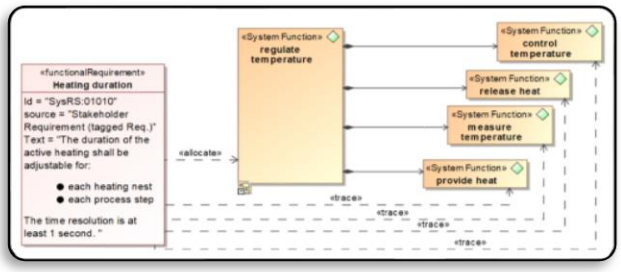
Having a look to the first pair of input and output, there is a necessity to define at least two potential subfunctions to transform the inputs into outputs. Here, one subfunction represents the movement needed to place the head into the correct position for picking the tip plate. The other subfunction represents the need to attach the tip plate to the head.
In the same way the potential subfunctions for the other pair can be defined. Here, the input regarding the control signal emerges a subfunction to control the movement and attachment, whereas the output requires a subfunction to send the fixation status to the device control.
Based on these deductions not only the needed subfunctions of the system function are further on defined, but also the corresponding link is inherently stated.



Having a look to the system function to "regulate temperature" being allocated to the "heating nest", here the closer look to the attached requirements leads to a further functional decomposition.

Guideline 2 – attached requirements

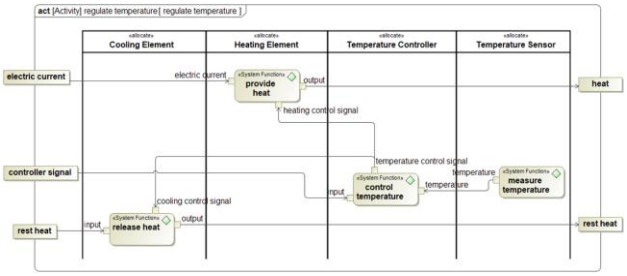
Question: "Does any linked functional requirement suggest or require a specific subfunction?"



Linked functional requirement: "heating duration"

Explanation/Description:

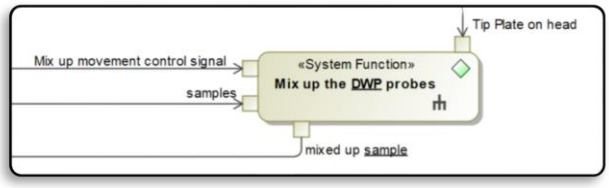
System or stakeholder requirements (directly or indirectly) linked to a specific function can provide further information about needed subfunctions. Here, the requirement "heating duration" and its description that "the duration of the active heating should be adjustable for each heating nest and each process step" already tells us that an active temperature measurement and control is necessary. Consequently, four subfunctions "control temperature", "provide heat", "measure temperature" and "release heat" are needed and thus can be defined.



In some cases, also the existing knowledge of the engineers about the product (a similar one or a similar function in a different context) is sufficient to define the subfunctions. Thus, for the main function to "mix up the DWP probes" a predefined solution is already known leading to a restricted solution-neutral description in the further functional decomposition.

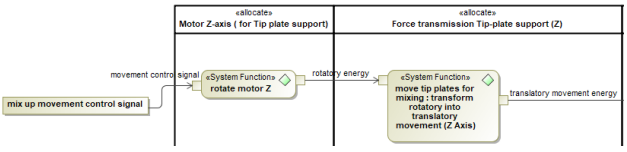
Guideline 3 – general product and engineering knowledge

Question: "Are there any subfunctions to be defined based on general product and engineering knowledge, experience or company-specific pattern?"



Explanation/Description:

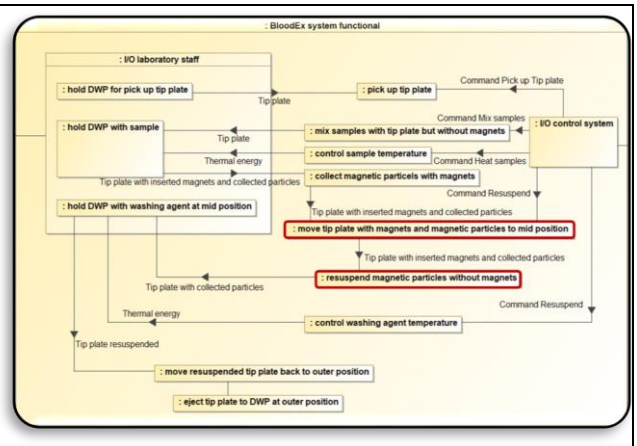
Engineers often doubt the sufficiency of "just" using their expertise, even it can be really useful to specifically decompose the system architecture. In the case to "mix up the DWP probes", the engineers recognized that the needed translational movement is most conveniently realized by an electric motor for such a type of system, so that at this point a solution-specific function can be defined. Additionally, the engineers know that the most convenient type of electric motors is using a rotatory working principle. In consequence, for the translational movement a function for transforming the rotational force into a linear force is required. As a result, this leads to a respective definition of a subfunction since it can be relevant on system level.



For the system model of the EXMO device, there was an older functional architecture from a previous generation as basic input. This model was not clearly decomposed in system layers, but it still allowed to extract some subfunctions as shown below.

Guideline 4 – information from reference products

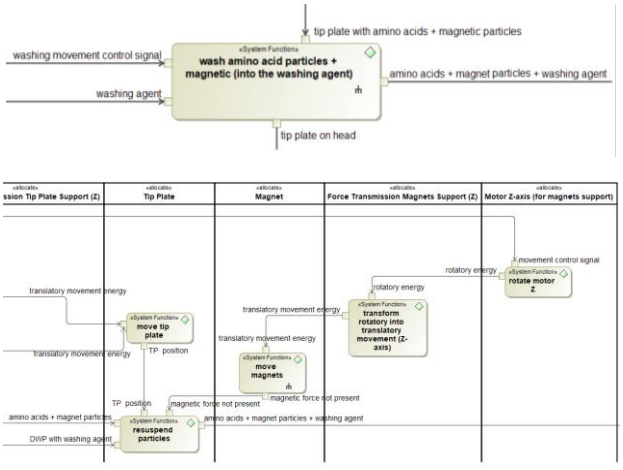
Question: "Can information from reference products (i.e., architectures of previous or similar generations) be at least partly used for decomposing the actual system function?"



Carry-over function 1: ”move tip plate with magnets and magnetic particles to mid position”
Carry-over function 2: “resuspend magnetic particles without magnets”

Explanation/Description:

As described above, the carry-over of subfunctions and mapping them to a function reduces the effort for functional decomposition. In case of a given EXMO reference system, a base functional architecture was already available when the modelling of the demonstrator started. Thus, some functions (e.g., “pick up tip plate”) were directly reused in the new functional architecture without any need for changes. Other functions (e.g., “move tip plate with magnets and magnetic particles to mid position” or “resuspend magnetic particles without magnets”) were identified as subfunctions of a function (e.g., “wash amino acids + magnetic particles”) from an upper system level and only get partly split into more separated functions covering the desired atomic character when allocating to specific logical elements. Consequently, the elaboration of the functional architecture based on a reference system eases the definition and decomposition process by having a proven template with a view of its scope as concrete provider of ideas.



However, there are also some *exceptions for decomposition*, i.e. for some functions the next decomposition layer does not necessarily coincide with the next system layer. This can occur when a system layer L(n+1) is needed by some functions but not by others. In this case, the function from system layer L(n) can be repeatedly allocated to another logical element from system layer L(n+1) without decomposing it, and thus the function will be explicitly allocated to two logical elements. Since these two logical elements belong to two different system levels and are themselves in a decomposition relation, this multiple allocation does not contradict the previously defined boundary conditions.

For the EXMO demonstrator a similar case could exists for the “control movement” function. If the system element “control unit” is first logically decomposed into different subsystems or rather modules on system level 3 (e.g., “control module 1” and “control module 2”) before specifying the logical subcomponents of the controllers, then the “control movement” function can be meaningfully decomposed only when specifying the functions of the particular controller, which is the case on system level 5 (see Fig. 6). Thus, the “control movement” function on system level 4 is allocated to the “movement controller” without any changes.

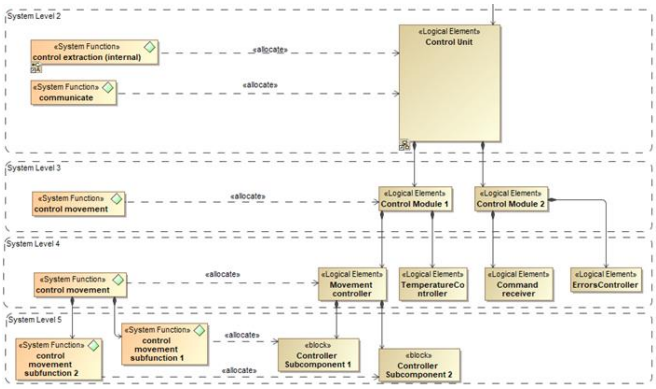


Fig. 6. Visualization of an exemplarily exception for decomposition

With that in mind, and answering the always arising question “*when is the functional decomposition finished?*”, the system and its functions can be decomposed until one of the following cases occur:

- For a given function, a logical element is found that can be already mapped to a specific engineering domain with concrete detailed design ideas, and thus the system element does not need to be furtherly decomposed
- A logical element is found that represents a subsystem whose functional decomposition is not relevant for the current system, but can be further modelled and decomposed in a separate model along with its subfunctions (note: considering SoS mindset)

Again, using the example of the EXMO demonstrator, for example the function “rotate z-axis motor” to support the tip plate is allocated to the logical element “motor z-axis”, where a further functional decomposition is not directly relevant for the entire EXMO system, but only for the independent development of the subsystem “motor z-axis”. Thus, the decomposition of the “motor z-axis” is only required if it cannot be mapped to a single engineering domain (e.g., the “motor” is still a mechatronic system) or it is not purchased, but developed in-house, so that a separate modeling in an own model becomes necessary.

V. Conclusion

Having evaluated the actual state of the art and stated the derived needs for a revised approach to systematically elaborate a functional architecture, this contribution first presents a general procedure with concrete guidelines to strategically define and decompose functions for both perspectives a greenfield and brownfield environment. By providing real assistance with defined guiding rules and control criteria, thus the mapping between functional and

logical interfaces can be tackled much more efficiently, which is partly shown on a practical example.

VI. Discussion and Outlook

With a specific view to project or company-specific topics, however, there are often some further restrictions or boundaries and their influence on the regular functional decomposition apart from the pure distinction between a complete new development (without any preparatory logical architecture) and adapted constructions (with an existing logical base architecture), such as make-or-buy decisions, product line engineering & reusability, predefined standard functions for internal collaborations with different company divisions as well as a technical vs. organizational or divisional relatedness.

Starting with the aspects of make-or-buy decisions, the system model and its functional decomposition is immediately affected depending on the decision of buying or developing subsystems. In the case of finding a logical element that can be mapped directly to a purchased technical component, only the top-level functions for this logical element are needed to be elaborated. The decomposition of the top-level functions is not required. However, if the company decides to develop a subsystem rather than purchasing it, then the further definition and decomposition of the functions are necessitated.

How functions are defined, decomposed and grouped by allocating them to logical elements may also depend on the nature of the product. If it is a one-off product or a product with few variants (e.g., specialized manufacturing equipment), then the reusability of the system functions and architecture plays a minor role. For each such system, here, new functions are required and a new system architecture needs to be created where the previously defined functions and their decomposition can be used only as a reference. However, if the system is to be part of a product line or many variants exist, then the direct reusability of common parts and thus of functions and parts of the system architecture is desirable.

Taking up this last aspect again from another perspective, standard functions may also be predefined at the highest level when collaborating with different company divisions. This could be a thought for an early design freeze with defined high-level functions before starting with a further functional decomposition of products whose core functionality just barely changes from generation to generation. Thanks to this predefinition, a clear functional scope for the product is available for all collaborating divisions right at the beginning of the modelling of the functional architecture, so that lengthy analyses of the high-level functional cut are not necessary anymore.

Looking to the aforementioned aspect of a respective relatedness, the functional decomposition and its allocation to logical elements can be further influenced not only by technical factors, but also by the company structure. For example, selected electronic functions can be deliberately allocated to one logical element in order to simplify the complex processes of developing electronic components by giving the responsibility for a further elaboration to a dedicated group of experts (e.g., for sensor technologies).

Moreover, and with a further regard to collaborations with different company divisions, internal engineering processes and standard practices can also affect the previously described functional decomposition, where system architects may prefer to divide functions according to their affiliation to a technical domain, since this can facilitate the subsequent product development within the technical domains. Illustrating this issue on the example of the EXMO function “control temperature” (see Fig. 7), in the current functional decomposition this function belongs to “regulate temperature”, since it is required to regulate the heating and cooling cycles of the “heating nest” (see Fig. 5). This type of decomposition can be described as “mechatronic decomposition”, since subfunctions are not separated according to engineering domains to which they belong.

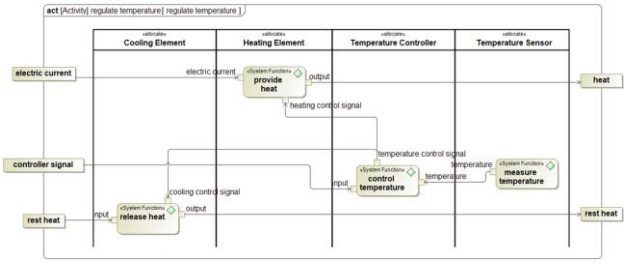


Fig. 7. Functional allocation of the subfunction “control temperature”

In contrast, another strategy would be to decompose the functions according to their respective technical domains or at least to focus more on these. Thus, the subfunction “control temperature” can still be allocated to the logical element “temperature controller”, albeit this is now part of the system “control unit” and no longer has a composition relationship to the “heating nest”. Since the logical system “control unit” is part of one engineering domain (namely electronics), it can now be used as subfunction in the function “control extraction (internal)”, as comparatively depicted in Fig. 8. Here, the information input needed for “regulate temperature” to control heating and cooling cycles would be provided by an object flow from “control extraction (internal)” (see Fig. 5).

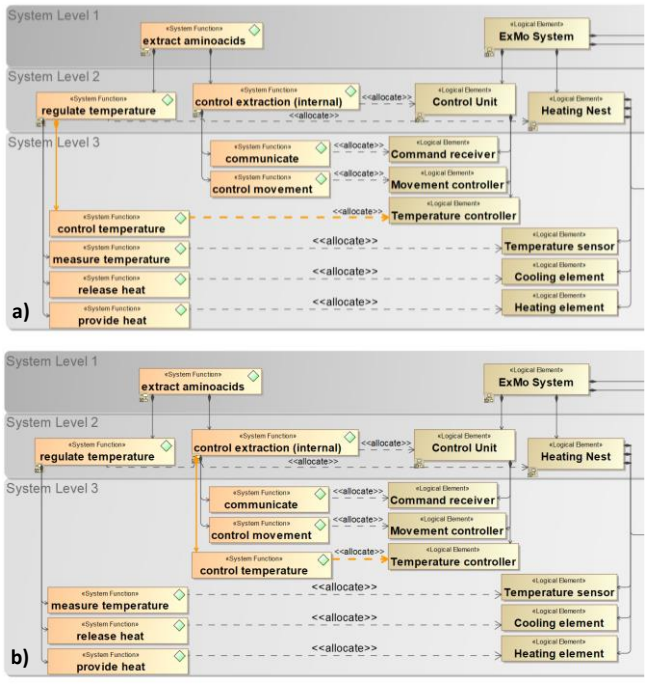


Fig. 8. Illustration of both collaboration-driven decomposition strategies: a) mechatronic decomposition, b) domain-oriented decomposition

In the end, the advantages and disadvantages of both decomposition strategies need to be evaluated by the

company's experts and how they would like to handle it further on. Notwithstanding this, this topic should be further analyzed based on further examples in order to gain a better understanding of the individual benefits and finally to derive further consultancy advices.

Given the fact that all these advanced topics need a further consideration also in association with the consequent application of an adaptable zigzagging procedure between requirements, functions and at least logical elements, the subsequently commenced research activities focus on the further integration of company- specific or rather case-specific circumstances into the previously introduced guidance to systematically define and decompose a functional architecture. Here, a human-centered modeling advisor accessing the current state of the model and all its stored data and project-specific knowledge is envisaged as an accompanied add-on directly within the operating systems modeling tool.

ACKNOWLEDGMENT

The presented article is part of the research project "CyberTech - Advanced Systems Engineering for the Design of Cyber-Technical Systems". This research and development project is funded by the German Federal Ministry of Education and Research (BMBF) within the "Innovations for Tomorrow's Production, Services, and Work" program (funding number 02J19B010) and implemented by the Project Management Agency Karlsruhe (PTKA). The authors are responsible for the content of this publication.

REFERENCES

[1] M. E. Porter, J. E. Heppelmann, "How smart, connected products are transforming competition," Harvard business review, vol. 92, pp. 64–88, 2014.

[2] S. Friedenthal, "Object Oriented Systems Engineering," Process Integration for 2000 and Beyond: Systems Engineering and Software Symposium. New Orleans, Lockheed Martin Corporation, 1998.

[3] A. Friedenthal, A. Moore, R. Steiner, A practical guide to SysML: The systems modeling language. Amsterdam: Elsevier, 2008.

[4] T. Weilkiens, Systems Engineering with SysML/UML: Modeling, analysis, design. Amsterdam: Elsevier, 2007.

[5] J. G. Lamm, T. Weilkiens, "Functional architectures in SysML," in Tag des Systems Engineering 2010, M. Maurer and S.-O. Schulze, Eds. München: Carl Hanser Verlag, 2010, pp. 109–118.

[6] J. A. Estefan, "Survey of model-based systems engineering (MBSE) methodologies," INCOSE MBSE Initiative. Seattle: International Council on Systems Engineering, 2008.

[7] B. Lucero, V. K. Viswanathan, J. S. Linsey, C. J. Turner, "Identifying critical functions for use across engineering design domains," Journal of Mechanical Design, vol. 136, pp. 121101/1–11, 2014.

[8] M. Agyemang, J. Linsey, C. J. Turner, "Transforming functional models to critical chain models via expert knowledge and automatic parsing rules for design analogy identification," Artificial Intelligence for Engineering Design, Analysis and Manufacturing, vol. 31, pp. 501–511, 2017.

[9] G. Telleschi, A. Caroni, E. Willingham, P.-H. Pradel, "Reference missile functional architecture, addressing design in a multinational defense company," in Proceedings of the 4th INCOSE Italia Conference on Systems Engineering (CIISE), E. Mancin, A. Garro, L. Tirone, D. Fierro, P. Gaudenzi, A. Falcone, Eds. Aachen: RWTH Aachen, 2018. pp. 31–37.

[10] S. Behere, M. Törngren, "A functional reference architecture for autonomous driving", Information and Software Technology, vol. 73, pp. 136–150, 2016.

[11] T. Weilkiens, SYSMOD - The systems modeling toolbox: Pragmatic MBSE with SysML, 3rd ed. MBSE4U, 2020.

[12] A. Albers, N. Bursac, E. Wintergerst, "Product generation development – importance and challenges from a design research perspective," in New Developments in Mechanics and Mechanical Engineering, 2015, pp. 16–21.

[13] I. M. Mactaggart, C. Eckert, H. Lockett, "Analysis of functional reference architecture through an industry lens," Proceedings of the International Conference on Engineering Design (ICED21), vol. 1, pp. 467–476, 2021.

[14] G. Pahl, W. Beitz, J. Feldhusen, K.-H. Grote, Engineering design: A systematic approach, 3rd ed. London: Springer-Verlag, 2007.

[15] N. Crilly, "Function propagation through nested systems," Design Studies, vol. 34, pp. 216–242, 2013.

[16] Verein Deutscher Ingenieure, VDI 2221: Systematic approach to the development and design of technical systems and products. Berlin: Beuth Verlag, 1993.

[17] M. Hundal, "A systematic method for developing function structures, solutions and concept variants," Mech Mach Theory, vol. 25, pp. 243–256, 1990.

[18] J. Hirtz, R. B. Stone, D. A. McAdams, S. Szykman, K. L. Wood, "A functional basis for engineering design: Reconciling and evolving previous efforts," Research in Engineering Design, vol. 13, pp. 65–82, 2002.

[19] K. Pohl, H. Hönninger, R. Achatz, M. Broy. Model-based engineering of embedded systems: The SPES 2020 methodology. Berlin, Heidelberg: Springer-Verlag, 2012.

[20] D. D. Walden, G. J. Roedler, K. J. Forsberg, R. Douglas Hamelin, T. M. Shortell, Systems engineering handbook: A guide for system life cycle. Hoboken: John Wiley & Sons, 2015.

[21] J. Holt. Systems Engineering Demystified. Birmingham: Packt Publishing, 2021.

[22] J. Tang, S. Zhu, R. Faudou, J.-M. Gauthier, „An MBSE framework to support agile functional definition of an avionics system," in Complex Systems Design & Management, E. Bonjour, D. Krob, L. Palladino, F. Stephan, Eds. Cham: Springer, 2018, pp. 168–178.

[23] G. A. Miller, "The magical number seven, plus or minus two: Some limits on our capacity for processing information," Psychological Review, vol. 63, pp. 81–97, 1956.