Artificial Intelligence for Engineering Design, Analysis and Manufacturing

www.cambridge.org/aie

Function modeling in model-based systems engineering using flow heuristics

Unal Yildirim[1,2,3], Felician Campean[3,4] and Amad Uddin[5]

[1]University of Leicester, Leicester, UK; [2]School of Automotive Engineering, Hubei University of Automotive Technology, Shiyan, China; [3]University of Bradford, Bradford, West Yorkshire, UK; [4]SAFI Verse Ltd., Bradford, UK and [5]Jaguar Land Rover, Coventry, UK

Research Article

Cite this article: Yildirim U, Campean F and Uddin A (2025). Function modeling in model-based systems engineering using flow heuristics. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, **39**, e26, 1–19
https://doi.org/10.1017/S0890060425100176

Received: 15 July 2024
Revised: 15 August 2025
Accepted: 13 September 2025

Keywords:
model-based systems engineering; function modeling; flow heuristics; system state flow diagram; system composer

Corresponding author:
Unal Yildirim;
Email: uy10@leicester.ac.uk

Abstract

Model-based systems engineering (MBSE) is increasingly used across industries for the integrated modeling of complex systems to support model-based development and provide enhanced traceability between requirements and verification and validation of the system. This paper seeks to strengthen the function modeling methodology in MBSE by introducing an approach based on flow heuristics guided by the System State Flow Diagram schema. This provides integrated function architectures with an enhanced integrity in MBSE. The approach is illustrated with a case study of an electric bicycle implemented in the MathWorks System Composer environment.

Introduction

Background and motivation

Models serve as the primary information resource in engineering product creation, effectively capturing the intricate interconnections across multiple disciplines – from strategy and design to production, service, and end-of-life considerations (IDE, 2021). The growing complexity of customer requirements, manifested through increased product variations and features, requires more efficient development and validation methodologies across the entire product lifecycle (Hoff and Scott, 2019; Borky and Bradley, 2019). This complexity is particularly pronounced during early conceptual design phases, where requirements uncertainty is highest, and during validation and verification activities where system integration challenges emerge (Campo et al., 2023). This evolution led to the emergence of model-based systems engineering (MBSE), a comprehensive approach that facilitates requirements identification, design, analysis, and validation of complex systems (Cameron, 2018; Brusa et al., 2018; Borky and Bradley, 2019). The widespread adoption of MBSE across industries (e.g., Ferrogalini et al., 2019; Davey, 2022) has yielded significant advantages for organizations, particularly in enhancing productivity through streamlined requirement traceability and improved interdisciplinary collaboration.

The evolution from traditional document-based approaches to model-centric development reflects the increasing complexity of modern engineering systems and the need for enhanced collaboration across multidisciplinary teams (Madni and Sievers, 2018; Estefan and Weilkiens, 2022). MBSE, coupled with simulation, multi-disciplinary analysis, and immersive visualization environments, has the capability to harness automation and computation for simulating dynamic system behavior, conducting multi-disciplinary analysis, visualizing, and managing system design throughout its lifecycle. The digital representation can facilitate a collective comprehension of the system among its stakeholders (INCOSE, 2021). This offers a great potential for a wider adoption of the MBSE methods and tools across industries, particularly during conceptual design and system validation phases (Davey, 2022; Zhang et al., 2022). The potential for wider MBSE adoption is most pronounced during conceptual design phases, where functional architectures can guide early system definition and requirements allocation, and during validation and verification stages, where model-based approaches enable systematic testing and analysis (Jacobs et al., 2022; Zhang et al., 2022). Additionally, MBSE methods show particular promise in system integration phases of complex multidisciplinary projects, where the digital representation facilitates cross-domain collaboration and reduces integration risks (Ferrogalini et al., 2019; Husung et al., 2021). One of the major limitations to this wider take-up of MBSE is related to tools and language constraints with associated problems such as complexity management (Chami and Bruel, 2018; Campo et al., 2023) to facilitate consistent deployment across multi-disciplinary systems with complex multi-physics behaviors (Madni and Sievers, 2018). A significant evolution of the MBSE methods and tools is needed to promote their use by systems design practitioners in industry. This requires a paradigm shift in the development of the MBSE methods and tools by enhancing their integration with "design thinking" and the array of methods and tools necessary for fulfilling product development (PD) tasks.

The field of design theory and methodology (DTM) has evolved over many decades to assist designers with how to design (rather than what to design) by introducing a rich collection of studies on design processes and activities with a focus on their application in different domains to ensure consistency and productivity. Therefore, there is a great potential in enhancing the effectiveness and applicability of MBSE methods and tools across complex multidisciplinary systems through key fundamental concepts underpinning the methods employed in DTM, such as function modeling frameworks for design (Pahl et al., 2007). This integration becomes particularly valuable during functional architecture development and requirements decomposition activities, where systematic DTM approaches can provide structured guidance for MBSE practitioners (Zhang et al., 2022). Importantly, this integration does not rely on detailed behavioral simulation capabilities typically associated with later design phases, but rather leverages MBSE tools' capacity for systematic functional modeling and architectural representation during early conceptual design (Drave et al., 2020). The DTM-MBSE integration focuses on structured function modeling approaches that can capture design intent and functional relationships before sufficient detail exists for behavioral simulation, thereby bridging the gap between conceptual design thinking and systematic model-based development. This approach recognizes that while detailed behavioral simulation requires extensive system definition typically available only in later design phases, MBSE tools can effectively support conceptual design activities through functional architecture modeling, requirements traceability, and systematic decomposition of design problems. The integration with DTM methods specifically targets this early-phase application of MBSE, where systematic function modeling can guide design exploration and decision-making before transitioning to detailed simulation-based validation.

"Function Modeling", defined by Erden et al. (2008, p. 147) as "the activity of developing models of devices, products, objects, and processes based on their functionalities and the functionalities of their subcomponents", has a critical role in the modeling and development of systems, as recognized by both DTM and MBSE research communities. A plethora of function modeling methods has been developed in the last decades in DTM research (see Tomiyama et al., 2009 for a review) and a set of criteria has been established for evaluating and benchmarking function models (Summers et al., 2017). Numerous scholars have introduced guidance for practitioners for the development of function models in a structured way, enhancing the consistency of the models developed by practitioners, e.g. Otto and Wood (2001).

Function modeling in MBSE mainly revolves around modeling tools provided by Systems Modeling Language – SysML (OMG, 2019), namely activity diagrams, sequence diagrams, and state machine diagrams. The consideration of functions as activities in activity diagrams enables the capture of the physical behavior of complex multidisciplinary systems in a broader context as compared to sequence diagrams and state machine diagrams. The extracted functions and functional requirements can directly be associated with a use case of the system of interest, describing its functionality in terms of the interaction of the user with the system. The physical architecture (PA) of the system can be described without defining a functional architecture (FA) for the system by associating functions to design solutions (i.e. component/subsystem) for the fulfillment of the functional requirements. This offers a practical way of managing the requirements; however, it generally impedes the use (e.g. for performance simulation) and re-use of the developed model, e.g. for product variants. This prompted further studies on the development of better frameworks for function modeling in MBSE, mostly based on using SysML modeling tools (e.g. Drave et al., 2020). These studies have mainly been focused on the use/reuse of function modeling in support of various PD activities, e.g. FMEA (Pearce and Friedenthal, 2013), safety analysis (Biggs et al., 2018) and development of Digital Twins (Bickford et al., 2020). There are also many good examples from academic and industry practice on the integration of MBSE with modeling environments supporting system performance analysis and physical behaviour simulation (e.g. Forlingieri and Weilkiens, 2022).

Research objective and methodology

While research in MBSE has advanced function modeling, the practice of developing separate functional models for different use cases, common in many applications, often leads to integration challenges and engineering change in downstream product development, and limits design flexibility and future system evolution (Kößler and Paetzold, 2017; Yildirim et al., 2017; Meißner et al., 2021). This raises a critical research question: "*How can an integrated FA of a system that effectively combines functions across multiple use cases be systematically developed?*" To address this question, this paper introduces a novel approach to creating an integrated FA that merges functions from different use cases into one comprehensive model. We establish systematic guiding principles, inspired by the well-established flow heuristics of Otto and Wood (2001), to support the integrated development of FAs for complex systems. This approach not only underpins physical architecture development but also bridges DTM research with MBSE practice, promoting more effective use of MBSE methods and tools.

This research employs a collective case study approach (Yin, 2018) to provide methodological validation of how flow heuristics can systematically support the development of integrated function architectures across multiple use cases. The cases were selected to demonstrate the methodology across varying levels of complexity, following a structured implementation process that included use case identification and flow heuristics application, and comparison with traditional approaches. Detailed guidelines were developed and validated through peer review to ensure reproducibility.

The research uses a systematic two-tier validation approach to evaluate the proposed schemata's effectiveness and scalability. The first validation tier uses desktop cases of a bread toaster and a coffee machine, as foundational studies combining basic use cases from the relevant previous works (Kurfman et al., 2000; Eisenbart et al., 2017) to illustrate the proposed schemata. These cases revealed shared functions and interactions previously obscured in separate models, while demonstrating improved consistency and completeness in the integrated approach. The second validation tier is based an electric bicycle (e-bike) case to systematically test the capability to handle increased complexity and multi-domain integration. This case specifically focused on managing concurrent operational modes with cross-domain interactions and complex dependency chains across use cases. Through rigorous peer debriefing and structured documentation, these validation tiers demonstrate both the validity and scalability to complex systems. The peer debriefing process involved systematic review sessions with the industry co-author, examining both theoretical foundations and practical implementation aspects with particular attention to industrial applicability across different complexity levels.

While acknowledging the co-author's dual role in the validation process, structured evaluation protocols were employed to ensure

https://doi.org/10.1017/S0890060425100176 Published online by Cambridge University Press

objective assessment. This included comparative analysis against traditional separate use-case modeling approaches to benchmark consistency, completeness, and model reusability. Specific validation criteria assessed the methodology's ability to: (1) reveal previously obscured shared functions and interactions between use cases, (2) demonstrate improved traceability between functional and physical architectures, and (3) effectively manage concurrent operational modes with cross-domain interactions. This dual academic-industry validation perspective strengthened the methodology's credibility by combining theoretical rigor with practical implementation considerations, particularly evident in the e-bike case study's complex multi-domain integration requirements.

System composer (SC), an MBSE modeling environment developed by MathWorks, was chosen for the demonstration of the e-bike case study due to its robust integration capabilities with other MathWorks tools, facilitating seamless multi-disciplinary analysis and simulation, which can be used in future work (e.g. execution of the developed functional architecture in simulation environments).

The next section introduces a literature review, while "*Flow heuristics for function modeling*" section outlines the function modeling methodology based on case studies of a bread toaster and coffee machine. "*Electric bicycle case study*" section introduces the demonstration of the methodology for function modeling in MBSE using the flow heuristics on the e-bike case study. The "*Discussion*" section introduces a discussion on the proposed methodology, followed by "*Conclusions*".

Literature review

Function modeling in engineering design

Function modeling in DTM focuses on the representation of a device-centric view (Chandrasekaran and Josephson, 2000) of systems by developing function chains based on operations on flows which describe "way of achievement" capturing functional knowledge (Kitamura and Mizoguchi, 2003). Many well-established function modeling schemes in various engineering disciplines, such as Otto and Wood (2001) and Ulrich and Eppinger (2003), are based on a flow-based function modeling methodology developed by Pahl et al. (2007) which defines a system in terms of the flows of energy, material, and signal (information), and decomposes the system's main function into subfunctions with reference to these flows. Pahl et al. (2007) introduced the concept of "main function" and "auxiliary function" in function modeling. While the main functions are associated with the fulfilment of the overall function of the system under consideration, the auxiliary functions contribute to the fulfilment of the main functions. Main and auxiliary functions constitute "Functional Architecture (FA)", describing functions and the relationships between them. Pahl et al. (2007) also introduced a way of developing "Physical Architecture (PA)" describing the physical hardware or platform needed for the fulfillment of functions in FA. They suggest finding working principles for the subfunctions and combining these principles into a working structure, that is, system synthesis. The concretization of the working structure leads to the principal solution.

Stone et al. (2000) expanded on the methodology of Pahl et al. (2007) by introducing a set of module heuristics to map FA to PA: dominant flow, branching flow, and conversion-transmission modules. These heuristics were introduced with the purpose of underpinning the development of product architecture based on a function model; however, they also shed some light on the way of developing a function model. By following a similar logic and approach with module heuristics of Stone et al. (2000), Yildirim et al. (2017) introduced function modeling heuristics based on the System State Flow Diagram (SSFD) methodology in order to provide a guidance to the practitioners in the development of function models for complex systems and to improve the consistency of the models developed by practitioners: main flow heuristic, connecting flow heuristic, branching flow heuristic and conditional fork node heuristic. Campean et al. (2018) discussed the use of the function modeling heuristics in the analysis of a complex system with multiple operation modes and with multiple levels of decomposition by introducing three types of nesting of function models.

While some function modeling schemes adopted the methodology of Pahl et al. (2007), there are some other modeling schemes introduced as variations of this methodology. For example, the Integration DEfinition for Function modeling-IDEF0 (Buede, 2009) introduced the flows for "controls" and "mechanisms" which complement the input/output flows of material, energy, and information of the system. The flows of "controls" guide the transformation process, whereas the flows of "mechanisms" (considered as a physical architecture (PA) element) are used to perform the function(s), and they can be associated with PA structures where a function is considered as a functional architecture (FA) structure.

As an alternative to the flow-based function modeling, the concept of state is used in function modeling, where a function is represented in terms of a state transition. Harel's (1987) state charts provide a basis for many well-established state-based methodologies. For example, the Contact and Channel Approach – C&C^2-A (Matthiesen and Ruckpaul, 2012) describes a function of a system as a sequence of at least two states where PA elements are incorporated in the introduction of the function through the Channel Support Structure (CSS). SSFD of Yildirim et al. (2017) integrated flow-based functional requirements reasoning with a state-based representation to provide a systematic approach for function modeling framework of complex multidisciplinary systems with multiple operation modes. Some approaches relate states to the concept of "behaviour" in function modeling, e.g. the Object–Process Methodology of Dori (2016). There are also approaches incorporating states into the function modeling of a system, such as the integrated function modeling approach of Eisenbart et al. (2017) which refers to "state" as a "view" in the function model of a system.

There are some other function modeling approaches that have been introduced for various purposes. For example, Muller et al. (2019) focused on the promotion of incremental product development by introducing enhanced function-means methodology, while Wichmann et al. (2018) proposed function integrity diagnosis and documentation method in the diagnosis of function integrity for risk identification and documentation purposes.

The evolution of function modeling in DTM has produced robust methodologies, particularly in flow-based and state-based approaches, with established heuristics for systematic model development. While these approaches offer valuable frameworks for functional decomposition and analysis, their integration with modern MBSE practices remains limited. The well-structured principles from DTM, especially the flow heuristics and systematic function decomposition approaches, provide a foundation

https://doi.org/10.1017/S0890060425100176 Published online by Cambridge University Press

that could enhance MBSE practices, particularly in developing integrated functional architectures across multiple use cases.

Function modeling in model-based systems engineering

Numerous scholars (e.g. Husung et al., 2021; Estefan and Weilkiens, 2022) have discussed the use and the benefits of MBSE methodologies to industry. For example, Davey (2022) discusses how MBSE is applied to various automotive systems analysis and design. Functions in MBSE provide a way of fulfilling engineering requirements that are derived from many places, including user needs and regulatory bodies. Requirements allocated to functions enable product teams to understand what criteria to use when designing specific functions which are decomposed from high-level functions and delivered by physical components that address the requirements allocated to these functions. Systems from one generation to the next will be functionally the same to some extent, and functions of FA of a system can be reused to support multiple generations of technology where PA of the system can be developed further with the evolution of technology (Lamm and Weilkiens, 2010). Maintaining a function focus throughout MBSE supports function failure mode analysis to identify the potential function failure modes through functional requirements. Consideration of possible failures in a design – like safety, performance, and reliability – enables engineers to plan how to alter the development/manufacturing process in order to avoid these failures (Siemens, 2019).

Discussion of functional approaches in the MBSE mainly revolves around SysML (OMG, 2019) as its use in the MBSE is ubiquitous in both academia and industry (Albers and Zingel, 2013; Lu et al., 2018). SysML's strength lies in high-level system architecture modeling through a set of pre-defined diagrams for various aspects of model-based development of multidisciplinary systems. Its behaviour diagrams (use case diagram, state machine diagram, sequence diagram, and activity diagram) support the capture of logical functional requirements across a system's lifecycle, while modeling of detailed dynamic behaviors, particularly for complex physical interactions, can be done through complementary modeling approaches. MBSE approaches function modeling from a different perspective as compared to approaches in DTM. An "overall function" in DTM is represented as a "Use Case (UC)" which drives the development of function modeling in MBSE. Each use case describes a functionality of a system which is achieved through the interaction between actor(s), which can be a user or other external entity, and the system under consideration. Sequence diagrams, activity diagrams, and state machine diagrams detail how a use case fulfils relevant functionality of a system by providing a function modeling methodology. A state machine diagram represents the state of a system, subsystem, or component throughout its use case in terms of state transitions, where transitions between states are denoted by lines. An activity diagram, similar to the use of a traditional functional flow diagram, illustrates the behavior of a system, subsystem, or component throughout its use case in terms of activities that requires the flow of inputs, outputs, and control. A sequence diagram represents a behaviour related to a use case in terms of a sequence of message exchanges representing the interactions between actors and the system. While the focus of the sequence diagrams is the mapping of message-based information interactions, Friedenthal et al. (2012) argue that these diagrams also have the capability to represent the flow of material and energy.

Many scholars explored the integration of relevant SysML behavior diagrams with function modeling methods introduced in DTM to enhance the practical applicability of function modeling methods and their transfer to MBSE. For example, Grobshtein and Dori (2011) introduced automatic generation of SysML views from an OPM model, whereas Zingel et al. (2012) aimed to enhance C&C^2-A in the function modeling of technical systems through its integration with SysML. Eisenbart et al. (2015) introduced a comparison between the IFM framework and SysML to provide a basis for the implementation of the former in SysML.

Instead of using SysML directly in the improvement of the practical applicability of function modeling, numerous scholars focused on the improvement of SysML behaviour diagrams in order to capture functional requirements in a more structured way. Sequence diagrams seem to be the focal point in this respect. For example, Zingel et al. (2012) and Zhu et al. (2019) introduced a methodology to develop function-based models from activity diagrams and sequence diagrams. Recently, Yildirim and Campean (2020) introduced the Enhanced Sequence Diagram by augmenting traditional sequence diagrams with flows/exchange-based information.

While MBSE and SysML provide comprehensive tools for system modeling, their approach to function modeling differs significantly from DTM methodologies. The use-case driven nature of MBSE function modeling, while practical for requirement management, often leads to isolated function models for different use cases. This highlights a critical gap in current MBSE practice: the lack of a systematic methodology for developing integrated functional architectures that can effectively combine functions across multiple use cases while maintaining the rigor of DTM approaches.

Function modeling challenges in MBSE

Many organizations follow a reuse approach to complete initial modeling on a particular project where a set of data is copied and pasted from one context to another (Chami and Bruel, 2018) to reallocate modeling resources to other projects (Purohit and Madni, 2022). The reused sub-systems contain information about the implementation of sub-functions, including intended input/output parameters of these sub-functions (Husung, 2023). Requirements can change throughout the development process, and the change of the copied source (e.g. input parameter of a function) can affect the implementation of the sub-function and therefore functionality of the associated sub-system. The functional architecture (FA) and the PA of the system need to be individually maintained while remaining mutually consistent (Madni and Sievers, 2018) which is a challenge due to increasingly complex systems with intricate interactions between various sub-functions and subsystems.

A function can often be fulfilled with different physical solutions from one or more domains. A function-oriented development process using MBSE methods offers a structured methodology to the development of systems through systematic development of functional architectures (Jacobs et al., 2022). This, in turn, underpins the reuse of function/function architectures, which are represented at different levels of abstraction. The adoption of standardized modeling languages (e.g. SysML) and frameworks enhances consistency and reuse of function models throughout the engineering process. However, the way of using the adopted language and framework in the development of a function model can still hinder the reuse of FAs across different stages of the engineering lifecycle. There is an ambiguity in natural language descriptions when it comes to defining functions. Albers and Zingel (2013) pointed out that the understanding of some terms, such as function, differs significantly even among Systems

https://doi.org/10.1017/S0890060425100176 Published online by Cambridge University Press

Engineers in academia and industry. For example, the activity diagram of SysML, which details the functionality of a system with reference to one of its use cases, does not specify how to define an activity, and this can lead to the interpretation of activities differently by different stakeholders and can lead to misunderstandings and potential inconsistencies in the function model. Stone et al. (2000) attempted to address this by introducing a so-called function library, which consists of function classes, basic functions, and synonyms, where one function is selected as the verb in the verb-object description of a sub-function. The applicability of this on MBSE modeling languages has not been tested systematically. The description and integration/reuse of the sub-systems in MBSE have been investigated by various scholars such as Husung et al. (2022). So far, very little attention has been paid to the reuse of functions/function architectures. Highly complex systems with numerous interconnected functions require the representation of functions at different levels of abstraction. MBSE modeling languages do not offer a particular methodology for the development of a function model and the transition of the developed model between levels of abstraction. Elements of the pre-defined diagrams can be used at practitioners` discretion. Function modeling methodology of Stone et al. (2000) is limited to the creation and the aggregation of function chains which are developed based on a black box model where they introduced module heuristics to map functional architecture (FA) to physical architecture (PA), e.g. the identification of a branching flow on the function model and the allocation of a module (sub-system) to this flow. Yildirim et al. (2023) introduced a methodology for FA development in MBSE in conjunction with a system use case where flows, such as branching flows, are considered during the development of the FA instead of when making architecture decisions. The proposed function flow heuristics underpin the reuse of individual functions and FAs in relation to the level of resolution of the analysis; however, this requires further validation, and further work is needed for the implementation of the proposed heuristics on the reuse of functions and FAs. Similarly, the reuse of a use case with its FA across new systems from an existing system needs to be investigated to facilitate function modeling in MBSE, as no research has been found on this to date.

These challenges in MBSE function modeling – particularly in managing complexity, ensuring consistency across abstraction levels, and facilitating reuse – underscore the need for a more systematic approach to functional architecture development. The absence of standardized methodologies for creating integrated functional architectures that can effectively combine functions across multiple use cases while maintaining consistency remains a significant challenge in current practice. This requires an integrated approach that combines the systematic rigor of DTM methodologies with the practical benefits of MBSE tools, particularly in developing comprehensive functional architectures that can better support complex system development and evolution.

Flow heuristics for function modeling

Functions as operations

As discussed in "*Function modelling in model-based systems engineering*" section, function representation in MBSE models revolves around behaviour diagrams, which are related to the DTM approaches centred on the conceptualization of function as operations on flows (Stone et al., 2000). In this work, we focus in particular on the System State Flow Diagram (SSFD) methodology (Yildirim et al., 2017), in which a function is conceptualized as state transition model, with states defined in relation to measurable attributes of objects, represented in a box, as shown in Figure 1a. The SSFD function representation schema is closely associated with Harel (1987)'s statecharts, which underpin the MBSE State Machine`s object-oriented programming formalism. Therefore, SSFD is a convenient DTM function model to associate with MBSE representations.

Like other MBSE tools, the MathWorks Systems Composer (SC) uses blocks for the representation of various conceptual entities, including "functions" as part of function architecture (FA), and "component/subsystem" as part of physical architecture (PA) (MathWorks, 2023). Function blocks are associated with functional requirements, which describe the desired behaviour and characteristics of a system, while component blocks correspond to hardware requirements that specify the physical components and constraints necessary for system operation. Each block contains ports that represent the input/output flow of information, material, or energy, described as measurable attributes of the flow. Figure 1b and Figure 1c show a function presentation and a component presentation, respectively, in SC. Comparing Figure 1a and Figure 1b, shows that the SC function block has a structure compatible with the SSFD, if the ports are assumed to be equivalent to the representation of the inputs and the output states as measurable attributes, in line with the SSFD. Similarly, a direct relationship between function and component can be established in SC by allocating functions to components through ports. As shown in Figure 1, the same input and output states of the function block in Figure 1b are allocated to a PA block (i.e., component/subsystem) in Figure 1c.

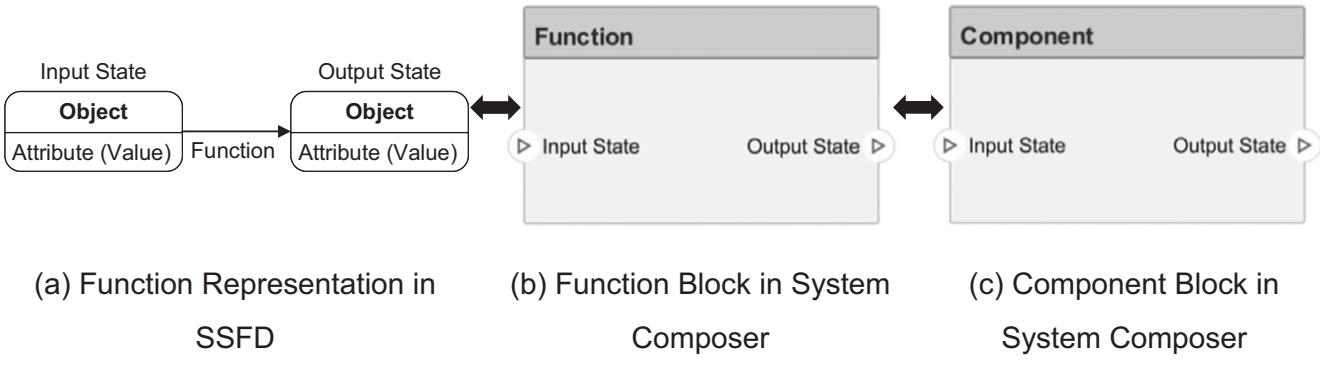

Figure 1. Function and component block representations in SSFD and System Composer.

https://doi.org/10.1017/S0890060425100176 Published online by Cambridge University Press

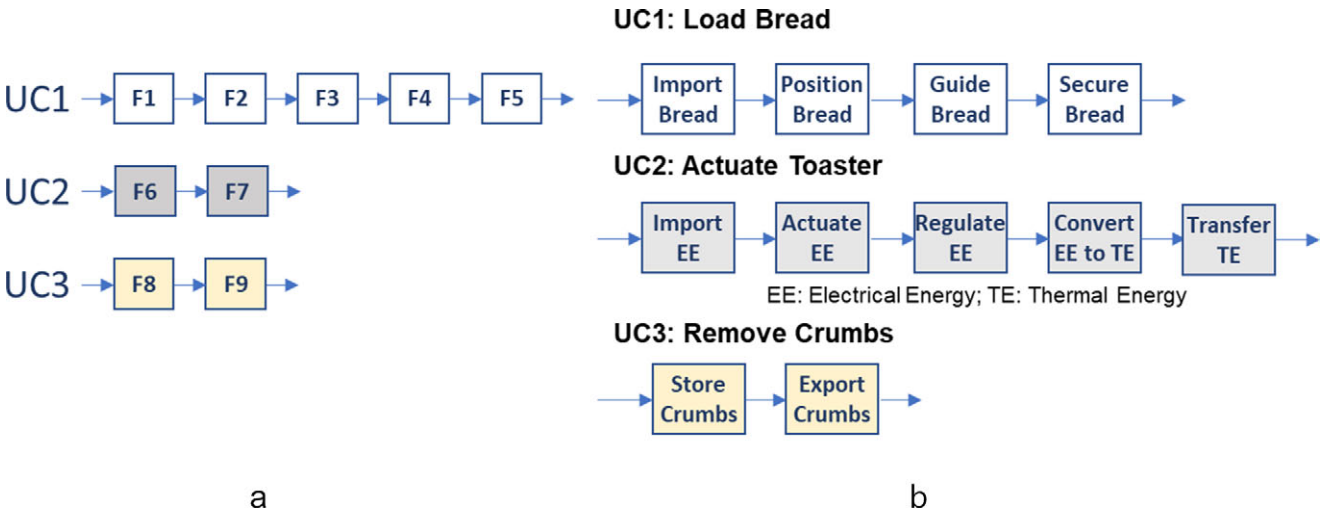

Figure 2. (a) schematic representation of function chains for the main flow heuristic ("F" denotes "Function") (b) three use cases of a bread toaster as an example.

The SC requirements toolbox supports comprehensive requirements management, including software requirements and non-functional requirements such as quality attributes, with full traceability between different requirement types. Stereotypes in SC represent system requirements by defining the overall system design with a common set of properties (such as mass) that can be specified with values to fulfil both functional requirements and stakeholder needs.

The representation of functions as operations on flows, on the basis of a state transition/transformation paradigm, underpins the capture of a functional requirement in a coherent way. For example, in the case of a conversion operation, a functional requirement can be articulated as "transform object <input state> to object <output state>". This functional requirement can directly be linked to a hardware requirement by allocating the FA block to a PA block along with ports.

Flow heuristics for the development of function architectures

This section introduces a set of function modeling heuristics in a prescribed order, following a similar logic and approach to that of Yildirim et al. (2017) and Stone et al. (2000), to represent the function model of a system using the concept of a function block introduced in Figure 1b in the formation of function chains leading to an integrated function architecture of a system with multiple use cases.

Main flow heuristic

Complex multidisciplinary systems generally have multiple modes of operation, corresponding to different use cases. Each operation mode can be independently described by a function chain. This heuristic provides a basis for the development of such a chain by identifying a linear function chain in relation to the fulfillment of the main function of the system, which is represented as a use case (UC) in a use case diagram. Figure 2a illustrates a generic schematic representation of the main flow heuristic for three use cases of a system. Figure 2b exemplifies this by representing three use cases of a generic bread toaster, showing the linear function chains as linked function blocks for three use cases identified as "load bread" (UC1), "actuate toaster" (UC2), and "remove crumbs" (UC3). In Figure 2b, the representation of the function chains is based on the Stone and Wood (2000)`s function blocks and the functional basis taxonomy for function representation, without explicit reference to the SSFD states (illustrated in Figure 1b). In this example, use cases UC1 and UC3 relate to transfer operations on a material flow (bread), whereas UC2 is associated with energy flow operations.

Connecting flow heuristic

The representation of the way of achievement of a UC as a linear (non-branching) function chain through the main flow heuristics captures the first-level system decomposition associated with the UC. Systems with a complex functionality require the aggregation of function chains corresponding to different main flows related to use cases/sub use cases of the system, leading to the synthesis of a system function model which reflects its multiple operation modes. The connecting flow heuristic underpins the aggregation of function chains of use cases of the same system developed through the main flow heuristic by connecting or concatenating function chains. Herein, we consider this in terms of connecting and concatenating two function chains.

Connection of function chains: A function chain of a use case can be connected to the function chain of another case in three distinct cases. a-c.1 in Figure 3 illustrates this schematically based on UC1 and UC3 in Figure 2a. A function chain may require additional resources for the fulfillment of a function, and this can be provided by a function chain of another use case of the same system. This link can be established by connecting the output flow of a function chain to the relevant intermediate or the exit function of another function chain, as shown in a.1 of Figure 3. In some cases, as shown by b.1 of Figure 3, a function can be reused between two function chains, meaning it serves as both an output in one chain and an input in another chain, and these function chains can be connected to each other on this basis. Function chains of two use cases can also be combined based on a function of the respective use case, which is associated with bringing two or more energies or materials together. This requires using this function in the aggregation of the inputs of function chains of these use cases, as shown in Figure 3c.1.

Figure 3a-c.2 shows examples for the representation of connection of two function chains in three distinct cases based on their schematic representation in a-c.1 of the same figure. Use cases are differentiated from each other with the colour of function boxes, as

https://doi.org/10.1017/S0890060425100176 Published online by Cambridge University Press

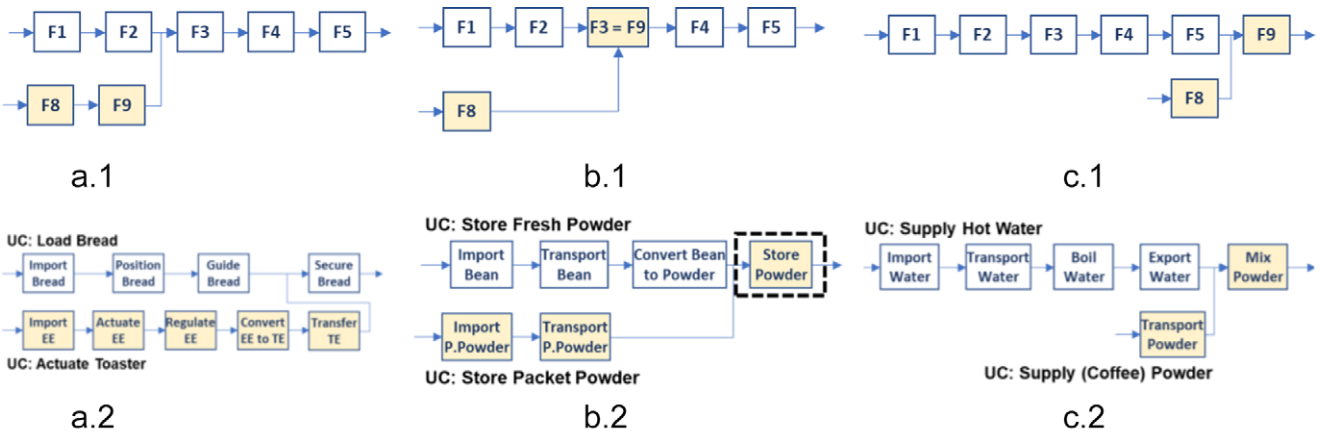

Figure 3. Schematic representation of connection of two function chains in three distinct cases (a-c.1) and respective examples based on use cases of a bread toaster (a.2) and coffee machine (b-c.2).

illustrated in Figure 2a. Figure 3a.2 shows function chains for two use cases of a bread toaster: "Load Bread" and "Actuate Toaster". The output of "Transfer TE" is connected to the input of "Secure Bread". Figure 3b.2 illustrates the function chains for use cases of "Store Fresh Powder" and "Store Packet Powder" of a coffee machine. These use cases relate to each other based on the reuse of "Store Powder" function block indicated with a dotted frame in b.2. In upper chain flow of b2, function "store powder" (counted as the fourth function)" is the last function with input and output states being powder. In the lower chain of b2, "store powder" is the last function (but counted as the third function) with input and output states being powder. In both upper and lower chains, "store powder" is the same common reusable function even though it is at the fourth place in the sequence of functions in "Store Fresh Powder" use case, whilst it is at the third place in "Store Packet Powder" use case. c.2 shows the combination of function chains for "Supply Hot Water" and "Supply (Coffee) Powder" use cases of a coffee machine based on the last function ("Mix Powder") of the function chain of "Supply (Coffee) Powder" use case.

Concatenation of function chains: Function chains can also be connected to each other through concatenation, that is, a linear function chain can be formed by connecting input/output functions of function chains to each other. There are two distinct cases in the concatenation of two function chains: i) there may be no overlap when there is no reusage of functions between function chains of UCs, ii) there is partial overlap if there is a reusage of at least one function between function chains of UCs. a.1 and b.1 in Figure 4 show schematic representations for (i) and (ii), respectively based on UC1 and UC2 in Figure 2a.

Figure 4a.1,b.1 illustrate two kinds of concatenation of two function chains; a.1 shows that the output of a chain can be linked to the input of another chain, and vice versa; a.2 exemplifies this based on two use cases of a bread toaster: "Load Bread" and "Remove Bread." Function chains can be connected to each other based on the reused function. An example of this is shown in b.2 on the basis of "Store Fresh Powder" and "Extract Powder" use cases of a coffee machine. The "Store Powder" function from the "Store Fresh Powder" use case serves as both an output function and an input function when connected to "Export Powder" in the "Extract Powder" use case. This function reuse pattern demonstrates how a well-designed function model can optimize system integration by maintaining consistent behaviour across different use cases, ensuring traceability between related operations, and reducing model complexity through strategic function sharing.

Branching flow heuristic

The fulfilment of some functions on a function chain cannot always be achieved without loss, and this loss(es) need to be mapped and accounted for, because the loss can cause problems elsewhere in the system or at the interface with other systems. Addressing this loss

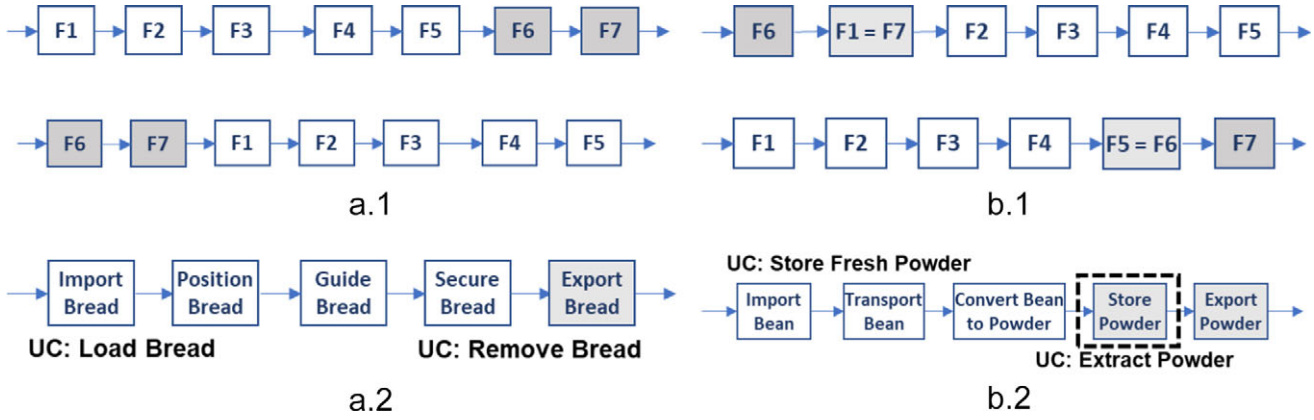

Figure 4. Schematic representation of concatenation of two function chains in two distinct cases (a-b.1) and respective examples based on use cases of a bread toaster (a.2) and coffee machine (b.2)

https://doi.org/10.1017/S0890060425100176 Published online by Cambridge University Press

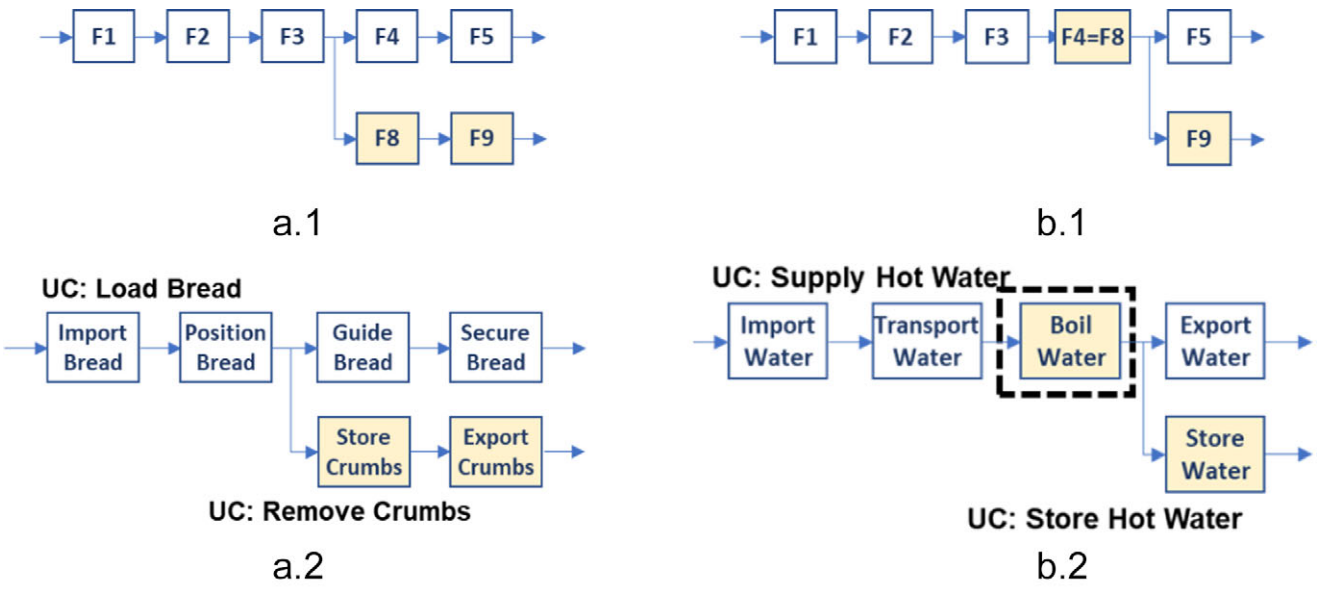

Figure 5. Schematic representation of branching flow for two distinct cases (a-b.1) and respective examples based on use cases of a bread toaster (a.2) and coffee machine (b.2).

can be associated with a use case of the same system. Similarly, unlike connecting and concatenating function chains, the linear flow of function chain of a use case may need to be branched out and linked with the function chain of another use case of the same system in order to add a new functionality to the system. Branching flow heuristic enables the representation of these two distinct cases by creating parallel function chains for more than one use case through bifurcating a flow(s) from a function chain of a use case. Figure 5a.1,b.1 represent this schematically based on UC1 and UC3 shown in Figure 2a.

Figure 5a.2 represents a branching flow from the function chain of "Load Bread" use case of a bread toaster and the allocation of this flow as an input to the function chain of its "Remove Crumbs" use case. Figure 5b.2 shows a branching flow from the function chain of "Supply Hot Water" use case of a coffee machine to its "Store Hot Water" use case based on the reuse of "Boil Water" function, shown in a dotted frame.

The main flow, the connecting flow, and the branching flow heuristics can be used incrementally to compose a complete function architecture of a system detailing the way of achieving its use cases.

Abstraction of functional architecture developed through flow heuristics

The allocation of requirements (including functional requirements derived from system functions and operational activities) to physical architecture (PA) blocks (e.g. Figure 1c) is common practice in MBSE, enabling traceability between system behaviour and physical implementation. More than one requirement is commonly allocated to the same PA block. The practitioners tend to conduct the design analysis based on the PA only without developing a structured function architecture (FA). The system decomposition is carried out based on the decomposition of each PA block to look at how all requirements can be delivered by sub-PA blocks. While this provides a strong traceability, this contradicts the basic principle that the FA should always be considered first for a better PD practice, as stated by numerous scholars (e.g. Lamm and Weilkiens, 2010; Dori, 2016; Zhang et al., 2022). The developed MBSE model will simply present how a selected PA delivers requirements. Not basing the development of the system model on a function architecture will create problems related to the integration between systems modeling tools and other tools in order to carry out some PD activities, such as using the generated MBSE model for simulation, FMEA, and safety analysis.

As in the MBSE practice, it is not straightforward to decompose a function model through different levels of abstraction in DTM research. The DSM approach can be used for developing modules, but this requires the consideration of PA first. Furthermore, significant effort is needed for the analysis of a wide range of possible design choices. The module heuristics of Stone et al. (2000) map functional structures to physical structures; however, the relation between the development of functional structures and the use of module heuristics is not well established. The proposed flow heuristics in this paper provides a basis for a much structured and simpler approach in the abstraction of functional architecture of a system across multiple levels supporting the progressive development of the physical architecture by following a similar logic of Yildirim et al. (2017) and Stone et al. (2000). Figure 6 shows abstraction of functions that are part of a function chain into a higher-level function for function chains developed through main flow heuristic (a), connecting flow heuristic (b-c), and branching flow heuristic (d) based on respective examples from the bread toaster and the coffee machine case studies presented in the previous section.

The series of sequence of operations on the main flow can be abstracted either in a single function block or in multiple function blocks (in particular for longer sequences) representing Functional Architecture (FA) at a higher level. Figure 6a illustrates this for the function chain of "load bread" use case of the bread toaster. Higher-level functions are deliberately named to directly correspond with their respective use cases to ensure clear traceability between requirements and implementation. While the names are identical, they serve distinct roles: the use case describes the user requirement, while the high-level function provides the actual implementation of that requirement. The same principle applies to function chains

https://doi.org/10.1017/S0890060425100176 Published online by Cambridge University Press

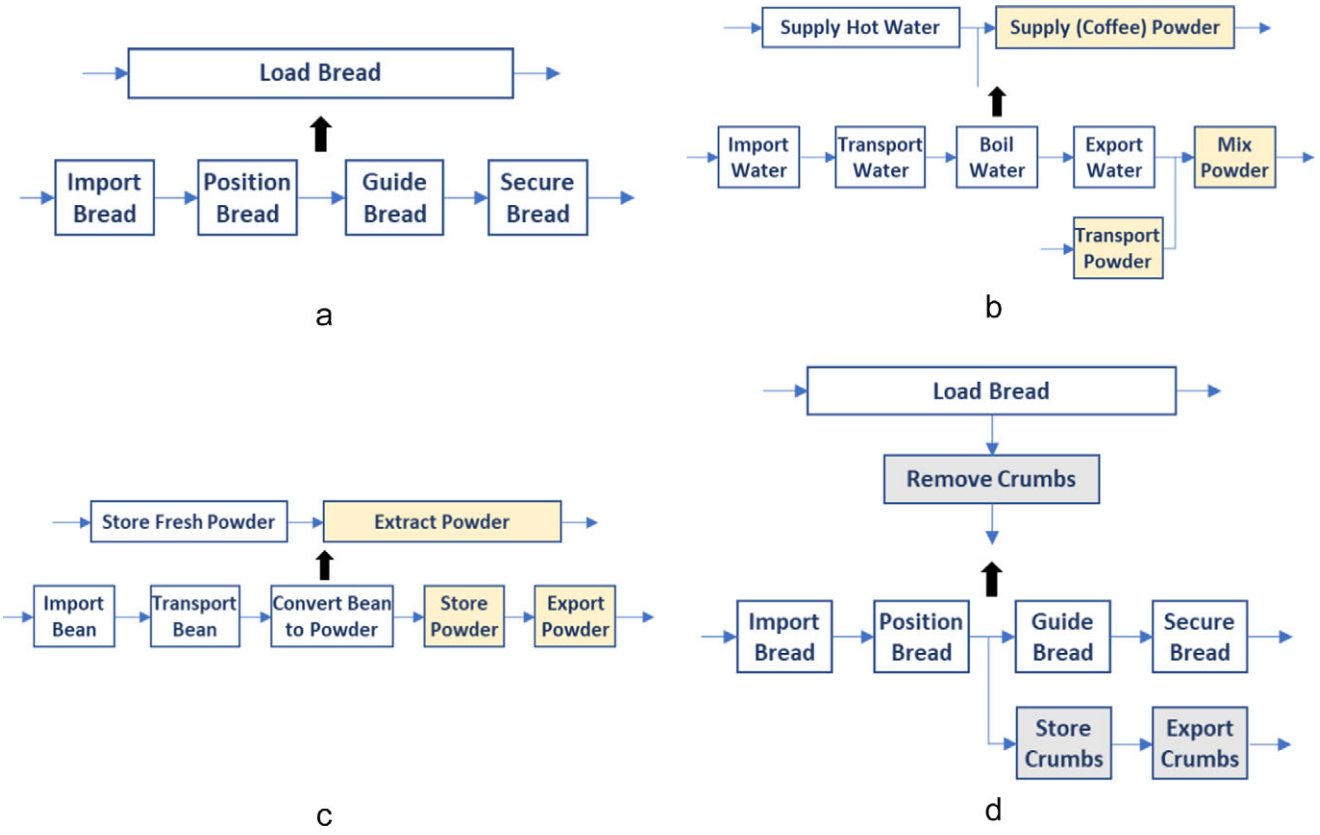

Figure 6. abstraction of function chains – developed through main flow heuristic (a), connecting flow heuristic (b-c), and branching flow heuristic (d) – into higher-level functions based on respective examples from the bread toaster (a-d) and the coffee machine (b-c).

developed through connecting and branching flow heuristics. Function chains that connect, concatenate, or branch from one use case to another within the same system can be abstracted into either a single function block or multiple function blocks, representing higher-level functional architecture. Figure 6b,c represents the abstraction of function chains developed through connection and concatenation into higher-level functions, respectively, based on function chains of two use cases of the coffee maker. Figure 6d introduces an example for the abstraction of function chains developed through branching flow heuristics based on two use cases of the bread toaster.

Each function block representing a high-level function of FA can be allocated to a component/subsystem block denoting a main element of the associated Physical Architecture (PA), see Figure 1b,c. In practice, function blocks of function chains developed through connecting and branching flow heuristics can be associated with the PA blocks of function chains developed through the main flow heuristic, depending on the architecture choice. However, it is recommended that both functional and physical coupling elements are kept as separate functions and physical elements where possible in line with the principles of Axiomatic Design (Suh, 1998). Each high-level PA block can be decomposed into sub-PA blocks addressing the fulfillment of functions in the developed FA.

Integrated MBSE function modeling methodology

Figure 7 illustrates the proposed MBSE function modeling methodology based on the flow heuristics, along with the methodological steps indicated on the diagram.

Step 1 – Use Case (UC) Analysis: The first step is to carry out a use case analysis of the system by developing its use case diagram.

Step 2 – Development of Functional Architecture (FA) using Flow Heuristics: The second step starts by developing an independent function model (as an FA) for each UC based on the main flow heuristic (sub-step 2.1). This follows the aggregation of the developed FAs into a unified FA by using the connecting and branching flow heuristics incrementally (sub-step 2.2). This requires focusing on how to join function chains by using the heuristics introduced above for relevant cases based on the complementarity of function blocks with their ports. For example, function chains of two use cases can be joined together through connection (e.g. b.1 of Figure 3 in the case of a reused function between function chains) and concatenation (e.g. a.1 of Figure 4).

Step 3 – Synthesis/Abstraction of Function Architectures: Once a unified FA has been developed by using the connecting and branching flow heuristics, this FA is abstracted into a FA consisting of higher-level functions by defining FA modules based on the principles outlined in "*Abstraction of functional architecture developed through flow heuristics*" section.

Step 4 – Allocation of FA to Physical Architecture (PA): The last step of the proposed framework starts with allocating a PA to the FA developed in the previous step by representing a PA based on the complementarity with function blocks of the FA, along with their ports (see Figure 1b,c). More than one function block of the FA can be allocated to a PA block depending on the choice of technology/architecture (sub-step 4.1). The allocation editor of SC enables the practitioner to map the FA blocks to the PA blocks in a matrix presentation (sub-step 4.2). This provides a basis for the

https://doi.org/10.1017/S0890060425100176 Published online by Cambridge University Press

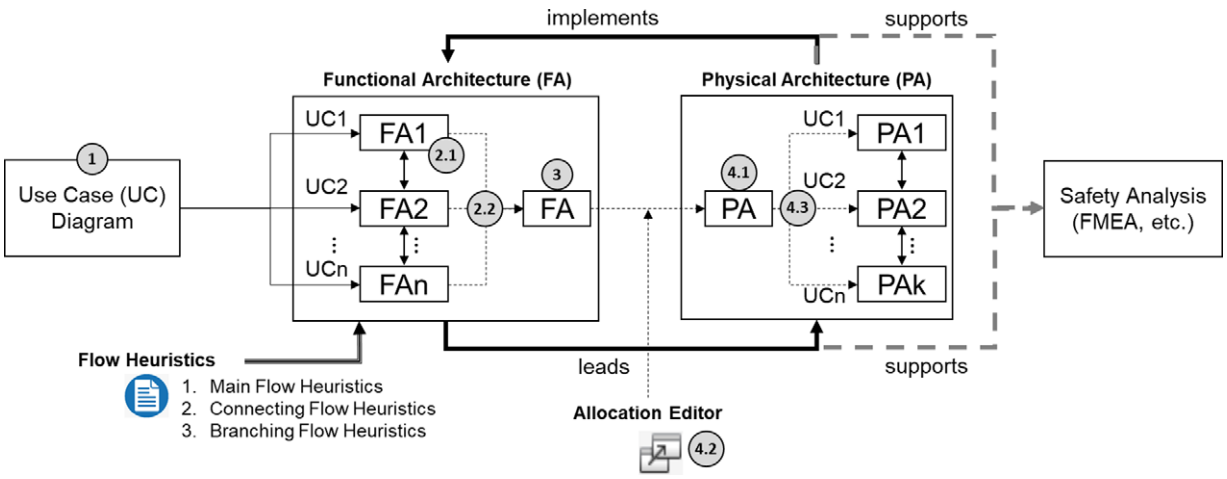

Figure 7. MBSE function modeling framework, along with the methodological steps.

decomposition of the developed PA into sub-PAs representing the inner structure of the PA blocks (sub-step 4.3) with reference to function chains developed through the flow heuristics in Step 2.

Electric bicycle case study

In this section, we validate the proposed methodology presented in Figure 7 through its practical application to an electric bicycle (e-bike) case study using the MathWorks System Composer (SC).

Step 1 – Use Case (UC) Analysis: The set of e-bike use cases considered in this case study for the purpose of representing the application of the methodology is shown in Figure 8 as a use case diagram. From the user perspective, the rider is associated with five main use cases: Pedal e-bike (UC1) where the e-bike is used like a normal push-bike; Pedal e-bike with power assistance (UC2) where

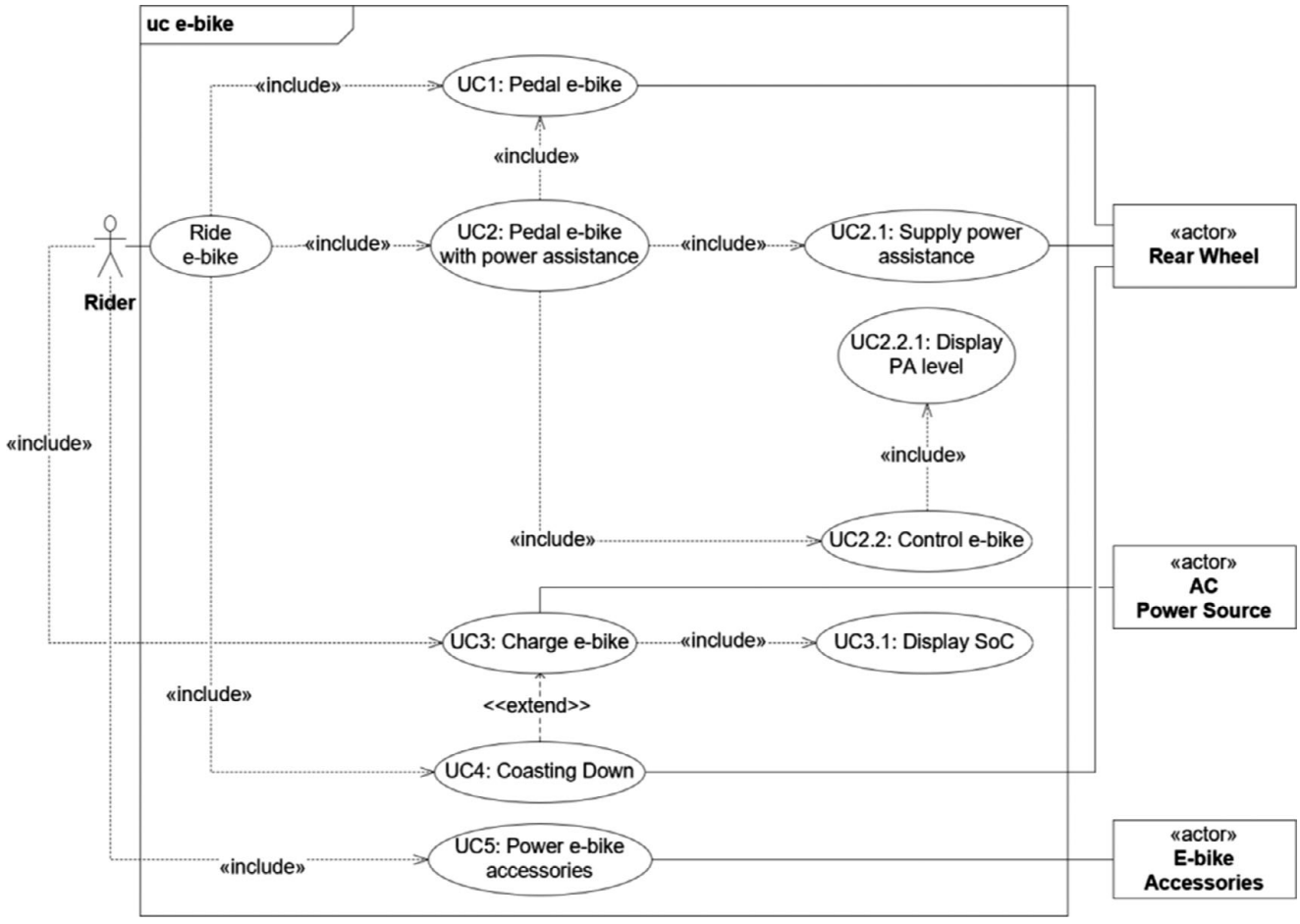

Figure 8. UC diagram of an e-bike, including the set of use cases considered in this paper.

https://doi.org/10.1017/S0890060425100176 Published online by Cambridge University Press

the e-bike provides the rider with power assistance if requested; Charge e-bike (UC3); Coasting Down (UC4) which refers to the act of riding a bike downhill without actively pedaling and Power e-bike accessories (UC5). UC2 includes two sub-use cases: supply power assistance (UC2.1) associated with the supply of power assist for propulsion by the e-bike system and Control e-bike (UC2.2), as the rider controls of the feature where the rider requests power assistance (PA) for additional propulsion. UC2.2 and UC3 include sub-use cases Display PA level (UC2.2.1) and Display State of Charge-SoC (UC3.1), respectively.

Use cases are related to one another by inclusion and extension relationships, as shown in Figure 8. An inclusion relationship means that a base use case can incorporate the functionality of another use case, known as the included use case. Whenever the base use case is executed, the included use case functionality is also invariably executed (Friedenthal et al., 2012). For example, Pedal e-bike (UC1) takes place whenever Pedal e-bike with power assistance (UC2) is executed. Similarly, Display PA level (UC2.2.1) is always performed along with the Control e-bike (UC2.2). An extension relationship represents a functionality that is not included in the standard base use case functionality. In contrast to an included use case, the base use case is not reliant on the extended use case. However, an extended use case might be influenced by the events occurring in its base use case (Friedenthal et al., 2012). For example, charge e-bike (UC3) is shown as an extension to Coasting Down (UC4). This means that the e-bike does not always charge its battery (UC3) when it is coasted down (UC4). The relevant parameter of the bike (e.g, velocity) and the environment (e.g., slope) affects the charging operation.

Step 2 – Development of FA using flow heuristics: This step starts with the development of an FA for each UC of the e-bike via the main flow heuristic, showing the "way of achievement" of these UCs in terms of the decomposition of activities as function flows. Figure 9 represents FA for each UC of the e-bike shown in Figure 8.

Figure 9 shows the FA of each UC in terms of SC function blocks introduced in Figure 1b where a function is articulated in verb-noun format and represented with its input ports and output ports. This follows the development of a unified FA using the connecting and branching flow heuristics. Figure 10 represents the implementation of the connection flow heuristics on the "connection" of FAs of relevant e-bike UCs.

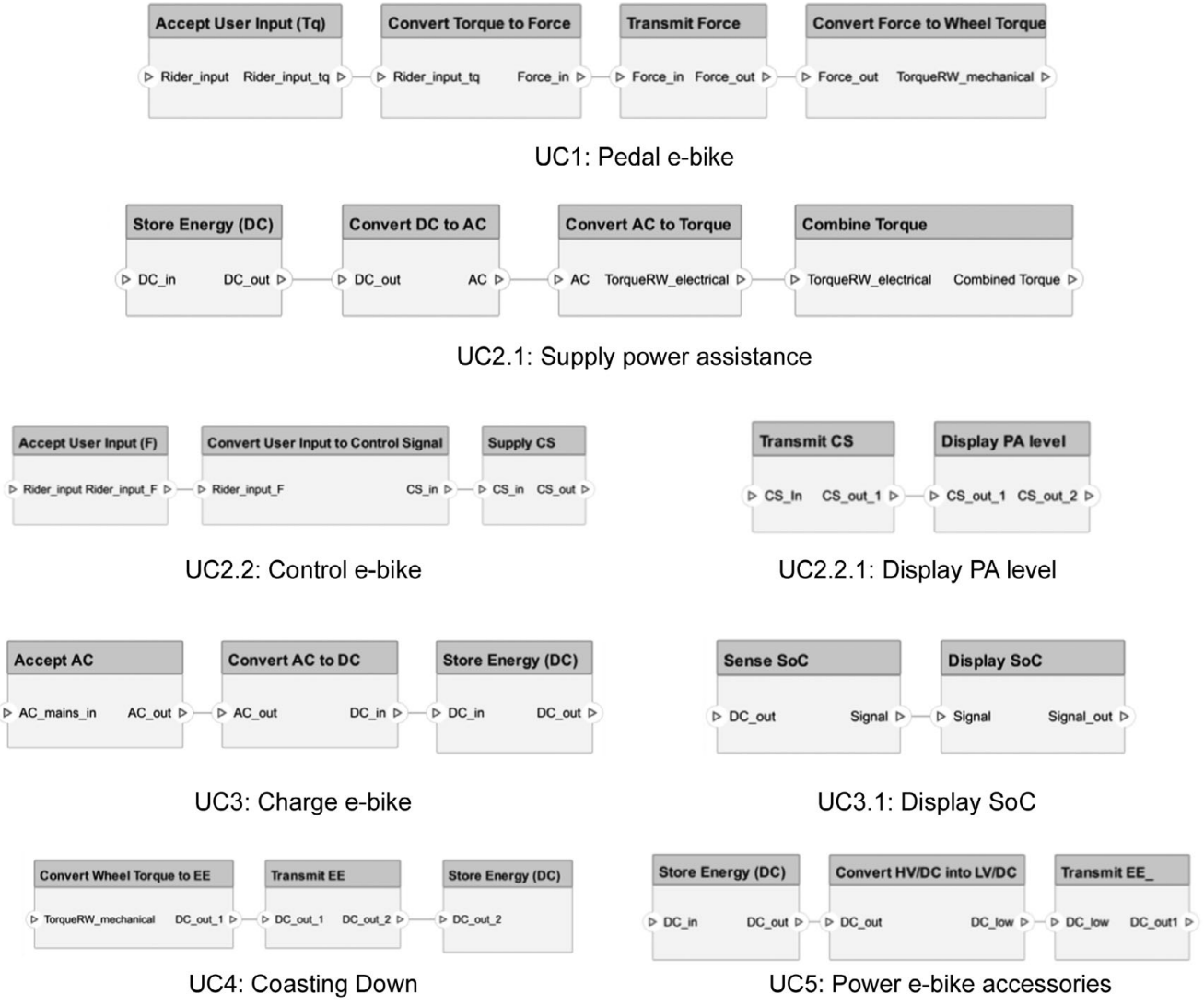

Figure 9. FA for each UC of the e-bike.

https://doi.org/10.1017/S0890060425100176 Published online by Cambridge University Press

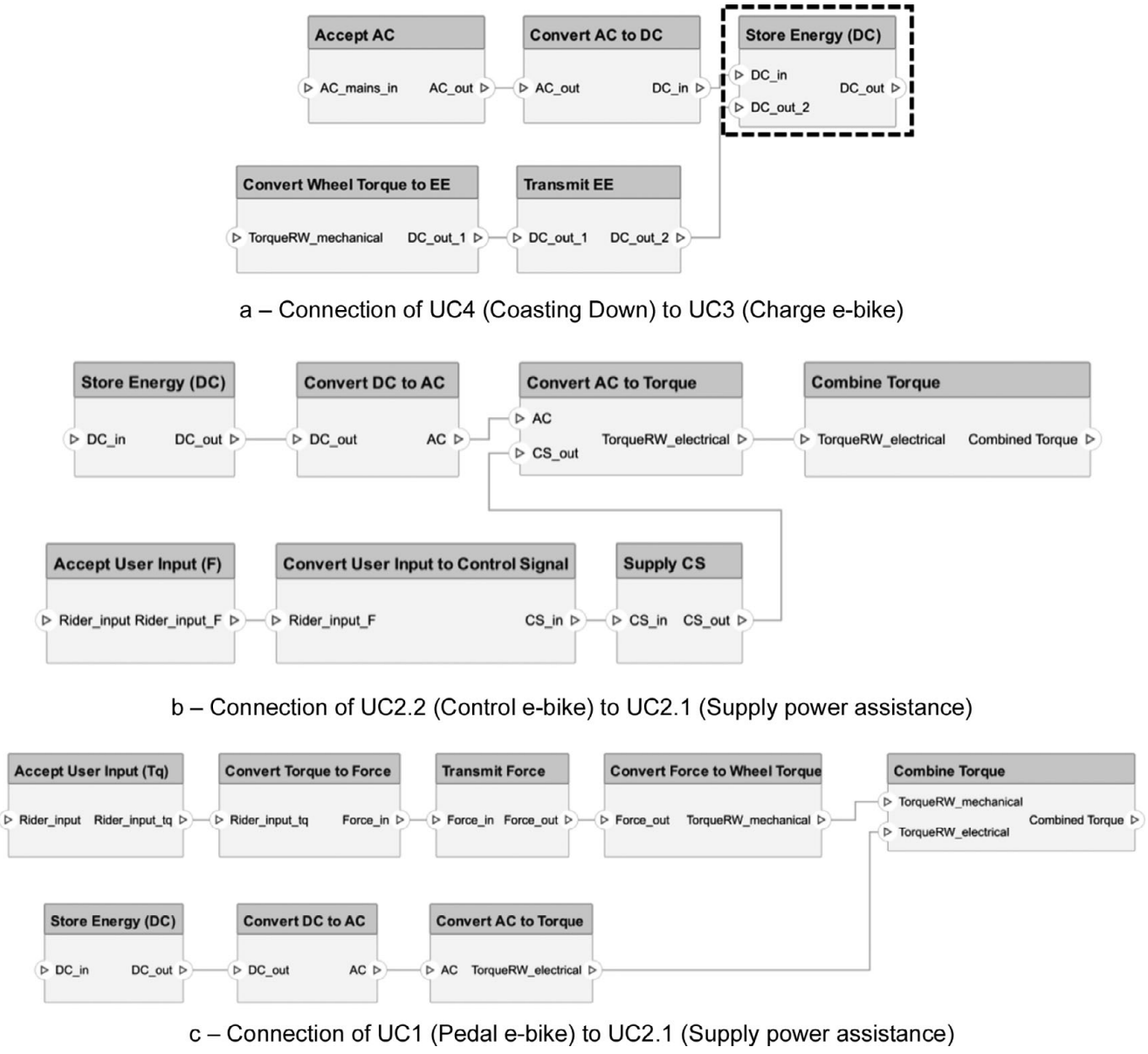

Figure 10. Connection flow heuristics on "connection" of FAs of relevant e-bike UCs.

The analysis of the states involved in the definition of function blocks facilitates the implementation of connecting and branching flow heuristics. For example, the final function blocks of UC3 and UC4 refer to the same energy flow, shown in a dotted frame in Figure 10a. This provides a basis for the connection of FAs of these UCs, around the same function block "Store Energy (DC)". It should be noted that this connection is established based on the value of attributes on the ports of the function blocks, to ensure that the combined FA is mathematically correct. The function block "Store Energy (DC)" generates the output "DC_out" based on the inputs "DC_in" and "DC_out_2" from UC3 and UC4 respectively. Similarly, the function block "Convert AC to Torque" of FA of UC2.1 incorporates the output of the function block "Supply Control Signal (CS)" of UC2.2 in Figure 10b. The function block "Combine Torque" is associated with FA of UC 2.1 and this function block combines the flow from the function block ""Convert AC to Torque" of the same FA with the flow from the function block "Convert Force to Wheel Torque" of FA of UC1 in Figure 10c. FAs of UCs can also be connected to each other via "concatenation" to form a linear function chain as discussed above. Figure 11 illustrates the implementation of the connection flow heuristics on "concatenation" of FAs of relevant e-bike UCs.

Use cases are differentiated from each other with dotted frames in Figure 11. The output of FA of UC3 is the input of FA of UC3.1, leading to the development of a linear function chain in Figure 11a. FA of UC4 is connected to FA of UC2.1 through the reused function block "Store Energy (DC)", as shown in Figure 11b. Connection flow heuristics in Figures 10 and 11 represent how FAs of relevant e-bike use cases are "connected" to each other. Figure 12 shows how FA of an e-bike use case branches out of the FA of its another use cases.

The output of "Convert User Input to Control Signal" function block of UC2.2. is shown as the input of the FA of UC2.2.1 in Figure 12a. FA of UC5 branches out of FA of UC2.1 based on the reused function block "Store Energy (DC)", highlighted with a dotted frame in Figure 12b. Figure 10–12 represent how to connect

https://doi.org/10.1017/S0890060425100176 Published online by Cambridge University Press

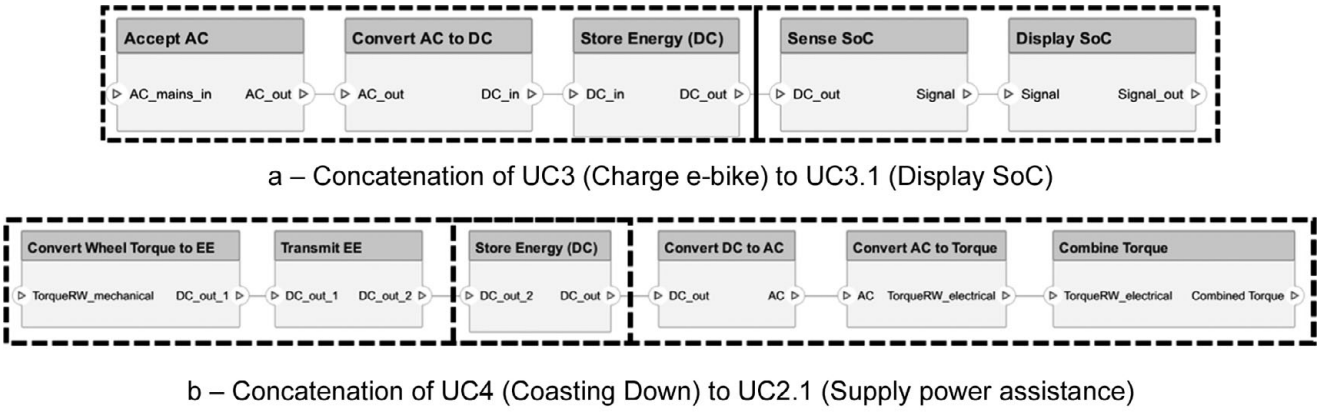

Figure 11. Connection flow heuristics on "concatenation" of FAs of relevant e-bike UCs.

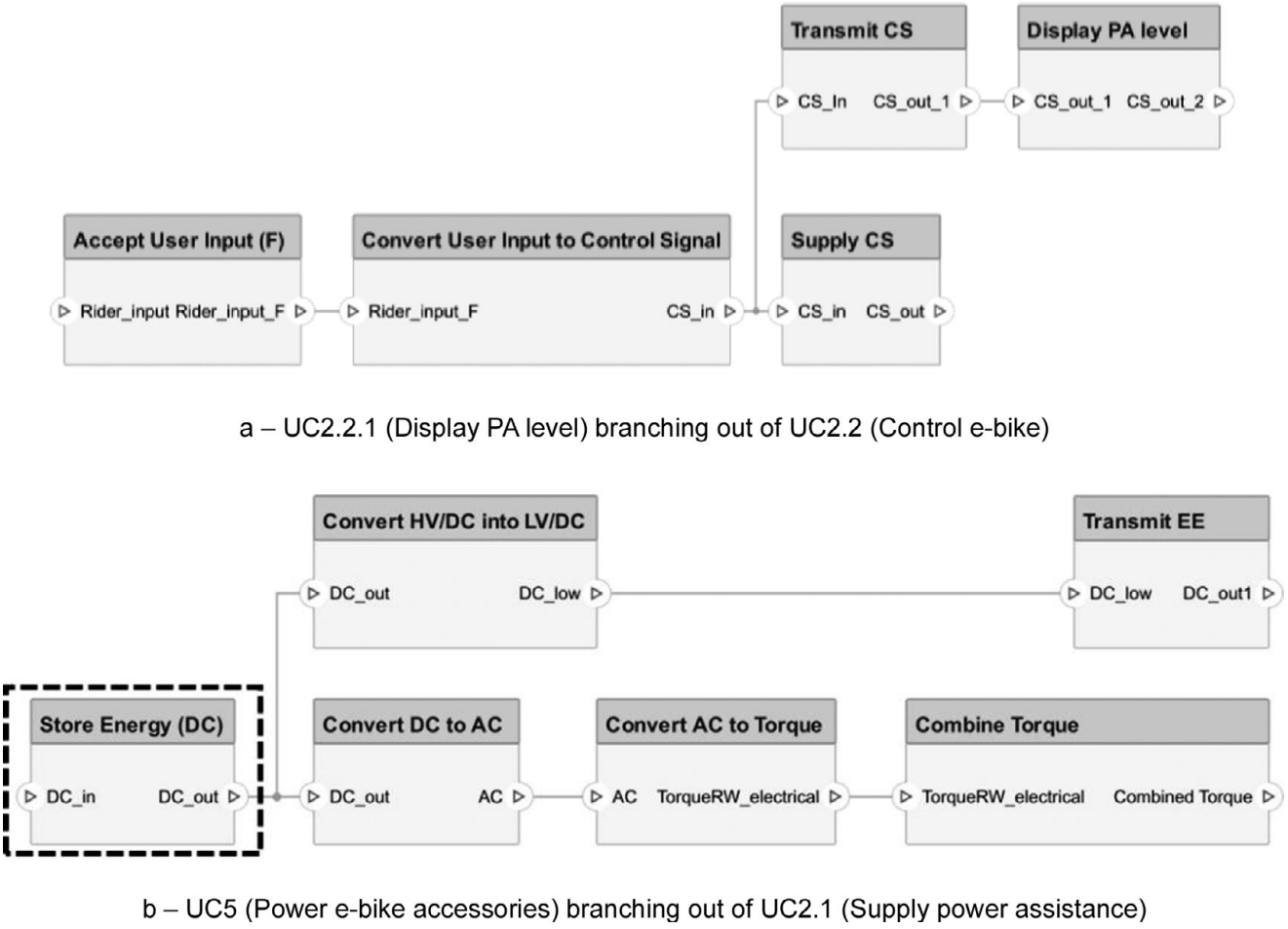

Figure 12. Branching flow heuristics on FAs of relevant e-bike UCs.

relevant FAs of e-bike UCs, shown via the main flow heuristic in Figure 9, through connecting and branching flow heuristics. Finally, in order to deliver all e-bike UCs through a single FA, we need to combine all FAs introduced in Figure 10–12 based on the complementarity of function blocks with their ports, as shown in Figure 13. The input and the output function blocks of the system are aligned to the left and the right, respectively. The consideration of the connecting and branching flow heuristics to perform the connecting and the branching of these FAs in Figure 10–12 facilitates this step.

Step 3 – Synthesis/abstraction of function architectures: The unified e-bike FA in Figure 13 introduces a detailed analysis of the flows from FAs of its use cases, resulting in a complex function structure. Higher-level function blocks can be defined based on this FA by abstracting relevant function chains as higher-level functions, based on the principles outlined in the previous section.

https://doi.org/10.1017/S0890060425100176 Published online by Cambridge University Press

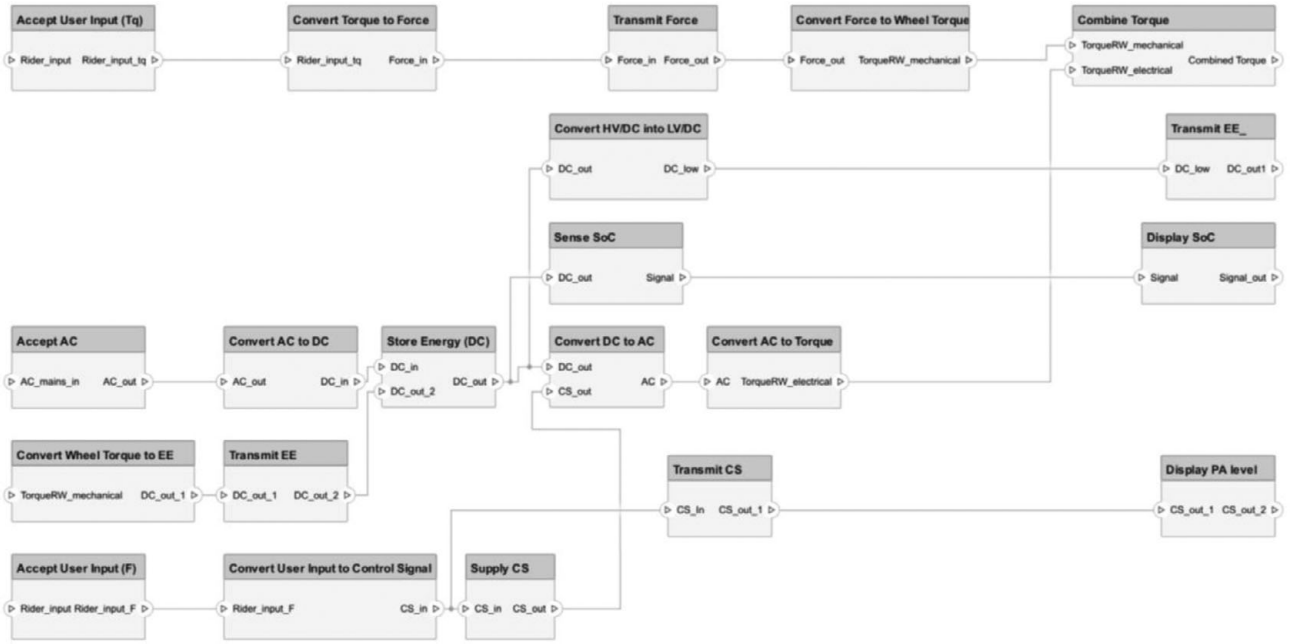

Figure 13. A unified ebike FA delivering its UCs in Figure 8.

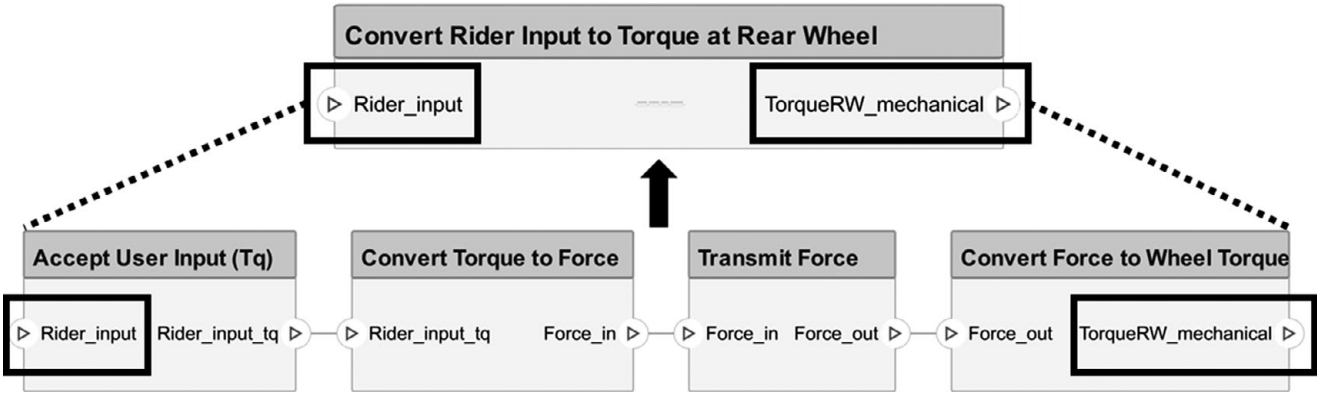

Figure 14. Abstraction of a function chain as a higher-level function block.

Function blocks that are part of a linear function chain can be conveniently abstracted into a higher-level function block, as shown in Figure 14.

Figure 14 illustrates the allocation of function blocks of UC1 (Pedal e-bike) to "Convert Rider Input to Torque at Rear Wheel" function block by taking the input of the first function block as the input and the output of the last function block as the output, as shown in the Figure. Figure 15 represents the abstraction of the unified e-bike FA in Figure 13 as higher-level function blocks.

The function block Store Energy (DC) is reused among many UCs, as shown in Figures 10a, 11b and 12b. This is represented as a separate entity in Figure 15 for this reason, and linked to higher-level function blocks through the flows.

Step 4 – Allocation of functional architecture (FA) to physical architecture (PA): A PA can be allocated to the e-bike FA, abstracted as higher-level function blocks in Figure 15. While more than one function block of an FA can be allocated to a PA block depending on the choice of technology/architecture, it is recommended to allocate one function block to one physical block where possible, coherent with the principles of Axiomatic Design. More than one function block in an FA can be abstracted as a higher-level function block if necessary, and then a physical block can be allocated to this function block by following the same principles. Figure 16 illustrates a PA based on the allocation of a physical block to each function block in Figure 15.

The allocation editor of SC enables to establishment of a directed relationship between FA elements and PA elements (including components and ports). Figure 17 shows an excerpt from the allocation of FA elements of the e-bike shown in Figure 15 to its PA elements shown in Figure 16 using the allocation editor.

Figure 17 shows the allocation of high-level function blocks defined in Figure 15 to conceptually defined physical elements in Figure 16. For example, "Convert Rider Input to Torque at Rear Wheel" high-level function block is allocated to "Pedal Drive System (PDS)" high-level physical block. The allocation editor encompasses the whole FA and PA elements (including components and ports). This enables the allocation of sub-elements of high-level function blocks to sub-elements of high-level physical

https://doi.org/10.1017/S0890060425100176 Published online by Cambridge University Press

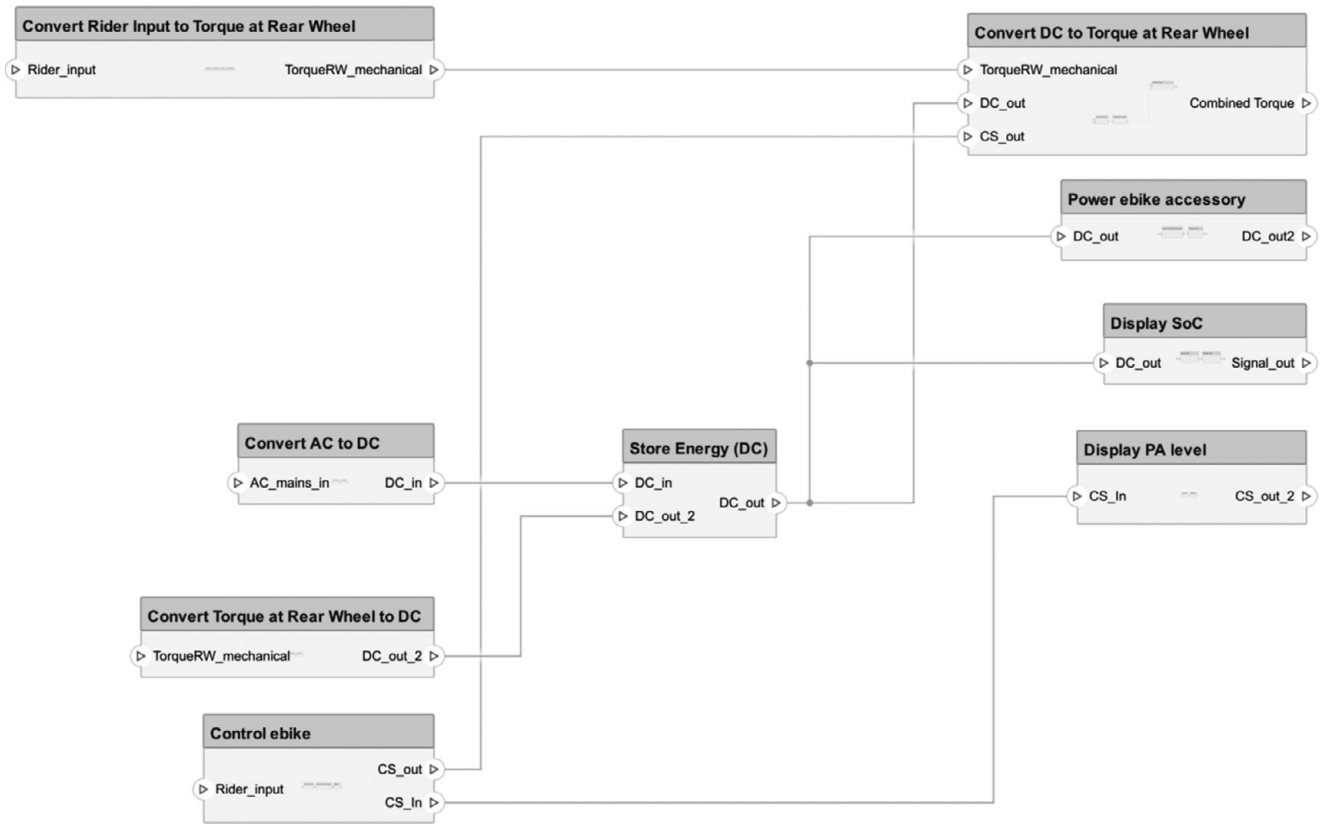

Figure 15. Abstraction of the unified e-bike FA as higher-level function blocks.

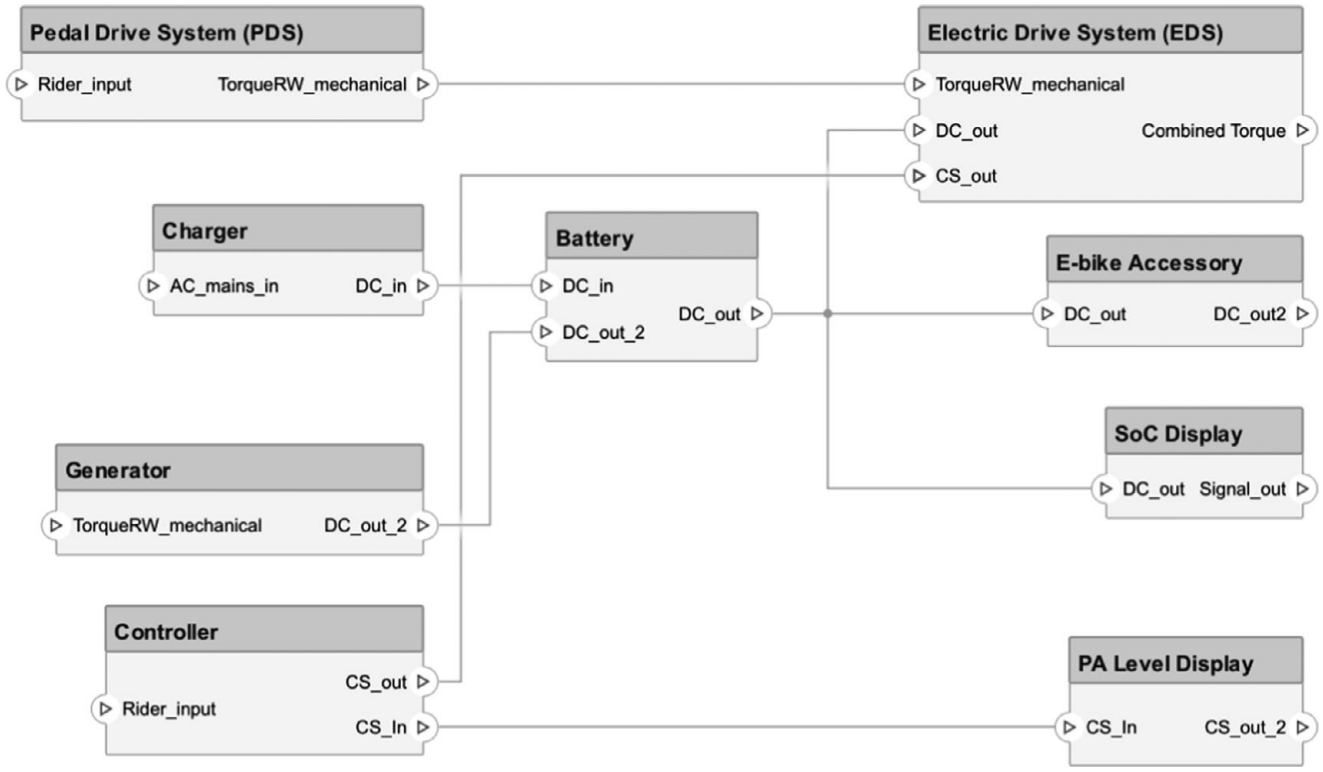

Figure 16. PA for the unified e-bike FA.

https://doi.org/10.1017/S0890060425100176 Published online by Cambridge University Press

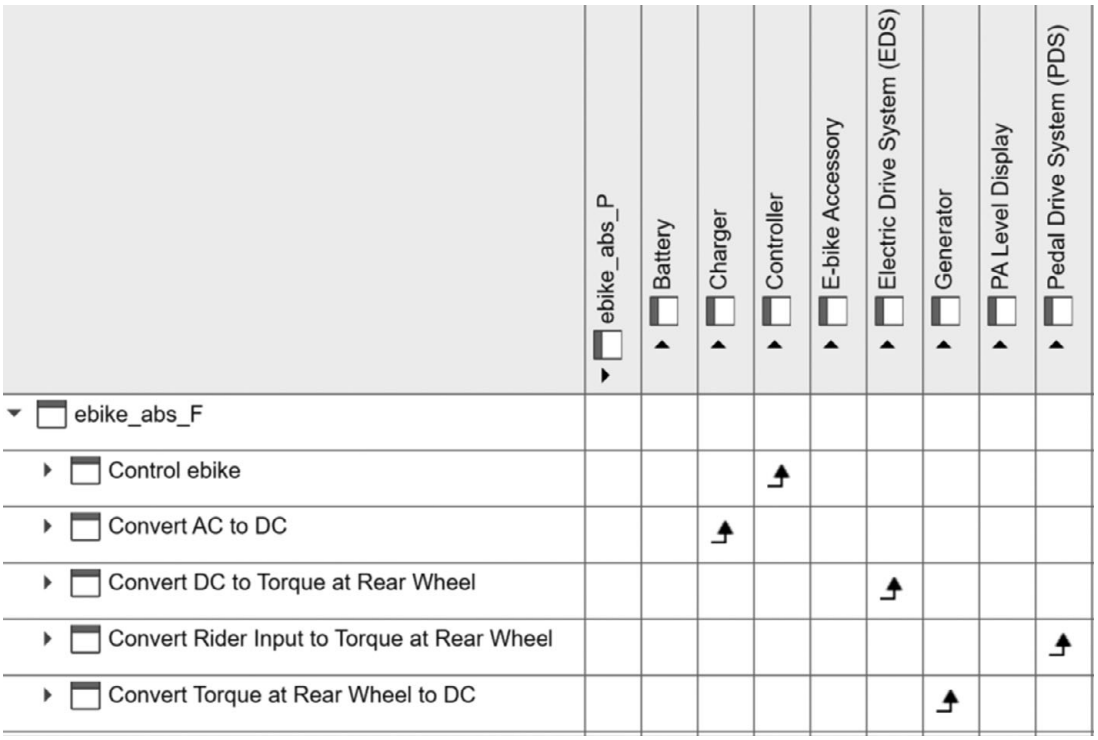

Figure 17. Allocation of e-bike FA elements to its PA elements.

blocks by following the same methodology described above. This provides a basis for the decomposition of the developed PA into sub-PAs representing the inner structure of the PA blocks with reference to function chains developed through the flow heuristics. For example, the PDS, as a high-level PA element, can be decomposed into sub-PA elements based on the FA of UC 1, as shown in Figure 18.

The allocation editor also promotes the consideration of different architecture solutions to the subfunction blocks (preserved as separate function blocks) associated with combinations of flows by creating scenarios where each scenario contains a set of allocations between the FA and PA models. For example, the PDS is named based on the architecture decision taken for "Accept User Input": "Pedal". Different architecture solutions can be considered, for

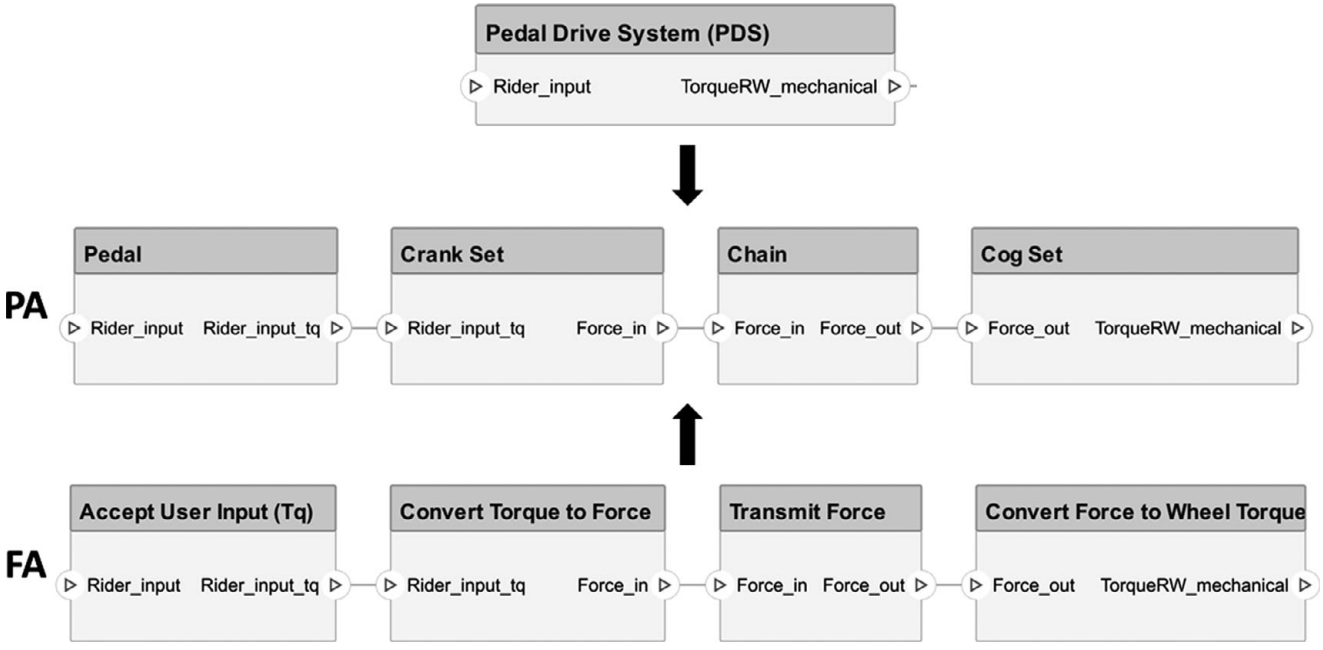

Figure 18. Allocation of FA sub-elements to PA sub-elements.

https://doi.org/10.1017/S0890060425100176 Published online by Cambridge University Press

example, by abstracting "Accept User Input" and "Convert Torque to Force" as a higher-level function block and allocating a PA block to this function block as part of a scenario.

Discussion

The implementation of the proposed methodology in system composer (SC) addresses key challenges in both DTM and MBSE approaches to function modeling. While DTM approaches like Pahl et al. (2007) and Stone et al. (2000) provide structured methodologies, they do lack integration with modern MBSE practices. Conversely, MBSE approaches using SysML, while comprehensive, require numerous elements to develop function models (Andersson et al., 2010; Cloutier, 2015) and can lead to ambiguous interpretations due to flexible semantics (Madni and Sievers, 2018). The methodology proposed in this paper bridges this gap by providing a streamlined approach that maintains model clarity while integrating DTM principles into MBSE practice.

The key theoretical contribution in this paper is the systematic methodology for developing integrated Functional Architectures, enabled by flow heuristics within an MBSE environment. This addresses a fundamental limitation in existing MBSE approaches where function modeling lacks systematic guidance for integrating functions across multiple use cases into coherent system architectures – an integration capability that traditional DTM approaches like Pahl et al. (2007) do not provide within MBSE contexts, and that current MBSE practices using SysML struggle to achieve without ambiguous interpretations (Madni and Sievers, 2018). This also addresses a more general problem in functional modeling where analysis often results in isolated function chains (Stone et al., 2000), which our connecting and branching flow heuristics specifically overcome by providing structured methods for cross-use case function integration that Stone et al.'s module-focused approach cannot systematically handle. The flow heuristics provide prescriptive MBSE practitioner guidelines for developing function models, starting with the main flow heuristic used to capture the main purpose of the new feature or system being developed within each of the relevant use cases under consideration. The branching and connecting flow heuristics can then be used to both complete the function model for a use case, where connecting resource flows or bifurcating function flows is necessary, and to join function models developed from the perspective of independent use cases, to ensure the consistency and integrity of the whole system architecture model. This provides a structured approach to combine individual functional models developed for different use cases into a coherent FA, incorporating flow considerations during the initial development process rather than as a later step during architecture decisions.

The methodology particularly lends itself to systems with multiple operation modes and clear flow-based interactions, as demonstrated through our case studies. The bread toaster and coffee machine examples validated the basic application of flow heuristics, while the e-bike case study demonstrated its effectiveness in handling more complex scenarios with multiple interacting subsystems. The approach is particularly suitable for mechatronic systems where energy, material, and information flows need to be clearly tracked and managed across different operational states. However, the methodology may have limitations for systems where flows are not the primary consideration or for systems with highly abstract or cognitive functions.

The implementation in SC enables precise specification of functional requirements through defined inputs/outputs of function blocks and supports sophisticated use case decomposition and FA aggregation techniques. The e-bike example demonstrates how the approach manages multiple interacting subsystems through structured integration of function models. The methodology's support for nested decomposition of FA and PA into subfunctions and subsystems, while maintaining clear traceability of requirements, addresses a key limitation in current practice where such decomposition often leads to fragmented models and loss of consistency between different functional views.

Conclusions

This paper has introduced flow heuristics and a systematic methodology for developing an integrated functional architecture (FA) in conjunction with use-case analysis, implemented within the MathWorks System Composer environment. The proposed methodology integrates well-founded approaches from Design Theory and Methodology function modeling practice into the Model-Based Systems Engineering process flow.

The methodology's effectiveness has been demonstrated through three case studies of increasing complexity, culminating in an e-bike system model. The approach underpins the mathematical correctness of the developed FA through complete representation of function blocks with defined inputs and outputs, supporting model execution in simulation environments. This can also support the development of executable system models by mapping requirements to physical architecture through FA, improving model use and reuse compared to direct requirement-to-PA mapping.

Future work will focus on multiple directions. Further validation of the proposed methodology is needed through analysis of existing functional models in the literature. We will explore whether additional heuristics, such as the conditional fork node heuristic, are needed for a complete set of flow heuristics. Furthermore, an investigation is required to understand how architecture choices at lower resolution levels affect FA invariance and how to effectively implement conditional FAs within complete system models. The primary focus will be on demonstrating the integration of the proposed MBSE methodology with simulation modeling for robustness-focused verification and validation. This will include leveraging the e-bike model in System Composer for simulating system dynamic behaviour with real-world operating conditions, supporting virtual testing across different system levels. Additionally, we can explore the implementation of flow heuristics in other leading MBSE support software solutions, such as PTC's Windchill Modeler and Dassault Systèmes' Cameo, to enhance the methodology's broader applicability and adoption across different platforms.

Acknowledgements. The research presented in this paper was partly supported by the Doctoral Research Start-up Fund of Hubei University of Automotive Technology, grant number BK202204. Their support is gratefully acknowledged.

References

Albers A and Zingel C (2013) Challenges of model-based systems engineering: A study towards unified term understanding and the state of usage of SysML. In: *Proceedings of Smart Product Engineering: The 23th CIRP Design Conference, Bochum*, pp. 83–92. https://doi.org/10.1007/978-3-642-30817-8_9

Andersson H, Herzog E, Johansson G, Johansson O. (2010) Experience from introducing unified modeling language/systems modeling language at Saab aerosystems. *Systems Engineering* **13** (4), 369–380.

https://doi.org/10.1017/S0890060425100176 Published online by Cambridge University Press

Bickford J, **Van Bossuyt DL**, **Beery P**, **Pollman A.** (2020) Operationalizing digital twins through model-based systems engineering methods. *Systems Engineering* **23**, 724–750. https://doi.org/10.1002/sys.21559

Biggs G, **Juknevicius T**, **Armonas A and Post K** (2018) Integrating safety and reliability analysis into MBSE: Overview of the new proposed OMG standard. *INCOSE International Symposium* **28**, 1322–1336. https://doi.org/10.1002/j.2334-5837.2018.00551.x.

Borky JM and Bradley TH (2019) *Effective Model-Based Systems Engineering*. Switzerland: Springer

Brusa E, **Calà A and Ferretto D** (2018) *Systems Engineering and Its Application to Industrial Product Development*. Switzerland: Springer. https://doi.org/10.1007/978-3-319-71837-8

Buede DM (2009) *The Engineering Design of Systems: Models and Methods*, 2nd Edn. Wiley Series in Systems Engineering and Management. Hoboken: John Wiley & Sons. https://doi.org/10.1002/9780470413791.

Cameron B (2018) *MBSE: The Next Revolution of Systems Engineering*. MIT, presentation.

Campean F, **Yildirim U and Henshall E** (2018) Synthesis of functional models from use cases using the system state flow diagram: A nested systems approach. In *Paper Presented at the 15th International Design Conference, Dubrovnik, Croatia 21–24 May 2018* pp. 2833–2844. https://doi.org/10.21278/idc.2018.0543.

Campo KX, **Teper T**, **Eaton CE**, **Shipman AM**, **Bhatia G and Mesmer B** (2023) Model-based systems engineering: Evaluating perceived value, metrics, and evidence through literature. *Systems Engineering*. **26**, 104–129. https://doi.org/10.1002/sys.21644.

Chami M and Bruel J-M (2018) A survey on MBSE adoption challenges. In *INCOSE EMEA Sector Systems Engineering Conference (INCOSE EMEASEC 2018)*, 5 November 2018- – 7 November 2018 (Berlin, Germany).

Chandrasekaran B and Josephson JR (2000) Function in device representation. *Engineering with Computers* **16**, 162–177. https://doi.org/10.1007/s003660070003.

Cloutier R (2015) Current Modeling trends in systems engineering. *Insight* **18**, 10–13. https://doi.org/10.1002/inst.12013.

Davey C (2022) Ford's connected-agile, model based systems engineering and simulation journey….so far. In *32nd Annual INCOSE International Symposium, Detroit, MI, USA, June 25–30*.

Dori D (2016) *Model-Based Systems Engineering with OPM and SysML*. New York: Springer. https://doi.org/10.1007/978-1-4939-3295-5

Drave I, et al. (2020) Modeling mechanical functional architectures in SysML. In *Proceedings of the 23rd ACM/IEEE International Conference MODELS '20*, pp. 79–89. https://doi.org/10.1145/3365438.3410938.

Eisenbart B, **Mandel C**, **Gericke K and Blessing L** (2015) Integrated function modelling: Comparing the IFM framework with SysML. In *Proceedings of the 20th ICED15. Milan, 27–30 July*.

Eisenbart B, **Gericke K**, **Blessing LTM**, et al. (2017) A DSM-based framework for integrated function modelling: Concept, application and evaluation. *Research in Engineering Design* **28**, 25–51. https://doi.org/10.1007/s00163-016-0228-1.

Erden M, **Komoto H**, **Van Beek T**, **D'Amelio V**, **Echavarria E and Tomiyama T** (2008) A review of function modeling: Approaches and applications. *AI EDAM* **22** (2), 147–169. https://doi.org/10.1017/S0890060408000103.

Estefan JA and Weilkiens T (2022) MBSE methodologies. In Madni AM, Augustine N and Sievers M (eds), *Handbook of Model-Based Systems Engineering*. Springer. https://doi.org/10.1007/978-3-030-27486-3_12-1

Ferrogalini M, **Linke T and Schweiger U** (2019) How to boost the extended enterprise approach in engineering using MBSE—A case study from the railway business. In Bonjour E, Krob D, Palladino L and Stephan F (eds.), *Complex Systems Design & Management. CSD&M 2018*. Cham: Springer. https://doi.org/10.1007/978-3-030-04209-7_7

Forlingieri M and Weilkiens T (2022) Two variant modeling methods for MBPLE at Airbus. *INCOSE International Symposium* **32**, 1097–1113. https://doi.org/10.1002/iis2.12984.

Friedenthal S, **Moore A and Steiner R** (2012) *A Practical Guide to SysML: The Systems Modeling Language*, 2nd Edn. Morgan Kaufmann.

Grobshtein Y and Dori D (2011) Generating SysML views from an OPM model: Design and evaluation. *Systems Engineering* **14**, 327–340. https://doi.org/10.1002/sys.20181.

Harel D (1987) Statecharts: A visual formalism for complex systems. *Science of Computer Programming* **8** (3), 231–274. https://doi.org/10.1016/0167-6423(87)90035-9.

Hoff U and Scott D (2019) *Automotive Trends Create New Challenges for Wiring Harness Development*. Siemens Digital Industries Software. Retrieved from https://siemens.com/electrical-systems.

Husung S (2023) *Model Based Systems Engineering for Efficient Reuse of Subsystems During Development*. https://www.tu-ilmenau.de/aktuelles/model-based-system-engineering-for-efficient-reuse-of-sub-systems-during-development (accessed on 11th Dec 2023).

Husung S, **Weber C**, **Mahboob A**, **Kleiner S** (2021) Using model-based systems engineering for need-based and consistent support of the design process. In *Proceedings of the International Conference on Engineering Design (ICED21), Gothenburg, Sweden, 16–20 August 2021*. https://doi.org/10.1017/pds.2021.598

Husung S, **Weber C and Mahboob A** (2022) Model-based system engineering: A new way for function-driven product development. In Krause D and Heyden E (eds.), *Design Methodology for Future Products*. Cham: Springer. https://doi.org/10.1007/978-3-030-78368-6_12

IDE-Institute of Digital Engineering (2021) *Digitalisation Roadmap*. https://roadmap.ide.uk/ (accessed on 17th Jan 2023).

INCOSE (2021) *Systems Engineering Vision 2035*. https://www.incose.org/publications/se-vision-2035 (accessed on 17th Jan 2023)

Jacobs G, **Konrad C**, **Berroth J**, **Zerwas T**, **Höpfner G and Spütz K** (2022) Function-oriented model-based product development. In Krause D and Heyden E (eds.), *Design Methodology for Future Products*. Cham: Springer. https://doi.org/10.1007/978-3-030-78368-6_13

Kitamura Y and Mizoguchi R (2003) Ontology-based description of functional design knowledge and its use in a functional way server. *Expert Systems with Applications* **24** (2), 153–166. https://doi.org/10.1016/S0957-4174(02)00138-0.

Kößler J and Paetzold K (2017) Integration of MBSE into existing development processes – Expectations and challenges. *In: Proceedings of the 21st International Conference on Engineering Design (ICED17), Vol. 3: Product, Services and Systems Design, Vancouver, Canada, 21*-25.08.2017.

Kurfman MA, **Stone RB**, **Van Wie M**, **Wood KL and Otto KN** (2000) Theoretical underpinnings of functional Modeling: Preliminary experimental studies. In *Proceedings of the ASME 2000 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. Volume 4: 12th International Conference on Design Theory and Methodology, Baltimore, Maryland, USA. September 10–13, 2000*. ASME, pp. 203–216. https://doi.org/10.1115/DETC2000/DTM-14563

Lamm JG and Weilkiens T (2010) Funktionale Architekturen in SysML. In Maurer M and Schulze S-O (eds.), *Tag des Systems Engineering*. München: Carl Hanser Verlag, pp. 109–118; English translation by J. Lamm.

Lu J, **Wen Y**, **Liu Q**, **Gürdür D and Törngren M** (2018) MBSE applicability analysis in Chinese industry. *INCOSE International Symposium* **28**, 1037–1051. https://doi.org/10.1002/j.2334-5837.2018.00532.x.

Madni AM and Sievers M (2018) Model-based systems engineering: Motivation, current status and research opportunities. *Systems Engineering* **21**, 172–190. https://doi.org/10.1002/sys.21438.

MathWorks (2023) System Composer, User Guide (R2023a). https://uk.mathworks.com/help/pdf_doc/systemcomposer/index.html.

Matthiesen S, **Ruckpaul A** (2012) New insights on the contact&channel-approach – Modelling of systems with several logical states. In *International Design Conference*. pp. 1019–1028.

Meißner M, **Jacobs G**, **Jagla P and Sprehe J** (2021) Model based systems engineering as enabler for rapid engineering change management. *Procedia CIRP* **100**, 61–66. https://doi.org/10.1016/j.procir.2021.05.010.

Müller JR, **Isaksson O**, **Landahl J**, **Raja V**, **Panarotto M**, **Levandowski C and Raudberget D** (2019) Enhanced function-means modeling supporting design space exploration. In *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 1–15. https://doi.org/10.1017/S0890060419000271

OMG (2019) OMG systems modeling language (OMG SysML™). *Specification Version* **1** (6). https://sysml.org/.res/docs/specs/OMGSysML-v1.6-19-11-01.pdf.

Otto K and Wood K (2001) *Product Design: Techniques in Reverse Engineering and New Product Development*. New Jersey: Prentice Hall

https://doi.org/10.1017/S0890060425100176 Published online by Cambridge University Press

Pahl G, **Beitz W**, **Feldhusen J and Grote KH** (2007) *Engineering Design: A Systematic Approach*, 3rd Edn. London: Springer. https://doi.org/10.1007/978-1-84628-319-2.

Pearce P and Friedenthal S (2013) A practical approach for modelling submarine subsystem architecture in SysML. In *Submarine Institute of Australia Science, Technology & Engineering Conference*. pp. 347–360.

Purohit S and Madni AM (2022) Employing digital twins within MBSE: Preliminary results and findings. In Madni AM, Boehm B, Erwin D, Moghaddam M, Sievers M and Wheaton M (eds.), *Recent Trends and Advances in Model Based Systems Engineering*. Cham: Springer. https://doi.org/10.1007/978-3-030-82083-1_4

Siemens (2019) *Model-Based Systems Engineering: Integrated MBSE in Automotive.* White Paper.https://www.plm.automation.siemens.com/media/global/en/Siemens%20SW%20Model%20Based%20Systems%20Engineering%20wp_tcm27-70391.pdf (accessed on 07th Dec 2023).

Stone RB and Wood KL (2000) Development of a functional basis for design. *Journal of Mechanical Design* **122**, 359–370. https://doi.org/10.1115/1.1289637

Stone RB, **Wood KL and Crawford RH** (2000) A heuristic method for identifying modules for product architectures. *Design Studies* **21** (1), 5–31. https://doi.org/10.1016/S0142-694X(99)00003-4.

Suh N (1998) Axiomatic design theory for systems. *Research in Engineering Design* **10**, 189–209. https://doi.org/10.1007/s001639870001.

Summers JD, **Eckert C and Goel AK** (2017) Function in engineering: Benchmarking representations and models. *AIEDAM* **31**, 401–412. https://doi.org/10.1017/S0890060417000476.

Tomiyama T, **Gu P**, **Jin Y**, **Lutters D**, **Kind C and Kimura F** (2009) Design methodologies: Industrial and educational applications. *CIRP Annals* **58**, 543–565. https://doi.org/10.1016/j.cirp.2009.09.003.

Ulrich KT and Eppinger SD (2003) *Product Design and Development*, 3rd Edn. New York: McGraw-Hill/Irwin.

Wichmann RL, **Gericke K**, **Eisenbart B and Moser H** (2018) A method for function integrity diagnosis and documentation: FIDD. In *Proceedings of the Design Society: DESIGN Conference*, pp. 1429–1440. https://doi.org/10.21278/idc.2018.0211

Yildirim U and Campean F (2020) Functional modelling of complex multidisciplinary systems using the enhanced sequence diagram. *Research in Engineering Design* **31**, 429–448. https://doi.org/10.1007/s00163-020-00343-8.

Yildirim U, **Campean F and Williams H** (2017) Function modeling using the system state flow diagram. *AI-EDAM* **31** (4), 413–435. https://doi.org/10.1017/S0890060417000294.

Yildirim U, **Campean F**, **Korsunovs A and Doikin A** (2023) Flow heuristics for functional modelling in model-based systems engineering. *Proceedings of the Design Society*; **3**:1895–1904. https://doi.org/10.1017/pds.2023.

Yin RK (2018) *Case Study Research and Applications*, 6th Edn. SAGE

Zhang Y, **Roeder J**, **Jacobs G**, **Berroth J and Hoepfner G** (2022) Virtual testing workflows based on the function-oriented system architecture in SysML: A case study in wind turbine systems. *Wind* **2**, 599–616. https://doi.org/10.3390/wind2030032.

Zhu S, **Tang J**, **Gauthier J-M and Faudou R** (2019) A formal approach using SysML for capturing functional requirements in avionics domain. *Chinese Journal Aero* **32** (12), 2717–2726. https://doi.org/10.1016/j.cja.2019.03.037.

Zingel C, **Albers A**, **Matthiesen M and Maletz M** (2012) Experiences and advancements from one year of explorative application of an integrated modelbased development technique using C&C^2-A in SysML. *International Journal of Computer Science (IJSC)* **39** (2), 165–181.

https://doi.org/10.1017/S0890060425100176 Published online by Cambridge University Press