Received: 3 December 2021 | Revised: 10 February 2024 | Accepted: 15 April 2024

DOI: 10.1002/sys.21761

REGULAR ARTICLE

WILEY

Development of functional architectures for cyber-physical systems using interconnectable models

Oliver C. Eichmann[1] | **Jesko G. Lamm**[2] | **Sylvia Melzer**[3] | **Tim Weilkiens**[4] | **Ralf God**[1]

[1]Institute of Aircraft Cabin Systems, Hamburg University of Technology, Hamburg, Germany

[2]Bernafon AG, Bern, Switzerland

[3]Institute of Information Systems, University of Lübeck, Lübeck, Germany

[4]oose Innovative Informatik eG, Hamburg, Germany

Correspondence
Oliver C. Eichmann, Institute of Aircraft Cabin Systems, Hamburg University of Technology, Hein-Sass-Weg 22, Hamburg 21129, Germany.
Email: oliver.eichmann@tuhh.de

Funding information
Bundesministerium für Wirtschaft und Energie, Grant/Award Number: 20K1510D; German Academic Scholarship Foundation; Innosuisse - Schweizerische Agentur für Innovationsförderung, Grant/Award Number: 44423.1 IP-LS

Abstract

Cyber-physical Systems (CPSs) are characterized by entities in both the physical and the virtual space, thus enabling an immersion of the physical world into the cyberspace. Connectivity via the cyberspace allows CPS cooperation for new services in product service systems (PSS). In consequence, cooperating CPSs act as actors with interest in the CPS in focus. Considering the needs of human actors and cooperating CPSs is a challenging task in CPS development because of many actors, interdepending CPS functions, and multiple CPS interfaces. For systems, the known Functional Architectures for Systems (FAS) method offers a solution approach for deriving functional system architectures from system use cases originating from human actors. For CPS development, this publication presents an expansion of the FAS method for developing functional architectures based on use cases originating from human actors as well as from cooperating CPSs and offers a model-based approach based on the method description. In the authors' opinion, interconnectable CPSs and models of cooperating CPSs can be integrated and interconnected with each other into a unifying aggregated model to represent the joint behavior of CPSs in an aggregated system. The paper explains this novel approach through a CPS functional architecture development example related to the prediction of remaining boarding time in an aircraft. The result is an approach that allows the consideration of initial CPS functions and new aggregated system functions, that pays special attention to the interconnectivity of CPSs and the required interfaces, and enables the systematic analysis of functions for the identification of redundancies.

KEYWORDS
cyber-physical systems, functional architecture, interconnectable models, model-based system architecture, product service systems

1 | INTRODUCTION

Increasing performance, miniaturization, and integration of microelectronics in mechatronic systems result in better information, communication, control, and computation capabilities[1] of mechatronic systems. Above all, the communication capabilities can enhance cooperation between multiple systems.[2] The resulting class of systems is composed of sensors, actuators, embedded computing systems, and—particularly important—a communication infrastructure and is denoted as the class of *Cyber-physical Systems* (CPSs), which exchange information for coordination and control.[3,4] For a comprehensive overview about CPS definitions please refer to *Henshaw 2016*.[5] Communication infrastructure and information exchange enable CPSs to network with each other and fulfill a new mission in the form of a new, larger system, denoted as an *aggregated system*. Rapid technological progress in miniaturization and digitization as well as ubiquitous and more powerful communication networks make the development of this type of aggregated systems a common but challenging development task in the industry.

One example of an extensive aggregated system in an aviation context is a learning galley catering system (LGCS) for improved in-flight catering demand predictions.[6] The LGCS automatically collects in-flight catering inventory data on the aircraft and transmits this data to the ground. On the ground, the collected data are analyzed and used for more accurate catering demand predictions for future flights for better availability of desired catering and less waste. Thus, aircraft galley providers offer a so-called product service system (PSS), in that they combine their tangible product in the form of an aircraft galley with a prediction service for in-flight catering. PSS are defined as a "market proposition that extends the traditional functionality of a product by incorporating additional services."[7] The functions for providing this PSS are distributed across a variety of CPSs on the aircraft and on the ground at the caterer and the airline. Identifying all the necessary functions is challenging due to the size of the aggregated system, with many actors overseeing only a subset of all the processes required for catering. A model-based functional analysis and architecture development is therefore a promising way to get an overview of all required functions and their interfaces. This article presents existing approaches for functional architecture development and extends the established FAS method to the CPS-FAS method for application to aggregated systems. Aggregated systems are to be distinguished from *Families of Systems* (FoS), that is, product-lines or domains in that assets are re-used unmodified and/or modified,[8] as aggregated systems in the scope of this article include diverse CPSs that do not share any assets.

Individual CPSs in an aggregated system may have overlapping functionality, leading to redundant presence of functions in this larger system. This is a well-known circumstance from the System-of-Systems (SoS) domain.[9] SoS are special cases of aggregated systems in that the so-called *constituent systems* are fully independent with respect to managerial and operational aspects.[10] In contrast and to be in full and central control of all the functions during the design of an aggregated system, functional considerations are needed for each CPS in both its own context and in the context of the aggregated system. Functional architectures structure the developed system based on its required system functions independent of a technical solution. They comprise functional elements and connections between the inputs and outputs of functional elements.[11] For an individual system, different approaches for functional architecture development exist, that is, the Integrated Systems Engineering and Pipelines of Processes in Object-Oriented Architectures (ISE&PPOOA),[12] Object-Oriented Systems Engineering Method (OOSEM),[13] and the Functional Architectures for Systems (FAS) method.[11,14] These approaches were elaborated as a reaction to a rapidly increasing number of functions and resulting complexity in the development of systems. Their objectives are to reduce design flaws and cost under given time and cost constraints, increase system performance, and reduce technology risk. Furthermore, they support clarity and traceability between functions and actor needs. They consider single systems with interfaces to actors and other systems,[11–14] as shown in Figure 1(A). The integration of CPSs into aggregated systems makes the development task even more challenging as the variety of functions and the number of interfaces continue to increase significantly. Consequently, functional architecture development approaches must be adapted for CPSs. Figure 1(B) illustrates the required scope of an approach for functional CPS architecture development, considering multiple interconnected CPSs in an aggregated system. This scope is not supported by established approaches to a full extent, as identified in a review presented in Section 2.2.

Existing methods could be used for the design of functional architectures of aggregated systems or contained CPSs, as indicated by the dotted frames in Figure 2. The result is either a functional architecture for an aggregated system (top of Figure 2) that does not consider the functions of contained CPSs or a functional CPS architecture (bottom of Figure 2) without relations to the aggregated system. The example in Figure 2 demonstrates the deficiency of current methods. The functional architecture for the aggregated system contains the functional elements (FE) FE1.1, FE1.3, and FE1.4 (top left of Figure 2), marked with a diagonal hatching, for satisfaction of stakeholder needs for the aggregated system. However, stakeholders of the CPS 1 have other needs, represented by the functional elements FE1.2, FE1.3, and FE1.4 (bottom left of Figure 2) that are marked with a diagonal hatching in the other direction. None of the architectures takes into account all the functions required by both stakeholder groups of the aggregated system and the CPS. The CPS-FAS method supports the consideration of needs by both stakeholder groups, resulting in the architecture in the middle of Figure 2. This architecture contains FE1.1, required by stakeholder needs of the aggregated system, and FE1.2 for meeting CPS stakeholder needs. FE1.3 and FE1.4 satisfy needs by both stakeholder groups.

The FAS method[11,14] is chosen for extension as it best meets the objectives for functional CPS architecture development [*cf.* objectives definition in Section 2.1] and requires the least amount of adjustment for functional CPS architecture development. In addition, an approach for describing the resulting interconnections in a single model using interconnectable models is needed. An extension of the FAS method for functional CPS architecture development is elaborated and presented in this paper in the form of the new *Cyber-physical Systems FAS method* (CPS-FAS method). A stepwise and detailed description guides system

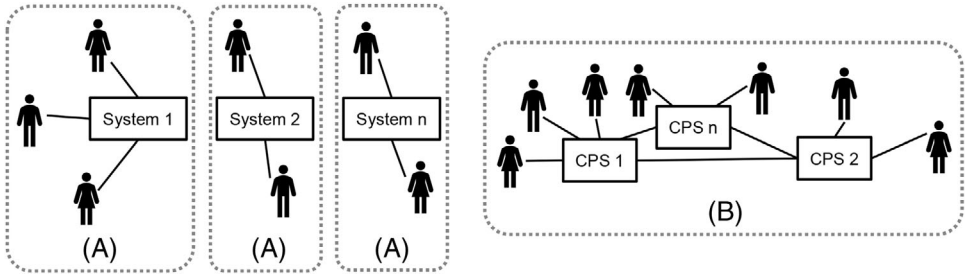

FIGURE 1 Scope of approaches for functional architecture development for single systems (A) and required approach for functional architecture development of interconnected CPSs (B).

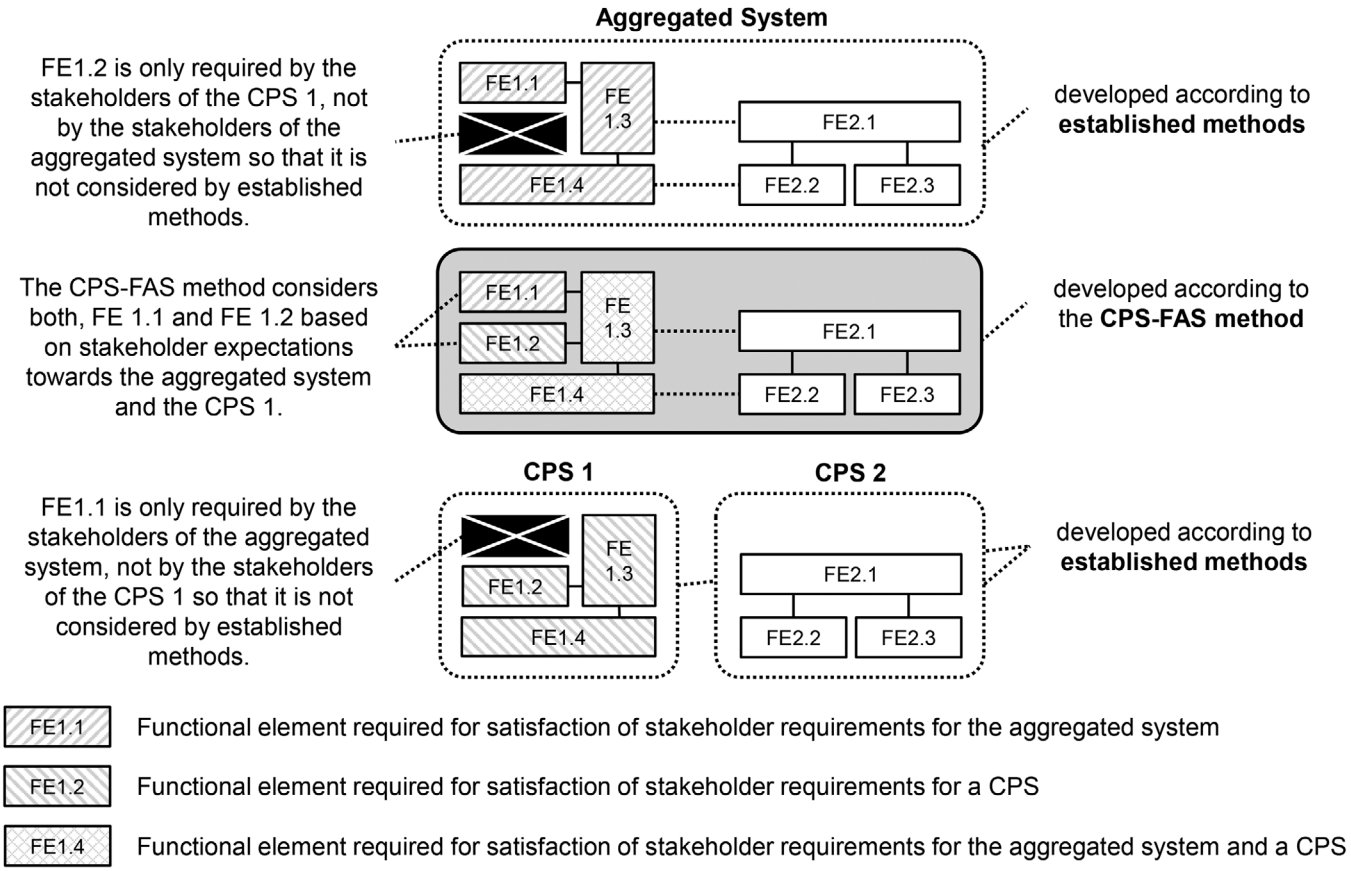

FIGURE 2 The joint consideration of functional architectures of aggregated systems and CPSs is the purpose of the CPS-FAS method.

architects and researchers for method application. The applicability of the CPS-FAS method with SysML is explained using the example of a *Departure Control System* (DCS), which is part of an aggregated system. The DCS is used by airlines for managing passenger data and will be explained in more detail in Section 5. In addition, a plugin for the modeling tools *MagicDraw* and *Cameo Systems Modeler*, a profile, and a model template were developed that support system architects and researchers in application of the CPS-FAS method. They are accessible on a public file hosting platform.[15]

Because of the complexity in an interacting set of multiple CPSs a model-based approach is chosen to enable information capturing and CPS functions identification. The functional architecture development is supported by interconnecting CPS models to create a single model of the aggregated system. Moreover, the novel CPS-FAS method focuses specifically on the communication aspects of CPSs and extends classic use case analysis. The method is developed for meeting the given collection of objectives and is introduced in general and generically in Section 3.1 without the use of the *Systems Modeling Language* (SysML) and related tools. Subsequently, the model-based development of functional CPS architectures using the SysML follows in Section 3.2. The model-based approach allows the automation of some method-related process steps. The potential for automation and an example

of an automated implementation are described in Section 4. Section 5 presents the application of the model-based CPS-FAS method for the development of functional architectures. The CPS-FAS method is applied to the development of a boarding time prediction system that estimates remaining time until completion of aircraft boarding, a smart inflight catering trolley, a document archive and library system, and a hearing healthcare system.

2 | DEVELOPMENT OF FUNCTIONAL ARCHITECTURES

Information, communication, computation, and control capabilities of CPSs demand a suitable functional architecture development approach. CPS characteristics must be considered and are translated into objectives that shall be achieved by an improved development approach. Section 2.1 presents identified objectives and is followed in Section 2.2 by a presentation of existing functional architecture development methods. A gap analysis of existing methods reveals opportunities for improvement in functional CPS architecture development, which are addressed in this article.

2.1 | Objectives

Different modeling activities in the research projects SiLuFra[16] and ConCabInO[17] have identified research needs concerning functional CPS architecture development. A secure air cargo transport chain including necessary infrastructure was developed in the SiLuFra project.[16] The SiLuFra infrastructure included multiple connected CPSs such as gates for the identification of cargo or systems for security checks. During the development of functional architectures for SiLuFra was noted that none of the analyzed methods supported the distinction of functional interfaces to numerous other CPSs. Furthermore, none of the methods supported the design of a functional architecture for the aggregated air transport system. The ConCabInO project[17] aimed at the design of optimized business processes in aircraft cabins using the benefits of CPSs, for example, a smart trolley. The smart trolley identifies and counts the in-flight catering it contains and connects to CPSs at the airline and catering providers to enable more accurate catering demand predictions. During the design of the smart trolley and other CPSs involved in the ConCabInO project none of the existing methods supported the identification of functions on the level of the aggregated system

In addition to the experiences from the projects, a literature review was performed to correlate and confirm the identified research needs in the area of CPS architecture design. Engell[18] describes system architectures for CPSs and the integration of new components as one of the key challenges in CPS research. The design and modification of a functional architecture should allow the addition and modification of functions with limited changes of many parts of the system. Colombo et al.[19] emphasize that the collaboration of heterogeneous CPSs in the cyberspace allows the creation of new functionalities, resulting in a challenging development task due to multidisciplinarity that must be encountered by engineering methods and tools for evolvable architectures. These methods and tools should consider the aggregated system as a set of cooperating entities.[19]

CPSs, being part of an aggregated system, are developed by multiple groups in a comprehensive development team, where the design of the aggregated system is a shared task. Experiences from collaboration of multiple groups in an extensive team show that the team performance is improved when all groups are aware of the common, overarching task and have more insights into the work of their collaborators.[20] Therefore, the CPS development results are expected to improve if development teams know the purpose of the aggregated system and have insights into the work of other teams in the form of functional architectures.

The experiences from the research projects SiLuFra and ConCabInO and the results of the literature review are translated into objectives that an approach for functional CPS architecture development should achieve:

1. Technological systems typically exist before their integration into an aggregated system and serve a specific purpose. This initial purpose must be considered when designing CPS tasks for the aggregated system. Consequently, the method shall provide a systematic way of deriving functional architectures with a modeling approach for serving both, individual and aggregated CPS purposes.
2. The second objective expresses the modeler's need to represent the interlinked CPSs in a model. Communication capabilities are characteristic for CPSs[1] and prerequisite for CPS cooperation in an aggregated system. The method shall have a focus on the interconnectivity of CPSs, allowing communication flows of a single CPS with its surroundings to be identified and described in a model.
3. The third objective is about the need to address interfaces between CPSs explicitly in the novel method, independent from their precise model representation. Communication capabilities of CPSs lead to multiple data and information exchanges between CPSs. Data and information exchange via the interfaces of CPS functions shall be specifiable during method application. Thus, function inputs and outputs are considered by the method. Since the second and third objective arise from characteristic abundance of interconnection topics in CPSs they are somewhat related.
4. CPSs in an aggregated system may have overlapping functionality. This leads to function duplicates and thus unnecessary complexity of the aggregated system. Therefore, the method shall allow the identification of redundant functions across an aggregated system of CPSs. It shall be possible to express which CPS provides the respective function in the context of the aggregated system.

2.2 | Existing approaches

Various architecture development approaches for self-contained and aggregated systems exist. The ISE&PPOOA, the OOSEM approach, and the FAS method are well-known methods for functional architecture development and are introduced below.

Fernandez and Hernandez suggest the Integrated Systems Engineering and Pipelines of Processes in Object-Oriented Architectures (ISE&PPOOA) methodology.[12] Part of the ISE&PPOOA methodology includes the following precondition and process steps for functional architecture development:

- **Precondition**: Operational scenarios are identified, and required system capabilities and quality attributes are specified.
- **Step #1**: High-level system functions are identified by analyzing similar systems and the inputs and outputs of the desired system in its context.
- **Step #2**: High-level functions are decomposed into subfunctions, thus resulting in a functional hierarchy that can be modeled in a SysML block definition diagram. Subfunctions can exist on multiple hierarchy levels. The correct degree of granularity depends on the allocation of subfunctions to building elements for the system solution.
- **Step #3**: Interfaces between functions are considered in N2 charts. For a number of N functions, N2 charts have N+1 columns and N+1 rows. The first row and the right-hand column contain entries for system inputs and outputs, respectively. Functions are entered on the diagonal of the remaining matrix elements, except the first row and the right-hand column. Interfaces between functions are indicated by entries in the matrix cell at the intersection of the row of the output function and the column of the input function. Reordering of functions in the matrix allows identifying potential functional clusters. It becomes obvious which functions are strongly connected to each other and which functions are independent. Strongly related functions should be allocated to the same physical element. Interaction between functions can be further specified using functional flows in SysML activity diagrams. The ISE&PPOOA methodology considers interfaces to external actors and other systems from N2 matrices as accept event and send signal actions in SysML activity diagrams.[12]

Potential for improvement with respect to CPS development: The ISE&PPOOA methodology is intended for the development of functional architectures for general systems. Although interfaces to external systems are considered as accept event and send signal actions, a white box view on the functional architecture of the aggregated system that is composed of the functional architectures of the included CPSs is not in the scope of the ISE&PPOOA methodology.

The Object-Oriented Systems Engineering Method (OOSEM)[13] defines logical architectures as a set of logical components. Logical components are abstractions of physical components and describe system functionality independent of implementation considerations and constraints. They are therefore similar to functional blocks, whereby the grouping criterion is the technical concept that represents the logical component.

Logical architecture development according to OOSEM contains the following precondition and steps:

- **Precondition**: Required system behavior is modeled in operational scenarios in the form of activity diagrams. These activity diagrams are divided by activity partitions that represent actors and external systems. Behaviors and interactions of actors, external systems, and the system under development are modeled as actions as well as control and object flows between actions. Actions are grouped into activity partitions and thereby allocated to the system under development, actors, and external systems. At this development step it is uncertain how the actions are performed by the system so that the actions are denoted as black box operations of the system.
- **Step #1**: The system is decomposed logically into external interface, application, and infrastructure components, resulting in a hierarchical representation of logical components.
- **Step #2**: Interactions and interfaces between logical components are identified for each system function. Interactions and interfaces are modeled in activity diagrams and in internal block diagrams, respectively. Each black box system operation from operational scenarios is refined by an activity diagram. These activity diagrams utilize activity partitions for grouping of actions into logical components.[13]

Potential for improvement with respect to CPS development: OOSEM considers both internal and external interfaces. Internal interfaces are between subsystems and external ones are between the system and users, external systems, and the environment. For developing functional CPS architectures in an aggregated system, a connection of the logical CPSs architectures for modeling structure and behavior of aggregated systems would be beneficial.

The FAS method introduced by Lamm and Weilkiens[11,21] provides a systematic approach for developing and modeling functional architectures. It is independent of any modeling language but suited for a model-based application, for example, using the SysML (Systems Modeling Language). Modeling support using the SysML is presented after a general description of the FAS method. The FAS method comprises the following precondition and steps:

- **Precondition**: The system context describes interactions of the system under development with its environment. The context indicates system boundaries and interfaces to external actors. System requirements are collected and refined by use cases. These use cases are further refined by activities, so-called *use case activities* in the context of the FAS method. Activities are interconnected for object exchanges representing flows of energy, material, and/or information.
- **Step #1**: Activities are grouped, for example, based on heuristics.[11] A suitable noun is assigned as a name to each resulting functional group.
- **Step #2:** Functional elements and functional interfaces form the functional architecture of a system. A functional element is created for each functional group in order to obtain a functional architecture. Functional interfaces are created between functional elements based on object exchanges in use case activities. A functional

TABLE 1 Mapping of FAS concepts to SysML.[21]

FAS concept	SysML element
Use case activity	Activity defining use case behavior
I/O activity	Activity whose call behavior actions are part of an activity partition with stereotype I/O
Core activity	Activity whose call behavior actions are part of an activity partition without stereotype I/O
Functional element	Part property typed by a functional block (block with FAS stereotype "functional block")
Functional group	Block with FAS stereotype "functional group"
Functional interaction point	Proxy port typed by an interface block
Functional parameter	Flow property of interface blocks that type functional interfaces
Functional connection	Connector between proxy ports of functional blocks
Functional interface	Functional connection including the connected functional interaction points, that means SysML connector and connected proxy ports

interface is created for each object exchange between activities that are assigned to different functional groups.

- **Step #3 (optional)**: The functional architecture can be reworked and refined.

Lamm and Weilkiens present a corresponding modeling approach.[11] Relevant elements for the model-based application of the FAS method are part of the FAS method domain model in Figure 3 and modeled as domain block. Domain blocks are used to model terms and their structure of a specific domain,[22] for example, aviation.

The domain model is divided into three areas containing elements for FAS method application. These areas comprise elements for requirements analysis (top of Figure 3), elements for the preparation of functional architecture development (middle), and elements for functional architecture development (bottom). The elements of the domain model are utilized to achieve the precondition (requirements analysis) as well as for the work in step #1 (functional grouping) and step #2 (functional architecture development) of the FAS method. Requirement types, use cases, and use case activities are used for requirements analysis to achieve the precondition of the FAS method. Functional groups in the middle of the diagram are defined for preparing the functional architecture development. They are mapped to functional elements at the bottom of the diagram. Besides functional elements, functional architecture development requires model elements for describing connections between functional elements. These are functional connections, interfaces, and parameters. Table 1 presents how elements from the FAS method and the associated domain are allocated to SysML elements. The modeling of functional elements with SysML blocks has been proposed by Lamm and Weilkiens,[23] by Korff et al.[24] and later also by Fernández-Sánchez et al.[25]

Use case activities can be allocated either to the system or its interfaces using partitions in SysML activity diagrams. Partitions that contain use case activities on system interfaces are marked with an «*I/O*» stereotype. SysML blocks with «functional block» and «functional group» stereotypes represent the functional elements and groups. Interfaces between functional blocks are modeled by means of proxy ports and connectors.

For model-based development of functional architectures with SysML using the FAS method, the following considerations should be taken into account for the previously described precondition and steps:

- **Precondition**: The system context is modeled as a block that is decomposed into external actors and the system. As part of the model-based requirements analysis, use cases are identified and modeled. Use cases are refined into use case activities that are described by an activity diagram with call behavior actions. Call behavior actions call further use case activities that are taking the role of sub-functions. Use case activities are either system activities or I/O activities, depending on their interface characteristics. System activities are mainly performed by the system without interaction with its environment, while I/O activities encompass interaction with external actors or systems. System activities and I/O activities are modeled in system and I/O activity partitions, respectively.
- **Step #1**: Each functional group has the stereotype «*functional group*», that is a specialization of the SysML stereotype "block." Use case activities are connected to functional groups using a trace relationship. An I/O group is created for each I/O activity during functional grouping. System activities are grouped and allocated to system groups.
- **Step #2:** In order to model functional blocks, model elements corresponding to the groups from step #1 are created with same name. Functional blocks have the stereotype «*functional block*» that is a specialization of the SysML stereotype "block." Part properties of the system block are created and interconnected via proxy ports and connectors in order to model flows of material, information, and/or energy in internal block diagrams. The interconnected part properties represent the functional architecture.

Potential for improvement with respect to CPS development: The FAS method provides a step-by-step approach for the development of functional architectures. Interfaces to other systems are considered in use case activities in I/O activity partitions. Using multiple I/O activity partitions for the consideration of interfaces to multiple

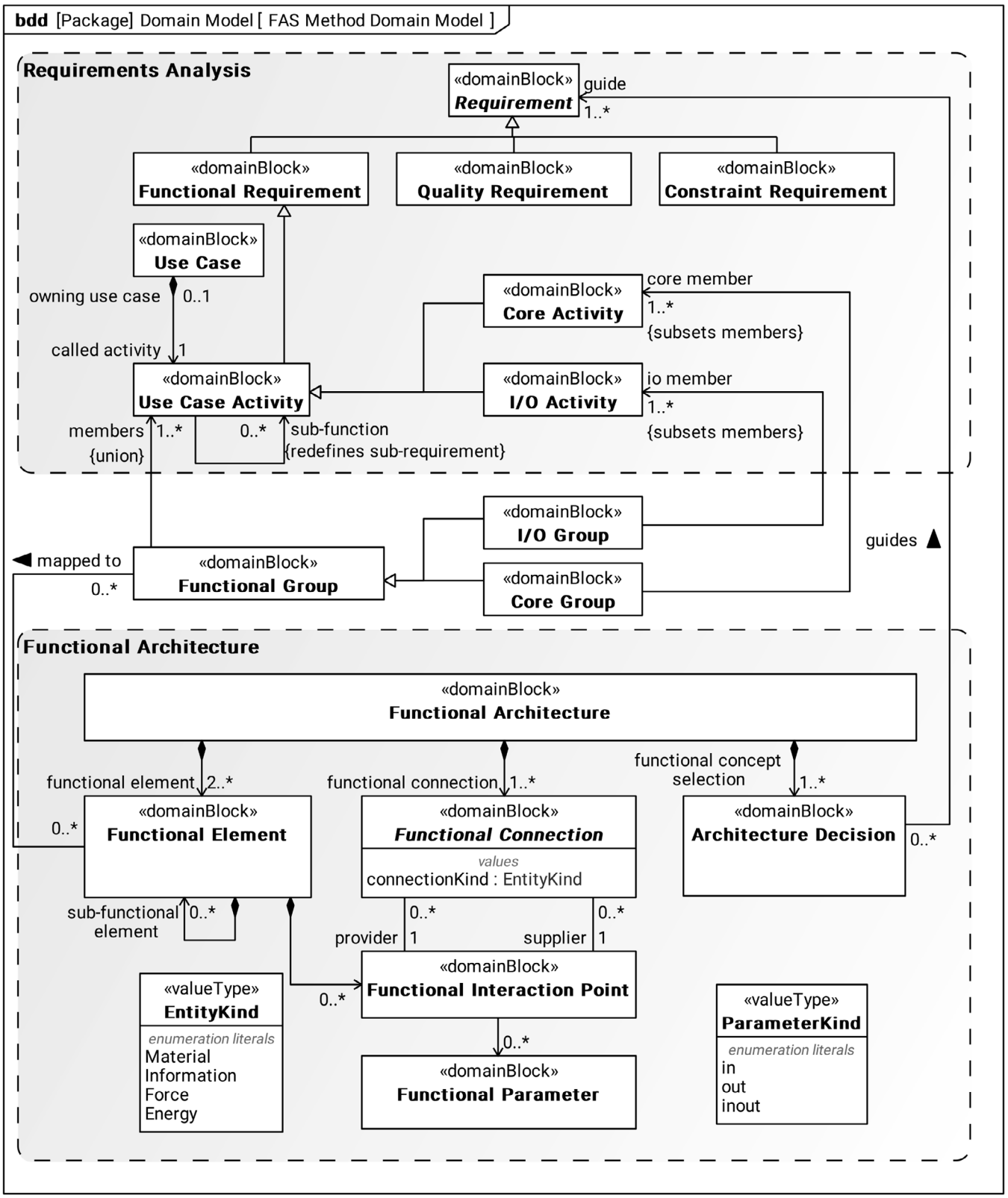

FIGURE 3 FAS method domain model, adapted from Weilkiens et al.[21]

CPSs would be beneficial. One should note that "internal" interactions of the aggregated system may be "external" interactions of an individual involved CPS. The I/O partitions should be used in the modeling of each respective individual CPS and hence relate to the external interfaces of an individual CPS (which may include interfaces to other CPSs that would then be "internal" intefaces of the aggregated system). In addition, use cases should consider human user needs as well as demands by other CPSs. The derived functional architectures of CPSs should be importable for modeling functional architectures of aggregated systems. Binder et al.[26] recognized the need for functional architecture development in aggregated systems and integrated the FAS method into the Reference Architecture Model Industrie 4.0 (RAMI 4.0) for developing functional architectures in Industry 4.0 systems. The diagrams of the FAS method are added to RAMI 4.0 for developing a System of Interest in an Industry 4.0 context. It is mentioned that the FAS method could also be applied to the supersystem, that is, the aggregated system. However, the utilization of the FAS method on aggregated system level and the transition between CPS and aggregated system level are not further described.

The FAS method is chosen for extension and tailoring to CPS needs because it fulfills the objectives from Section 2.1 to a large extent. Although the ISE&PPOOA approach supports the identification of redundant functions by means of N2 charts, the FAS method is chosen for extension as it allows for tracing redundant functions to the original use cases that led to the implementation of the function in the given CPS.

3 | FUNCTIONAL ARCHITECTURE DEVELOPMENT FOR CYBER-PHYSICAL SYSTEMS

Section 2.2 introduces approaches for functional architecture development and justifies the choice of the FAS method for an extension to the CPS-FAS method. The subsequent Section 3.1 introduces the CPS-FAS method in general and generically without modeling support using the SysML and respective tools. Instead, fundamental principles of the developed method are introduced. Intensive communication between various CPSs results in a higher degree of complexity which is difficult to cope with in conventional, document-centric approaches. Hence, the principles of the CPS-FAS method have been transformed to a model-based approach in the next step. This model-based approach is able to cope with complexity by utilizing the SysML and is described in Section 3.2.

3.1 | The CPS-FAS method

A CPS-FAS method meeting the objectives given in Section 2.1 has to address the functional design of one or multiple CPSs. The inherent communication capabilities of CPSs will enable their interconnection with other CPSs during the integration into an aggregated system. Figure 4 shows a graphical representation of the CPS-FAS (top of Figure 4) and the FAS method (bottom of Figure 4) and considered elements, that is, actors, CPSs, use cases, functional groups, and functional elements. The associated process for application of the CPS-FAS method is presented in Figure 5. Lowercase letters indicate elements of the CPS-FAS method in Figure 4 and related process steps in Figure 5. The process includes steps at the aggregated system and CPS levels, separated by a horizontal line and is described in more detail below.

The cooperation of multiple CPSs requires an overview of relevant CPSs and interacting human actors. Therefore, application of the CPS-FAS method starts with identification of actors of the aggregated system and CPSs that constitute the aggregated system, as shown in part (a) of Figure 4. Connecting lines represent interfaces for the exchange of information, energy, and/or material between actors and CPSs.

Services required by actors of the aggregated system are expressed as a set of use cases, as presented in part (b) in Figure 4 and on the left-hand side of Figure 6. Use cases describe the intended use of the aggregated system and are refined by use case activities (right-hand side of Figure 6). Latter ones are connected to each other by flows of information, energy, and/or material.

The aggregated system cannot provide the functionality itself. Instead, the contained CPSs provide desired functionality through cooperation. In consequence, the aggregated system activities are allocated to respective CPSs, as shown in part (c) of Figure 4 and in Figure 7.

Activities on aggregated system level are considered as use cases on the CPS level so that functions requested from the aggregated system are considered for functional architecture development of the CPSs.

Before being integrated into an aggregated system, CPSs provide services in their original context so that there are native use cases as well as natively required functions involving human actors. Allocating additional activities from aggregated system use cases leads to additionally required functions and the reuse of initially existing CPS functions within the aggregated system. Both function types, native functions and additionally required functions for CPS usage in an aggregated system, must be considered during the development of functional CPS architectures. Additionally, required functions result from integration into an aggregated system and are considered by creating use cases for allocated aggregated system activities (top of Figure 8). A CPS use case is created for each aggregated system activity. Native system functions stem from use case identification according to the conventional FAS method[11] considering human interaction (bottom of Figure 8).

On the CPS level, these use cases are further described using activities [*cf.* right-hand side of Figure 8], identical to use case description in the conventional FAS method. Activities that refine both use case types, that is, use cases resulting from the integration into an aggregated system and native use cases expressing human actor needs, are considered consistently in subsequent identification of functions during functional grouping which is explained below.

Activities related to use cases may overlap and thus cannot be conclusively translated into a well-organized set of related functions. In consequence, a functional grouping [*cf.* part (e) in Figure 4 and left-hand

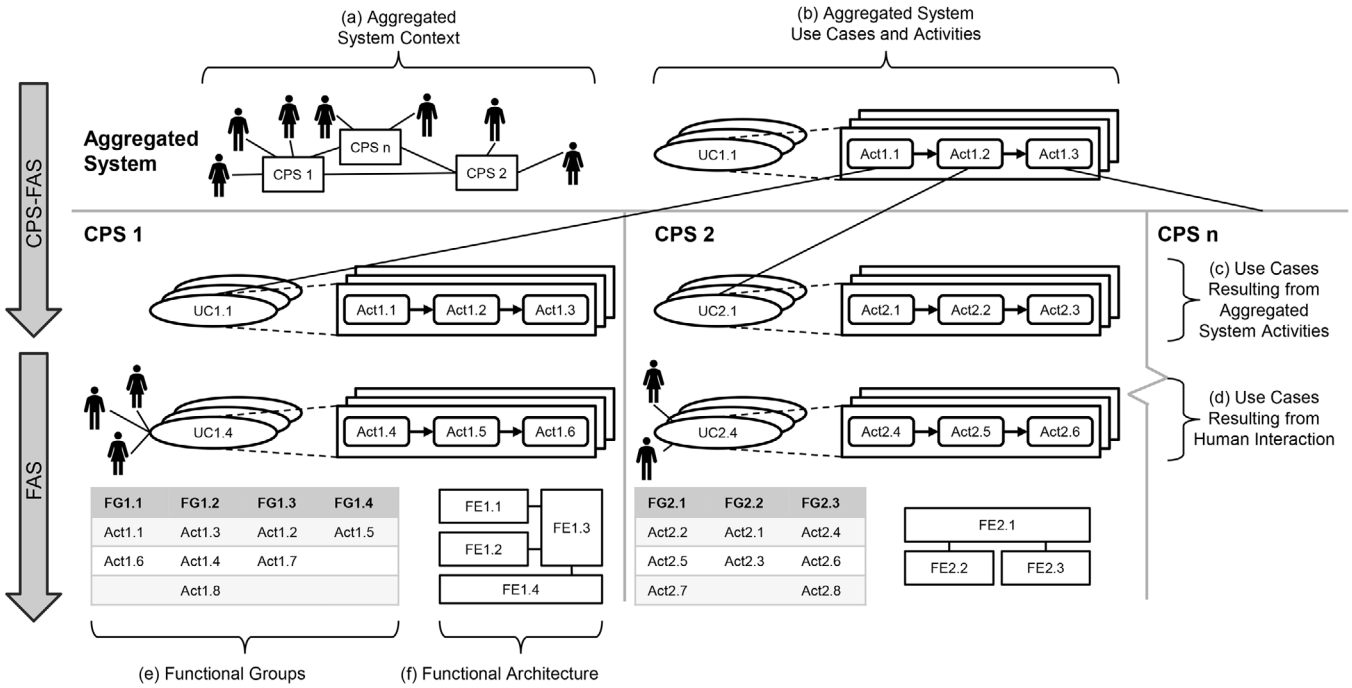

FIGURE 4 Graphical representation of the CPS-FAS method, adapted from Eichmann et al.[27]

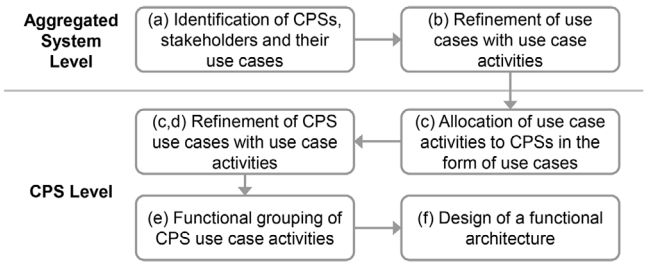

FIGURE 5 Process for application of the CPS-FAS method.

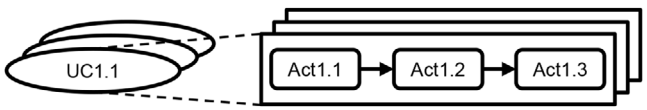

FIGURE 6 Description of aggregated system services by use cases and use case activities.

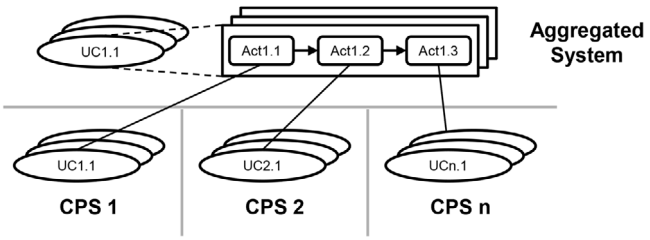

FIGURE 7 Allocation of aggregated system use case activities to CPSs.

side of Figure 9] clusters similar activities into functional groups so that a condensed and conclusive set of required functions is elaborated that comprehensively covers required functionalities from human user and aggregated system perspectives.

Heuristics for functional grouping are presented by Lamm and Weilkiens[11] and consider, for example, functions that are relevant to same system actors, function calls, functions that share data, and existing system grouping criteria. Use case activities with a relation to other CPSs or actors are grouped by I/O functional groups for indicating their interface characteristics.

A functional element of the same name is created for each functional group [*cf.* part (f) in Figure 4 and right-hand side of Figure 9]. Functional elements are the building blocks for the functional architecture. Connections between functional elements are generated based on the flows between use case activities [*cf.* right-hand side of Figure 8]. Interfaces between functional elements in the functional architecture are created for each information, energy, and material flow between activities that are allocated to different functional groups.

In conclusion, the CPS-FAS method starts with the identification of CPSs, actors, and their use cases for an aggregated system [Figure 4(a)]. Use case activities describe these use cases in more detail [Figure 4(b) and Figure 6] and are traceable to CPSs by means of use cases [Figure 4(c) and Figure 7]. Use cases are detailed by means of use case activities on CPS level [Figure 4(c,d) and Figure 8]. They are grouped by functional groups [Figure 4(e) and Figure 9, left]. For each functional group, a functional element with the same name is created. Interfaces between functional elements are based on flows between use case activities. Functional elements and their interfaces represent CPS functional architectures [Figure 4f and Figure 9, right].

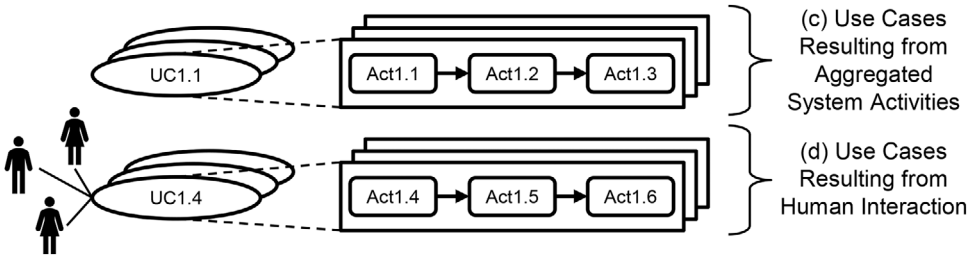

FIGURE 8 Description of desired CPS behavior by use cases and use case activities.

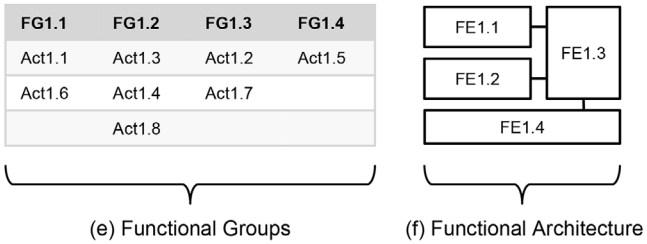

FIGURE 9 Functional grouping of activities and functional architecture derivation.

3.2 | The CPS-FAS method with SysML

Implemented CPSs already possess a distinctive system structure and provide multiple CPS functions. When integrating an existing CPS into an aggregated system, the extended functionality results in a more extensive system structure and thus a more challenging development task. Developing such functional architectures in a document-centric CPS-FAS approach may result in inconsistencies or design flaws. Taking the example of adding use case activities for a more detailed use case description, a document-centric approach must rely on the skills of a system architect who has to consider this added activity in functional grouping and the following architecture development as well. In all these development steps, inconsistencies and/or design flaws may occur. To support the prevention of these mistakes for obtaining a complete and consistent representation of the functional CPS architecture, the newly developed CPS-FAS method supports a model-based approach, which would allow for the use of model checkers and other means to maintain consistency.

A model is an abstract representation of reality[13] and allows the creation of different views on the information stored in a model repository, whereby all views can be kept consistent. Changes or additions in a particular model view are stored in the central model repository and can therefore be made visible in all other model views, with the appropriate kind of automation. Taking up the example of an additional use case activity and relating this to a model-based approach: Adding a use case activity in an activity diagram stores this activity in the model repository. The additional activity is displayed automatically in a model view for functional grouping.

In a model-based CPS-FAS approach, it is possible to check the model automatically to make sure that all use case activities are a member of a functional group. And automation is not restricted to model checking. A model-based approach offers various possibilities for automation, as shown in Section 4, for example, the support for functional architecture derivation after functional grouping. Due to the cyber-physical nature of the systems, use cases need to be interpreted in a more abstract form in this approach. Unlike traditional use case analysis that is typically human-centric, also the use of a CPS from another techno-centric system can be similarly modeled as a use case, in which the mentioned technical system becomes the main actor.

In systems modeling, the SysML is a well-established modeling language for model-based systems engineering. With the publication of the SysML specification in 2007, the modeling of CPSs using the SysML has started[28,29] and is still a research topic.[30–35] While early research focuses on developing or analyzing architectures or mapping them to other system development processes, contemporary literature presents newly defined meta-languages or SysML profiles for developing CPS architectures[31] or developing CPSs in the context of SoS.[34] Nevertheless, these approaches do not present how to integrate different independently modeled CPSs or CPSs developed in different engineering domains into an overall aggregated system. This is exactly the point where the CPS-FAS method comes in. In the following and for simplicity reasons, it shall be assumed that all CPSs and other systems to be integrated are described by a functional architecture model that has been derived according to the FAS method. Whether such models are available as a part of a description of an off-the-shelf CPS or whether they need to be created by organizational units involved in carrying out the integration of the aggregated system has no relevance for the application of the method.

In the CPS-FAS method using SysML, an aggregated system context, as shown in part (a) of Figure 4, is created and described by an internal block diagram first. Part properties in the internal block diagram represent actors and CPSs that are part of the aggregated system. When using the suggested model structure that integrates interconnectable CPS models into an aggregated system model, part properties of CPSs can have the type of the system blocks that are contained in CPS models. After context modeling, aggregated system use cases are modeled in use case diagrams and connected to actors by means of association relationships, as presented in part (b) of Figure 4 and in Figure 6.

Aggregated system use cases are refined into use case activities for further explanation of the use cases. The details of use case activities are clustered by activity partitions. CPSs that are affected by the respective use case are allocated to the appropriate activity

partitions. Activities are created in CPS models and selected as behavior for call behavior actions in aggregated system use case activities. This procedure allows traceability between aggregated system and CPS entities. Moreover, activities on the CPS level that are used as behavior in a call behavior action on aggregated system level can be used for functional architecture development of CPS. These activities resulting from aggregated system use cases coexist besides activities resulting from native CPS use cases. For the distinction between conventional CPS use cases and use cases that result from CPS integration into aggregated systems, new modeling conventions to address communication and interconnection capabilities of CPSs are described as follows:

- A use case involving the communication infrastructure with non-human interactions of the system shall be modeled with a stereotype «cps-machine» that has been newly introduced to indicate that a dominating machine-to-machine interaction is represented [*cf*. part (c) in Figure 4].
- A use case involving a predominant amount of human interactions with the system in similar system functions as before shall be modeled with a stereotype «cps-human» that has been newly introduced to reflect the dominance of human interaction [*cf*. part (d) in Figure 4].

Both stereotypes are extensions of the metaclass "use case" and support the modeling of CPS use cases with the focus on machine interactions, considering the fact that use cases in classical systems are only used to capture human interactions with systems. Differentiating use cases by means of the introduced stereotypes allows insights into which functions require special attention due to a higher degree of human interaction, while cps-machine use cases may be easier to automate. In addition, similar use cases that are required by both humans and external systems can be distinguished.

A typical CPS property is to provide access to its core functionality via its communication interfaces—and often such core functionality is also accessible via a direct human-machine interface but can be also based on a machine-machine interaction. In other words, a particular system function can be performed by both: by direct human interactions or via interactions with other CPS over communication links. In consequence, the extension of the FAS method with a special focus on both use case types leads to two distinctive use cases with different actors. Therefore, it is suggested (*i*) to model all use cases, (*ii*) classify modeled use cases as cps-human or cps-machine, and then (*iii*) create generalized use cases of a set of cps-human and cps-machine use cases. The generalized use cases are modeled as abstract use cases because only the specializations of the use case exist in the real world.

Section 5 presents application of the CPS-FAS method using the example of an aggregated system for remaining aircraft boarding time predictions. The *Departure Control System* (DCS), briefly introduced in Section 1, is used as an example for a CPS with cps-human and cps-machine use cases. A sample cps-human use case describes passenger check-in at the terminal by airport personnel. A corresponding cps-machine use case is passenger check-in via a remote web application that acts as an external system with respect to the DCS. Both use cases are specializations of the abstract use case *Check-In Passengers*.

Use cases like "Check-In Passengers—Machine" are described in more detail by use case activities. Use case activities of different CPSs can be connected via call behavior actions, pins, and object flows for modeling processes that run across different CPSs in an aggregated system. Analogously, the functional architecture of an aggregated system can be modeled by interconnecting functional architecture models of individual CPSs that have been obtained via the FAS method. This is done via connectors between the proxy ports representing communication interfaces of the CPSs. These proxy ports can be automatically created by a computer program that interprets the corresponding activity pins used for modeling the aforementioned object flows between use case activities. This will be demonstrated by means of creating a sample implementation of the computer program in the modeling tool plugin *FAS for MagicDraw*,[15] which will be explained in Section 4.

Overall, a functional model of CPSs and their interconnection in an aggregated system can be obtained via the CPS-FAS method by following the subsequent sequence and steps:

Obtain functional CPS models

For each of the relevant CPSs, do the following:

- **Precondition**: The system context for the CPS is modeled according to Section 2.2.
- **Step A.1**: Model use cases of the CPS and ensure that variants of a given use case with more dominant human interaction or more dominant machine interaction are identified and modeled via different specializations of the same use cases, distinguished by assigning the stereotypes «cps-human» and «cps-machine», respectively.
- **Step A.2**: Refine CPS use cases by means of use case activities and the corresponding sub-activities as described in the preconditions for the FAS method from Section 2.2.
- **Step A.3**: Apply the steps of the FAS method from Section 2.2 to derive the functional architecture of the CPS.

Obtain the functional use case model of the aggregated system

After modeling the functional architectures for CPSs considering their original human use cases, use cases for the aggregated system are identified:

- **Step B.1**: Model use cases of the aggregated system.
- **Step B.2**: Refine the use cases of the aggregated system by means of use case activities.
- **Step B.3**: Refine each use case activity of the aggregated system in an activity diagram that uses call behavior actions to call the corresponding use case activities of the involved CPSs. Activity partitions can be used to show the allocation of calls to CPSs.

Compose a functional architecture model of the aggregated system from the functional architecture models of the CPSs

The CPS activities resulting from human and machine use cases are allocated to functional groups as described in the conventional FAS method in Section 2.2. The functional architectures of the individual CPSs can be combined as described in the following steps:

- **Step C.1**: Compose a system block representing the aggregated system, comprising the topmost blocks of the functional architectures of CPSs in the aggregated system.
- **Step C.2**: Show the resulting part properties of the aggregated system, and optionally their internal structure, in an internal block diagram.
- **Step C.3**: Connect the part properties of the aggregated system by means of proxy ports and connectors, where the proxy ports represent the flows of energy, material, and information into or out of the corresponding CPSs, respectively.

Following the sequence of the CPS-FAS method results in a single model that contains the functional architecture of all CPSs and their interconnections. Modeling CPSs as well as their joint behavior and functional architecture in an aggregated system includes models of multiple systems since it is unlikely that all systems are modeled in a single model. Individual models with a link to the aggregated system model allow usage of CPS models with a well-defined system boundary for subsequent development steps. For this reason, a model structure for the aggregated system model is suggested. Each CPS model should import the FAS profile and the CPS-FAS profile so that all required stereotypes for modeling according to the CPS-FAS method are available. In addition, each CPS model should import an interfaces model. Domain blocks are used in the interfaces model for specifying pins and activity parameter nodes in CPS activities as well as flow properties in the functional architectures. The interfaces model should be imported in an editable way so that domain blocks are synchronized in all CPS models. All CPS models are imported into an aggregated system model. Since the FAS profile and CPS-FAS profile as well as the interfaces model are imported into all CPS models, they are also part of the aggregated system model. If architecture frameworks such as UAF[36] are used, the functional architecture developed using the CPS-FAS method can be integrated at the behavioral view level. Since the CPS-FAS method uses the widely used SysML, the prerequisites for integration into the SysML-based UAF, for example, are in place and can be investigated in future work.

Verification and validation activities evaluate development results and respective integrated systems. Verification determines "whether the finished product satisfies the specific requirements for which it was built"[37,38] and thus answers the question "Was the product built (written, built, coded, assembled, and integrated) correctly"?[37,38] Examining if all requirements, refining use cases, and use case activities are met by the identified functional groups, blocks, and interfaces can verify the designed functional architecture. Traceability can be established by so-called satisfy relationships between these model elements. For verification purposes, dependency matrices, relation maps, and queries can determine unsatisfied requirements. For more detailed verification purposes, all requirements and their satisfy relationships can be checked systematically.

The satisfaction of actors is checked in validation activities and answers the question "Was the right product built"?[37,38] The functional architecture can be validated by allocation of the functional blocks and interfaces to Rapid Prototyping hardware, as demonstrated for a seat occupancy detection system for aircraft cabins by Lamm et al.[39] In addition, elements of the functional architecture can be presented to and discussed with human actors as suggested by Weilkiens et al.[21]

4 | AUTOMATION FOR FUNCTIONAL CPS ARCHITECTURE DEVELOPMENT

Model-based development of functional architectures for CPSs according to the CPS-FAS method contains many steps that can be automated to reduce modeling effort and to increase added value of the method. Potentials for automation would be the same as for the traditional FAS method. They are, for example, automatic diagram initialization and functional blocks creation based on functional groups. The FAS plugin for the modeling tools *MagicDraw* and *Cameo Systems Modeler* by No Magic / Dassault Systèmes supports step A.3 of the CPS-FAS method with SysML [*cf*. Section 3.2]. Step A.3 refers to the conventional FAS method that is described in Section 2.2. The plugin provides the following functions for automated support of method application according to the precondition and process steps from Section 2.2:

- Each use case is described by use case activities that are modeled in SysML activity diagrams. The plugin creates use case activity diagrams for each use case. An activity partition with «*I/O*» stereotype applied is created for each actor and external system that is connected to the respective use case [*cf*. precondition].
- The plugin creates a functional group for each «*I/O*» partition from use case activity diagrams [*cf*. step #1].
- The systems engineer adds functional groups to the model that already contains a system and I/O functional groups. Subsequently, the plugin creates a functional block for each functional group. The functional block has the same name as the functional group [*cf*. step #2].
- The plugin creates functional interfaces between functional blocks. A functional interface is created if there is an object flow between two use case activities that are traced to different functional groups. Consequently, the functional interface is created between the functional blocks that are based on these functional groups [*cf*. step #2].
- A functional system context displays the system and its interfaces to actors and external systems. The plugin creates a functional system context that represents the system, its actors, and external systems

as well as interfaces between them. Information about actors and external systems is taken from use case diagrams [*cf*. part (c) and (d) in Figure 4]. The system, its actors, external systems, and their connections are represented by properties and connectors in an internal block diagram [*cf*. step #2].

- The plugin creates an internal block diagram for the functional architecture. Part properties in the internal block diagram are typed by functional blocks and connected to each other by means of connectors [*cf*. step #2].
- A behavior view presents the functional system behavior and is created by the plugin. Use case activities are presented in activity diagrams and are allocated to functional blocks using activity partitions.

A "Do all creations at once" function of the plugin bundles the functions for creating functional blocks, functional interfaces, functional system context, and connectors between actors and systems in the context. It will allow to join all automatically executable parts of the procedure into one quick process step with minimum user interactions for a fast prediction of results, but without a possibility to influence all individual steps of applying the method. The plugin and additional information are available at the file hosting service SourceForge in the project *FAS for MagicDraw*. There are also templates for modeling functional system architectures according to the conventional FAS method and the new CPS-FAS method as well as profiles containing the required stereotypes.

5 | APPLICATION AND RESULTS

The CPS-FAS method was applied in various projects for the development of functional CPS architectures. In the following, CPS-FAS method application and results of four illustrative projects are presented. In a first example project, a boarding time prediction system, both the application and the results are described in more detail. In three further projects, a smart inflight catering trolley,[17] a document archive and library system,[40] and a hearing healthcare system[41] mainly the results will be briefly quoted.

The boarding time prediction system consists of a mobile application for the flight crew and an additional back-end for estimating and displaying the expected remaining aircraft boarding time. It communicates with the departure control system (DCS), a sensor floor, and passenger seats. These systems and the boarding time prediction system form an aggregated system as shown in the context in Figure 10.

DCS manage information for passenger check-in, printing of boarding cards, baggage acceptance, boarding, and load control. A sensor floor detects passenger positions and movements in the aisle of an aircraft. Passenger seats are equipped with a seat occupancy detection system for identifying their occupancy status. Seat occupancy detection systems are also used by seat belt warning systems in passenger cars. Schultz[42] improves the prediction of passenger boarding processes during aircraft turnaround with seat occupancy detection systems. Functional architecture development of a seat occupancy detection system for boarding simulation validation is presented by Lamm et al.[39]

The development of the boarding time prediction system begins with identification of passengers and cabin crew as well as boarding time prediction system, seat, sensor floor, and departure control system as relevant actors and CPSs in the aggregated system context [*cf*. part (a) in Figure 4]. Use cases for the aggregated system [*cf*. part (b) in Figure 4] encompass, *inter alia*, "Support Boarding" and "Predict Remaining Boarding Time." Joint behavior of the systems to be integrated into the aggregated system is modeled in a SysML activity diagram. The diagram section in Figure 11 belongs to the "Predict Remaining Boarding Time" use case on aggregated system level and shows DCS and the boarding time prediction system behaviors modeled as call behavior actions.

These call behavior actions on aggregated system level are the basis for identifying machine and human use cases on CPS level [*cf*. Section 3.2]. Figure 12 shows a use case diagram section for the boarding time prediction system. The call behavior action ": Predict Remaining Boarding Time—Human" from the aggregated system-level activity diagram in Figure 11 leads to the CPS use case with the same name presented in the CPS use case diagram in Figure 12.

The human actor *Cabin Crew* is interested in a visualization of the current boarding status to determine aisle congestions and possible conflicts between passengers as well as in remaining boarding time predictions. Therefore, information about seat occupancy, aisle occupancy, and seat booking status is required from information sources. If there were other CPSs with interest in the remaining boarding time, the remaining boarding time prediction would be an example for a machine use case. Activities explain the use cases in more detail, are linked to the use cases, and represent the called behavior of call behavior actions that describe joint system behavior [*cf*. Figure 11]. Figure 13 shows the activity diagram related to the *Predict Remaining Boarding Time—Human* use case. The activity diagram illustrates the *Predict Remaining Boarding Time—Human* activity that is called by the call behavior action of the same name in Figure 11.

The activity diagrams are divided into activity partitions. Activity partitions with «*I/O*» stereotype collect functions interfacing to a human actor or external system, that is, *Booking Status Information Source*, *Aisle Occupancy Information Source*, *Seat Occupancy Information Source*, and *Cabin Crew*. The activity partition without stereotypes applied contains actions performed by the system independent of an actor. The activity diagram in Figure 13 shows that information about seat booking, seat occupancy, and aisle occupancy is obtained at system interfaces. This information is collected, remaining boarding time is predicted, and the information is rendered to provide visual information to the *Cabin Crew*. Next, activities from activity diagrams are traced to functional groups. There is one functional group per I/O interface and additional functional groups are added depending on

[1] The figure shows only a portion of the acitivity diagram. The object flows to the : *Seat Occupancy Information* and : *Aisle Occupancy Information* pins of the : *Predict Remaining Boarding Time - Human* action have their origin in other actions that are not shown in this section.

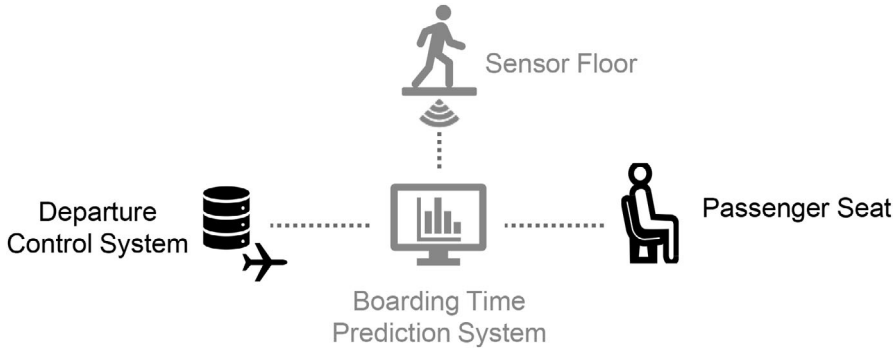

FIGURE 10 Context of the boarding time prediction system with legacy systems (black) and new systems (grey).

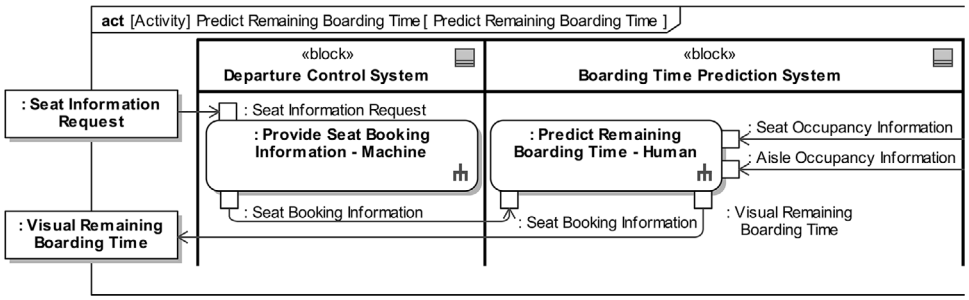

FIGURE 11 Section of joint behavior of multiple CPSs for the aggregated system use case "Predict Remaining Boarding Time"[1] .

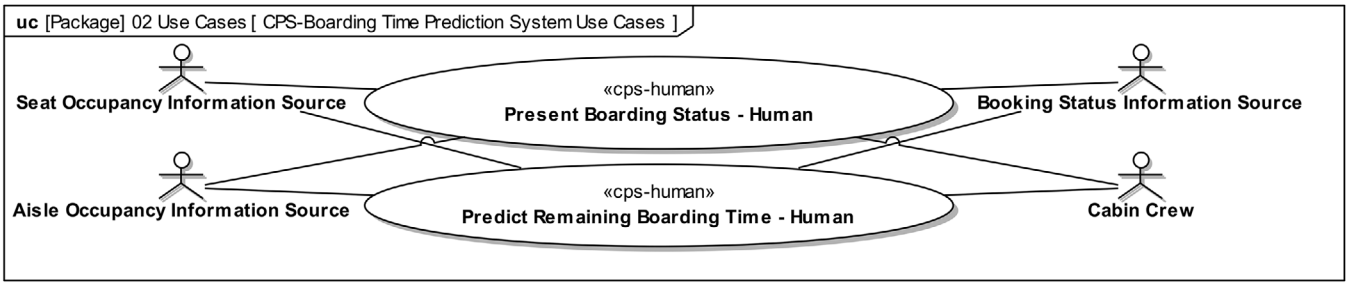

FIGURE 12 CPS use case diagram section for the boarding time prediction system.

reasonable commonality of activities. Besides the aforementioned I/O interfaces, *Information Collection* and *Remaining Boarding Time Estimation* are functional groups for the boarding time prediction system. Functional groups are the basis for the creation of functional blocks. These functional blocks have proxy ports and flow properties in correspondence to action pins and object flows in the previously modeled activity diagrams. Part properties are typed by functional blocks and form a functional architecture, as shown in Figure 14.

The external system *Aisle Occupancy Information Source* is connected to the part property with the type *I/O Aisle Occupancy Information Source*. The output of this part property and the part properties with the types *I/O Booking Status Information Source* and *I/O Seat Occupancy Information Source* are connected to input proxy ports of the part property with the type *Information Collection*, that is, based on the functional group *Information Collection*. Functional architectures for remaining CPSs of the aggregated system are developed in the same way.

Observations: In the past, the functional architecture for the boarding time prediction system was designed using the conventional FAS method. Application of the FAS method resulted in many discussions about the system boundary and the question what systems are part of the boarding time prediction system. On the one hand, the whole boarding process should be covered by the use case activities so that activities by the aisle occupancy information source, the booking information source, and the seat occupancy information source were included. On the other hand, these systems were already in place and were not part of the boarding time prediction system, but of a higher-level aggregated system so that they should not be part of the functional architecture of the boarding time prediction system.

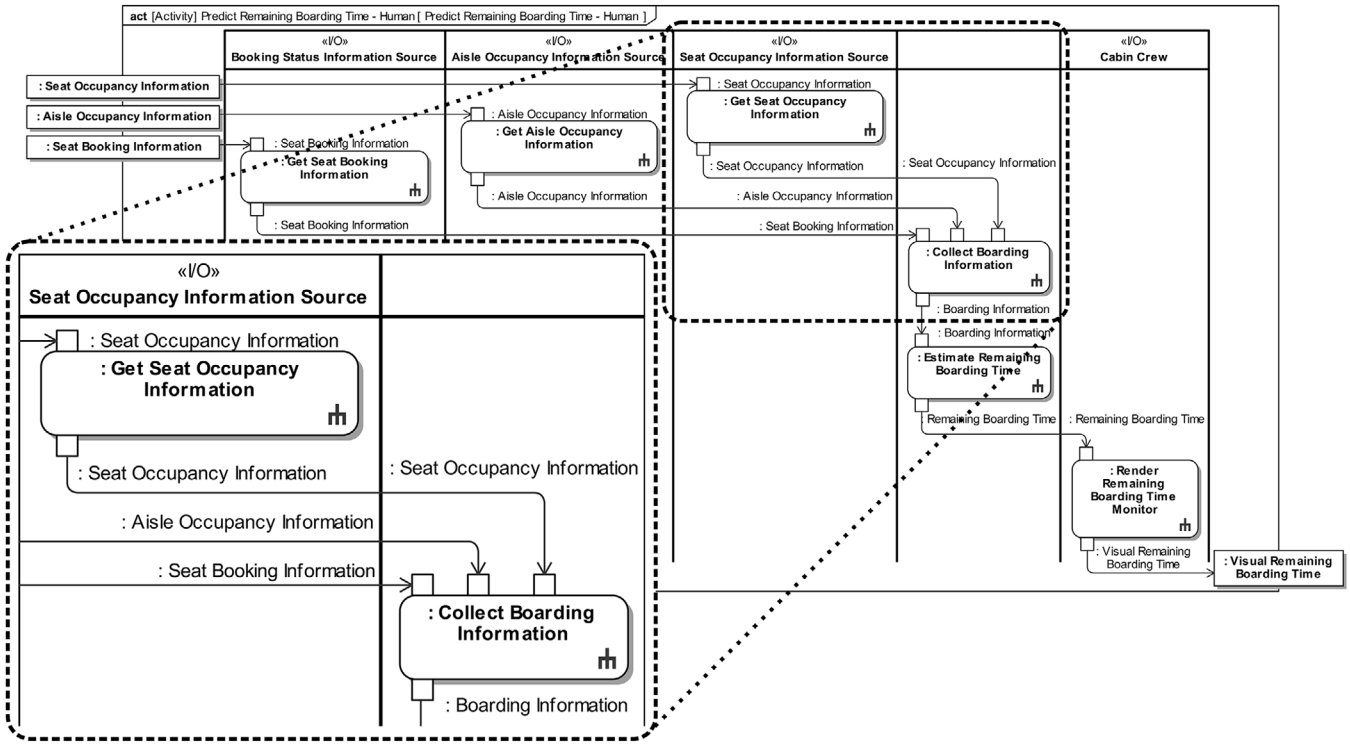

FIGURE 13 Activity diagram for predict remaining boarding time—Human use case for the boarding time prediction system.

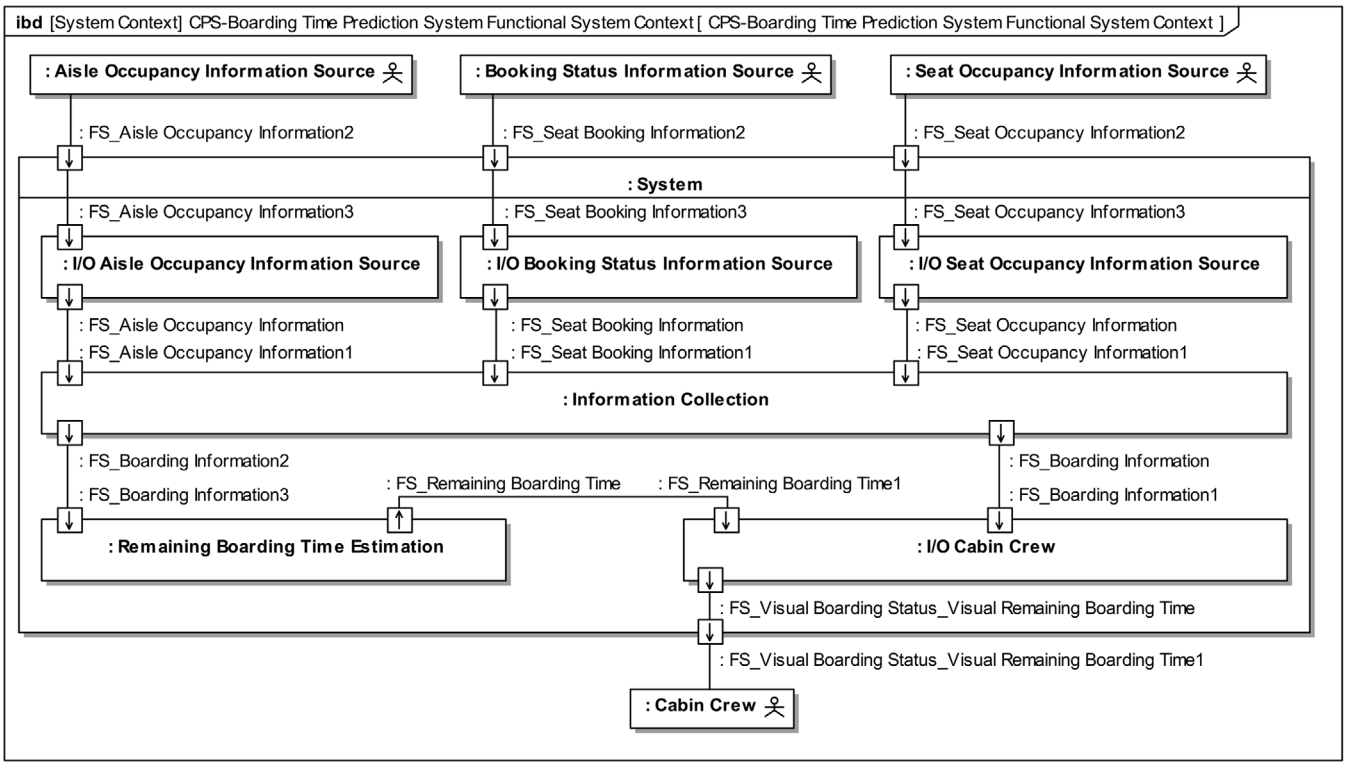

FIGURE 14 Functional architecture for the boarding time prediction system.

Consequently, the system boundary of the use case activities was larger than the system boundary of the functional architecture. The functional architecture of the boarding time prediction system developed according to the FAS method only considered the system functions and one interface to the environment so that the joint action of the functions in the aggregated system could not be analyzed. The CPS-FAS method improves the situation significantly. It allows the analysis of the whole process and the design of a functional architecture on aggregated system level. The functions are allocated to constituent CPSs so that the context of functional CPS architectures is transparent and interfaces to functional architectures of other CPSs are visible, thus increasing the understanding of and the exchange between development teams. In addition, the FAS plugin supports the architecture design of the boarding time prediction system and automates the time-consuming steps of the CPS-FAS method. Practice effort for plugin usage paid off because of less required modeling time and fewer modeling errors due to automation of error-prone and tedious modeling processes, for example, functional block initialization after functional grouping. However, CPS-FAS modeling can also be done without the plugin.

In this subsequent example, results from the inflight catering trolley development are given. Inflight catering trolleys are used for catering storage and transport. The smart trolley[17] extends the trolley functionality and is able to detect the trolley content and provide inventory reports to other CPSs for more accurate provision of future inflight catering. Before development of the CPS-FAS method, the design of the smart trolley was conducted according to the conventional FAS method. Consequently, use cases were described on system level. Use case activities contained two partitions that represented activities with a relation to system interfaces in an I/O partition and all other activities in a system partition. The functional architecture contained one functional I/O group that was connected to all external systems and human actors and did not differentiate between different flow types, for example, data flow to a touch display for communication with the cabin crew and to the cabin management system that collected the inventory data from multiple trolleys on board of an aircraft.[17] Development according to the CPS-FAS method allows the identification of use cases on aggregated system level and the allocation of aggregated system use case activities to CPSs that are interacting in the aggregated system, for example, the crew display and cabin management system.

Observations: It can be detected that utilization of the CPS-FAS method considers interaction in an aggregated system in more detail and allows systematic distribution of functions on aggregated system level among the interacting CPSs. On CPS level, the CPS-FAS method distinguishes different interfaces to external systems and human actors in use case activities through utilization of multiple I/O partitions. Accordingly, the functional architecture contains multiple I/O functional groups and represents system interfaces in more detail. Application of the conventional FAS method resulted in one functional I/O block that represented multiple interfaces to different systems, that is, the crew display and the cabin management system. This I/O block had to be divided manually into more accurate functional I/O blocks. In conclusion, application of the CPS-FAS method to the development of a smart inflight catering trolley resulted in improved functional considerations and distribution at aggregated system level and reduced the effort for interface modeling in functional CPS architectures.

In another project, the CPS-FAS method was used for the design of document archive and library systems.[40] At the Centre for the Study of Manuscript Cultures (CSMC) at the Universität Hamburg a steadily growing number of autonomously developed archive and library systems, that have been developed as CPSs, are aggregated into a centrally accessible information system. For the development of the aggregated system the open-source web-based database management system HEURIST[43] was used. While the functional architectures of these CPSs were similar based on the tool selection, the challenge was to integrate the CPSs into an aggregated system. In this context, the aggregated system is a federated data management system that provides a federated search. The federated search function appears in the aggregated system, but should also be available to all CPSs. The CPS-FAS method has helped to focus on communication aspects during the development process. The aggregated system was specified as a federated database system to provide federated searches and tested through simulations with real database interactions, as shown by Melzer et al.[40] In addition to communication functions, new functions such as the federated search were added. However, these also had an impact on the existing search function of the individual CPSs and consequently on their functional architecture. The changes mainly concerned the interfaces and the processing of new inputs and outputs.

Observations: By systematically developing the functional architectures with the CPS-FAS method, it was possible to work in a targeted manner making the aggregated system extensible by enabling the simple integration of additional CPSs. A CPS does not necessarily have to be developed with HEURIST. The integration of other autonomously developed archive and library systems was also considered. The development of a physical architecture based on the functional architecture is currently being carried out in further work.

In addition, the CPS-FAS method supported the PASTOR project[41] for the development of novel CPSs in the form of a hearing instrument and a diagnostic software for implementing a new concept that was expected to provide relief to certain individuals with tinnitus. Both CPSs are independent from each other and collaborate during a hearing instrument fitting use case.

Observations: The application of the CPS-FAS method improved the understanding of involved development parties about their role in the process, desired functions they had to implement, and the purpose of both, the aggregated system and the included CPSs.[41]

Application of the model-based CPS-FAS method to the development of a boarding time prediction system, a smart inflight catering trolley,[17] a federated document archive and library system,[40] and a hearing healthcare system[41] is used in the following to review the achievement of the objectives. Other project-specific CPS-FAS application results will be the subject of further publications (unpublished results of ongoing work). The following list comments on the

achievement of the objectives from Section 2.1 and compares the CPS-FAS method to the non-extended FAS method.

1. The method combines the existing FAS approach with an approach for treating CPSs both individually and in interaction in an aggregated system. Identifying an aggregated system context, aggregated system use cases, and their use case activities enhances the existing FAS method. Use cases resulting from aggregated system use case activities are the starting point for application of the CPS-FAS method that treats CPSs both individually and in interaction in an aggregated system. Functional architectures are derived for each CPS in the aggregated system.
2. The interconnectivity of CPSs is considered by introducing individual I/O partitions for each actor and external system in use case activity diagrams of CPSs. Functional CPS architectures can be integrated in order to obtain a functional architecture of the aggregated system according to steps C.1, C.2, and C.3 in Section 3.2. The FAS method considers system interfaces in a single I/O partition so that less distinction between interfaces is made in functional architectures.
3. Data and information exchanges between CPSs and CPS functions are specified in the model by means of domain blocks. Domain blocks typify pins of use case activities [*cf*. Figure 4(b,c,d) and Figure 11] and flow properties of interface blocks. These interface blocks specify proxy ports of functional blocks [*cf*. Figure 4(f) and Figure 14].
4. Redundant functions within a CPS are identified during functional grouping. Required aggregated system functionality is distributed by allocation of aggregated system use case activities to CPSs. However, similar functions in different CPSs may exist. The application showed that redundant CPS functions are sometimes necessary, for example, all CPSs require a power supply and this function cannot be allocated to a single CPS and provided to other CPSs. Identifying unnecessary function duplicates in different CPSs can be part of future research. A possible approach is functional grouping on the CPS level that should consider activities from other CPSs as well. Then, similar functions can be identified and allocated to only one CPS. Especially the unique feature of data—availability at multiple places at the same time—offers potentials for function sharing of CPSs when considering computation tasks. The FAS method supports the identification of non-redundant functions for a self-contained system with few interfaces but does not address the possible redundancies of functions at CPS level.

The distinction between human and machine use cases as well as the introduction of actor and external system-specific I/O partitions in activity diagrams extends the FAS method for developing and modeling functional architectures for CPSs that are characterized by interfaces to multiple systems. Modeling joint behavior of multiple CPSs is supported by means of project usages in *MagicDraw* and *Cameo Systems Modeler*. Project usage should be accompanied by agreements between modelers to ensure model compatibility. A shared library of commonly used types or domain objects will be needed.

6 | SUMMARY, CONCLUSION, AND OUTLOOK

The FAS method supports functional architecture development for self-contained systems with a limited number of interfaces to actors and other systems. The CPS-FAS method enables developing functional architectures and creating functional models for individual CPSs as well as for aggregated systems comprising multiple CPSs. The method considers interconnectivity of CPSs by specifying stereotypes and a specialization scheme for use cases. The distinction of use cases supports the analyst who will need to cover machine-to-machine communication during use case analysis. The method also addresses possible redundancy of functions across an aggregated system. Therefore, one representation of each function is selected and modeled as a functional group after analysis of use case activities.

Validation of functional architectures developed using the CPS-FAS method is not in the scope of this publication. Lamm et al.[39] propose an approach for validation of functional architectures. They introduce a function-oriented system demonstrator in which functional blocks are implemented directly in hardware and software. This way the detailed design of the system is skipped and a cost-efficient demonstrator is implemented. Application of this approach to the CPS-FAS method could be investigated to support the validation of functional CPS architectures.

Modeling CPS machine use cases, activities, and deriving CPS functions provided to other systems requires a prior vision of CPS cooperation. It also requires the creation of a shared library of types of domain objects for specification of exchanged information, energy, and material. Developing complex and more extensive system networks such as a System of Systems requires a top-down approach for identifying system network functions and allocating these functions to systems.

For future projects, modeling according to a model structure that is the same across all modeled CPSs is recommended. Therefore, the model template and profiles used for the presented work have been made available via the FAS profile for *MagicDraw*, which is a part of the FAS plugin.[15] The model template provides a model structure and the profiles contain required stereotypes for CPS-FAS application such as «cps-human» and «cps-machine». The given version of the FAS plugin also contains automation for generating ports that are required for interconnecting functional architectures of different CPSs. The corresponding automation has been developed as a side product of the presented work, providing the sample implementation mentioned in Section 4.

The application of the CPS-FAS method to more functional architecture development projects and the quantitative evaluation, for example, with respect to redundancy reduction, is subject to further research and publications. For validation of the designed functional architecture, the behavior of the functional blocks could be modeled, for example, in activities and in state machines, so that an executable model could demonstrate the behavior of the identified functions. Furthermore, the use of SysML and contemporary MBSE tools offers the advantage of automated validation according to validation rules. As an

example, these rules could concern the interfaces between the functional blocks that should be of the same type. It is expected that the presented modeling approach will not only work for aggregated systems consisting of few CPSs, but also for modeling more complex aggregated systems, like SoS. Future work will refine the method and study the corresponding functional modeling of SoS. While this paper has shown a way to derive the functional architecture of an aggregated system from the functional architectures of the CPSs it consists of, a functional architecture in the SoS case would be derived from the functional blocks of the SoS constituents. However, it would also be possible to derive it using the FAS method on the use case activities of the SoS. It needs to be investigated how the described two different functional architectures of the SoS relate to each other. In addition, vulnerabilities may occur as CPSs being part of an aggregated system have multiple interfaces for the exchange of data and information. The combination of the CPS-FAS method with model-based System Security Engineering approaches[44–46] is another topic for future research.

The CPS-FAS method integrates CPS models into an aggregated system model. Individual models in the application example have been developed using the modeling tool *Cameo Systems Modeler*. Extensive aggregated systems may comprise CPSs that are modeled using different modeling tools and thus require model integration. Integration of models from different modeling tools is another field of future research.[47]

ACKNOWLEDGMENTS

We thank Abdallah Altalmas for his valuable work on the FAS plugin for *MagicDraw* and *Cameo Systems Modeler* and the sample implementation of additional functions supporting the development of functional architectures for aggregated systems. We like to thank Markus Walker and Stephan Roth for the permission to use advance information from work on the second edition of the Model-Based System Architecture book, presented in Figure 3 and Table 1. This work was supported by the LuFo V-2 project "Connected Cabin—Information Centric Operation of Future Conntected Cabin (ConCabInO)", funded by the Federal Ministry for Economic Affairs and Energy based on the decision by the German Bundestag under the project number 20K1510D. The PASTOR project was co-financed by Innosuisse under the project number 44423.1 IP-LS.

Open access funding enabled and organized by Projekt DEAL.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the corresponding author upon reasonable request.

ORCID

Oliver C. Eichmann https://orcid.org/0000-0003-2064-3436
Jesko G. Lamm https://orcid.org/0000-0002-6425-5884
Sylvia Melzer https://orcid.org/0000-0002-0144-5429
Tim Weilkiens https://orcid.org/0000-0002-5633-5294
Ralf God https://orcid.org/0000-0002-3272-8907

REFERENCES

1. Talcott C. Cyber-physical systems and events. In: Wirsing M, Banâtre J-P, Hölzl M, Rauschmayer A, eds. *Software-Intensive Systems and New Computing Paradigms: Challenges and Visions*. Springer; 2008:101-115.
2. Isasa JAE, Larsen PG, Hansen FO. A holistic approach to energy-aware design of cyber-physical systems. *Int J Embedded Syst*. 2017;9(3):283-295.
3. Lee EA. The past, present and future of cyber-physical systems: a focus on models. *Sensors (Basel)*. 2015;15(3):4837-4869. doi:10.3390/s150304837
4. Cyber-physical systems — merging the physical and virtual worlds. In: *Cyber-physical systems: Driving force for innovation in mobility, health, energy and production*. Springer; 2011:15-21.
5. Henshaw M. Systems of systems, cyber-physical systems, the Internet-Of-Things…Whatever next? *Insight*. 2016;19(3):51-54.
6. Oppermann S, Hoth J, God R, Project Report for the Research Project "Lernendes Galley-Catering-System (LGCS): Teilvorhaben: Entwurfsmethoden und Architekturen für lernende, künstlich intelligente Systeme in der Kabine".
7. Baines TS, Lightfoot HW, Evans S, et al. State-of-the-art in product-service systems. *Proc Inst Mech Eng Part B J Eng Manuf* 2007;221(10):1543-1552.
8. Clark JO, System of systems engineering and family of systems engineering from a standards perspective. In: 2008 IEEE International Conference on System of Systems Engineering (SoSE): Monterey, California, USA, 2 - 4 June 2008. IEEE; 2008:1-6.
9. Flanigan D, Brouse P. System of systems requirements capacity allocation. *Proc Comput Sci*. 2012;8:112-117.
10. Maier MW. Architecting principles for systems-of-systems. *Syst Eng*. 1998;1(4):267-284.
11. Lamm JG, Weilkiens T. Method for deriving functional architectures from use cases. *Syst Eng*. 2014;17(2):225-236.
12. Fernandez JL, Hernandez C. *Practical model-based systems engineering*. Artech house; 2019.
13. Friedenthal S, Moore A, Steiner R. *A practical guide to SysML*. Elsevier Science; 2014.
14. Lamm JG, Weilkiens T. Happy Birthday! 5 Jahre Funktionale Architekturen nach FAS. *Tagungsband zum Tag des Systems Engineering (Hrsg.: Chr. Muggeo, SO Schulze), S*. 2015:59-68.
15. Gesellschaft für Systems Engineering e.V. FAS Plugin for MagicDraw. Accessed 2021. https://sourceforge.net/projects/fas4md/
16. Melzer S, Wittke U, God R, Sichere Luftfracht-Transportkette: Modellbasierte Architektur- und Lösungsspezifikation: TUHH Final Report of the Research Project Sichere Luftfracht-Transportkette: Konzepte, Strategien und Technologien für Sichere und Effiziente Luftfracht-Transportketten (SiLuFra) SiFo BMBF; 2017.
17. Melzer S, God R, Connected Cabin—Spezifikation und Integration Cyber-Physischer Betriebs- und Geschäftsprozesse: TUHH Final Report of the Research Project 'Information Centric Operation of Future Connected Cabin (ConCabInO)' LuFo V-2 BMWi; 2020.
18. Engell S, Paulen R, Reniers MA, Sonntag C, Thompson H, Core research and innovation areas in cyber-physical systems of systems. In: International Workshop on Design, Modeling, and Evaluation of Cyber Physical Systems; 2015:40-55.
19. Colombo AW, Karnouskos S, Kaynak O, Shi Y, Yin S. Industrial cyber-physical systems: a backbone of the fourth industrial revolution. *EEE Ind Electron Mag*. 2017;11(1):6-16. doi:10.1109/mie.2017.2648857
20. McChrystal SA. *Team of teams: new rules of engagement for a complex world*. Portfolio/Penguin; 2015.
21. Weilkiens T, Lamm JG, Roth S, Walker M. *Model-based system architecture*. 2nd ed. Wiley; 2022.
22. Weilkiens T, *Systems Engineering mit SysML/UML: Anforderungen, Analyse, Architektur*. 3., überarbeitete und aktualisierte Auflage. dpunkt.verlag; Ciando; 2014.

23. Lamm JG, Weilkiens T. Funktionale Architekturen in SysML. In: Maurer M, Schulze S-O, eds. *Tagungsband zum Tag des Systems Engineering*. Carl Hanser Verlag; 2010:109-118.
24. Korff A, Lamm JG, Weilkiens T. Werkzeuge für den Schmied Funktionaler Architekturen. In: Maurer M, Schulze S-O, eds. *Tagungsband zum Tag des Systems Engineerings, Hamburg, 9.-11. November 2011*. Carl Hanser Verlag; 2011:3-12.
25. Fernandez-Sánchez JL, García-Muñoz J, Gómez-Pérez JP. La ingeniería de sistemas y su aplicación a un vehículo aéreo no tripulado. *DYNA-Ingeniería e Industria*. 2012;87(4):456-466.
26. Binder C, Draxler D, Neureiter C, Lastro G, Towards a Model-Centric Approach for developing Functional Architectures in Industry 4.0 Systems. In: 5th IEEE International Symposium on Systems Engineering (ISSE); 2019.
27. Eichmann OC, Lamm JG, Melzer S, Weilkiens T, God R, CPS-FAS method: deriving functional architectures for cyber-physical systems using interconnectable models. Tag Des Systems Engineering 2020 Online Conference; 2020.
28. Huynh TV, Osmundson JS, An integrated systems engineering methodology for analyzing systems of systems architectures. In: Asia-Pacific Systems Engineering Conference, Singapore; 2007:77.
29. Rao M, Ramakrishnan S, Dagli C. Modeling and simulation of net centric system of systems using systems modeling language and colored Petri-nets: a demonstration using the global earth observation system of systems. *Syst Eng*. 2008;11(3):203-220.
30. Gezgin T, Etzien C, Henkler S, Rettberg A, Towards a rigorous modeling formalism for systems of systems. In: 2012 IEEE 15th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops; 2012:204-211.
31. Canedo A, Schwarzenbach E, Faruque MAA, Context-sensitive synthesis of executable functional models of cyber-physical systems. In: 2013 ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS); 2013:99-108.
32. Bryans J, Fitzgerald J, Payne R, Kristensen K. 2.2. 2 maintaining emergence in systems of systems integration: a contractual approach using SysML. *INCOSE International Symposium*. 2014;24:166-181.
33. Ingram C, Fitzgerald J, Holt J, Plat N. Integrating an upgraded constituent system in a system of systems: a SysML case study. *INCOSE Int Symp*. 2015;25:1193-1208.
34. Lollini P, Mori M, Babu A, Bouchenak S, AMADEOS SysML profile for SoS conceptual modeling. In: *Cyber-physical systems of systems*. Springer; 2016:97-127.
35. Huang P, Jiang K, Guan C, Du D, Towards modeling cyber-physical systems with SysML/MARTE/pCCSL. In: 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC). 2018;Vol. 1:264-269.
36. Object Management Group. Unified Architecture Framework Modeling Language (UAFML); 2022. https://www.omg.org/spec/UAF/1.2
37. VDI-Richtlinie 2206. *Entwicklungsmethodik für Mechatronische Systeme*. Beuth-Verlag; 2004.
38. Engel A. *Verification, Validation, and Testing of Engineered Systems*. Wiley; 2010.
39. Lamm JG, Melzer S, Hintze H, God R, Ein Modellbasiertes Vorgehen zur Verifikation und Validierung von Funktionalen Architekturen. In: Schulze, Tschirner et al. (Hg.) 2019 - Tag des Systems Engineering: 43-52.
40. Melzer S, Thiemann S, Möller R, Modeling and simulating federated databases for early validation of federated searches using the broker-based SysML toolbox. In: 2021 IEEE International Systems Conference (SysCon); 2021:1-6.
41. Lamm JG, Hu S, Hockley NS, Caversaccio M, WimmerW. MBSE case report: successful functional allocation in a novel scenario of hearing health care. Tag des Systems Engineering 2022: Tagungsband Paderborn, 16.-18. November 2022:176-180.
42. Schultz M. *The Seat Interference Potential as an Indicator for the Aircraft Boarding Progress*; 2017.
43. HEURIST. A unique solution to the data management needs of Humanities researchers. Accessed 2021. https://heuristnetwork.org/
44. Hintze H, God R. Using model-based security engineering in the development of complex aircraft cabin systems. *SAE Int J Aerosp*. 2015;8(1):89-96. doi:10.4271/2015-01-2445
45. Hintze H, Speichert JP, God R. The risk matrix as an integral part of a SysML-based security engineering approach in the development of complex aircraft cabin systems. In: 2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC). IEEE; 2018:1-9.
46. Nguyen PH, Ali S, Yue T. Model-based security engineering for cyber-physical systems: a systematic mapping study. *Inf Softw Technol*. 2017;83:116-135. doi:10.1016/j.infsof.2016.11.004
47. Dungern von O, Alt O. Specification Integration Facility—Wozu braucht man SpecIF neben SysML? Tag Des Systems Engineering 2020 Online Conference; 2020.

How to cite this article: Eichmann OC, Lamm JG, Melzer S, Weilkiens T, God R. Development of functional architectures for cyber-physical systems using interconnectable models. *Systems Engineering*. 2024;1-19.
https://doi.org/10.1002/sys.21761