Model-based Development of a System of Systems Using Unified Architecture Framework (UAF): A Case Study

Oliver C. Eichmann
Institute of Aircraft Cabin Systems
Hamburg University of Technology
Hamburg, Germany
oliver.eichmann@tuhh.de

Sylvia Melzer
Institute of Aircraft Cabin Systems
Hamburg University of Technology
Hamburg, Germany
sylvia.melzer@tuhh.de

Ralf God
Institute of Aircraft Cabin Systems
Hamburg University of Technology
Hamburg, Germany
ralf.god@tuhh.de

***Abstract*—In the development of safety- and security-relevant systems the V-model is established providing verification possibilities at each development stage. Usually, methods and tools of Model-based Systems Engineering (MBSE) are used in combination with the V-model for the development of single and self-contained complex systems. Nowadays, ubiquitous connectivity leads to a high degree of communication between systems enabling their cooperation for the provision of new services in a so-called System of Systems (SoS). In contrast to conventional systems engineering new methods and tools are required for service enabling SoS. In order to fulfill requirements of System of Systems Engineering (SoSE) the Object Management Group (OMG) developed the Unified Architecture Framework (UAF) for representation of enterprise architecture. This paper presents an approach for model-based development of SoS using UAF according to the V-model. In addition, an application of this new method shows differences between single system and SoS development methods.**

***Index Terms*—Cyber-Physical Systems, System of Systems, Hardware Integration, V-model, Message Broker**

I. Introduction

In the past, companies developed and sold mainly isolated systems taking into account a limited number of interfaces to other systems as displayed in Fig. 1 below. Nowadays, increased connectivity drives companies to digitize their processes and to offer services instead of physical products. Increased connectivity of systems enables communication between systems, resulting in much more information exchange and allowing new capabilities in a so-called *System of Systems* (SoS). The SoS definition "applies to a system-of-interest whose elements are themselves systems; typically these entail large scale interdisciplinary problems with multiple, heterogeneous, distributed systems" [1]. The SoS capabilities are used for service creation by utilizing several connected systems in an SoS as displayed in the upper part of Fig. 1. For this reason, companies must evolve from system suppliers to service providers in order to maintain their market share. Car manufacturing companies are an example for this transformation by changing from car manufacturers to mobility suppliers by offering car sharing services [2]. Car sharing services utilize the systems "car", "booking portal", and "users' smartphones". In the past systems engineering was sufficient for system development and faces challenges in the support of service engineering in SoS today. Therefore, *System of Systems Engineering* (SoSE) often uses previously developed military frameworks like *Department of Defense Architecture Framework* (DoDAF) and *Ministry of Defence Architecture Framework* (MODAF) for defining an SoS architecture [3]. The *Object Management Group* (OMG) developed the *Unified Architecture Framework* (UAF) integrating concepts and elements of DoDAF and MODAF. UAF used in this case study is customized for civil users and enterprise requirements and is based on the *Systems Modeling Language* (SysML) [4].

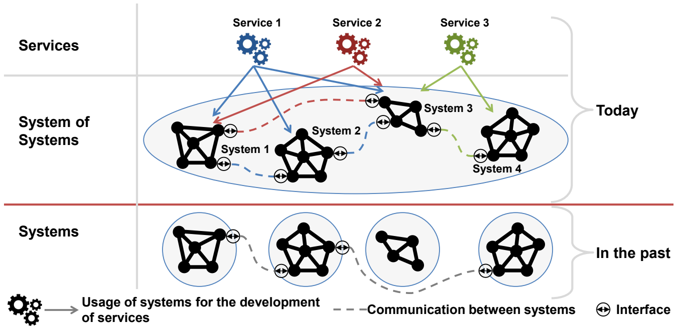


Fig. 1. Context of *System of Systems Engineering* (SoSE).

Large-scale complicated security- and safety-relevant systems are often developed according to the V-model methodology for verification and validation purposes [5]. In verification is examined if achieved results satisfy earlier defined requirements answering the question "Are things done right?". The question "Are the right things done?" is considered in validation by examination if final results fulfill the desired purpose [6]. If SoS entities are also security- and safety-

This work was supported by the LuFo V project "Sichere Luftfracht-Transportkette: Konzepte, Strategien und Technologien für sichere und effiziente Luftfracht-Transportketten (SiLuFra)", funded by the Federal Ministry for Economic Affairs and Energy based on the decision by the German Bundestag.

relevant development according to V-model is suggested. For this reason this paper presents a model-based approach for SoSE by using elements and views of UAF according to the V-model. In order to validate an SoS in the design phase by means of rapid prototyping, early and simple hardware integration is enabled by a broker-based *SysML Toolbox* using the open source message broker RabbitMQ [7], [8].

II. Fundamentals

In this section fundamentals for SoS development using UAF and the V-model are described. Therefore, the first part introduces UAF and the second subsection describes the V-model. The third part presents a hardware integration concept for rapid prototyping using the message broker RabbitMQ.

A. Unified Architecture Framework (UAF)

UAF architecture models provide a means to develop an understanding of complex relationships between organizations and systems in SoS and enable the analysis of these systems in order to ensure that they meet the expectations of the user community [4]. The UAF specification supports frameworks like DoDAF 2.02, MODAF, Security Views from Canadas *Department of National Defense Architecture Framework* (DNDAF), and the *North Atlantic Treaty Organization (NATO) Architecture Framework* (NAF) 3.1 which are used in military communities to develop standardized and consistent architectures. Besides its support for already existing military frameworks UAF aims at providing a framework for civil users in order to support their technical processes by means of systems engineering. In order to meet the needs of companies for developing services in SoS in a model-based way the UAF is based on SysML and contains a set of viewpoints and views for the description of a system for different stakeholders using a predefined architecture [4].

B. V-Model

In aviation industry systems development requires collaboration of many disciplines, *i.e.* mechanical engineering, electrical engineering, and informatics. Systems specification is an important aspect for successful collaboration. Special requirements regarding safety and security lead to development methodologies for early verification and validation of developed systems [9]. Development according to rules and recommendations of the so-called V-model (*cf.* Fig. 15) enables required early verification and validation. The approach starts with elicitation of requirements which are the basis for product validation after development. The design process can be ordered in the stages

- (A) requirements analysis,
- (B) functional analysis,
- (C) high-level design,
- (D) low-level design.

System specification in the formerly mentioned development stages is followed by domain-specific development. Afterwards, properties defined in system design are validated by means of the previously introduced stages in system integration. The overall process is supported by modeling, by model analysis, and finally ends with product delivery [5], [10] and further lifecycle support.

C. Hardware Integration using Message Broker

In the design phase of the systems engineering process *Model-based Systems Engineering* (MBSE) methods and tools enable simulation, validation, system safety and security analyses, and executable specifications. Hardware integration into models allows rapid prototyping already in early stages of the development process and fosters a deep understanding of SoS in the design phase. Rapid prototyping reduces misconceptions, error rate, and consequently error induced costs. Design and implementation of service-oriented systems typically demand easy to use and configurable messaging architectures. A standardized communication approach is the developed broker-based *SysML Toolbox* [11] using the open source message broker software RabbitMQ. The *SysML Toolbox* is applied to the model-based development of SoS and is an implementation of six different messaging paradigms of RabbitMQ into the modeling tool Cameo Systems Modeler.

Cameo Systems Modeler offers the usage of implementation-specific *Unified Modeling Language* (UML) elements, *i.e.* opaque behavior. An opaque behavior element can be used in activity diagrams in order to realize the required communication between systems or services via a RabbitMQ server. The implementation specific model elements can be integrated via drag-and-drop into the SysML model as opaque actions in order to create a communication network easily. Fig. 2 and Fig. 3 exemplarily present the two fundamental opaque actions *sendMessage.bsh* for sending and *receiveMessage.bsh* for receiving messages. The *sendMessage.bsh* element needs a message, a broker configuration, and a queue configuration as input parameters. The *receiveMessage.bsh* element requires broker and queue configurations and delivers the message as output value.

In order to enable communication between hardware and the model a hardware-specific messaging application has to be developed using the RabbitMQ API with the same configurations of the broker-based *SysML Toolbox* as defined in the

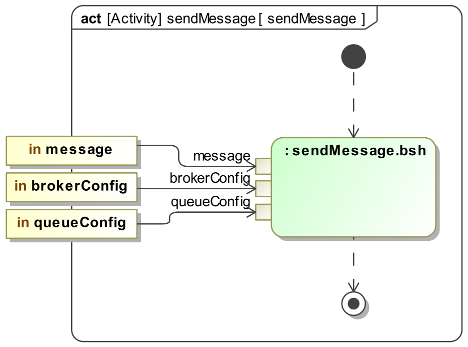


Fig. 2. Created opaque action *sendMessage.bsh*.

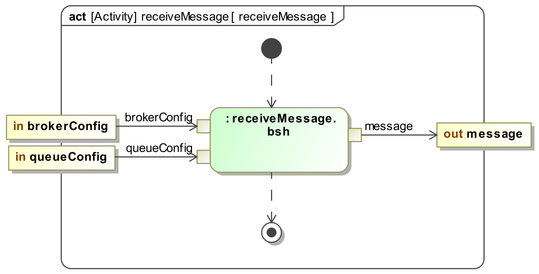


Fig. 3. Created opaque action *receiveMessage.bsh*.

SysML model of an SoS. More technical details are described in [11].

III. Case Study

The purpose of this paper is to investigate an approach for model-based development of SoS using the UAF according to the V-model. In this section the example of a secured air cargo transportation system is used for a case study.

A. Secured Air Cargo Transport Chain

The task of an air cargo transport chain is to transport goods reliably and efficiently from one location to another. The air cargo transport system has always been a target for criminal and terrorist attacks. The first known assassination attempt on commercial aviation with a bomb in a cargo compartment took place in 1933 on a United Airlines flight from Newark to Oakland. Thereby the passenger aircraft crashed over Indiana [12]. In order to avoid exploitation of vulnerabilities in the air transport system, and thus the air cargo transport chain, measures have been continually defined and regulations updated to protect the air cargo transport chain and counteract threats as effectively as possible [13], [14].

Many actors participate in the air cargo transport chain. For the shipment of cargo from Germany (EU) to the world the actors shipper, forwarder, cargo handling agent, ground handling agent, airline, and consignee are involved.

The air cargo transport chain offers different possibilities for securing and transportation options. So-called unsecured cargo can be secured by a check from regulated agents, coming along with additional cost and time effort in the shipment process. For this reason many companies register as known consignors. Known consignors ensure that air cargo is protected against unauthorized access and manipulation on their premises. Therefore, cargo secured and sent by known consignors does not require additional checks by regulated agents [15]. During the SiLuFra (Secured Air Cargo Transport Chain) project [16] a secured air cargo transport chain system, called *Trusted Forwarder System* (TFS), was developed according to the premisies of a single system with two stations (*cf.* Fig. 4). At station 1 a known consignor provides cargo. Afterwards, a forwarder picks-up cargo from station 1 and transports cargo to station 2. At station 2 a cargo handling agent performs a security check, accepts cargo, and a second forwarder transports cargo to the next station. A detailed description of the single system development concept and methods is presented in [16] and [17].

B. Development of a Secured Air Cargo Transport Chain

Within the research project SiLuFra the TFS was developed using the modeling language SysML, the V-model methodology, and the tool Cameo Systems Modeler [17].

The purpose of the trusted forwarder concept is to improve the transport processes of the three actors known consignor, forwarder, and cargo handling agent with respect to efficiency and security. Due to the security aspect in the project the processes of the so-called “actor” threat, *i.e.* a possible attacker, was taken into account in the analysis and development of the system. The determination of all stakeholders took place according to [18] and [19].

1) Requirements Analysis: For the success of the project it is essential that all stakeholder needs are sufficiently met. Therefore, all stakeholders, who have requirements concerning the system or who are interested in the system, must be identified. Afterwards, requirements are identified, queried, documented, and structured corresponding to respective stakeholders. In order to mitigate potential threats, measures have been specified in the form of so-called security requirements using the model-based security engineering process in accordance with [10] for focusing on the security aspects of the air cargo transport chain.

The security requirements for a TFS are “identity check”, “loading secured cargo”, and “transparency of secured cargo transport”. The derived functional requirements are ”person identification”, “cargo identification”, and “eID combination”, *i.e.* coupling person and cargo identities to a token. For representing the requirements so-called requirement diagrams are used.

Use cases describe requirements with respect to the view of each stakeholder. The above mentioned requirements are derived from the use case “secure and efficient transport of cargo”. The use cases are presented in use case diagrams.

2) Functional Analysis: Refinement for each use case is made by defining activities. Activities are presented in activity diagrams. The activities for pick-up cargo at the known con-

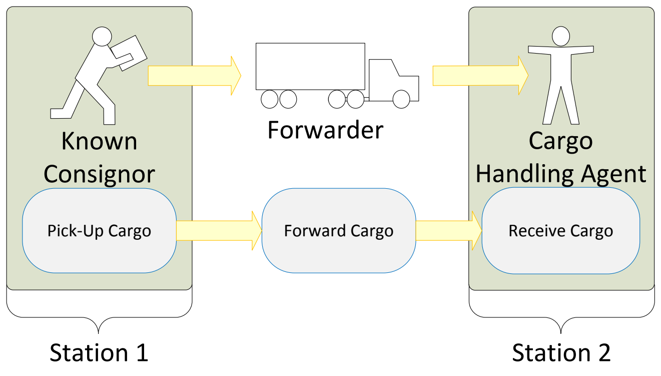


Fig. 4. Secured air cargo transport chain with two stations.

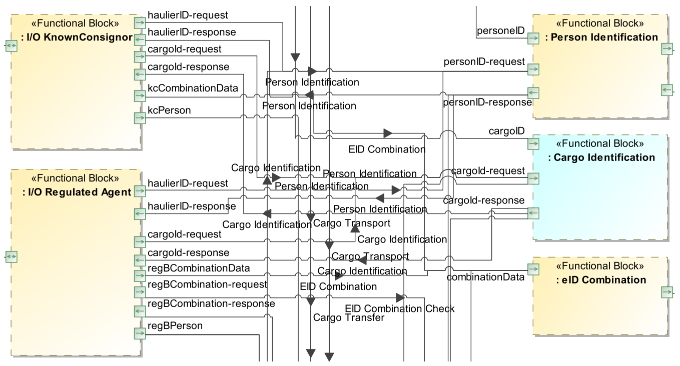


Fig. 5. Representative part of the functional architecture of the Trusted Forwarder System (TFS) with the *part properties* “Person Identification”, “Cargo Identification”, and “eID Combination” as well as “I/O Known Consignor” and “I/O Regulated Agent” modeled in an internal block diagram.

signor are “identify person”, “identify cargo”, “merge person and cargo data”, and “transport cargo”.

To be able to assign an activity to a corresponding function, functions were defined. According to the FAS method [20] each function is presented as a *block*. These blocks are called *functional blocks*. For the use case “transporting cargo securely and efficiently”, among other things, the activity “identify cargo” and the functional block “cargo identification” were defined.

In a dependency matrix the relations between activities and functional blocks were defined. The functional architecture of the TFS was modeled in an internal block diagram (ibd) (*cf.* Fig. 5).

3) High-Level Design: The physical architecture is based on the functional architecture and specified in High-Level Design. First, the TFS structure is developed in package diagrams and block definition diagrams in order to allocate functional blocks to real systems and system components. Therefore, technologies for realizing the functional blocks “Person Identification”, “Cargo Identification”, and “eID Combination” of the “Functional System” Trusted Forwarder System are modeled in a block definition diagram (*cf.* Fig. 6).

4) Low-Level Design: The instance of a physical architecture is displayed in an internal block diagram. The enlargement of the TFS physical structure in Fig. 7 contains among others

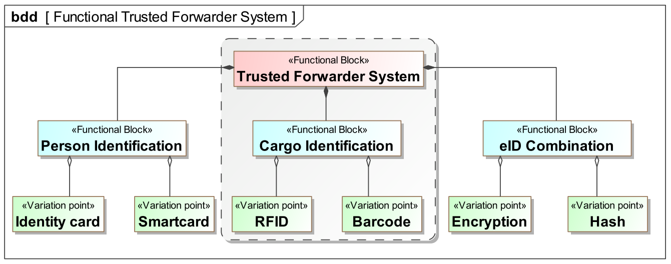


Fig. 6. A block definition diagram (bdd) represents technologies for realizing the functional blocks “Person Identification”, “Cargo Identification”, and “eID Combination” of the “Functional System” TFS.

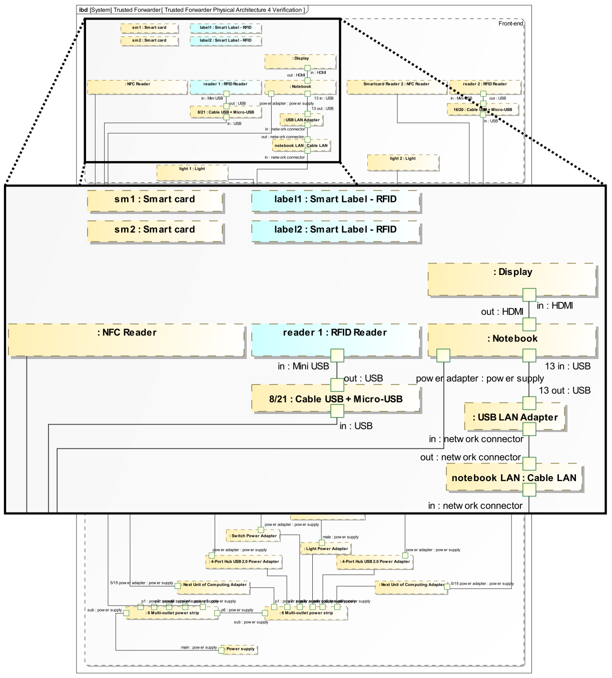


Fig. 7. Physical structure of the TFS presented in an internal block diagram including an enlargement of smart cards, smart labels, an NFC reader, *etc*.

RFID Tags and RFID reader for cargo identification, smart cards, smart labels, an NFC reader for person identification, and their connections. The interactions between actors are presented in sequence diagrams. A detailed description is given in [16] and [17].

Simulation of the developed TFS enabled behavior specification verification. Validation was supported by implementing a RabbitMQ broker for communication between model and hardware elements. In the SiLuFra project the TFS was developed according to the rules of model-based development of single and independent systems. The next section describes model-based SoS development using UAF. Section V concludes with a comparison of both approaches.

IV. Model-based Development of SoS combining UAF and V-Model

Considering the progressive digitization of processes and the development of services the TFS is a representative example for developing cross-system services instead of a single, self-contained system. The acceptance of cargo, its transport, and security checks can be seen as services within a *Trusted Forwarder SoS*. With regard to the whole cargo transport network many stations form the SoS for the provision of services. In the following model-based development of the TFS as SoS using UAF in combination with V-model development process is presented. Cameo Systems Modeler (version 19.0) was used as modeling tool.

A. Requirements Analysis

The suggested model-based approach for developing SoS starts with requirements elicitation and analysis using Strategic Structure views. The *TrustedForwarderSoS* provides the services “Acceptance of cargo and transport” and “Acceptance of cargo, transport, and security check” satisfying the requirements “Identity check”, “Loading secured cargo”, and “Transparency of secured cargo transport” as displayed in Fig. 8. The requirement “Identity check” contains a demand for an identity check of all involved entities, *i.e.* all forwarders in charge. The requirement “Loading secured cargo” demands secured cargo transport while “Transparency of secured cargo transport” fosters a new concept for a transparent and traceable cargo transport process at all times.

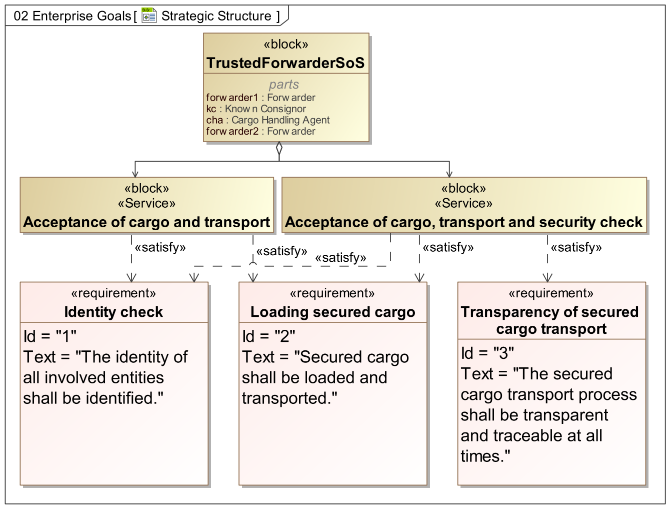

02 Enterprise Goals [Strategic Structure]
«block»
TrustedForwarderSoS
parts
forw arder1 : Forw arder
kc : Know n Consignor
cha : Cargo Handling Agent
forw arder2 : Forw arder
«block»
«Service»
Acceptance of cargo and transport
«block»
«Service»
Acceptance of cargo, transport and security check
«satisfy»
«satisfy»
«satisfy»
«satisfy»
«satisfy»
«requirement»
Identity check
Id = "1"
Text = "The identity of all involved entities shall be identified."
«requirement»
Loading secured cargo
Id = "2"
Text = "Secured cargo shall be loaded and transported."
«requirement»
Transparency of secured cargo transport
Id = "3"
Text = "The secured cargo transport process shall be transparent and traceable at all times."

Fig. 8. *Strategic Structure* view contains System of Systems and its requirements.

B. Functional Analysis

According to the functional analysis in the V-model approach SoS capabilities instead of system functions are defined in Strategic Taxonomy view. In [4] a capability is defined as “an enterprise’s ability to achieve a desired effect realized through a combination of ways and means (*e.g.*, capability configurations) along with specified measures”. For capability definition the former elicited requirements are linked to the capabilities required for their satisfaction. In this example the air cargo transportation system has the capability “Secured Cargo Transport” consisting of the capabilities “Person Identification”, “Cargo Identification”, and “eID Combination” as depicted in Fig. 9.

C. High-Level Design

The Operational Structure view provides an overview of operational architecture and considers exchanges between systems in an SoS. For operational architecture description a block definition diagram (bdd) is used [4] as displayed in Fig. 10. The air cargo transportation system contains the operational performers “Forwarder”, “Known Consignor”, and “Cargo Handling Agent” allocated to the respective capabilities. Information exchange in the SoS is described in an internal block diagram (ibd) depicted in Fig. 11. There is

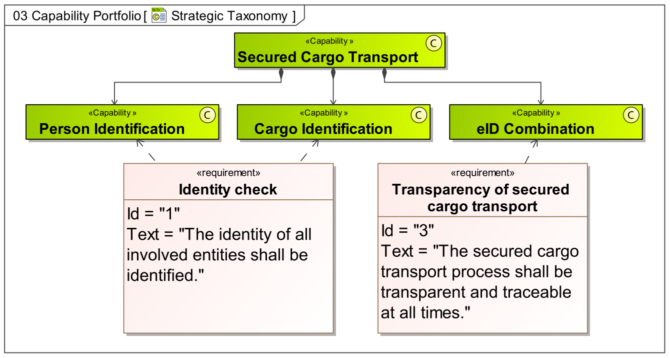

03 Capability Portfolio [Strategic Taxonomy]
«Capability»
Secured Cargo Transport
«Capability»
Person Identification
«Capability»
Cargo Identification
«Capability»
eID Combination
«requirement»
Identity check
Id = "1"
Text = "The identity of all involved entities shall be identified."
«requirement»
Transparency of secured cargo transport
Id = "3"
Text = "The secured cargo transport process shall be transparent and traceable at all times."

Fig. 9. *Strategic Taxonomy* view displays requirements and their allocation to capabilities.

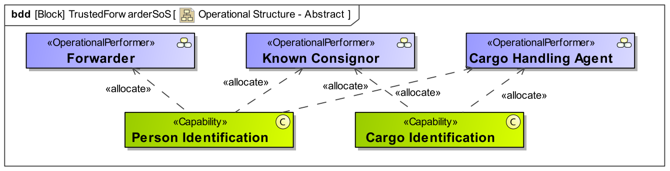


Fig. 10. Operational Structure view in a block definition diagram presents the allocation of capabilities to operational performers.

a forwarder (forwarder1) exchanging information with the known consignor (kc), the cargo handling agent (cha), and a second forwarder (forwarder2). Elements in the ibd are part properties of the operational performers defined in the bdd.

D. Low-Level Design

In low-level design Operational Process view is used for further detailing the exchanges between operational performers. Based on the defined exchanges in high-level design processes between operational performers are established. In Fig. 12 the sequence diagram shows that (1) the first forwarder (forwarder1) picks up cargo and (2) transports it to a second forwarder (forwarder2). The forwarder2 transports cargo to the next station (3), and (4) finally the cargo handling agent (cha) receives cargo.

E. Implementation

In the implementation stage processes defined in low-level design are implemented using a rapid prototyping approach. The formerly introduced broker-based *SysML Toolbox* enables easy hardware integration at an early stage in the design phase. Fig. 13 displays an activity diagram (act) for the process pick-up cargo. Blue boxes indicate call behavior actions containing opaque behaviors for using the broker-based *SysML Toolbox* as already introduced in Figure 2 and Figure 3. When simulating the processes developed in activity diagrams opaque behaviors are executed and send and receive messages to the RabbitMQ server. An exemplary definition of opaque behavior is depicted in Fig. 14.

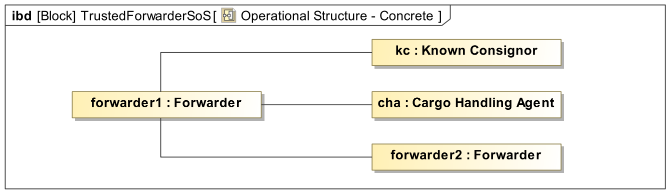

ibd [Block] TrustedForwarderSoS [Operational Structure - Concrete]
kc : Known Consignor
forwarder1 : Forwarder
cha : Cargo Handling Agent
forwarder2 : Forwarder

Fig. 11. Operational Structure view in an internal block diagram contains operational performers and their exchange.

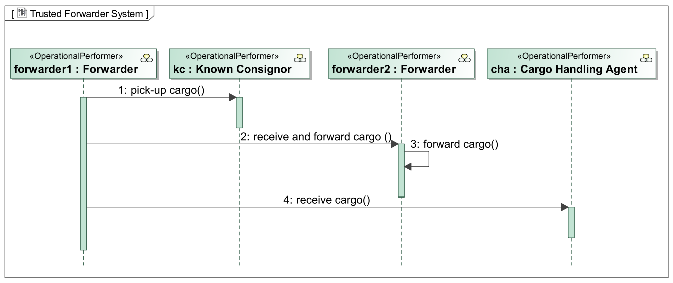

[Trusted Forwarder System]
«OperationalPerformer» forwarder1 : Forwarder
«OperationalPerformer» kc : Known Consignor
«OperationalPerformer» forwarder2 : Forwarder
«OperationalPerformer» cha : Cargo Handling Agent
1: pick-up cargo()
2: receive and forward cargo ()
3: forward cargo()
4: receive cargo()

Fig. 12. Operational Process view of the TFS.

V. Application and Discussion

Within the research project SiLuFra (Secured Air Cargo Transport Chain) [16] the security and efficiency of air cargo transport chain should be improved by supporting new services at each cargo receiving department. As described in [16] and the case study presented in Section III of this article the TFS was developed as one self-contained system, although each cargo receiving department offers rather services in an SoS than functioning as a single system. In the following the approaches for the development of a single, self-contained system using the V-model and SysML (Single system method, abbreviation: SiS method) and for the development of services in an SoS using the V-model and UAF (SoS method) are compared at each level of the design process.

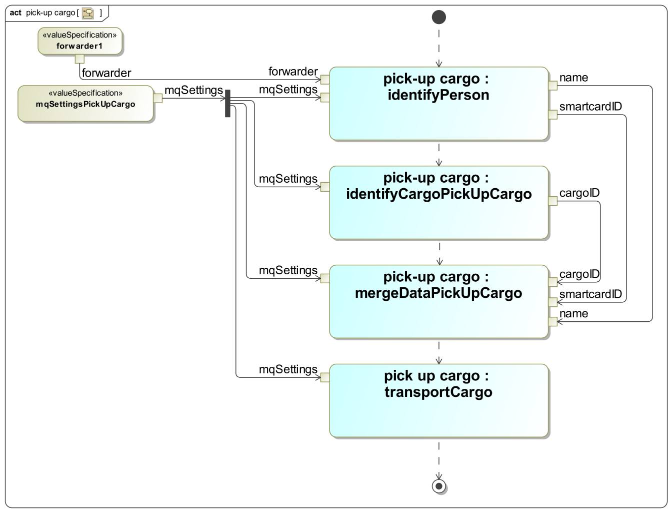

act pick-up cargo [ ]
«valueSpecification» forwarder1
forwarder
«valueSpecification» mqSettingsPickUpCargo
mqSettings
forwarder
mqSettings
pick-up cargo : identifyPerson
name
smartcardID
mqSettings
pick-up cargo : identifyCargoPickUpCargo
cargoID
mqSettings
pick-up cargo : mergeDataPickUpCargo
cargoID
smartcardID
name
mqSettings
pick up cargo : transportCargo

Fig. 13. Specification of the opaque behavior for the “identify person” activity. The code language is Beanshell. The message “get” is sent to the RabbitMQ server via command *SendCommand*.

A. Requirements Analysis

SiS method: Requirements are defined with SysML and presented in requirement diagrams or requirement tables. Initiation of requirements is performed by use cases in use case diagrams.

SoS method: The requirements are modeled in SysML and presented in the Strategic Structure view. In addition, SysML blocks represent the services of the SoS.

Discussion: Within the SoS method use cases are not used as an initiator of requirements. In some practical studies [21]–[24] it was shown that the initiation via use cases contributes to a better understanding of requirements. Therefore, it is suggested to additionally model use cases in the Strategic Structure view.

B. Functional Analysis

SiS method: The operational processes describe the behavior of the TFS at different abstraction levels using activity diagrams.

The functional structure and functional architecture of the whole system is presented in block definition diagrams and internal block diagrams. The interfaces to persons are considered as I/O blocks and parts, respectively. In order to receive the functional structure and architecture, processes at system level are identified and analyzed before.

SoS method: The capabilities of the TFS are presented in the Strategic Taxonomy view. In this view the capabilities of all cargo stations are explicitly considered.

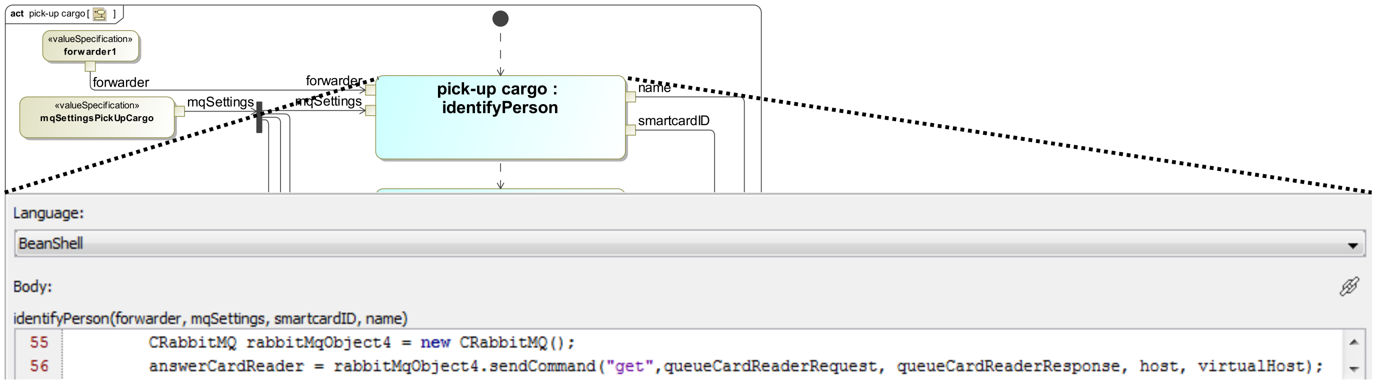
Discussion: At this level, the method of single system development delivers detailed functional information, whereby this approach requires nor a service-oriented functional design neither clear differentiation between subsystems of SoS. A proactive support by methods and tools is a prerequisite for service development in SoS. Thus, the usage of a special Strategic Taxonomy view with a focus on SoS design is suggested.

C. High-Level Design

SiS method: Physical architecture including interfaces to system actors is derived from functional architecture. Technologies for realization are allocated to functional blocks and further detailed in low-level design.

SoS method: Operational Structure view in block definition diagrams displays allocation of capabilities to operational performers. Internal block diagrams focus on exchange between operational performers.

Discussion: The consideration of operators and their needs is an essential condition for developing services in SoS which consider different operators. During TFS development according to SiS method the operators (actors) were incorrectly designed at the Functional Analysis step as a part of a system. A workaround was to use the definition of an I/O operator which represents an interface between operator and



identifyPerson(forwarder, mqSettings, smartcardID, name)

```
        CRabbitMQ rabbitMqObject4 = new CRabbitMQ();
        answerCardReader = rabbitMqObject4.sendCommand("get",queueCardReaderRequest, queueCardReaderResponse, host, virtualHost);
```

Fig. 14. Specification of the opaque behavior for the activity “identify person”. The code language is Beanshell. The message “get” is sent to the RabbitMQ server via command.

system. This approach was semantically right, however, to systems engineers not intuitive. The usage of Operational Structure view in SoS method for representing operational performers enabling the provision of capabilities and their required exchange allows a better overview in case of multiple systems and is therefore suggested for SoS.

D. Low-Level Design

SiS method, SoS method, and Discussion: The interactions and system-level processes are represented in the SysML sequence and activity diagrams both in SiS and SoS method. At low-level the SoS method does not differ from the SiS method. In both cases the V-model can be implemented as described before.

E. Verification and Validation

SiS method: Activity diagrams containing opaque behavior elements are used for hardware integration and validation. Verification is also supported by means of dependency matrices improving traceability between model elements, *e.g.*, functional blocks and system blocks.

SoS method: Validation according to rules of V-model is supported by hardware integration using activity diagrams as displayed by horizontal arrows in Fig. 15. Fig. 15 shows the demonstrator containing the model, a graphical user interface (GUI), and the systems station 1 and station 2 as required in a secured air cargo transport chain depicted in Fig. 4. Status indication lights, smart cards, RFID tagged cargo items as well as smart card readers and RFID readers are used as hardware elements in combination with the model and the GUI for process validation.

Moreover, elements are traceable across all development stages due to hierarchical model architecture as displayed in Fig. 16. Each element is connected to model elements in adjacent development stages enabling traceability from services in Strategic Structure view across requirements, capabilities, operational performers down to part properties in Operational Structure view.

Discussion: Both methods can be used in combination with hardware integration. However, SoS method offers advantages in traceability beyond system’s perspective up to service development in contrast to SiS method. New services can be developed by using existing system model elements and adding new services, requirements, and capabilities.

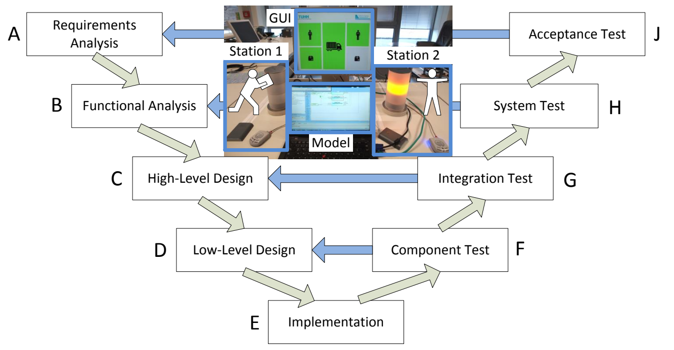


Fig. 15. Validation in the V model displayed by horizontal arrows. In System Test a model contains the developed process and interacts with the systems station 1 and station 2 as well as a graphical user interface (GUI).

VI. Conclusion

This paper introduced an approach for SoS development using UAF in accordance with the V-model development process. Therefore, UAF views Strategic Structure, Strategic Taxonomy, Operational Structure, Operational Process, and the SysML diagrams sequence diagram and activity diagram were mostly used for modeling a Trusted Forwarder SoS. In order to evaluate the suggested model-based SoS method, the approach was compared to the previously developed SiS method [17] using the example of a secured air cargo transport chain.

Since this paper focuses on service specification and implementation in SoS further effort can be made regarding the verification and validation of the developed SoS which is symbolized by the right branch of the V-model (*cf.* Figure 15). Future work will provide details for UAF application in respective development stages, *e.g.*, requirements analysis methods in development stage *A - Requirements Analysis* and methods for functional architecture design for SoS in development stage *B - Functional Analysis*.

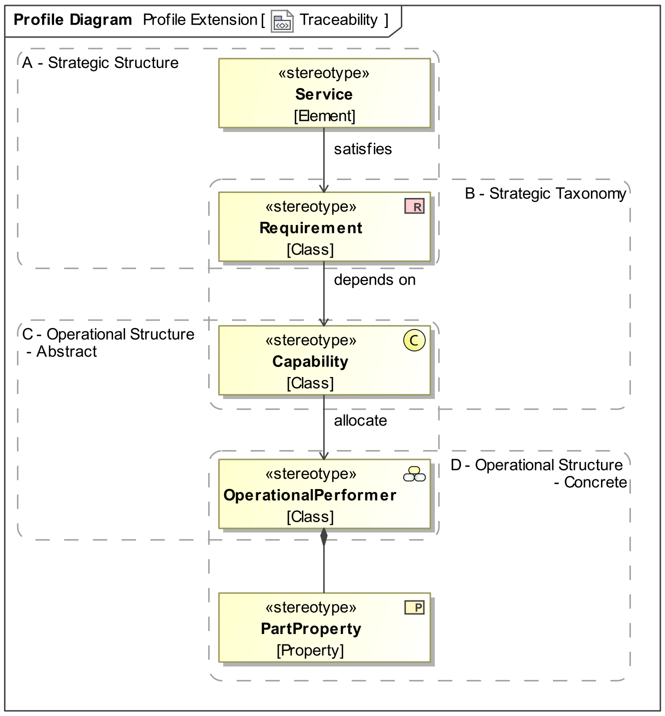

Profile Diagram Profile Extension [Traceability]
A - Strategic Structure
«stereotype» Service [Element]
satisfies
B - Strategic Taxonomy
«stereotype» Requirement [Class]
depends on
C - Operational Structure - Abstract
«stereotype» Capability [Class]
allocate
D - Operational Structure - Concrete
«stereotype» OperationalPerformer [Class]
«stereotype» PartProperty [Property]

Fig. 16. Model element allocation to views. Traceability is achieved by connecting model elements to elements in adjacent development stages.

The application results demonstrate that in comparison to SiS method model-based development of SoS using UAF focuses more on service development addressing companies' needs for evolving from system to service providers. New services are often based on existing systems emphasizing the need for service and capability consideration in SoS development methods besides conventional system design.

References

[1] D. D. Walden, G. J. Roedler, K. Forsberg, R. D. Hamelin, and T. M. Shortell, eds., *Systems engineering handbook: A guide for system life cycle processes and activities; INCOSE-TP-2003-002-04, 2015*. Hoboken, NJ: Wiley, 4. edition ed., 2015.

[2] H.-J. Bullinger and A.-W. Scheer, *Service Engineering — Entwicklung und Gestaltung innovativer Dienstleistungen*, pp. 3–17. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003.

[3] C. Piaszczyk, "Model Based Systems Engineering with Department of Defense Architectural Framework," *Systems Engineering*, vol. 14, no. 3, pp. 305–326, 2011.

[4] Object Management Group, "Unified Architecture Framework (UAF)," Version 1.0, November 2017.

[5] Verein Deutscher Ingenieure, "Design methodology for mechatronic systems," ICS 03.100.40; 31.220, June 2004.

[6] C. Ebert, *Systematisches Requirements Engineering: Anforderungen ermitteln, spezifizieren, analysieren und verwalten*. Heidelberg: dpunkt-Verl., 4th ed., 2012.

[7] S. Boschi and G. Santomaggio, *RabbitMQ Cookbook*. Birmingham and Mumbai: Packt Publishing Ltd., 2013.

[8] E. Ayanoglu, D. Nahum, and Y. Aytas, *Mastering RabbitMQ*. Birmingham, UK: Packt Publishing Ltd., 2015.

[9] H.-H. Altfeld, *Commercial aircraft projects: Managing the development of highly complex products*. Farnham, Surrey, England and Burlington, VT: Ashgate Pub., 2010.

[10] H. Hintze and R. God, "Using model-based security engineering in the development of complex aircraft cabin systems," *SAE International Journal of Aerospace*, vol. 8, no. 1, pp. 89–96, 2015.

[11] S. Melzer, J. P. Speichert, O. C. Eichmann, and R. God, "Simulating Cyber-Physical Systems Using a Broker-Based SysML Toolbox," 2019.

[12] W. Moser, "United Flight 23 to Chicago: The First Airline Terrorism?," 2011. https://www.chicagomag.com/Chicago-Magazine/The-312/September-2011/United-Flight-23-to-Chicago-The-First-Airline-Terrorism/.

[13] Federal Aviation Administration (FAA), "A Brief History of the FAA," jan 2017. https://www.faa.gov/about/history/brief_history/.

[14] International Air Transport Association (IATA), "Cargo Security," 2019. https://www.iata.org/whatwedo/cargo/security/.

[15] Luftfahrt-Bundesamt, "General Information about Possibilities for Air Freight Shipment."

[16] S. Melzer, U. Wittke, and R. God, "Sichere Luftfracht-Transportkette: Modellbasierte Architektur- und Lösungsspezifikation, TUHH Schlussbericht im Vorhaben Sichere Luftfracht-Transportkette: Konzepte, Strategien und Technologien für sichere und effiziente Luftfracht-Transportketten (SiLuFra)," *SiFo BMBF*, 2017.

[17] S. Melzer, U. Wittke, H. Hintze, and R. God, "Physische Architekturen variantengerecht aus Funktionalen Architekturen für Systeme (FAS) spezifizieren," in *Tag des Systems Engineering, Herzogenaurach, 25.-27. Oktober 2016* (S.-O. Schulze, C. Tschirner, and C. Muggeo, eds.), München: Hanser, 2017.

[18] International Organization for Standardization, "ISO/IEC-15288:2008 Systems and Software Engineering - System Life Cycle Processes," 2008.

[19] T. Kiehl and R. God, "Modeling Top-Level Requirements for a Tangible User Interface in the Aircraft Cabin," (Hamburg), pp. 275–284, 2015.

[20] J. G. Lamm and T. Weilkiens, "Happy Birthday! 5 Jahre Funktionale Architekturen nach FAS," *Tagungsband zum Tag des Systems Engineering (Hrsg.: Chr. Muggeo, SO Schulze), S*, pp. 59–68, 2015.

[21] Institute of Aircraft Cabin Systems, "KomKab (Kommunizierende Kabine)," 2016, unpublished results.

[22] Institute of Aircraft Cabin Systems, "ConCabInO (Information Centric Operation of Future Connected Cabin)," 2016, unpublished results.

[23] N. David, F. Laukotka, and O. Wölm, "Entwurf und Demonstration eines Smart Trolley für ein intelligentes Catering-Konzept." Entwurf und Demonstration eines Sitzbelegterkenungssystems in einer Flugzeugkabine, Systemtechnisches Entwicklungsprojekt, Hamburg University of Technology, 2017, unpublished results.

[24] C. van der Hoek, E. Aydin, and I. Y. Karahan, "Entwurf und Demonstration eines Sitzbelegterkenungssystems in einer Flugzeugkabine." Entwurf und Demonstration eines Sitzbelegterkenungssystems in einer Flugzeugkabine, Systemtechnisches Entwicklungsprojekt, Hamburg University of Technology, 2018, unpublished results.