ASTES

Advances in Science, Technology and Engineering Systems Journal
Vol. 7, No. 3, 19-31 (2022)
www.astesj.com
Special Issue on Multidisciplinary Sciences and Engineering

ASTES Journal
ISSN: 2415-6698

Towards a Model-based and Variant-oriented Development of a System of Systems

Sylvia Melzer[*1,2], Stefan Thiemann[3], Hagen Peukert[3], Ralf Möller[1]

[1] *Universität zu Lübeck, Institute of Information Systems, Lübeck, 23562, Germany*

[2] *Universität Hamburg, Centre for the Study of Manuscript Cultures, Warburgstraße 26, 20354 Hamburg, Germany*

[3] *Universität Hamburg, Center for Sustainable Research Data Management, Monetastraße 4, 20146 Hamburg, Germany*

ARTICLE INFO

Article history:
Received: 15 February, 2022
Accepted: 14 May, 2022
Online: 25 May, 2022

Keywords:
System of Systems
SysML
Information System
Variants

ABSTRACT

The development of an aggregated system consisting of autonomously developed components is usually implemented as a self-contained unit. If such an aggregation is understood as a system of systems (SoS) that communicates via interfaces with its autonomous subsystems and components, the interfaces and communication exchange should play a central role in the architectural design. In fact, complete and exact interface specifications simplify loose coupling of independent systems into an aggregation. Since an SoS consists of variant and non-variant subsystems, the main challenge in SoS development is the identification of all true variants and its deviating attributes within an SoS. If the system variants are identified at an early stage of the development process, redundant work in the interface design can be substantially reduced. This paper presents an efficient method to identify SoS variants with regard to life cycle management and it shows how to configure a variant-oriented SoS with a standardized communication interface. For the development, the forward-looking model-based systems engineering approach is recommended to create executable specification parts and to detect errors early on through simulations.

1 Introduction

The *Centre for the Study of Manuscript Cultures* (CSMC) at Universität Hamburg hosts a steadily growing number of autonomously developed database systems, e.g., for the projects *Epigraphic database of ancient Asia Minor* (EDAK) (https://www.epigraphik.uni-hamburg.de), *Going From Hand to Hand: Networks of Intellectual Exchange in the Tamil Learned Traditions* (NETamil) (https://www.manuscript-cultures.uni-hamburg.de/netamil/), and *Thesaurus Defixionum* (TheDefix) (www.thedefix.uni-hamburg.de). As a first priority requirement, the database schema reflect the high data variety of these research projects while maintaining the same overall structure. If these databases are now combined into an aggregated information system, new functionalities that are designed to the overall structure and not to the peculiarities of each schema can be defined as it is the case for federated searches. In addition, one can very well imagine that new database systems would like to connect to the aggregated information system later on as long as the structure remains clear.

An illustration for the usefulness of an aggregated information system are trove discoveries, whose associated parts, for some reason, are scattered at different places in the world as it often happens for old manuscripts. Such script fragments are administered in different information systems. As an example, one fragment AO 29196 [1] is located at the Louvre and the counterpart of this fragment, KUG 15 [2], is located in Germany. Indeed, both fragments were discovered without using federated search queries, but for analyzing data from different databases it would be desirable to find related data in an aggregated information system. This example highlights the need to combine, analyze, and query data from different database systems.

The requirement for the development of an aggregated system is, on the one hand, to develop autonomous systems in such a way that the variant parts are not implemented redundantly and, on the other hand, that external databases can be added to the aggregated system without much effort.

A systematic approach to model variant parts was developed at the *Institute of Product Development and Mechanical Engineering Design* (PKT) at the Hamburg University of Technology. The variant-oriented approach is called *integrated PKT* approach (see

[*]Corresponding Author: Sylvia Melzer, Universität Hamburg, Centre for the Study of Manuscript Cultures, Warburgstraße 26, 20354 Hamburg, Germany, sylvia.melzer@uni-hamburg.de

[3]). The integrated PKT approach aims at satisfying a wide variety of customer requirements while developing a component. Ideally, the approach applies to a product family that is developed within one organization, for which the marketed products are supposed to have as few variants as possible. Also, adding an external sub-product to the product family at a later date must be considered very early on in the planning phase. Early consideration of coupling systems or products can lead to modularization of systems to support the approach. The integrated PKT approach also supports modularization. Nevertheless, it is not always possible to extend the product family by a new variant. Another core idea of the integrated PKT approach, besides the variant-oriented development of products, is the combination of the database into an SoS and keeping the main focus on the development of a communication interface. While there are some good approaches to SoS development already, an approach that considers system variants during development of an SoS is to the best of our knowledge not yet available.

For the development of an SoS, model-based methods using the *Systems Modeling Language* (SysML) are increasingly used. Users also benefit from the general *Model-Based Systems Engineering* (MBSE) advantages, such as making complexity manageable. The first model-based SoS developing methods are evolving, cf. [4] as well as [5].

In this paper, we present a provident, model-based, and variant-oriented approach to develop new functions for an aggregated information system so that all functions can be used simultaneously to the benefit of all database systems.

The paper is structured as follows. In Section 2 and Section 3 we give an overview on related work and preliminaries for developing a model-based and variant-oriented SoS. In Section 4 we describe how to develop variant-oriented and sustainable information systems such that as many customer requirements as possible are considered while increasing the number of variants and reducing the necessity of redundant information system development.

In Section 5 we present, first, how variant and SoS relevant requirements are elicited during the requirements engineering process, and second, how to design the structure of an SoS. In Section 6 we describe modeling and simulating the systems' behavior to execute a federated search as an example. The application of our new approach and its results are presented in Section 7. We like to close in Section 8 with a summary and a preview of future work.

2 Related Work

Despite its frequent usage, there is little agreement nowadays on a concise and general definition of the term *system of systems* (SoS). Some approaches of SoS distinguish between SoS and traditional systems. These approaches elaborate specifically on the heuristics of SoS development. They also emphasize the differences to traditional systems. More particularly, it has been noted that the architecture of an SoS aims at optimal communication for the vast majority of all SoS (see e.g. [4, 6, 7]).

The application in [4] illustrates the development of a *Trusted Forwarder System* (TFS) for a secured air cargo transport chain as an SoS using a set of standards that enable useful communication between existing and newly developed components. For the TFS, a communication standard is used that satisfies the new requirements. Thus, after SoS integration, the systems and components are enabled to follow their original tasks without diversion.

In [7], the authors show how, from autonomously developed database systems, an aggregated information system enables relatively simple data exchange through a standardized communication interface. If the individual systems are combined into a federated system via a communication interface, new functions can be added easily, such as the federated search functionality. With the new search function, the individual systems can perform searches in all databases (as opposed to only one database) provided that access rights are correctly assigned. The new search feature can be considered as a new service relevant for a broader range of users.

Both aggregated systems, the TFS and the aggregated information system, were initially developed as single systems, cf. [8, 9]. The single system and the SoS development of the TFS were compared in [4]. It was observed that the SoS development approach mainly is advantageous for traceability beyond the system perspective to the service. The advantages of re-usability of a system, which was developed as SoS, are plain to see: Communication interfaces give more flexibility to add new functionalities or remove subsystems from the overall system.

In [6], the author recommends a stable architectural design for SoS. Such stability can be achieved by admitting independently, i.e. autonomously, developed systems in the architectural design together with a communication interface.

Model-based approaches, such as the *Variant Modeling with SysML* (VAMOS) presented in [10], pure::variant (`https://www.pure-systems.com/purevariants`), the *Variety Allocation Model* (VAM) (variant-oriented developing process of the integrated PKT approach) with SysML [11], exist to identify possible variants in the early phase of system development. The methods VAMOS and VAM with SysML can be represented in the SysML modeling tool *Cameo Systems Modeler* by extending the language elements. Pure::variant can be used as a stand-alone entity for variant modeling. In this paper we decided for VAMOS, since it was already applied in [7] for the development of a cross-domain information system.

Cameo Systems Modeler and the broker-based SysML Toolbox have been successfully used for simple modeling of communication networks in several projects such as SiLuFra [12], ConCabInO [13], KomKab [14], and KMUDigital [15].

3 Preliminaries

This section describes the languages, methods, and tools proposed for a model-based and variant-oriented development of an SoS.

3.1 Modeling Languages

According to the recommendations of MBSE, systems are described or documented using semi-formal modeling languages such as the *Unified Modeling Language* (UML) or the SysML.

Systems Modeling Language SysML was specified by the *Object Management Group* (OMG) to support the model-based devel-

opment of complex systems during the system development process. SysML is a subset of the standardized language UML 2 including some additional extensions. A SysML model can be used to describe the structure as well as the behavior of a system, and can be used to simulate the behavior of systems. In this paper, the focus of variant modeling is on structure. For variant behavior modeling, further challenges are to be expected (cf. [16]), which should be addressed separately due to the complexity of SoS development.

Variant Modeling with SysML A variant is characterized by a base model and differentiating parts, where the base model represents the core of the system and the differentiating parts represent the distinctions of the system components (see [17]). In [10], the author specified VAMOS to model variants with SysML. To use this, the existing model elements in SysML are extended (see Figure 1). The two model elements *Package* and *NamedElement* are extended with the stereotypes *Variant*, *Variation*, *VariationPoint*, and *VariationElement*. The *Variation* is a stereotype of the *Metaclass Package*, which contains all the elements of an option of *Variation*. A *Variation* is also a stereotype of the *Metaclass Package*, which contains multiple variation packages. A *VariationPoint* is a stereotype of the *Metaclass NamedElement*.

VAMOS is suitable for systems with some variability. Furthermore, VAMOS is applicable for the use of structural elements. For the description of variant system behavior, VAMOS can be applied conceptually. Practical applications usually exploit extensions of further SysML language elements related to the behavior necessitating the variant behavior description (see [16]).

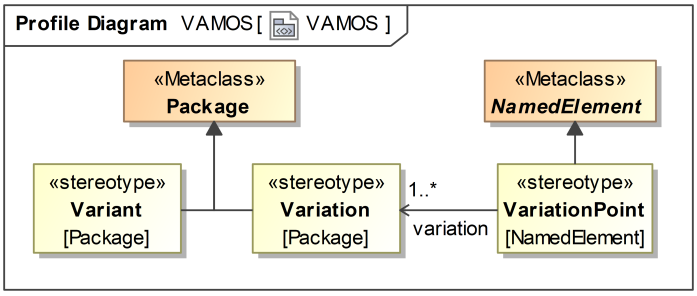

Figure 1: Profile diagram: Variant Modeling with SysML (VAMOS) for the development of a cross-domain information system

3.2 Methods

Context-Based Requirements Engineering The identification of variants should be done as early as possible, so it is necessary that views of all stakeholders involved are considered during the requirements engineering process. The associated systems have to be identified during the SoS context description process. In [18], the authors have defined the *Approach for Context-Based Requirements Engineering* (ACRE) ontology with the goal to capture the requirements of all stakeholders and manage them during the entire system development process. A fundamental approach of the ACRE ontology is the context, since it emphasizes and defines the view of use cases and requirements. Depending on the use cases, some new model-based systems engineering approaches add specific contexts defined for system or product development, e.g. the life phase modularization context in [8], the variety context in [11], and the system of systems context in [19]. In [20, 21], a supplemented ACRE ontology with the SoS aspect are presented and also introduce new terms for system development. Adding the variety context to the SoS approach results in a new variant-oriented approach, which is described in this paper.

V-model For all IT projects in the federal public administration, the V-model is a mandatory procedural standard. The processes of the V-model, inspired by the V-model XT (see www.v-modell-xt.de), can be described as follows: analysis of requirements, functional analysis, high-level design, low-level design, implementation, component test, system test, integration test, and acceptance test. Verification and validation also belongs to the processes.

We argue that the V-model is a good basis for system development, so we have used this approach to develop information systems. For a variant-oriented development and implementation of a communication interface, it is important to consider in the individual process steps like the high-level design, that, among other properties, combined approaches are used to develop the SoS efficiently and correctly. Approaches for the development of an SoS are described in Subsection 5.2.

Broker Federation The brokerage network enables the creation of message routing networks, in which messages in one broker are automatically routed to another broker. These routes may be defined, e.g., between exchanges in the source and destination brokers, or from a message queue in the source broker to an exchange in the destination broker [22]. The principle of coupling systems via a broker federation is a practically proven approach that is used in many applications. In this paper, broker federation is used to create a communication interface between the systems to develop the SoS.

3.3 Tools

Communication Tool The open-source message broker RabbitMQ (https://www.rabbitmq.com/) can be used to create communication networks. RabbitMQ uses the *Advanced Messaging Queuing Protocol* (AMQP) as a standardized communication technology. AMQP defines three components which are essential to implement a message-based architecture. 1) The *message queue* stores messages which can be consumed by client applications. 2) The *exchange* receives messages from publisher applications and routes these to message queues. 3) The *binding* defines a relationship between a message queue and an exchange. Using these components, classic communication paradigms can be implemented and used such as 1) send and receive, 2) work queues, 3) publish and subscribe, 4) routing, 5) topics, and 6) request and reply.

In [4, 7, 9, 15], the authors show that the developed communication interfaces with RabbitMQ can be used for implementing real software or hardware in the model with little effort. For this reason, we choose RabbitMQ to support a communication interface for the individual systems that become part of the SoS.

Modeling and Simulation Tool Cameo Systems Modeler (version 2021x) is a modeling and simulation tool that was originally

developed specifically for the development of systems using the SysML. To simulate behavioral diagrams, Cameo Systems Modeler uses a subset of the UML elements on the OMG *Foundation Subset for Executable Models* (fUML) and *W3C State Chart XML* (SCXML) standards. The broker-based SysML Toolbox is an extension of the Cameo Systems Modeler and provides the integration of real software and hardware [15]. The Toolbox also offers predefined SysML elements that can be used to create database interactions. The SysML Toolbox contains an implementation of these six messaging paradigms. These communication paradigms are implemented via the SysML element *opaque action* and the usage of the Java-like scripting language *BeanShell*.

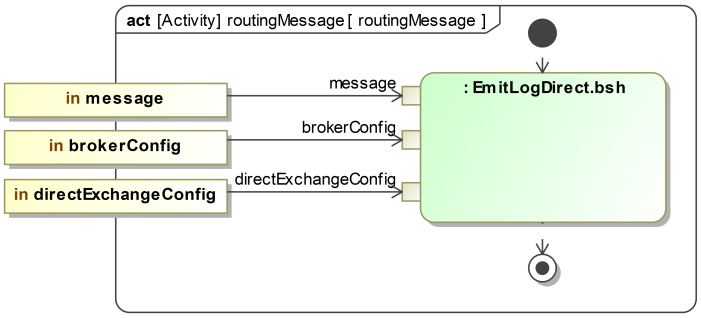

Figure 2: The opaque action *EmitLogDirect.bsh* for sending a routing message to the RabbitMQ server

An implementation of the routing communication paradigm as an opaque action element and the respective BeanShell code are presented in Figure 2 and in Figure 3. The opaque action is called *EmitLogDirect.bsh*. The other paradigms are also available as opaque actions. These opaque actions are implemented as drag-and-drop communication elements, with the aim to increase efficiency and to avoid coding effort.

Language:

BeanShell

Body:

EmitLogDirect.bsh(message, brokerConfig, directExchangeConfig)

```
18 try
19 {
20     ConnectionFactory factory = new ConnectionFactory();
21     factory.setHost(host);
22     factory.setVirtualHost(virtualHost);
23     factory.setPort(port);
24     factory.setUsername(userName);
25     factory.setPassword(pw);
26     Connection connection = factory.newConnection();
27     Channel channel = connection.createChannel();
28
29     try {
30        channel.exchangeDeclare(EXCHANGE_NAME, "direct", true);
31     } catch (IOException e1) {
32        // TODO Auto-generated catch block
33        e1.printStackTrace();
34     }
35
36     String severity = routingKey;
37
38     channel.basicPublish(EXCHANGE_NAME, severity, null, message.getBytes("UTF-8"));
39     print(" [x] Sent :'" + message + "'");
40
41     channel.close();
42     connection.close();
```

Figure 3: The source code for sending a routing message to the RabbitMQ server

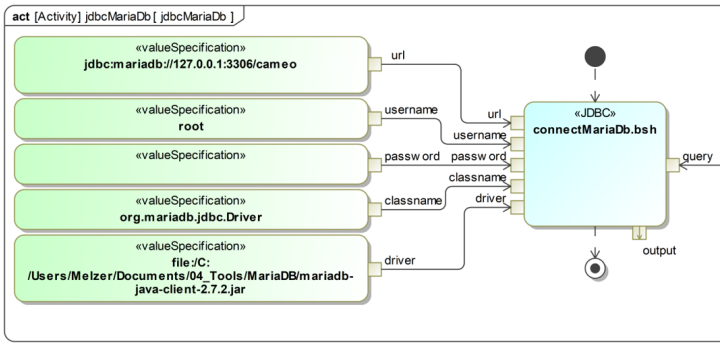

Figure 4: The opaque action *connectMariaDb.bsh* for sending a message to the database MariaDB, source: [23]

The following SysML blocks are used to define input and output parameters: *MessageBroker*, *MessageQueue*, and *MessageExchange*. The *MessageBroker* contains the properties: *host*, *virtualHost*, *port*, *username*, and *password*, which are input parameters for the opaque behavior *EmitLogDirect.bsh*. The properties of the MessageExchange are *exchangeName* and *routing key*. In order to set individual configurations, it is possible to create instances of the SysML blocks. An instance of the *MessageBroker* is *brokerConfig*. An instance of the *MessageExchange* is *directExchangeConfig* (see Figure 2). More details of the broker-based SysML Toolbox are given in [15].

For modeling database expressions, the extension of the broker-based SysML Toolbox can be used or replicated. The predefined database expressions for creating, manipulating, and querying databases are implemented as opaque actions. These predefined actions can also be used as drag-and-drop elements (cf. [23]).

Language:

BeanShell

Body:

connectMariaDb.bsh(url, username, password, classname, driver, query, output)

```
20 import org.mariadb.jdbc.Driver;
21
22 ConnectionMariaDB c = new ConnectionMariaDB();
23 conn = c.connectDb(classname, url, username, password, driver);
24
25 if(conn!=null) {
26     // create the beanshell statement
27     Statement st = null;
28     try {
29         st = conn.createStatement();
30     } catch (SQLException e) {
31         // TODO Auto-generated catch block
32         e.printStackTrace();
33     }
34     // execute the query, and get a beanshell resultset
35     ResultSet rs = null;
36     try {
37         rs = st.executeQuery(query);
38     } catch (SQLException e) {
39         // TODO Auto-generated catch block
40        e.printStackTrace();
41     }
```

Figure 5: The source code for sending a message to the database MariaDB

Figure 5 shows that the opaque action element *connectMariaDB.bsh* has the input values *classname*, *url*, *username*, *password*,

and *driver* to create a database connection. In addition, the BeanShell code is required to send a request to the database (Figure 5, line 29) and get a response (Figure 5, line 37).

The source code in the opaque behaviors is tested by running the simulation. The code can therefore be adopted when implementing, e.g., systems or SoS.

Database Management System The open-source web-based database management system Heurist was specially developed for the Humanities. Heurist allows researchers without prior IT knowledge to develop databases, store and search their data, and publish it on an automatically-generated website.

4 Information System Development

Data projects in the Humanities depict a perfect test scenario for information system development for two reasons. First, the projects tend to be comparatively small and, second, both data and usage show high degrees of heterogeneity. This phenomenon, known as the *long-tail* problem of the Humanities, is due to an institutional decision of most universities to subsume all kinds of subjects under one departmental unit called Humanities. Left the reasons aside, software architects and researchers alike find themselves in the situation to cope with the high variability of requirements, software quality attributes, and missing standards. From a point of view of information system development, one can think of several solutions for data heterogeneity. In fact, they can also all be viewed as an SoS. An analysis of the current situation reveals three strategic strains of data management. Firstly, isolated applications fully independent and maintained by decentralized units such as a single chair. Second, single data applications implemented with a set framework such as *My Content Repository* (MyCoRe) managed centrally. And third, a globally maintained platform with limited but extensive data curation functionality for archiving, publishing, and analyzing data such as Heurist. Since sooner or later, the isolated applications are transferred to one of the centralized data solutions, we will take a closer look at the two later approaches.

MyCoRe The framework MyCoRe (https://www.mycore.de/en/) contains all the functionality of a data repository. Some public institutions such as libraries and universities implement instances of MyCoRe to administer publication inventories and research data. As a typical client-server application, it can be used to host any kind of data. Among the main configurable components of the MyCoRe system are a Solr (https://solr.apache.org/) search engine, a data base access handler via Hibernate, a system management for user rights and access as well as a content store. Interfaces to external systems are restricted to library formats Z39.50, but also comprise REST, OAI and SWORD. Generally all documents and metadata are saved as XML, however, some information is stored in relational database tables for reasons of performance and modifiability. Other interfaces include information exchange to the application layer, that is, a layout engine rendering XSL stylesheets and some functionality to configure the data model as well as other system variables.

Although the structure of a MyCoRe database is known and could be used for automatic retrieval, the data models of a MyCoRe application are very flexible and represented without a standard as a XML schema definition. Its retrieval and analysis depend on how different data models are related to each other and which structural information on how to process the data is *hidden* in the application. Generally it is possible to parse the data model schema definitions and based on this information automate the data retrieval. Yet, for MyCoRe applications that make use of several data models whose interaction and processing became part of the business logic of the program, a semi-automated retrieval process seems to be the only doable solution.

It is a valid data management strategy to have these projects set up as independent MyCoRe instances if larger amounts of data need to be handled or if many users with many different tasks and views on the data require clear and comprehensible workflows. It ensures more flexibility while keeping data maintenance and server administration on an acceptable workload. Although the structure of the data, its formats and processing, is the same for all instances and it therefore has a lot of technical scalability potential, the operation of many MyCoRe instances still leads in the long run into maintenance problems if new versions have to be adjusted to the specific needs and the changing requirements of the project stakeholders. Thus, if specific needs such as a federated search are desired, this cannot be easily added. The implementation of a new function would have to be done for all instances. And if there are variant instances, a new function would have to be developed separately for each instance.

An elegant way around the growing maintainability dilemma is to find a new optimum between usability and scalability. More specifically, it means trading off the flexibility of front end layouts and some cut back on performance to integrate projects into one platform. Indeed, the tendency to focus on services rather than entire system development plays a role in the design decisions of SoS. A practical solution is to devise a system that allows for just so much adjustability as necessary for requirements satisfaction (variant-oriented system development), but leave the components responsible for all other quality requirements untouched. Heurist can be seen as such a way in the middle. Within the approach of SoS, one could push it a step further and classify data projects according to their requirements or one could also embrace all smaller projects into a new platform solution, such as Heurist, and leave the few projects with a large data inventory on MyCoRe instances to keep performance on an acceptable level.

Heurist The data management system Heurist is suitable for variant-oriented system development such as presented in [7, 9]. Even if the development of the systems, here database instances of Heurist, hold the same functions, these can be used to create a project-specific database autonomously. If the individual database instances were to be combined into an aggregated system, it would be possible to develop the complete system as a single system, as a product family or as an SoS. However, the system development of a single system has little flexibility to make extensions. Single systems cannot be used for different purposes as variants as effortless and cost-saving than SoSs.

With Heurist, for each project a project-specific web page can be constructed as a variant with the same functional range. In order

to create a website, the search area, the display of a result set, the display of the contents as well as the integration of a map can be straightforwardly arranged. The view can either be programmed with PHP or implemented via an editor interface.

To sum up, Heurist makes it possible to create a database instance as a variant and supports further development with individual properties.

4.1 Information Systems

The three information systems EDAK, TheDefix, and NETamil are autonomously developed information systems at the CSMC using the tool Heurist while the *Collection of Greek Ritual Norms* (CGRN) is an external application that was not modeled in Heurist.

The first three information systems mentioned above represent how systems can be developed in a variant-oriented manner. The CGRN system represents a system which becomes part of the SoS without being an instance of Heurist.

In practice, other systems are often developed as different instances, but they should also have the possibility to use the same functionality if required. Then, it is desirable that these systems can also be integrated into the existing overall structure without having a complete redesign of the SoS.

In what follows it is shown that both variant and non-variant systems can be part of the SoS and thus all these systems can use the federated search function.

NETamil During the project NETamil at the Universität Hamburg a repository was created containing digital images of classical Tamil manuscripts on palm leaves and on paper from Indian and European libraries along with a descriptive catalog, e-texts along with critical editions and annotated translations. The data was originally stored in a Word document.

In general, the database schemes can either be created individually or they can also be converted into well established XML standards such as *Text Encoding Initiative* (TEI) (`https://tei-c.org`). The TEI format is more common used in the humanities for data storage and exchange.

The automatic transformation of XML-encoded formats into a Heurist database instance has the feature of transferring a large data set into a new database instance in short time. In this paper, the created database systems have been automatically created using a word to TEI transformation process [24].

EDAK During the project EDAK the Department of History at the Universität Hamburg created an epigraphic database of ancient Asia Minor. This database contains a collection of Greek and Latin inscriptions in the area of modern-day Turkey. The data are stored in the format EpiDoc to enable easier data exchange between machines. EpiDoc is a widely used scheme for encoding scholarly and educational editions of ancient documents. It uses a subset of the TEI's standard for the representation of texts in digital form [25].

The EDAK Information system has been automatically created using an EpiDoc to Heurist transformation process.

TheDefix The database TheDefix contains curse inscriptions of the ancient world. The data are represented in a project-specific scheme. In Figure 6 the information system for the *TheDefix* project is presented: the search area is located at the left, in the middle is the result set, and on the very right the project specific data representation (text and map representation) are displayed.

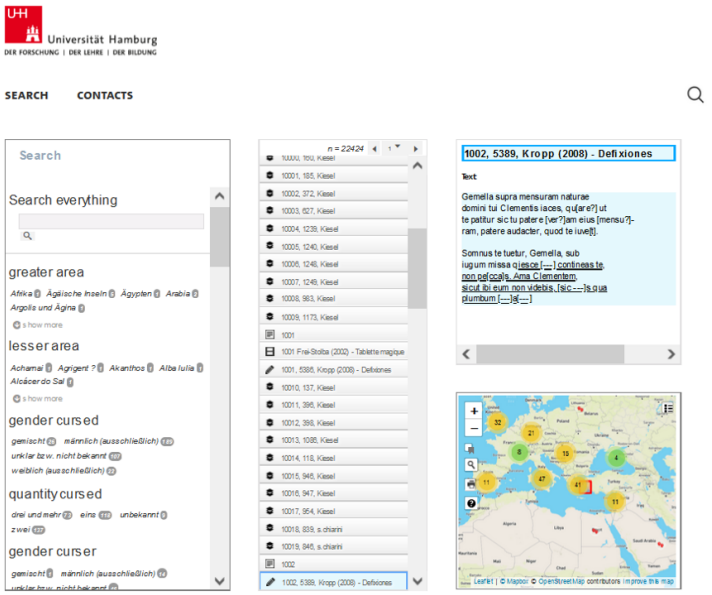

Figure 6: TheDefix Information System

CGRN The CGRN presents epigraphical data on a website. Its primary goal is to gather epigraphical material for the study of Greek rituals and to make these sources widely available [26]. The data are additionally stored in the EpiDoc format.

Merging information systems into an aggregated system, in general, requires addressing the complex issue of information integration. "Information integration is the merging of information from heterogeneous sources with differing conceptual, contextual, and typographical representations" (see [27]). For computers it is difficult to merge information without the knowledge of the syntax, semantics, model, and access of the data representations (see [28]). The approaches therefore require a standardized framework for representing data that, while supporting autonomy to some degree, can make heterogeneity manageable.

In the next chapters, we will reveal how to integrate different autonomously developed information systems, which can also be physically located in different places, as one SoS, taking into account that variant parts are not developed redundantly.

Product family The integrated PKT approach includes the VAM which, in a hierarchical approach of four levels, is used to develop a variant component for each custom-relevant differentiating property, whenever possible in a 1:1 relationship. This approach is very suitable if as many different customer requirements as possible have to be satisfied while still remaining competitive. The VAM approach was also transformed into a model-based approach, using VAMOS to represent the variants. It was observed that the structured package overview in the SysML model avoids redundancies

and improved transparency and traceability for large and complex projects (see [11]).

The integrated PKT approach aims at developing a product family comprising variants. The idea of developing an SoS in a standardized manner is obvious. However, it must be ensured that the development of product families also involves the development of variant hardware components and not just a communication interface. In addition, one has an influence on all systems with the development of product families. If, however, an information system were designed as an SoS consisting of both internal and external systems, it would be recommendable for a clear focus on the communication interface. This recommendation must be taken into account when the SoS is actually modeled on variants.

5 Variant-oriented SoS Development

In this section we present how context information relevant to the SoS and the variants are identified as part of the requirements engineering process ACRE. Additionally, it is described how to design the structure of an aggregated information system. Finally, this chapter depicts how to create a communication network as well as how to simulate the common function *federated search* using Cameo Systems Modeler and the broker-based SysML Toolbox.

5.1 Variant-Based Requirements Engineering

For successful system development it is essential that the needs of all stakeholders are sufficiently satisfied. Therefore, it is necessary to have identified all persons and institutions that have requirements or interest in the system. The respective requirements of all identified stakeholders are collected, documented, and structured according to the ACRE ontology, presented in [18], with the goal to identify the requirements of all stakeholders and to be able to manage them throughout the system development process.

For the variant-oriented development of an information system as an SoS, a lean version of the ACRE ontology was specified and used for modeling an aggregated information system. The ACRE ontology with SoS and variety contexts is presented in Figure 7.

An (abstract) requirement has the specializations business, functional, and non-functional requirements. A requirement description explains requirements, where some rules are applied. The rules could be that requirements have to be formulated in accordance with the ISO 29148:2011 [29] and RFC2119 [30] to use the linguistic syntax profitably. A requirement description is elicited from one or more *Source Element*(s). Source elements can be standards, conversations, requirement lists, and specifications among others.

The contexts are defined as:

- A system of systems context defines views on aggregated systems.
- A system context contains views of system, subsystem, assembly, and component.
- A stakeholder context defines views on different stakeholders.
- A variety context defines views on system variants.

Use cases are validated by one or more *Scenario*(s). A *Scenario* describes the “what ifs” in a semi-formal or formal way. SysML activity and sequence diagrams can present *Semi-formal Scenario*s to describe communication processes and interactions between elements in the system. SysML parametric diagrams present the “what ifs” formally. Both scenario types support the analysis of “what ifs” to validate the use cases. Through simulation, the modeled scenarios can test the interactions between all participants within the communication network.

The developed systems have a satisfying relationship with the requirements they meet. A *System of System* element is a generalization and has two or more systems as parts.

It should be noted that there are a number of other approaches to the requirements engineering process. However, it is crucial that the variety context will be considered during the requirements engineering process. Contextual information has to be added in all other approaches as well. The ACRE ontology has already been successfully applied in many projects over several years using the SysML [11, 13, 14]. Due to the well-known and proven approach of applying ACRE with a variety context in a model-based way during system development, the ACRE approach was chosen for the development of an SoS.

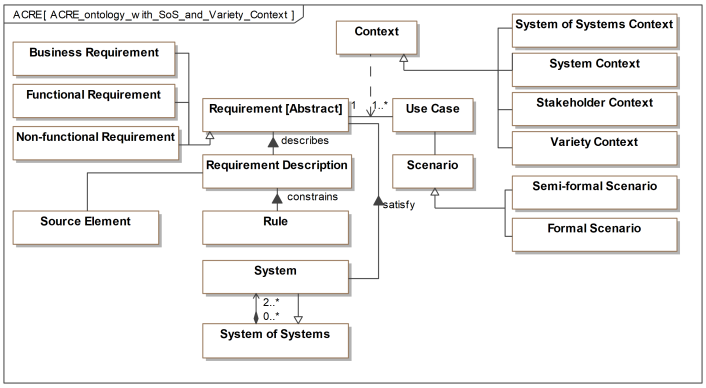

Figure 7: ACRE ontology with systems of system and variety contexts

5.2 SoS Development

Heurist can be used to create variant database instances and is realized as a client-server architecture. Although established design patterns are missing in the still evolving software, further development tends in the direction to have Heurist fully operational as an *Model-View-Controller (MVC)* application. The MVC separates an application into three main logical components: the model, the view, and the controller. Each of these components are built to handle specific development aspects of an application. In the context of creating an information system, the model represents a data scheme, the view a graphical user interface, and the controller accepts user inputs and converts it to commands for the model or view.

The new planned architectural approach is important when it requires adding another layer, the SoS layer. The development is currently still in the conceptual phase. As of now, Heurist is initially used for variant system development and the SoS layer is first tested out through simulations and prototype implementations.

In addition, the Universität Hamburg operates Heurist as a pub-

lic institution, which recommends the use of an adjusted version of the V-model. It follows that further adjustments will be made to Heurist in the area of verification and testing.

6 Modeling and Simulation of an SoS

Modeling and Simulating of an SoS using the SysML and the tool Cameo Systems Modeler has the advantage to test the system's behavior before implementation because the specification is executable. "This quality of executable specifications promises to remedy the most serious problem of software – its lack of correctness and reliability." [31]

In the following we present how to develop an executable specification for an SoS during the requirements engineering process.

6.1 Requirements Profile

For the special requirements (business, functional, and non-functional), new stereotypes were defined as an extension of the *Extended Requirement* stereotype. An *Extended Requirement* is a standard requirement extension that adds some properties to a requirement element. The requirements are devised in accordance with the ISO 29148:2011 and RFC2119.

6.2 SoS Profile in SysML

For representing Systems and SoS the new stereotypes *System* and *System-of-Systems* are defined as extensions of the *Metaclass Class*, see Figure 8.

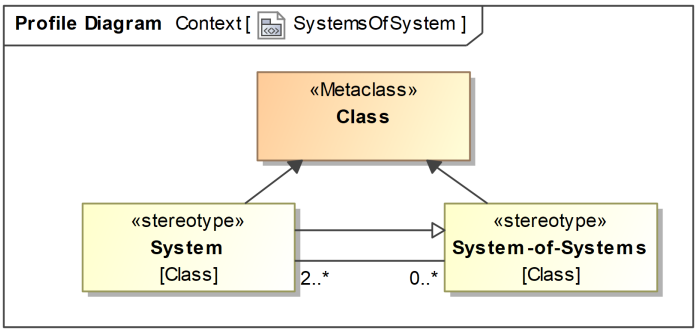

Figure 8: Profile Diagram: new stereotypes *Systems of System* is a specialization of a system and has an association to the stereotype *System*

6.3 Variety Profile

The VAMOS profile which is presented in Figure 1 is used for variant modeling. Figure 9 presents a concrete application of VAMOS.

The package *Variation 1* has the stereotype *Variation* and contains the System *Heurist*. The packages V1, V2, and V3 are variants of *Variation 1*. The variant *V1* contains the System EDAK, the variant *V2* contains the system TheDefix, and the variant *V3* contains the system NETamil, respectively.

One way to introduce a redundancy-reduced communication interface for all variants is to add a SysML port element to the Heurist system. All variants inherit the port via the specialization. However, if an external system were added to the aggregated system at a later point in time, a separate communication interface would have to be implemented for this external system. This is precisely the crux of the matter. If a communication interface is to be offered for internal variant systems as well as for external systems, the communication interface should be inherited by the systems via a specialization using an SoS element.

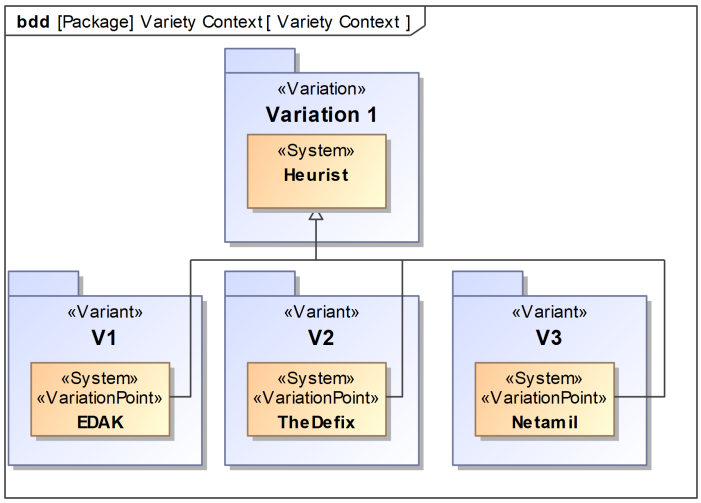

Figure 9: Representation of three variants using VAMOS

6.4 Use Cases

Figure 10 shows the representative use cases for different search functionalities while considering the variety and SoS contexts. The main actor is a CSMC user. The CGRN, EDAK, NETamil, and TheDefix users are specializations of the CSMC user.

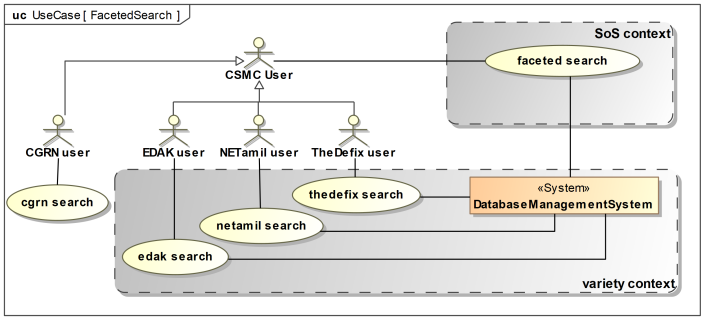

Figure 10: Use case diagram with variety and SoS context

One can see that CGRN users do not belong to the variant context as the other three users but all users also have an association to the use case faceted search. As described in Section 4, the three systems should be developed as variant systems using Heurist, while the development of the CGRN system was done externally. Nevertheless, all systems should be networked so that each system can use the federated search function.

A CSMC user can execute a faceted search. The specialized users can also execute this search while all users have (project-)specific search functionalities, e.g., EDAK users search for specific names mentioned in editions or for object types of inscriptions (use

case: *edak search*), NETamil users look up which word occurs in which poem and in which line (use case: *netamil search*), and TheDefix users want to know the curse id of curses (use case: *netamil search*). The different contexts are presented in SysML use case diagrams.

6.5 Scenarios

Each use case can be validated by one or more scenarios. The scenarios can be represented in behavioral diagrams such as activity or sequence diagrams. We use activity diagrams for modeling and simulating, e.g., the federated search functionality.

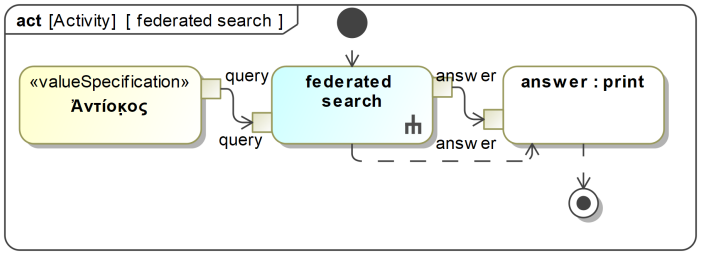

Figure 11: Search for the word “Antiochus” (engl.)

Figure 11 depicts the word “Antiochus” (engl.). It is the input value (=query) for the federated search activity. Behind the federated search activity is a more detailed federation process. As a result, the responses of all databases are printed.

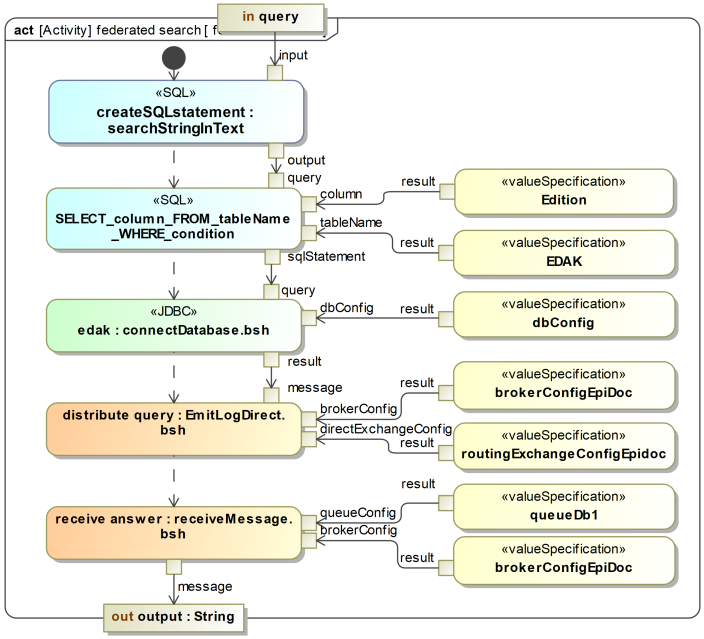

Figure 12: Faceted search actions

Figure 12 illustrates the faceted search process in more detail. The query is the input value. The first action *createSQLstatement:searchStringInText* creates the SQL statement *“Text LIKE ’%Antiochus%’;”*. The action *SELECT_column_FROM_tableName_WHERE_condition* creates the SQL expression *SELECT Edition FROM EDAK WHERE ’%Antiochus%’;*, which is the input value for the next action. The action *distribute query:EmitLogDirect.bsh* sends the SQL expression to a server. The action *receive answer:receiveMessage.bsh* sends a response from a server. The opaque action *EmitLogDirect.bsh* can also contain a forwarding process to another database. To realize a federated search, a script was implemented and must be active on the server side. In fact, the script calls the requests from the server (query queue), processes the schema mapping, and passes the response to the server (response queue). We implemented the server side scripts in Java. The source code is very similar to that of Beanshell (see https://www.rabbitmq.com/tutorials/tutorial-three-java.html). However, other programming and script languages can also be used such as Python, PHP, C#, or JavaScript (see https://www.rabbitmq.com/getstarted.html).

It should be noted here that the scenario at hand already incorporates decoupling of the systems using a communication interface. In a very early phase of system development, communication could take place directly with the database. And yet, communication interfaces are to be used in the development of SoS. Briefly put, this has already been taken into account in the scenarios. As intended by ACRE, the use cases were validated by the scenarios during the requirements engineering process.

6.6 Communication Interface

Communication interfaces ensure the coverage of the need for information and are used for data exchange. For creating communication networks, RabbitMQ is used as an *Application Programming Interface* (API) for SoSs. RabbitMQ offers broker federation and therefore allows the exchange between source and destination brokers, or from a message queue in the source broker to an exchange in the destination broker (see [15]). To model these communication interfaces the stereotype *interfaceBlock* is used. One port of the SoS has at least this communication interface to establish a communication network between the systems which are part of the SoS.

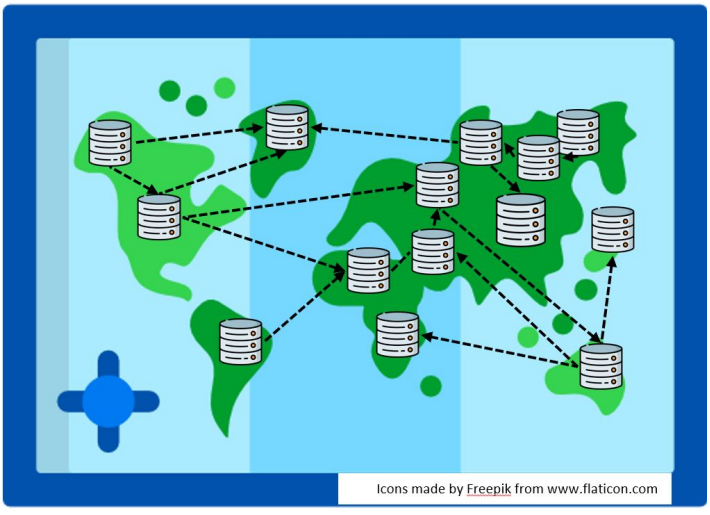

Figure 13: Federated Search Network

Figure 13 illustrates a communication network between loosely-coupled systems which are to be transferred to an aggregated system including a communication interface. By coupling the systems, e.g., federated searches can be realized. The idea is to provide each participant with its own RabbitMQ message broker to easily realize this communication network.

6.7 *Structure of the CSMC Information System*

Figure 14 illustrates the structure of the SoS named *CSMC Information System*. The SoS has the parts of systems *EDAK*, *TheDefix*, *NETamil*, and *CGRN*. These systems are also specializations of the SoS and inherit all activities of the SoS. In this case, federated search is part of each system. The SoS has a communication interface which is modeled as a port. The systems *EDAK*, *TheDefix*, and *NETamil* also inherit all elements of the system *Heurist*. The system *CGRN* is not a specialization of the system *Heurist* and thus does not inherit all activities of the Heurist system, but only those of the SoS.

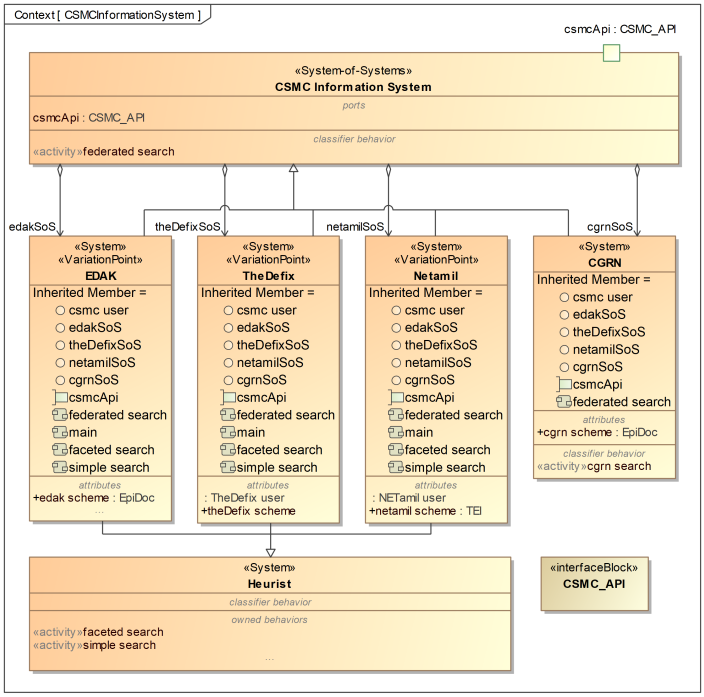

Figure 14: CSMC Information System

Figure 15 gives another overview about the dependencies *generalization* and *inherit* members between the SoS and the systems. In the allocation diagram it can be seen at a glance which systems inherit which activities or which do not. When adding more activities to an SoS or when adding more external systems, this overview can be used to quickly determine which elements will be added to a system when it becomes part of an SoS.

In the development of interfaces, the allocation diagram is an excellent way to illustrate the dependencies of all the systems involved. In the diagram, the separation between the interfaces of the SoS or other interface dependencies can be clearly highlighted.

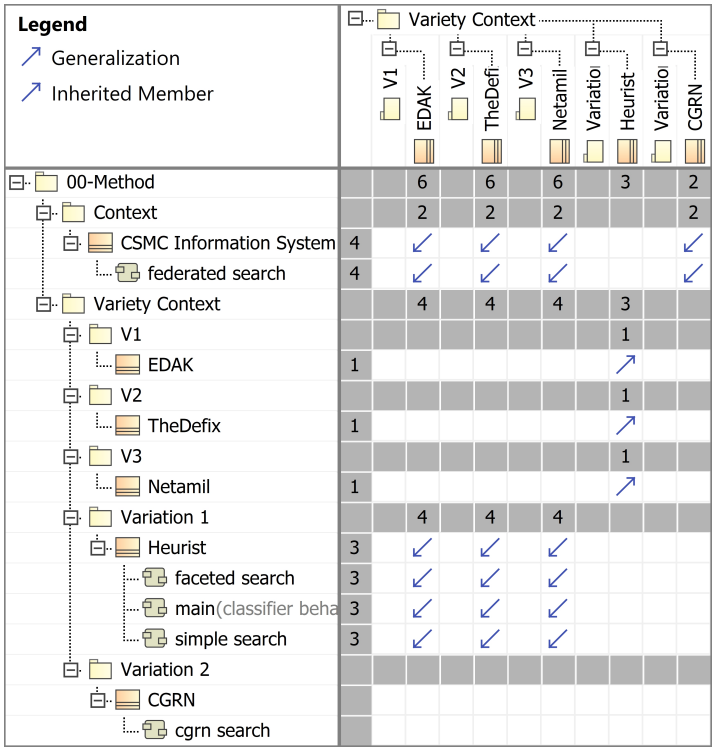

Figure 15: Allocation diagram which represents the dependencies *generalization* and *inherit* members

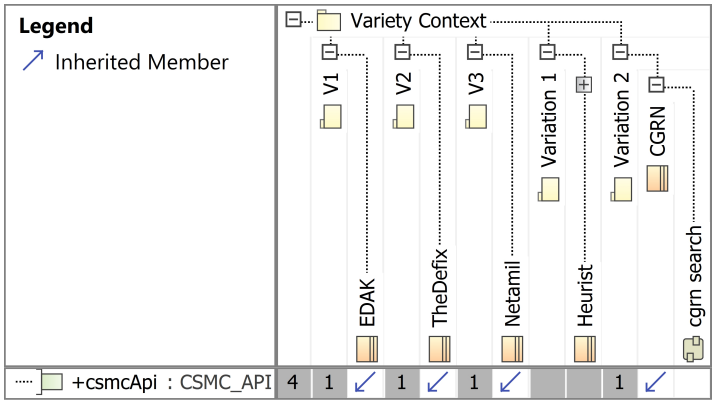

Figure 16: Allocation diagram which represents the systems which have the SoS communication interface

7 Application and Results

We evaluate our approach by a feasibility study. For this purpose, we use a notebook where the tool Cameo Systems Modeler (version 2021x) and the broker-based SysML Toolbox, a RabbitMQ server (version 3.8.9), and MariaDB (10.5.6) are installed. We emulate the databases EDAK, and NETamil on the database MariaDB which represents the Heurist database instances. On a Raspberry Pi 4 we also installed a RabbitMQ server and MariaDB where the database CGRN is simulated. Both RabbitMQ servers are configured with particular message queues, exchanges, and bindings as follows.

The message queues *queueDb1* for EDAK, *queueDb2* for NETamil, and *queueDb3* for CGRN are defined. They are all in the

same virtual host *dbFederatation* (see Figure 17).

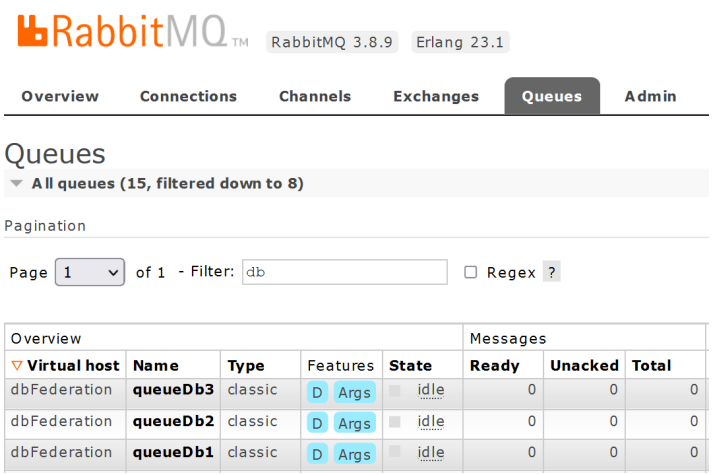

Figure 17: Defined queues on a RabbitMQ server

The exchange is called *db.direct*. The bindings with the particular routing key are: queueDb1 → epiDoc, object, query; queueDb2 → object, queueDb3 → epiDoc, object.

The EDAK data model is represented by the entity type "description." A description has the attributes "identifier", "description_id", "edition", "category", "region", "location", "find spot", "text", and "date." Each "description" has the unique identifier "description_id."

The CGRN data model is represented by the entity type "description." A description has the attributes "idno", "date", "provenance", "support", "layout", "bibliography", "text", "translation", "traduction", "commentary", "publication", "authors", and "project director."

The NETamil data model is represented by the entity types "poem", "commentary", and "dictionary." A poem has the attributes "edition", "transliteration", "word_by_word_translation_into_english", "translation_into_english", and "source."

We simulate federated searches, such as presented in Figure 12. During the simulation the query *SELECT Edition FROM EDAK* is sent to the EDAK database via the opaque action *edak:connectDatabase.bsh*. The SQL expression is published via the opaque action *distribute query EmitLogDirect.bsh*. The databases EDAK receives this expression via the opaque action *receive answer:receiveMessage.bsh*. The database CGRN is queried with the same SQL expression because of the defined routes in the RabbitMQ servers. For the search query "Antiochus", written in Greek language, (*SELECT Edition FROM EDAK WHERE '%Antiochus%';*) we received 1 answer from EDAK and 0 answers from CGRN. For the search query "Zeus", written in Greek language, (*SELECT Edition FROM EDAK WHERE '%Zeus%';*) we received 5 answers from EDAK and 1 answer from CGRN (see Figure 18).

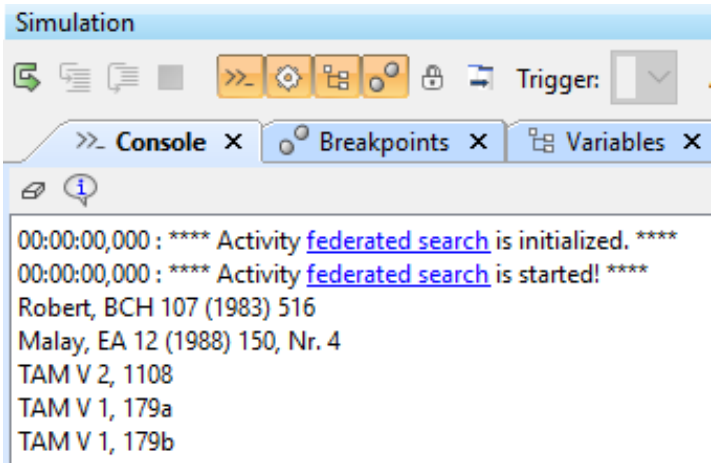

Figure 18: Database results for the search query "Zeus" written in Greek language

For the search query *SELECT COUNT (e.Date) AS Number, e.Date AS Date FROM EDAK e GROUP BY e.Date* we receive the following results (excerpt):

```
+--------+----------------+----------+
| Number | Date           | Database |
+--------+----------------+----------+
|     31 | 4. Jh. v. Chr. | EDAK     |
|    200 | 1. Jh. n. Chr. | EDAK     |
|    818 | 2. Jh. n. Chr. | EDAK     |
|    642 | 3. Jh. n. Chr. | EDAK     |
|    156 | 4. Jh. n. Chr. | EDAK     |
+--------+----------------+----------+
|      1 | ca. 250-200 BC | CGRN     |
|      2 | ca. 350-300 BC | CGRN     |
+--------+----------------+----------+
```

The responses returned by EDAK and CGRN show that the date is differently represented in both databases. The date differs in language and representation (indication as century or year). When the query is filtered by year, one of the two databases returns an empty result set as response. A translation of the date representations can lead to a complete answer. A mapping between the representation of the date is required to ensure *correct* query results. Schema mapping is generally required when defining federated search queries.

This simulation example also presents that queries from EDAK are answered using both the EDAK and the CGRN databases. NETamil is not involved in this specific query process because of the missing routing key in the RabbitMQ configurations. At this point it makes no sense in terms of content. If one wants to compare another repository with Tamil poems, a route can be defined via the RabbitMQ configuration that supports the sensible federated search. In this example, the Cameo System Modeler's console represents the CSMC information system, which receives all responses from the various databases from the federated search. If either a Heurist database instance or an externally developed system is to be aggregated to the SoS, this can be realized by installing a RabbitMQ broker, programming a script for publishing and receiving messages from the broker, and setting the broker configurations.

In this way, new databases can be added so that our principle "*bring your own database* is supported. Then all new systems of the

SoS will also benefit from the federated search. Consider that the challenges of information integration must be resolved for mapping to use federated searches successful.

In the humanities, as well as other fields, it is important that existing functions such as federated searches can be used without large expenditure of resources. After all, resources are limited. Existing data can thus be enriched with further information in a short time and users can focus more on editing content. Using the allocation diagram to keep an overview of all systems to be aggregated also helps to keep track of the growing number of systems (cf., Figure 15 and Figure 16). All systems, whether variants or not, can be specifically developed and integrated into the overall SoS. Without considering this overview, existing systems could be produced mistakenly from scratch simply because they are unknown and as such a variant is regarded as an external (unregistered) system. As a positive effect of the present approach, one has the advantage of being able to cooperate with external parties whether their system can also be developed as a variant of one's own system, so that the same communication interface (*csmcInterface*) can be used.

At the Universität Hamburg Heurist has already been set up and it is being used for the autonomous development of information systems. More than ten information systems have already been created. When changing functions, such as the web page design, the allocation diagram can be used to see which systems are affected.

The special feature of the model-based documentation of SoSs and the variants with VAMOS make it convenient for the developers to get an overview between the different diagram types and the filter properties for displaying data. Developers also see which systems are relevant and whether changes have an impact on the system properties or not. It is self-explanatory to switch between a display of specific SoS or variant elements, or view everything together in one diagram, as shown in Figure 15. The model-based and variant-oriented approach also ensures that the interfaces of the systems are not developed redundantly. The realization of a communication interface using RabbitMQ supports the aggregation of decoupled systems by implementing message scripts that are publicly available.

It is planned to transfer the prototypically implemented CSMC Information System with the presented communication interface into a product. Regarding the product development, the variant-oriented approach points in the right direction as to even more relevant parameters, such as performance and security attributes, that should be tested in advance.

The approach at hand also fits in other areas of system development, e.g., in the aviation industry. There, it has already been shown that the networking of systems after the digitization of business processes can be helpful to automate ordering processes between supply chain tiers [32]. It would be conceivable to design the ordering process as part of the SoS and thereby identify the variants in the ordering process as well as the external systems that want to become part of the SoS. As an assumption, there will be more individual solutions that will be aggregated into an SoS. Here, too, the approach offers the advantage of keeping an overview of all systems and working towards a common interface in a targeted manner so that the connection to the SoS can be made with little resource effort.

8 Conclusion and Outlook

In this paper, we presented how to develop an aggregated system, which, understood as an SoS, was put into practice in a model-based and variant-oriented way. The aim was to identify the number of variants easily at an early stage of the requirements engineering process so that the development of elements has neither to be done holistically nor redundantly. For this purpose, we used the ACRE ontology with the extended contexts on SoS and variety, and applied this approach with the SysML tool Cameo Systems Modeler as well as the VAMOS profile. In addition, we defined an SoS profile which helped us to distinguish between developing variants and merging variants as well as non-variants into an SoS. For the implementation of a communication interface, RabbitMQ was used as a message broker, which allows loosely coupled systems to be brought together in a simple way. The variant database systems were developed with Heurist which on the one hand supports the development of automated database systems and on the other hand keeps the heterogeneity under control, often resulting from the many requirements. The prototype implementation showed us that this path is promising and should be further pursued.

The advantage of merging multiple database systems is that functions such as a federated search can be implemented, however, the problem of data integration between all the database instances must be solved beforehand so that a search query does not lead to a faulty response. Therefore it must be ensured that the data or their representations have the same syntax, semantics, model, and access. At the CSMC, a feasible study is currently in progress.

Conflict of Interest The authors declare no conflict of interest.

References

[1] E. Werner, "Clay Tablet (AO 29196) from the Louvre in Paris (3D model)," 2020, doi:http://doi.org/10.25592/uhhfdm.918.

[2] E. Werner, "Clay Tablet (KUG 15) from the University Library Giessen (3D model)," 2020, doi:https://doi.org/10.25592/uhhfdm.766.

[3] D. Krause, G. Beckmann, S. Eilmus, N. Gebhardt, H. Jonas, , R. Rettberg, "Integrated Development of Modular Product Families - a Methods Toolkit," in In T.W. Simpson, J. Jiao, Z. Siddique, K. Hölttä-Otto (Eds.): Advances in product family and product platform design: Methods & applications, 245–269, Berlin Springer, 2014.

[4] O. C. Eichmann, S. Melzer, R. God, "Model-based Development of a System of Systems Using Unified Architecture Framework (UAF): A Case Study," in Proceedings of 2019 IEEE International Systems Conference, IEEE, 2019.

[5] C. B. Nielsen, P. G. Larsen, J. Fitzgerald, J. Woodcock, J. Peleska, "Systems of Systems Engineering: Basic Concepts, Model-Based Techniques, and Research Directions," 2015.

[6] M. W. Maier, "Architecting principles for systems-of-systems," Systems Engineering, **1**, 267–284, 1998.

[7] S. Melzer, H. Peukert, H. Wang, S. Thiemann, "Model-based Development of a Federated Database Infrastructure to support the Usability of Cross-Domain Information Systems," in Proceedings of 2022 IEEE International Systems Conference, IEEE, 2022.

[8] O. C. Eichmann, S. Melzer, M. Hanna, R. God, D. Krause, "A Model-Based Approach for the Development of Modular Product Families Considering Different Life Phases," in Proceeding EMEA Systems Engineering Conference, EMEASEC 2018 / TdSE 2018, 2018.

[9] S. Melzer, S. Thiemann, R. Möller, "Modeling and Simulating Federated Databases for early Validation of Federated Searches using the Broker-based SysML Toolbox," in Proceedings of 2021 IEEE International Systems Conference, IEEE, 2021.

[10] T. Weilkiens, Variant Modeling with SysML, MBSE4U Booklet Series, 2016.

[11] T. Bahns, S. Melzer, R. God, D. Krause, "Ein modellbasiertes Vorgehen zur variantengerechten Entwicklung modularer Produktfamilien," in Tagungsband zum Tag des Systems Engineering (Eds.: Chr. Muggeo, S.O. Schulze), Carl Hanser Verlag GmbH & Co. KG, 2015.

[12] R. God, S. Melzer, U. Wittke, "SiLuFra Schlussbericht - Sichere Luftfracht-Transportkette: Konzepte, Strategien und Technologien für sichere und effiziente Luftfracht-Transportketten; Teilvorhaben: Modellbasierte Architektur- und Lösungsspezifikation; Laufzeit des Vorhabens: 01.07.2013 - 31.08.2016," 2016.

[13] R. God, S. Melzer, "Teilvorhaben: Spezifikation und Integration cyber-physischer Betriebs- und Geschäftsprozesse : Schlussbericht : Laufzeit des Vorhabens: 01.05.2016-30.09.2019 : Berichtzeitraum: 01.05.2016-30.09.2019, Spezifikation und Integration cyber-physischer Betriebs- und Geschäftsprozesse," Technical report, Technische Universität Hamburg, Institut für Flugzeug-Kabinensysteme, Hamburg, 2020, doi:10.2314/KXP:1726105857.

[14] R. God, U. Wittke, S. Melzer, C. Witte, "KomKab Schlussbericht - Kommunizierende Kabine; Teilvorhaben: Digitaler Ramp-Agent; Laufzeit des Vorhabens: 01.01.2016 - 31.03.2019," 2019.

[15] S. Melzer, J. P. Speichert, O. C. Eichmann, R. God, "Simulating cyber-physical systems using a broker-based SysML Toolbox," in AST 2019 - 7th International Workshop on Aircraft System Technologies, Hamburg University of Technology, 2019.

[16] D. Arndt, S. Melzer, R. God, M. Sieber, "Konzept zur Verhaltensmodellierung mit der Systems Modeling Language (SysML) zur Simulation varianten Systemverhaltens," in Tagungsband zum Tag des Systems Engineering (Eds.: S.O. Schulze, C. Tschirner, R. Kaffenberger, S. Ackva), Carl Hanser Verlag, 2017.

[17] S. Melzer, R. God, T. Kiehl, R. Möller, M. Wessel, "Identifikation von Varianten durch Berechnung der semantischen Differenz von Modellen," in Tagungsband zum Tag des Systems Engineering (Eds.: M. Maurer, S. O. Schulze), Carl Hanser Verlag GmbH & Co. KG, 2014.

[18] J. Holt, S. A. Perry, M. Brownsword, Model-Based Requirements Engineering, volume 9 of *Professional Applications of Computing Series*, Institution of Engineering and Technology, Stevenage, 2012.

[19] O. C. Eichmann, S. Melzer, F. Giertzsch, R. God, "Stakeholder Needs and Requirements Definition During Service Development in a System of Systems," in Proceedings of 2020 IEEE International Systems Conference, IEEE, 2020.

[20] J. Holt, S. Perry, R. Payne, J. Bryans, S. Hallerstede, F. Hansen, "A Model-Based Approach for Requirements Engineering for Systems of Systems," IEEE Systems Journal, **9**(1), 252–262, 2015, doi:10.1109/JSYST.2014.2312051.

[21] J. Holt, S. Perry, SysML for Systems Engineering: A Model-Based Approach, Computing, Institution of Engineering and Technology, 2018.

[22] A. Qpid, "Messaging built on AMQP," https://qpid.apache.org/releases/qpid-cpp-master/cpp-broker/book/chap-Messaging_User_Guide-Broker_Federation.html, 2015, accessed January 22, 2022.

[23] S. Melzer, O. C. Eichmann, H. Wang, R. God, "Modeling and Simulation of Database Interactions," Tag des Systems Engineering 2021 (TdSE2021), 2021, doi:10.25592/uhhfdm.9696.

[24] S. Schiff, S. Melzer, E. Wilden, R. Möller, "TEI-based Interactive Critical Editions," in 15th IAPR International Workshop on Document Analysis Systems, Lecture Notes in Computer Science (LNCS), Springer, 2022.

[25] T. Elliott, G. Bodard, E. Mylonas, S. Stoyanova, C. Tupman, S. Vanderbilt, et al., "EpiDoc Guidelines: Ancient documents in TEI XML (Version 9)." Available: https://epidoc.stoa.org/gl/latest/., (2007-2022), accessed January 22, 2022.

[26] J.-M. Carbon, S. Peels, V. Pirenne-Delforge, "A Collection of Greek Ritual Norms (CGRN)," Liège, http://cgrn.ulg.ac.be, consulted in 2020, 2016–2020, online; accessed 10 December 2021.

[27] W. Hao, S. De-wen, F. Xujian, X. Haitao, "Application of information fusion technologies for multi-source data," Journal of chemical and pharmaceutical research, **5**, 2013.

[28] U. Leser, F. Naumann, Informationsintegration: Architekturen und Methoden zur Integration verteilter und heterogener Datenquellen, dpunkt.verlag GmbH, 2007.

[29] ISO/IEC/IEEE 29148:2011(E), "Systems and software engineering - Life cycle processes - Requirements engineering," https://wiki.unix7.org/_media/ict/lib/iso-iec-ieee-29148-2011.pdf, accessed January 22, 2022.

[30] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels," Harvard University, https://datatracker.ietf.org/doc/html/rfc2119, 2017, accessed January 22, 2022.

[31] N. E. Fuchs, "Specifications are (preferably) executable," Softw. Eng. J., **7**, 323–334, 1992.

[32] H. Wang, S. Melzer, "Simulation of Ordering Processes across different Supply Chain Tiers in the Aviation Industry," in 2022 IEEE International Systems Conference (SysCon) (IEEE SysCon 2022), Montreal, Canada, 2022.