Data Management Architecture for Service-oriented Maritime Testbeds

Julius Möller, Dennis Jankowski, Arne Lamm and Axel Hahn

Abstract In recent years, numerous new approaches that rely on data-intensive methods have been developed for maritime assistance systems, leading to a compelling need for more elaborate verification and validation procedures. Modern testbeds that can meet these demands are often developed separately from the system itself and provided as generically usable services. However, the joint usage of such testbeds by multiple stakeholders from research and industry confronts them with various challenges in terms of data management: Data control and protection is required to preserve possible competitive advantages or comply with legal framework conditions. The resulting decentralization in data management complicates collaboration, especially in the joint processing and analysis of testbed data. In this paper, we present a decentralized software system, which can deal with these challenges by modelling interrelationships between the stakeholders in a data space, considering their various interests. With the help of a modular data management architecture, the organization of a testbed data basis, as well as the support of verification and validation processes and the evaluation of data streams is made possible. This is achieved with a workflow model for mapping complex and distributed data processing steps. We demonstrate the applicability of the system in an application scenario for the development of a maritime assistance system.

***Index Terms*—testbed, data ecosystem, data management, maritime data, verification and validation, data space**

I. Introduction[1]

THE utilization of different types of testbeds to support all steps of development and operation in engineering processes of complex systems is a standard procedure. According to IEEE a testbed is defined as "an environment containing the hardware, instrumentation, simulators, software tools, and other support elements needed to conduct a test" [1]. As new technological developments approach modern engineering processes, the complexity and the architecture of testbeds has changed in the past years. Especially in the area of vehicular testbeds, that are utilized in engineering processes for cars, planes, vessels etc., the availability of new sensor technology and computational power has influenced requirements for testbeds drastically: With an increasing amount of data being processed in all areas of engineering, new challenges emerge for the processes of data management and distribution. Particularly, the maritime domain is currently experiencing a shift of paradigms with the advent of Maritime Autonomous Surface Ships (MASS), remote shipping operations and highly available or high bandwidth communication channels such as IP-over-Satellite, LTE or VDES.

The complexity of systems that build on these technologies often requires a deep understanding of the environment in which they act before the development process can start [2]. Nowadays, data-driven analysis of environmental data is frequently used to build models and to gain knowledge about the system's domain. Furthermore, as different (existing) sub-systems are often combined to deal with complicated tasks, such as collision avoidance or object identification, a collaboration of different parties from industry and academia for their development is not unusual (cf. [3]). This has multiple implications for the joint usage of a single testbed by different stakeholders, such as conflicts of interests. Additionally, monitoring and analyzing the behavior of several different sub-systems, that are organized in a decentral setup is a highly complex task and adds a significant amount of effort to the amount of work that is already required to develop the system itself. Therefore, the development of testbeds is often separated from the actual system development. Previous work has shown that generically usable, maritime testbed architectures are already able to deal with the integration of complex Systems-of-Systems into an existing testbed from an operational perspective (e.g. in [4]). However, the task of managing produced data artifacts from environmental sensors, as well as data from a system under test and making them securely

[1] This paper was submitted for review on May 27, 2022.

Julius Möller is with the Carl von Ossietzky University of Oldenburg, 26129 Oldenburg, Germany (e-mail: julius.moeller@uol.de).

Dennis Jankowski is with the German Aerospace Center (DLR) SE, 26121 Oldenburg, Germany (e-mail: dennis.jankowski@dlr.de).

Arne Lamm is with the German Aerospace Center (DLR) SE, 26121 Oldenburg, Germany (e-mail: arne.lamm@dlr.de).

Axel Hahn is with the German Aerospace Center (DLR) SE, 26121 Oldenburg, Germany (e-mail: axel.hahn@dlr.de).

available only to authorized parties was yet entirely left to the users.

In this article, we discuss how decentrally organized maritime data can be made available for a data-driven research and development process in the testbed context. First, the basic methods of research and development data management will be discussed and a connection to data management concepts in maritime testbeds is established. We represent the testbed as a data ecosystem, considering modelling it on architectural level as a data space and propose an architecture for the management of a testbed data basis, as well as for the support of verification and validation processes and the evaluation of data streams from testbed data. In contrast to existing data management systems in testbeds, which are not able to deal with decentralized data sources and processing or multiple stakeholders, we base our concept on a distributed workflow model for mapping complex data processing activities, while preserving the stakeholders' sovereignty over their data. Also, the proposed architecture is aimed at a service-oriented usage by following the Testbed-as-a-Service paradigm. The system is secured by a testbed-wide authentication of users and a fine-grained authorization mechanism for the available data resources. Finally, the designed system is prototypically implemented and evaluated in a case study.

II. The Testbed as a Data Ecosystem

In the recent years, the paradigm of Everything-as-a-Service (XaaS) has extensively been used in the domain of modern systems and software development, especially in the area of systems using Service Oriented Architectures (SOA) [5] and cloud infrastructures [6]. The process of providing specific functionalities as services on-demand has the potential to deliver high performance and easy access to relevant resources at a low cost [7]. Systems that are based on SOA can bring together functionalities that are often distributed over different physical locations or under the control of different entities. This pattern also has been adapted in the area of testbeds. A generic testbed or a Testbed-as-a-Service architecture is able to dynamically integrate a System under Test into its SOA [8], [9]: Several services for monitoring and analyzing the SuT are provided and can easily be utilized by the users of the testbed to test, monitor or evaluate their systems, which reduces the work of implementing a testbed for a system in parallel to the system development itself. However, opening a testbed for multiple users in a service-oriented manner introduces several challenges for the management of data resources, which are a main concern in the service-oriented testbed. The following sub-sections will give an overview of these challenges.

A. Data Management Tasks in Testbeds

Vehicle development is becoming more and more complex as the level of technology increases. Manufacturers are no longer only concerned with hull development [10]. Instead, the pure driving functionality and the technical systems required for this, such as control units, are becoming subdivided more and more into technical and logical system development [11]. To deal with the complexity and to accelerate development even with additional effort, parallelization is taking place in the form of partitioning the individual components, which leads to integration effort when the system is put into operation. This type of system development is also known as simultaneous engineering [12], often involving multiple contractors that collaborate in a system development process. In order to be able to implement parallel development, a higher level of coordination is required upstream via interfaces and information flows, which is also maintained during system development. Accordingly, coordination likewise takes place less over documents but mainly over product models especially in model driven development. This kind of the model-based systems engineering has further advantages, which are meaningful for the integration, test and later also for the operational phase in addition to the design phase. The wide-spread approach of X-in-the-loop testing requires well-defined interfaces and strict modularization in order to gradually replace the initially virtual components with physical subcomponents and still be able to maintain the data flows [10]. Additionally, the costs involved in the development of an environment capable of providing means to evaluate complex assistance systems is also a driving factor for the service-oriented and shared usage of testbeds.

Therefore, a core contribution of a testbed is the provision of a data management system with the services required for the entire system development process from the design phase to the operational phase. Accordingly, the services for the evaluation of a SuT, are divided into the categories "Modeling and Knowledge generation", "Verification and Validation" and "Operation and End-to-end Certification":

- **Modelling and Knowledge generation**: Starting at the early stages of system development or even before that, a testbed can be utilized to provide environmental data for the generation of knowledge or building models of the environment (cf. [13], [14]). In the maritime domain, traffic data, weather data or data from test carriers may, e.g., be used to gain insights into traffic flow patterns, vessel behavior or potential risk factors from weather. These insights are required by new approaches for the development of traffic management systems [15], vessel behavior prediction methods [16] or weather-based route planning models [17]. Managing this type of data comes close to classical approaches for data management and should be based in accordance to common data management principles, such as the FAIR guidance principles [18].
- **Validation and Verification (V+V):** According to [19], validation is the process of checking whether the expectations of stakeholders are satisfied by a system and verification is the process of checking whether the defined requirements are fulfilled by the system. Testbeds are commonly utilized in both processes as a controlled and well-observable environment during the system development

process. Test outcomes depend on the SuT behavior during a test, which is often assessed with the help of data that is generated by the SuT and environmental sensors. As different SuTs introduce a broad range of technologies into the testbed, managing data from V+V activities is a new challenge. Especially recording and analyzing data during field tests comes with the difficulty of streamed data, that requires procedures to manage it during the runtime of a test. As the developed systems get more and more complex these days (e.g., by the usage of AI technologies), defining strict rules for testing single functionalities of a SuT is not always applicable. A well-known approach to deal with these issues is the scenario-based testing, where representative test scenarios are used as a reference for real-world occurrences and are derived from historical data [20] or by expert knowledge [21].

- **Operation and End-to-End Certification:** When the system development process is completed, a testbed may be used for certification. Regarding data management, it is, e.g., recommended by certification agencies to store and document data for the certification of autonomous and remotely operated ships [22]. Additionally, functions in developed systems are also constantly being improved and expanded even after approval and during operation. For this reason, another challenge is just coming up for testbeds during operation: In addition to the general monitoring and development of triggers to monitor correct operation, the question of the reliability of over-the-air updates of critical software artifacts plays an increasingly important role, followed by the assurance of certification in the form of module-based post-certification. Also, testbeds are commonly used for demonstration of specific systems in a controlled environment (see e.g. [23]), where a specific, defined set of data is exported and presented or visualized.

B. Data Ecosystems

When it comes to the exchange of data between many different stakeholders in a network of independent actors, a central exchange point for sharing or trading data is often hard to establish. Reasons can be trust issues [24], loss of sovereignty [19] or legal issues such as data protection regulations, organizational or technical challenges and many others [26]. Therefore, decentral structures to share and exchange data are often utilized. According to Hugoson [27], decentralization means that involved parties in a system can make decisions locally. In the case of a testbed there may be different stakeholders with their own data requirements: A testbed user typically introduces their own data sources to the testbed, that are required for the SuT to work as expected. On the other hand, they may also request data from the testbed infrastructure, which is provided by the testbed operator / owner. Nevertheless, both stakeholders can decide on their data sources locally. Furthermore, Hugoson emphasizes in [27], that a system in a decentralized structure must fulfill some requirements that enable them to interact with other systems in the same structure. Otherwise, the system would be isolated from the decentralized structure.

A more specific model of the interaction in a decentralized data exchange network is given by the model of the *data ecosystem*. In general, this term describes an overarching, socio-technical network [28]. Oliveira et al. conducted a literature review in [29] to elaborate the main aspects of a data ecosystem. It is mentioned that a data ecosystem consists of multiple actors that have different interests, capabilities, and requirements and thus embody various roles. An actor can represent one or more roles at the same time and is connected to other actors through cooperation or competition. Additionally, each actor consumes, produces or provides data and other related resources such as services or infrastructure to the ecosystem [29]. Finally, data exchange is commonly based on standardized interfaces, licenses, and quality examinations. According to these aspects, a generic testbed with multiple collaborating parties can also be seen as a *data ecosystem*. A detailed analysis of these aspects for a service-oriented testbed is conducted in the following subsection.

C. Stakeholders in the Testbed

Inside a data ecosystem, different interests need to be considered. In the literature, role models have been discussed to categorize those interests and represent their relations. In accordance to the work of of Xu et al. [30], Oliveira et al. [31] and the International Data Space (IDS) reference architecture [32], we assume that stakeholders in data ecosystems can be categorized into some forms of a data providers, who by any means generate and/or provide the data and data consumers that use the data for their purposes. Most other stakeholders in the models of Xu et al. and the IDS are intermediaries in the process of collaborative data processing. These intermediaries seem to be specific to the architecture and fulfil tasks like data (service) distribution, data discovery and establishment of trust. Subsequently, we assume that the stakeholders in a data ecosystem can be characterized by the roles of the data provider, data consumer and intermediary. These roles may include some more specific sub-roles, which are not considered in detail in the continuation of our work.

A mapping of these abstract roles to stakeholders in the testbed is depicted in Figure 1: Starting with the data providers, the testbed owner can be seen as a classical, long-term data provider in a testbed. As the testbed is operated by its owner, statically available data sources, such as sensors or historical data bases are made available by the owner to the users of the testbed. In the maritime domain, typical static sensors would, e.g., include shore-based AIS, RADAR, cameras, or weather sensors. These static data sources are typically consumed by the users of the testbed. However, the generic sensors of the testbed owner may not fulfill the requirements of all users in the testbed. In these cases, users would integrate their own equipment to the testbed and provide it to authorized other testbed users, for collaboration in system development. It is important to notice that testbed users may want to use a testbed collaboratively, but not share

This article has been accepted for publication in IEEE Open Journal of Intelligent Transportation Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/OJITS.2022.3207235

all their data with other participants. This is due to the fact, that data may represent a valuable asset in the economic competition, which should not be exposed to competitors. To avoid conflicts and facilitate data exchange, the testbed owner normally has an interest in providing a neutral communication infrastructure, and therefore also acts as an intermediary in the testbed data ecosystem and fulfills the role of a data trustee. Sometimes this role can also be outsourced to a trusted third party.

In the context of testbeds, we generally differentiate between two kinds of data consumers: testbed users and observers. Depending on the type of use, the testbed users can be further categorized into different subgroups. Thus, a testbed user may be interested in the data of the testbed itself, e.g., to develop models or to identify patterns in the data that can be obtained from the testbed. This mainly applies to data scientists or data engineers. However, a testbed user may also be interested in evaluating a SuT by using the testbed infrastructure. This aim is more likely to be pursued by system engineers or data management engineers. A system engineer mainly acts as a data consumer, retrieving data from the testbed and feeding it into the SuT. This activity is complemented by the data management engineer, who is initially responsible for providing interfaces for exchanging data with other systems but is also interested in managing data flows within the testbed to analyze the system behavior during test runs. In particular, these stakeholders may have a business background and be interested in maintaining control over exchanged data. Furthermore, there are also entities in the testbed that can be classified as either only data provider or only data consumer: DaaS providers extend the available data in the testbed by providing additional (paid) data resources. In the maritime domain, this could, e.g., be high-precision chart data or weather predictions.

Finally, testbed observers have an interest in obtaining selected subsets of the testbed data and evaluating or visualizing it. Potential actors are experts in a system certification process or observers of a demonstration. It is particularly important for the certifiers, but also for other testbed users, to be able to trace the data provenance. This is the only way to make test processes traceable and reproducible.

D. Data Spaces

A classical architectural pattern to deal with the decentralized structure of stakeholders and their relations in a data ecosystem are so-called *data spaces*. In the original definition, given by Franklin et al. in [33], a data space is defined as a coexistent amount of data, which is connected by a supportive system. This supportive system – the Data Space Support Platform (DSSP) – is the main approach for data integration and management. It must support a wide variety of different data types and formats, offer methods for querying, updating and managing data sources in the data space. Due to the challenges of providing consistent data access in a data space, the DSSP always aims at approximative solutions for its functionalities. In previous work [34], we discussed the challenges of developing such a system. Furthermore, the DSSP must be able to provide solutions to improve data integration over time and also add new data sources to the data space incrementally [35]. This way, the technical and organizational heterogeneities can be considered.

E. Service Oriented Architecture vs. Microservices

Recent trends in service-oriented software development have been altering the concepts of SOA to be more dynamically scalable and more flexible in terms of intra-service communication, deployment, and implementation platforms, which has been leading to the utilization of *microservices*. Microservices are organized as smaller and independent services that run in separated processes and communicate via lightweight communication protocols. They are typically deployed in container runtimes, providing an adaptable abstraction layer per service, that makes it possible to use different implementation platforms for different microservices. Other advantages are the easy setup of microservice-based applications due to their containerization and the reduced complexity by providing simpler functionalities per microservice. [36]

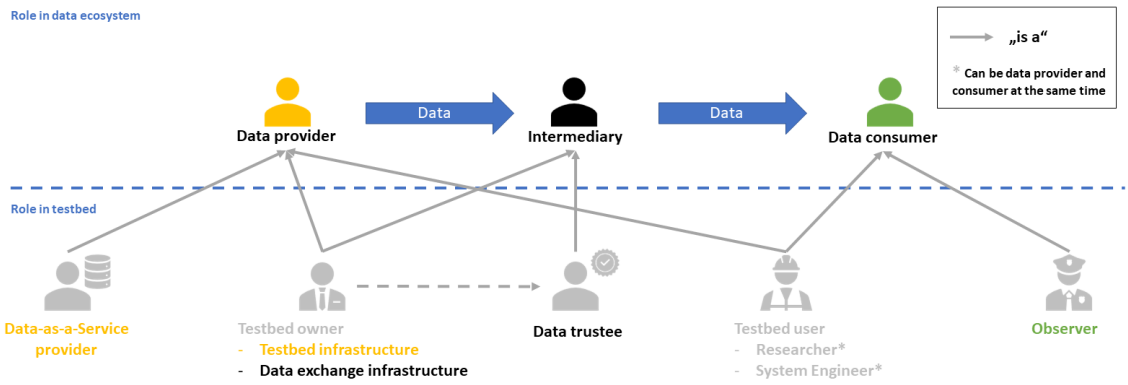

Figure 1: Stakeholders in data ecosystems and their mapping to stakeholders in the testbed.

Furthermore, SOA often requires a central *orchestration component*, that manages process flows and data exchange between different services [37]. From the perspective of a testbed data ecosystem, the more decentralized nature of microservices, which normally do not require central coordination (cf. [37]) and can easier be scaled, is a relevant solution to consider in the development of testbed data infrastructures. As testbeds also are being used in several different configurations, it would make more sense, to put the composition intelligence into to the services itself (referred to as the ability of *choreography* by [36] and [37]), than into a central orchestration component, that would be subject to ongoing change requests by the testbed users, as they frequently introduce new systems to the testbed.
Kubernetes is a frequently utilized platform, that helps to manage containerized microservices on a distributed infrastructure. This includes tasks like automatic scaling, restarting containers, providing networking functionalities or monitoring the containers. [38]

III. Requirements for a Maritime Testbed Data Management System

Based on the main task of enabling data exchange between the testbed infrastructure, its users and between different users respectively, several use-cases can be derived from the different data management tasks and stakeholders, that could be identified in section II. Starting with the testbed owner, it is of importance to provide a centrally accessible infrastructure to the users, which can deal with the decentralized structures in the testbed. Additionally, a testbed owner also wants to use the infrastructure to provide maritime data from static data sources of the testbed. This data is then used by the testbed users for modeling, knowledge generation or as an input to a SuT. The main interest of testbed users is to use the testbed for field tests. Also, for testing complex maritime assistance systems, the approach of scenario-based testing is often utilized, in which the necessary tests are given by representative scenarios [39]. Then, users or systems also need to use the infrastructure to dynamically exchange data from test executions with other entities. These data exchange processes may have to be documented thoroughly for subsequent analyses or certification processes. It should also be possible to only select and retrieve a subset of the processed data, for specific demonstration or certification activities. Finally, enabling external service providers to offer their data products in the testbed would also lead to a more efficient utilization of data in the testbed. Combining the insights from these use-cases leads to the conclusion, that a Testbed Data Management System (TDMS) must fulfill the following requirements:

- **Architectural core requirements**: It must be possible to integrate arbitrary data sources in the testbed to the TDMS and to process data queries on them, regardless of who operates them [40]. To enable easy access, the TDMS must manage the metadata of the available data sources [41] and map queries to them respectively [42]. This functionality should consider the fact, that some of the data sources are only allowed to be accessed by specific parties.
- **Data sharing**: Exchanging data through an intermediate system always involves trusting a third party. Therefore, a data provider must be able to control how exchanged data is being processed by the TDMS and under what conditions it is being done [43]. To maximize the control over its data, a data provider may not transfer large amounts of data from its data basis to the TDMS for a long-term period, but answer data queries directly, making the requested data only available for the requesting user. A role-model helps to distinguish between different access rights for different data sources (cf. [44]).
- **Data processing**: A TDMS should support the entire data lifecycle from acquisition to the final result (cf. [45]). It must be possible to seamlessly integrate data processing steps, that are carried out decentrally in the testbed. Therefore, a TDMS should provide interfaces that can be used by external tools and systems, without losing track of the data manipulation processes [46]. Furthermore, the TDMS must be able to track the execution of field tests and manage scenario or test execution data in the scenario-based testing approach.
- **Trust, security, and traceability**: To resolve trust issues between users of the testbed, the TDMS must provide means to secure communication channels. Furthermore, it must be possible to configure data access rights with different entities in a fine-grained manner (cf. [25]). Additionally, for the sake of data provenance, it should be possible to document the processing chain of a data set (cf. [47]).
- **Maritime domain**: Literature studies have shown, that data-based research and development in the maritime domain is often based on similar data sources, some of the most popular being the Automatic Identification System (AIS), weather data, or technical data from ships [48], [49]. Properties of these types of data (geospatial references, timestamps, etc.) should be utilized to provide better data integration capabilities and pre-processing functionalities by the TDMS.

IV. Related Work

In the following section, related work from several fields is examined. After evaluating conventional data management systems, DMS for data ecosystems are examined. Then in section C, existing maritime testbeds are examined based on a testbed register managed by the International Association of Lighthouse Authorities (IALA) [50]. In addition, a selection of testbeds from the automotive domain was analyzed in section D. In sections C and D only related work that includes dedicated concepts for data management was considered.

A. Conventional Data Management Systems

Several systems for traditional, centralized management of research data have been established and are used by numerous research institutions, often to publish research datasets. A basic comparison of popular systems of this type is presented by Amorim et al. in [51]. Most of the examined conventional research data management platforms like Figshare, Zenodo, CKAN, DSpace, ePrints and EUDAT are aiming at long-term preservation of research datasets in large repositories. These systems manage resulting data at the end of a data-driven process. For the most part, they are also designed to store the corresponding data and make it available as a kind of publication (similar to scientific publications). The documentation or recording of earlier states of the data in the data processing process (raw data, intermediate processing steps) is usually not considered. The mere fact that conventional research data management systems prefer to store the data in a centralized way already contradicts the principle of decentralization in the service-oriented testbed context. The aggregation of all testbed data in a central instance would violate the sovereignty interests of the data providers and it might not be possible to satisfy data protection regulations.

B. Data Management in Data Ecosystems

Current scientific literature also discusses approaches which enable the management of scientifically or industrially used data in a data ecosystem. In one of the earlier works in this area, Elsayed and Brezany describe a platform architecture for managing scientific data in a data space [52]: In this process, they characterize the "Scientific Data Space" by input data that is used for an experiment, analysis methods that are used in experiments and data sets that represent intermediate results of experiments. Here, the various data sources in the data space are searched and indexed by the data space indexer. A data space browser enables browsing of the data and can retrieve data from the data space via a query processor and SPARQL queries. However, this assumes that data in the data space has standardized metadata in the OWL format. With a further abstraction, additional metadata about the linkage of resources in the data space is then managed in an RDF database to represent scientific data processing operations. [52]

The concept of data management in data spaces was further developed by Curry in [53], who presented the approach of a Real-time Linked Data Space. This is an adaptation of conventional data spaces to scenarios in which data is processed live and event based. A platform solution (Data Space Support Platform) provides functionalities and services for managing the data space. In addition to search functionalities and access controls, these include services that support the streaming of data and complex event processing.

The International Data Space (IDS) Association is aiming to standardize data space architectures. For this purpose, an architecture model has been developed that can represent a wide variety of stakeholders and interests. The core of the concept is a decentrally organized system that enables data exchange in data ecosystems by means of standardized connectors and a central broker. For this purpose, various standardized models are used to exchange metadata and to define the framework conditions for data exchange. Further components are identity providers and an app store for the application-oriented provisioning and retrieval of data assets. [54]

From this, it can be seen that data spaces very much rely on the concept of service-orientation. Depending on the exact use-case both service-oriented architectures as well as microservice are conceivable.

C. Maritime Testbeds

As part of the **Smart Ship Application Platform (SSAP)** projects, an architecture was developed for exchanging data between ships at sea and coastal stations. An approach was followed in which the entire data of the testbed is managed in coast-side master databases with standardized interfaces and an onboard data server for ship-specific applications. Here, the centralized master databases are aggregating data across multiple vessels. The focus was specifically on IoT data and typical maritime data such as weather, traffic and engine data. Furthermore, the data is exchanged in a standardized way in a classical service-oriented architecture with multiple data services and can be controlled by the ship owners. In addition to storing data on the ship and shore side, live data can also be streamed. Data recording and transmission is based on ISO 19848 and IEC61162 standards. A provenance management component is not provided. [55]

The **datAcron** project focuses on the management and analysis of heterogeneous data related to surveillance systems from the maritime domain and aviation. Selected maritime data sources include AIS data, chart data, ship routes, ship data, (accident) reports, weather data and data on specific geographic areas (e.g. fishing areas). In particular, a distinction is made between "data-at-rest" (archived data) and "data-in-motion" (live/streaming data). The data is processed according to a classical scientific workflow. In this approach, all data is transformed into RDF format using a previously defined ontology and stored centrally. [56], [57]

The goals of the **Sea Traffic Management (STM)** initiative are to increase safety in maritime traffic by reducing the workload on ship bridges due to better organization of relevant information and to increase the efficiency of shipping through digital infrastructures. Within the STM Validation project, various concepts and services of STM have been investigated and evaluated by using several testbeds. Especially, the three areas of collaborative decision making in ports, route management systems, and traffic simulation were considered. [58]

In the area of collaborative decision making in ports, a platform solution has been developed that enables the sharing of data while considering different interests and security aspects. A platform-based solution helps in the sovereign exchange of data between different participants. For this purpose, the data ecosystem of maritime stakeholders is analyzed, and a role distribution is derived. The "core provider" provides the essential, standardized infrastructure to exchange and manage data and integrate new services. Furthermore, data providers can use connectors to make their data available and service providers

can provide services that use the existing data and make it available to service consumers in a specific form. Thus, a classic service-oriented architecture with an *orchestration* platform (cf. section II.E) is implemented in which the core provider stands as an intermediary between data consumers and providers. However, the whole platform is mainly focused on operational usage and does not provide support for managing test related data or tracing data provenance. [59]

The **eMIR testbed** is a generic testbed that is intended to enable the development and testing of highly automated and autonomous systems. The goal is to provide a testbed architecture that is as sustainable and interoperable as possible and thereby to simplify the integration of systems to be tested in the testbed. The focus was placed on the aspects of the development of cyber-physical systems of systems (CPSoS), which involve special requirements with regard to safety, reliability, validation and verification, and the complex interaction of several subsystems. eMIR consists of various simulation components that can simulate traffic, environment, and sensors, a reference waterway in the area of the German Bight and the Elbe River that records real data of maritime traffic and environmental conditions, a research boat, a mobile ship bridge, and a near-collision database. [4] Some selected data of the testbed is analyzed and stored in a central data warehouse. For this purpose, a classical scientific data processing process is applied, in which AIS and RADAR data are recorded, cleaned and merged. For detecting critical traffic situation on the sea, the data is merged into individual vessel tracks, analyzed, and categorized based on different types of vessel encounter situations. The raw data, tracks and near-collision encounter situations are stored centrally in different databases. However, this workflow only covers a portion of the data from the eMIR testbed; specific data management processes for other data sources, such as additional sensors, are not considered. Although some of the components are provided as single services, the architecture mainly focuses on the integration of a SuT via a central data flow management component and does not support the independent usage of all services. [60]

D. Automotive Testbeds

The **C-Vet** testbed at the University of California, Los Angeles is a generic testbed for testing models and protocols used in connected vehicles. It enables both the exploration of communication channels for connected vehicles (physical communication layer, protocol layer, and specifically data exchange via 802.11p) and the experimental validation of mobility models. The latter aims at testing within a realistic environment so that influences such as traffic density, realistic movement patterns and other traffic conditions can be considered in experiments and realistic behavior patterns in traffic can be investigated. The testbed consists of hardware and software for testing new network layers and protocols, a testbed-wide mesh network, several vehicles equipped with sensors, sensors for monitoring the environment (traffic, air quality, and imaging data from stereoscopic cameras), a web interface for managing the testbed components, and a central live database for aggregating the recorded testbed data. The architecture is non-service-oriented and focusses on the extendibility of existing components. [61]

The **VFbed testbed** is a test environment for testing concepts in the field of "Vehicular Fog Computing". In this case, calculations are outsourced to a network layer between the end device and the cloud (the so-called "edge"). In addition to communication between the vehicles in the testbed (vehicle-to-vehicle), communication between the vehicle and the infrastructure (vehicle-to-infrastructure) also plays an important role. Both the test vehicles (through on-board units) and the infrastructure (so-called "Fog Nodes") are connected to the backend of the testbed. Instead of real cars, miniature robots are used, which use a Raspberry Pi for control. The VFbed test field is strongly oriented towards use as a service: Users of the test field can book, configure and expand test environments fully automatically. Nevertheless, the architecture does not support live data management. It can be assumed that an SOA has been implemented to realize VFbed, but too few details are given about the type and nature of the interfaces and communication processes. [62]

One of the few papers dealing exclusively with the topic of data management in automotive testbeds or test environments was published in 2019 by Klitzke et al. They address the question of how data can be organized for real-world test drives. This involves the identification and management of data in the form of scenarios. For this purpose, a **Vehicle Data Management System (VDMS)** was developed, which divides the historical raw data from the test carrier vehicles into scenes, aggregates them and enriches them by further processing and data from external data sources (e.g. map data). The overall architecture and processing modules are designed to transform the data from tests into a scenario representation and make it centrally available to build a scenario catalogue, identify relevant scenarios for certifying SuTs, identify relevant system parameters in specific scenarios, and evaluate the performance of real-world tests. For data management, a three-layer module-based monolithic architecture is presented, consisting of a data layer (storing the testbed data), a module layer (data management e.g. aggregation, enrichment, compression) and an interface layer (querying and analyzing). [63]

HarborNet is a cross-domain testbed which is mainly used to investigate network protocols for connected vehicles. As a data management solution for managing the recorded tesbed data, a monolithic cloud-based backend system was developed, which stores the data from the testbed in a NoSQL database in the form of a data warehouse and makes it available via a web API. Furthermore, several monitoring components are used and real-time evaluation of experiments, as well as delayed analysis is possible. [64]

E. Summary

For examining existing solutions, approaches in the context of decentralized research and development data management, conventional research DMS and DMS in data ecosystems were considered first. Although conventional research DMS can fulfill basic requirements for merging data from different data sources and offer the possibility of

storing further metadata on the data processing history, they are not suitable for the use in decentrally organized testbeds due to their centralized nature.

In the data ecosystem domain, approaches like the IDS reference architecture can satisfy requirements related to trust and security in decentralized structure. They also already focus strongly on service-orientation. However, for the use in the research and development context, there are some isolated approaches so far, but they cannot cover the requirements of the user side due to a lack of testbed-specific methods for e.g., (testbed) data streaming, collaborative system engineering models or testcase-oriented data management.

Even directly related approaches for the data management of testbeds can only partially fulfill the defined requirements (c.f. section III). For example, while simple and static processes are used for data processing, general applicability or extensibility on an architectural level and the ability to facilitate collaboration of multiple stakeholders in a service-oriented environment is not given. The modular and sovereign connection of new data sources is also not fully supported, and data provenance information is not documented consistently. Also, only very few of them are utilizing SOA and none of the analyzed architectures utilized the advantages of microservices and frameworks like Kubernetes. As a result, the investigated architectures cannot be used to provide data in a testbed to external users in a service-oriented way while considering the decentralized context. This leads to a limited feasibility and efficiency for collaborative research and development processes. Finally, it can be stated that there is a need to conceptualize a new DMS for testbeds.

V. System Design of a Testbed Data Management System

Based on our assumptions in sections II and III, we interpret the modern testbed ecosystem as a data space, used by different stakeholders, that are exchanging their data and collaborate in modelling, development, verification and validation, certification and demonstration of new systems. In the following sections we adapt the functionalities from a classical DSSP and present a system, that can support data access and exchange in the testbed data space.

A. Methodology

For the design of a TDMS, several abstraction layers are considered during the system design, starting with the implications of the testbed data ecosystem, by examining the interests of different types of stakeholders and their interrelations. Additionally, as discussed in sections II and III, data spaces are chosen as an architectural model that can reflect the decentralized structures in a service-oriented testbed.

In this section, we aim to consider all the specific requirements of the different groups of stakeholders in the testbed, while at the same time avoiding designing a too complex monolithic architecture. Therefore, a layer-based approach for deriving the architecture of a TDMS is being used. Layer-based architectures are a well-known approach and can guarantee a "separation of concerns" in such a way, that responsibilities are clearly assignable to a specific layer, and that development, testing and management tasks can be divided into tasks with lower complexity [65].

Section VI gives a short summary of the used technologies and the implementation of the proposed architecture with the concept of microservices and the Kubernetes framework. Figure 2 shows an overview of the methodology.

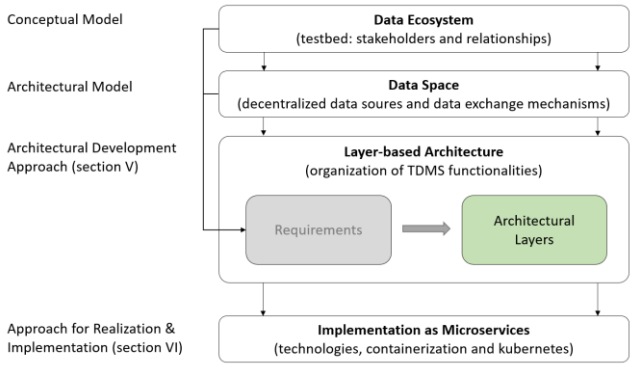

Figure 2: Overview of the methodology – conceptual and architectural model, development approach and realization.

Therefore, all architectural layers of the TDMS will be logically separated and will communicate with standardized interfaces: Each layer is built around a central request handler component, which interprets the calls from other layers or entities and forwards them to the desired internal components of the layer. Table 1 gives an overview of the architectural layers, their functionalities, and the associated requirements from section III.

Considering the interests of data sovereignty of the data providers in the testbed, the layer integrating the actual data from the testbed to the TDMS must deal with strong decentralization. Guaranteeing an independent control of the data sources is an important factor to satisfy the sovereignty requirements. This is realized in the so-called "Connector Layer", which provides data integration abilities and the "Connector Access Layer", which coordinates data requests to the different data sources, as they are expected to run on multiple different infrastructures from different data providers. After being able to access the decentralized data sources, the next challenge comes with the provision of requested data, while at the same time supporting maritime data processing operations by different stakeholders and making them traceable. This is the task of the "Data Processing Layer". Finally, the third layer of the proposed architecture, the Data Distribution Layer, must deal with authentication of users and authorization of data requests. This is mandatory for enforcing the role model and preventing unauthorized data access. The layers of the TDMS are discussed in detail in the following section and are depicted in Figure 3.

Architectural Element	Functionalities	Requirements covered
Connector Layer	Integration of decentralized data sources.	Core requirements on (maritime) data integration and access.
Connector Access Layer	Managing data access (queries and search requests).	Core requirements on data integration and access.
Data Processing Layer	Provision of requested data and data workflow management.	Data lifecycle support, integration of external processing operations, workflow- and scenario management.
Data Distribution Layer	Authentication, authorization, and user management.	Control of data exchange, access management, enforcement of role-model.

Table 1: Architectural layers of the TDMS and their functionalities.

In the following, sections B, C, and F describe the architectural structure of the proposed layers and their purpose by referring to the structure of the *data space* and the stakeholders with their use-cases in the *data ecosystem*. Sections D and E focus on the realization of requirements that are specific to the *testbed* data ecosystem.

B. Connector Layer and Connector Access Layer

Architecture. As different decentrally organized data sources in a data space come with the problem of technical heterogeneity, a first layer of the TDMS architecture must provide integration functionalities in such a way, that data in all data sources can be accessed, while at the same time guaranteeing the control over the data to the data providers. In data spaces, the architectural pattern of the connector can fulfill these requirements. A connector is a "standardized interface for receiving, sending and transforming data sets for communication" [66]. These interfaces are controlled by the data providers and can be used to enforce data usage policies [66]. In the testbed, we assume that processes for data acquisition are implemented outside the connectors (e.g., by reading and preprocessing sensor data) and that data is exchanged via the TDMS and not directly sent to the users. For this reason, our version of the connector only handles and sends data to the TDMS.

To manage the decentralized connectors, a counterpart is required at the TDMS side. The connector access layer collects metadata about the decentral connectors and their data schemes and forwards generic data queries from users of the TDMS to the desired connector with the help of a metadata service. It also supplies a verification functionality to make sure that connectors are not receiving incorrect queries.

Rationale. In this configuration, a connector has two main purposes. Firstly, it controls incoming data requests from the users and can reject them if they violate the data usage policy of the data provider. Secondly, a connector acts as a technology abstraction layer: A generic user request is forwarded by the system to a compatible connector, which will translate the data request to a native query, e.g., SQL or MongoDB query, which can be answered by the data source. It is then transformed to a generic container format and sent back to the user. We already discussed a detailed version of this approach in previous work in [34].

C. Data Processing Layer (DPL)

Architecture. As a next step, the Data Processing Layer provides a possibility to transport data from a connector to the user. A direct connection between user and connector is not realized in our scenario, as collaborative data processing involving multiple users would become very complicated and the main load for exchanging data would be put on the decentrally organized connectors. Because of this, the Data Processing Layer of the TDMS is utilized as an intermediate system to exchange data between different stakeholders. The architectural structure of the DPL is implemented as a *Kappa* data architecture, as described in [67]: Data is processed as streams in a *speed layer*, which is supported by a data *ingestion tool*. The ingestion tool temporary stores the data stream and therefore makes it easily available at a later point in time for repurposing or sharing in a collaborative setup. Also, the Kappa architecture can support many processes out-of-the-box, that rely on live (streamed) data.

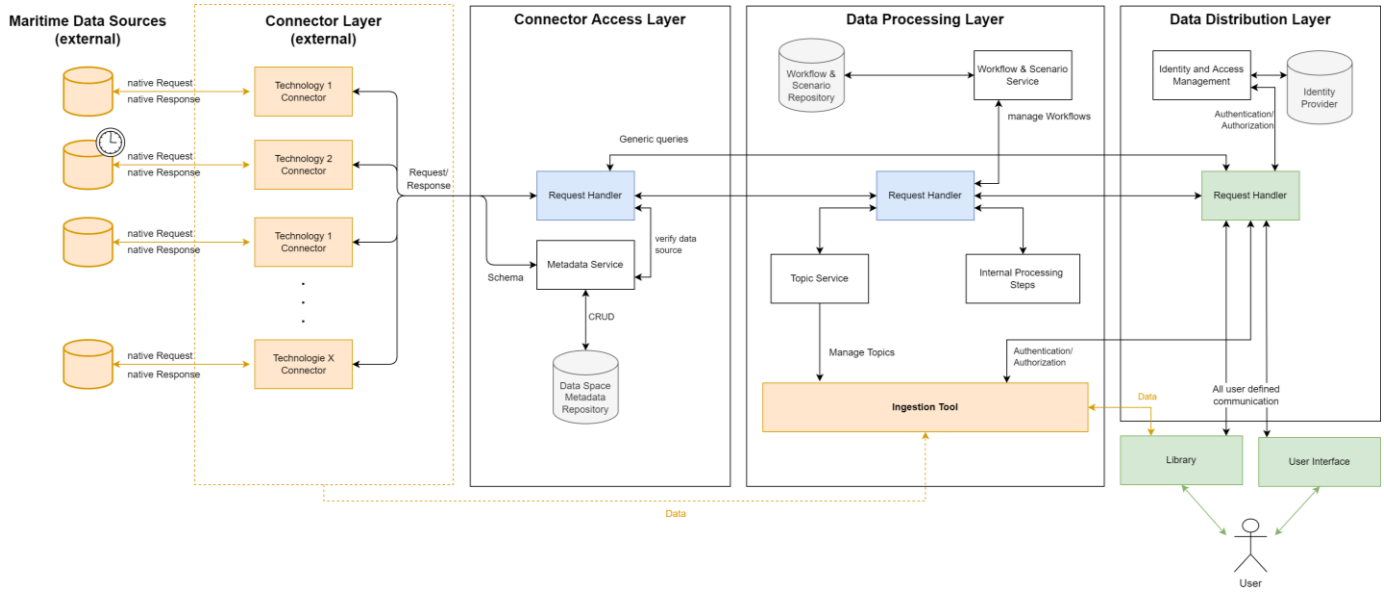

Figure 3: System architecture of the TDMS.

This article has been accepted for publication in IEEE Open Journal of Intelligent Transportation Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/OJITS.2022.3207235

Rationale. The management of live data is a relevant factor in maritime testbeds, as test executions typically produce large streams of data, supplied by a SuT, its input and output variables, environmental sensors, or other monitoring data. Data transferred to the ingestion tool does not necessarily have to come from a connector, it could also be an artifact in the further data processing workflow, which a system or a user applies to the data. For example, the ingestion tool could be used at the time of a test execution to hold data that represents internal states of a system under test, to better monitor its states and rate the test success at runtime of the system. Utilizing this tool for answering data queries and as a medium for data artifacts from data processing reduces the complexity of the TMDS architecture and facilitates data sharing in the testbed in several different use-cases.

Additional services support data processing in the *speed layer* of the kappa architecture. The speed layer can be seen as another decentralized component of the TDMS: Data consumers would utilize the TDMS for retrieving and exchanging data artifacts, but processing of the data still happens locally. In this way, the architecture can dynamically be extended based on different requirements of different types of data consumers. However, using the TDMS as a main component for data processing as part of a SuT itself would not make sense, as the system cannot rely on a testbed component for later usage in a production environment.

To organize data from different processes in the ingestion tool, different artifacts of data are managed in *topics* by the Topic Service. A topic is a logical unit of thematically related data (cf. [68]). This could, e.g., be the result of a data query, monitoring information about an internal state of a SuT in the testbed, or the output of a prediction model. The remaining services of the DPL are discussed in the following two subsections.

D. Workflow Model in the DPL

Apart from the architectural considerations on data source integration and answering of data queries, a testbed data management system should provide an overarching concept of data organization: Using topics as thematically related units makes the management of data in a maritime testbed easier but is still not sufficient to model the complex relations in a data space with multiple data providers, consumers, systems-of-systems, and data exchange processes. To solve this issue, it is required to add context to the data topics and connect multiple related topics to each other. Therefore, we introduce a simple data workflow model to describe complex processing chains and make them available as a whole to users of the TDMS. In the previous section, it was mentioned that the speed layer, which contains the actual data manipulation operations is a decentralized component. Because decentral components are not under the control of the TDMS, we can only store meta-information about these operations: We define a *processing step* as the application of a transformation to one or more data topics by a stakeholder in the testbed, resulting in a new data topic. By chaining data artifacts and processing step metadata, complex workflows can now be managed by the TMDS in a graph-like data structure. Finally, references to the data producers and consumers will also be stored to complete the provenance information of the data.

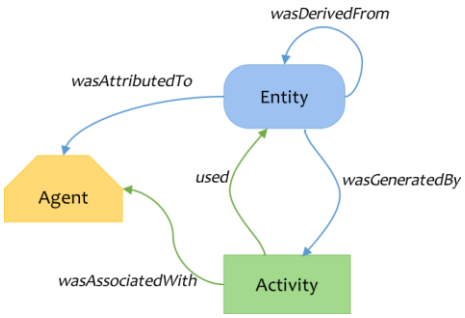

Figure 4: PROV-DM Model for documenting data provenance graphs [69].

A formal description of such a model is also given by the PROV-DM standard [70]. Like our conceptual model it uses three core elements to represent provenance information (cf. Figure 4). In the interest of providing a standardized model, we utilize the PROV-DM model to serialize the provenance information in the TDMS, describing a data topic as an *Entity*, a processing step as an *Activity* and data producers and consumers as *Agents*. Managing the provenance information of the workflows in the DPL is done by the workflow and scenario service.

E. Data Management Components for V+V in the DPL

Scenario Elicitation. The central step in the utilization of testbeds for engineering processes is the execution of testcases. Looking at complex cyber-physical systems, this process is more and more relying on scenario-based testing methods [71]. From the perspective of data, especially the tasks of scenario elicitation based on historical data, and the execution of scenarios in the testbed are relevant (cf. Figure 1 in [39]). Looking at processes for the elicitation of scenarios, Neurohr et al. state in [39] that these processes are use-case specific and require the incorporation of expert knowledge. However, with the proposed collaborative workflow model from section D, the TDMS can support a decentralization of the process to different stakeholders.

On the other hand, the local management of data sources may lead to different data models and varying data quality. Also, many approaches show, that raw maritime traffic data typically needs be preprocessed in similar ways. An example of AIS and RADAR data used for scenario elicitation is given by [20]. These reoccurring data (pre-)processing tasks typically represent the first processing steps in a data workflow for scenario elicitation. To further support users of the TDMS in this regard, we introduce the concept of *internal processing steps*. An internal processing step is a processing step, managed and executed on the TDMS infrastructure. Typical internal processing steps would solve frequent data (pre-)processing tasks. These could be provided in a parameterizable way by the testbed owner, as a service for the testbed users. In this way, typical preprocessing can be generalized to increase the efficiency of the scenario elicitation process.

Scenario Instantiation. As a core functionality, the TDMS should also support the management of data from

test/scenario executions. In the following, we propose a methodology for managing scenario execution data, based on a test specification: When executing a test-scenario, an abstract representation of a scenario is *instantiated*. From the perspective of data, this can be seen as the acquisition of several new artifacts of data, describing the test-environment and SuT behavior itself. As our understanding of data processes in the testbed is expressed by the workflow model, instantiating a scenario means instantiating a workflow, thus adding data to the topics of the workflow, which consists of topics, processing steps and associated users, as described in the previous subsection (cf. Figure 5).

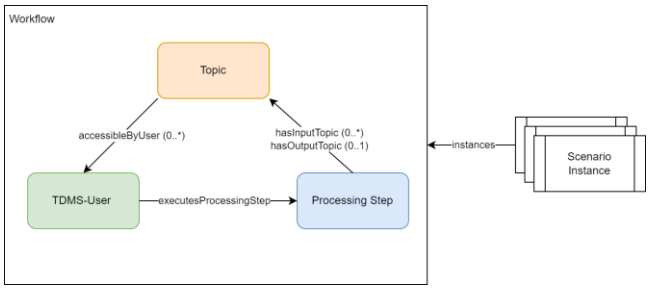

Figure 5: Metadata Model for Workflows and Scenarios.

According to IEEE, there can be different levels of test specifications (test design specification, test procedure specification, test case specification) that specify the inputs, predicted results, execution conditions and general procedures and approaches to implement and conduct tests [1]. In the proposed methodology, a test engineer would define a workflow based on his knowledge of the SuT and the requirements from the test specification documents. This is depicted in Figure 6: From the conceptual knowledge about the system and its input and output variables, it is possible to define a fine-granular workflow that maps desired data topics in the data processing chain that should be evaluated, to data topics in the TDMS-managed workflow. Information about the processing steps inside the system is added to enrich the metadata of the workflow and can be defined by the different stakeholders. Such a workflow may not only incorporate data directly related to the SuT, but also environmental data, that influences input variables or the system behavior in general. This brings the benefits of easier analysis of system behavior and the possibility to share insights, build automated tools for generating test reports or rating test success automatically. Furthermore, also a black box of data processing can be monitored live and different stages of data processing can be managed separately in the workflow. Due to the temporary storage capabilities of the Kappa architecture, test data may also be replayed or merged with additional data for enriching tests. An example is given in section VII.

However, to apply this methodology in a real-world setup, it must be taken care of the fact, that testing is often expensive, and a tight schedule for test cases must be adhered to. This leads to multiple tests being executed at the same time, or in quick succession. Storing data in a data workflow for a larger test campaign, may lead to large amounts of recorded data, which cannot be separated into single test-cases anymore. Therefore, it is required to store additional metadata for the scenario instances' starting and ending times. As common ingestions tools like Kafka store the data in the topics as immutable logs, with the possibility to only append new data, the TDMS stores the current position of the log as a marker, when starting and ending a scenario. Later, when a scenario is exported or replayed, this metadata can be used to retrieve the correct section of each topic in the workflow.

F. Data Distribution Layer (DDL)

Trust and Data Protection. When data is being exchanged via the ingestion tool, a data provider must trust the TDMS, that only authorized users can access and process this data.

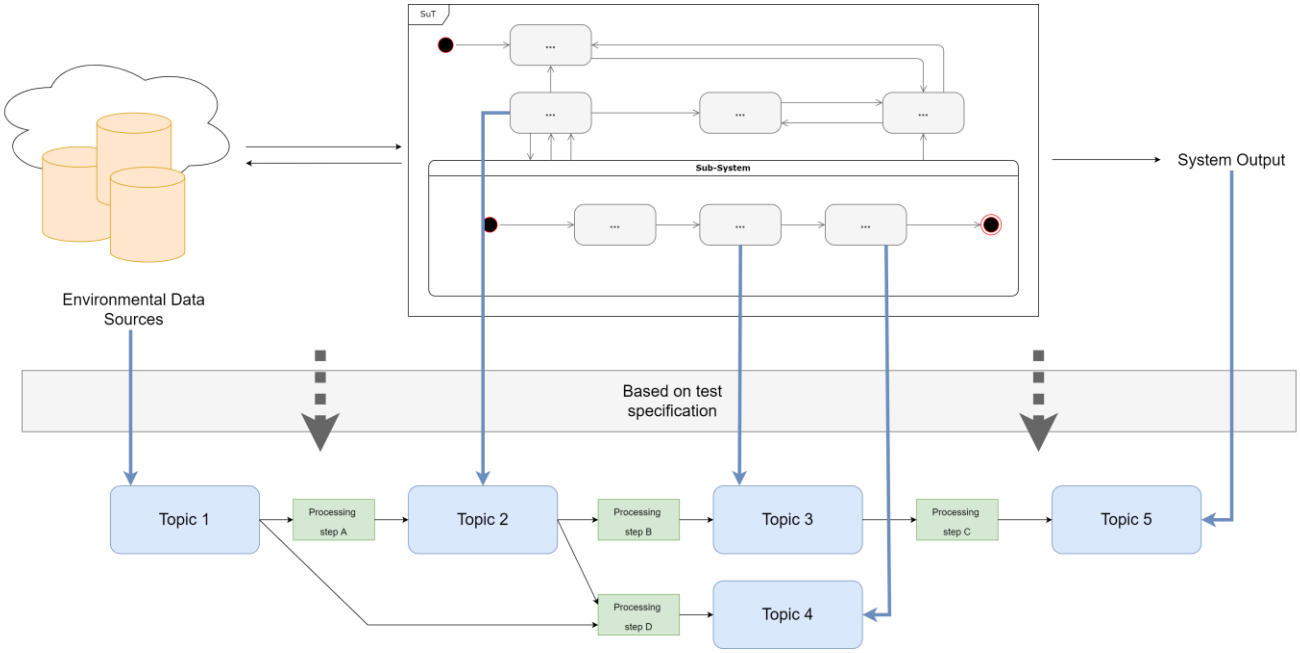

Figure 6: Derivation of a Data Workflow from a SuT based on a Test Specification.

This article has been accepted for publication in IEEE Open Journal of Intelligent Transportation Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/OJITS.2022.3207235

However, trust issues are more likely to appear between different users of the testbed, than between users and testbed owner. In a service-oriented testbed, users would sign a contract with the testbed owner regulating usage conditions and guaranteeing legal security regarding data protection. Furthermore, a provider of a TDMS typically is not in economic competition with users of the testbed. Nevertheless, it is in the responsibility of the testbed owner, to secure the TDMS against cyber security threats and especially unauthorized access to data. To do this, a third layer, the Data Distribution Layer (DDL) is introduced. Except for the ingestion tool, the DDL provides the only interface publicly available to testbed users from outside the TDMS. All data queries, workflow modifications or other functionalities must be requested from the DDL. After a successful authentication of a user and authorization, a request is forwarded by the DDL internally to the corresponding layer. This minimizes the attack surfaces and makes monitoring of suspicious activities far easier.

Architecture. For authentication of users, a local identity provider (IdP) is set up in the testbed by the testbed owner. According to [72], an identity provider is a trusted entity providing mechanisms for authentication. The IdP can be used to authenticate users at the interfaces of the TDMS and may also be utilized for other (non-data related) services in the testbed. The local IdP (or an external solution) may also already exist as an important part for accessing any types of the services in the service-oriented testbed in a single-sign on fashion.

Regarding authorization, there are three resources that need to be protected: data sources (connectors), topics (including the actual data) and workflows (as a collection of metadata, including scenarios). For the data sources, a Role Based Access Control (RBAC) is proposed (cf. [73]). RBAC represents an easy way of determining if a specific user is allowed to access a data source in the testbed. The TDMS can check if a requesting user owns the designated role to access the requested data source. More fine-granular access policies allowing or restricting specific queries may then be enforced at the side of the connector, directly by the data provider. For topics and workflows, it would not make sense to create designated roles for accessing them, as they are created more dynamically than data sources and access rights might also be changed by users itself. For this reason, it is required to manage access information separately for each resource in a Discretionary Access control (DAC, cf. [73]).

To make interactions with the interfaces of the TDMS easier available for developers or test engineers, libraries may be provided for different programming environments. Additionally, a Web UI is provided to monitor workflows and scenarios during a test execution.

VI. Technologies and Deployment

In the following section, the technologies used to implement the proposed concept as a proof-of-concept are disccused. The presented prototype was mainly implemented in Java.

A. Overview

Based on the advantages for the application in service-oriented testbeds discussed in section II.E, a microservices architecture was utilized for the implementation of the TDMS and the associated services (see Figure 7): First, each layer was implemented as a single service. To achieve the choreography property of microservices, lightweight REST-interfaces are exposed where necessary (depicted in green). The interfaces are provided in such a way, that services can directly communicate with each other without the need for a central orchestration component. For example, the extension of the architecture by new connectors is possible at runtime without the modification of any other components. Besides the REST-interfaces, the services that exchange data from the testbed also communicate with the ingestion tool (Apache Kafka, depicted in red). For the test setup, all the services run in a microk8s[2] Kubernetes cluster with a built-in container registry. The registry is also utilized by the DPL to dynamically instantiate new container instances for the internal processing steps via the Kubernetes API (depicted in blue, cf. section V.E). The services are organized in four different Kubernetes namespaces and only expose the marked interfaces (depicted as white dots) to the testbed.

B. Spring Boot Web API

The communication between all components is realized via standardized REST APIs. Spring Boot is utilized as a library

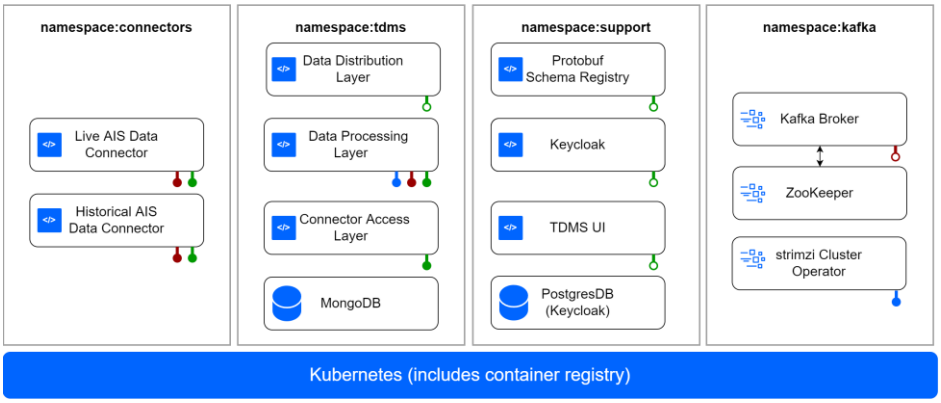

Figure 7: Microservices Deployment of the TDMS and Associated Components in Kubernetes.

[2] https://microk8s.io/. Accessed: August 18, 2022.

to provide the basic API implementation and as a framework for dependency injection in the code. Furthermore, the API endpoints are exposed compliant to the OpenAPI [74] specification. Communication with the public endpoints is only possible TLS-encrypted via HTTPS and login is realized via OAuth2 [63]. A Java client-library that is used to access TDMS functionalities and the Web-UI are an abstraction layer of the DDL-API, which is the only publicly available layer beside the ingestion tool.

C. Apache Kafka

As the ingestion tool, an Apache Kafka cluster is utilized. Kafka is a distributed event store and stream-processing platform [76]. It acts as a base for exchanging data in the testbed, while supporting stream (live) data processing and has client-libraries for many different programming languages. Furthermore, it is possible to have multiple topics that can be protected with a fine-granular access control. In our implementation, we specifically use the strimzi Kafka distribution [77], as it natively supports the Kubernetes API and the integration of an external identity and access management via the OAuth2 protocol. For every requested new data topic, a Kafka topic is instantiated and made accessible to the corresponding user.

D. Keycloak

As a Single-Sign-On (SSO) solution for all the services in the testbed, Keycloak is being used. Keycloak is an identity and access management platform that supports common authentication and authorization protocols such as OAuth2 or SAML [78]. In the implementation of the TDMS prototype, Keycloak is mainly used for authentication of the testbed users and to manage authorization for the Kafka cluster.

E. MongoDB

MongoDB [79] is a NoSQL database and is used as the database backend of the TDMS. Its document-oriented storage capabilities fit well to the hierarchical nature of provenance data. The MongoDB stores information regarding the testbed data sources and workflow- and scenario metadata. Furthermore, the workflow-related DAC data is also stored in the MongoDB.

F. GraphQL / Protobuf

Data querying to connectors is implemented in GraphQL (cf. [80]). GraphQL is a flexible query language, often used for the provision of data APIs. The connector access layer uses GraphQL-schemas of the data sources to match a query against them and forward it to a data source in the testbed that can answer the query. Besides queries for batch data, it also supports streaming queries. Finally, the requested data is dynamically encoded using Protobuf, which is a highly efficient data serialization format. Protobuf schemas are dynamically being generated and shared in the testbed data space with the help of a schema registry.

G. Maritime Data Connectors

For demonstration of our approach, we implemented two connectors to supply AIS data to the testbed users via the TDMS. Firstly, a PostgreSQL connector was implemented to make historical data available. The historical AIS connector is able to answer data queries and filter the data by several different attributes such as geographical position, ship speed, ship heading, time range and ship identifier. Additionally, a RabbitMQ connector for a live AIS-Data source was implemented, that provides data from the testbed and is also able to utilize the same attributes (except the time range) for the supplied data.

H. Decentral Deployment

To use the TDMS operationally in the service-oriented testbed, it cannot necessarily be assumed that all data providers and consumers trust a single central Kubernetes deployment (provided by the testbed owner) of such a system. As all the architectural layers are also logically separated in the implementation, it is possible to provide individual parts of the system in a modular fashion and to have them provided by specially certified or trusted parties. For example, a data trustee, as the provider of the DPL, or a metadata broker to provide the connector access layer. Finally, a provider of the DDL coordinates access to the layers and could resolve any conflicts that may arise, since it is able to log all requests to the TDMS. In this way, required layers could be instantiated multiple times, depending on the stakeholders' needs.

VII. Evaluation: Insights to a Maritime Collision Avoidance System

As the range of possible applications of the proposed system is very broad, we illustrate several use-cases with the example of a maritime assistance system for collision avoidance as an evaluation.

A. Introduction to Case Study

The Maritime Traffic Collision Avoidance System (MTCAS) is an assistance system, capable of detecting critical maritime traffic situations, predicting vessel behavior and negotiating a conflict resolution with multiple vessels [81]. As part of our evaluation, we specifically focus on MTCAS' functionality to detect critical encounter situations. This is done by a rule-based approach, as described in [41]: Based on the remaining predicted time and distance until an accident happens, five categories of so-called *escalation states* (1 – clear, 2 – recommendation, 3 – danger, 4 – last minute maneuver, 5 – accident) are utilized to rate the current collision risk with every vessel perceived by MTCAS. MTCAS uses a complex data processing workflow to transform raw traffic data (in the format of NMEA0183, cf. [82]) to an internal sensor data model, followed by a traffic situation model and a behavior prediction to classify the current collision risk as escalation state.

The goal of the proposed case study is to use a scenario-based testing approach to validate and verify the collision risk assessment functionality of MTCAS. As an input data stream for the system, we first utilize a third-party simulation and later enrich the data from the simulation with real live data from the testbed.

This article has been accepted for publication in IEEE Open Journal of Intelligent Transportation Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/OJITS.2022.3207235

B. *Setup for Collaborative Testing*

In the described setting, three stakeholders collaborate to test the risk assessment functionalities of MTCAS: the testbed owner (providing the live data from the testbed), the provider of the simulation environment and the engineers of the MTCAS system itself. Figure 8 shows the testing setup with the TDMS: Data produced by the simulation component, as well as live data from the testbed is pushed to a topic at the TDMS, where it is consumed by MTCAS as an input data stream. This way, incoming data can be dynamically added from simulation and/or the testbed. In a production environment, the input from the TDMS would be replaced by a real NMEA0183 data source on a ship. In contrast to using the TDMS as a middleware for data exchange between different stakeholders, the internal data processing states are only mirrored to the TDMS for analysis by other systems or users to minimize the dependency of the actual SuT on the TDMS. For our tests, the investigated data workflow consists of the NMEA0183 input data stream, the internal representation of the MTCAS sensor data model, the systems perception of the overall traffic situation, results of the behavior prediction for each ship and the escalation states as an output of the system. As MTCAS is implemented in Java, the TDMS Java client library is utilized to transmit the data from the system to the TDMS. Finally, input and output data are being transmitted continuously, while the internal states of the system are sampled as Java objects every 500ms, converted to Protobuf messages and then sent to the TDMS. As an additional application, the proposed setup can be used without the simulation or the live testbed data source after a single run of the test-scenario, by replaying the data from the associated data topic directly to the system under test. This can be useful for situations where repeating a specific test-case is very expensive or time-consuming, e.g., when executing tests with multiple vessels.

C. *Separation of Data Access*

In the proposed setup in Figure 8, the input data topic represents a central topic, to which all the three stakeholders need access. A live traffic data stream from the testbed may specifically be requested by the system engineer to test their system in a scenario with real-world surrounding traffic, while the provider of the simulation is required to access the topic to write simulation data of the scenarios. Only the MTCAS system and its engineer should be able to read the provided data. Additionally, the internal states of the MTCAS system should also only be available to the system engineer to protect innovative value of system internals. Table 2 shows how these relations can be modelled with the RBAC and DAC systems, introduced in section V.F.

	Access to live data (RBAC)	**Access to simulated data (RBAC)**	**NMEA0183 Topic (DAC)**	**MTCAS Topics (DAC)**	**Workflow Metadata (DAC)**
Testbed owner	(Yes)	No	- / Write	- / -	- / -
Simulation provider	No	(Yes)	- / Write	- / -	- / -
MTCAS System (engineer)	Yes	Yes	Read / Write	Read / Write	Read / Write

Table 2: Role-based (RBAC) and Discretionary Access Control (DAC) policies for MTCAS test workflow.

For additional data protection, the stakeholders may introduce custom encryption to the exchanged data independently from the TDMS.

D. *Management of Test Scenarios*

To conduct the scenario-based evaluation of MTCAS, we used five common traffic situations and recreated them in the simulation: As MTCAS bases its behavioral analysis of ship movements on the COLREGs [83], we utilized a crossing situation, a head-on situation, an overtake situation a situation, where two ships are passing and a situation in which a ship is following another ship. The scenarios were played sequentially in the simulation, without restarting the main components of MTCAS. To manage the data recording, the Web-UI of the TDMS (see Figure 9) was used to quickly start and stop recordings during the test, and to export them as json-Files for further analysis.

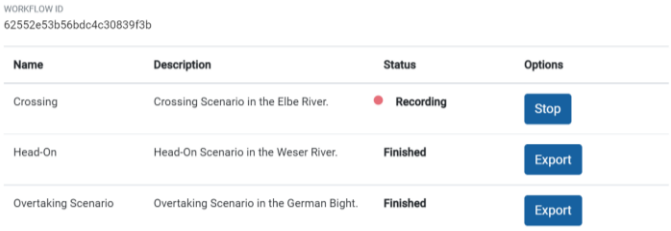

Figure 9: UI for the Management of Scenario Recordings.

Furthermore, we looked at an example with data of a real collision accident between the FU SHAN HAI and the

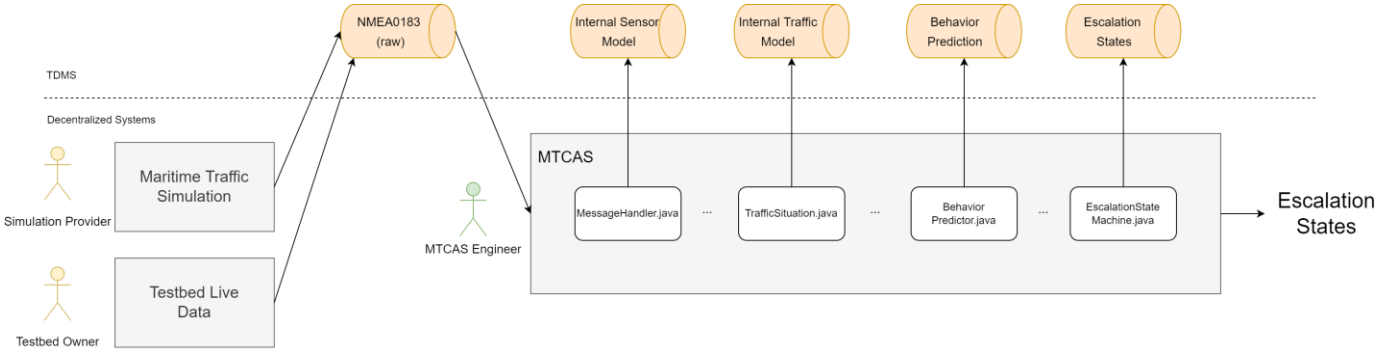

Figure 8: Test setup for the evaluation of MTCAS.

GDYNIA in 2003 north of Bornholm in the Baltic Sea [84], from which the results are presented in the following section.

E. Data Workflow Analysis

This section shows how the workflow and scenario-oriented data management concept of the TDMS can be utilized to analyze SuT behavior in the testbed: The workflow approach allows to analyze the generated data from the testcase in detail. For this, the described historical collision accident scenario was replayed in the simulation. It was then recorded with the workflow described in section B, and exported in json-Files (one file for each data topic) to conduct the analyses. This gives a detailed view on how MTCAS would have behaved in the scenario. For the documentation of the workflow, PROV-DM compliant metadata is also supplied by the Workflow and Scenario management component of the TDMS, as shown in Figure 10.

```
…
<rdf:Description rdf:about="https://testbed.emaritime.de/provenance
    /MTCASInternalModel">
    <rdf:type rdf:resource="https://www.w3.org/ns/prov#Entity"/>
    <prov:wasGeneratedBy rdf:resource="https://testbed.emaritime.de
        /provenance/transformToMTCASModel_ProcessingStep"/>
</rdf:Description>
<rdf:Description rdf:about="https://testbed.emaritime.de/provenance
    /mtcas_engineer">
    <rdf:type rdf:resource="https://www.w3.org/ns/prov#Agent"/>
</rdf:Description>
<rdf:Description rdf:about="https://testbed.emaritime.de/provenance
    /transformToMTCASTrafficModel_ProcessingStep">
    <rdf:type rdf:resource="https://www.w3.org/ns/prov#Activity"/>
    <prov:wasAssociatedWith rdf:resource="https://testbed.emaritime
        .de/provenance/ mtcas_engineer"/>
    <prov:used rdf:resource="https://testbed.emaritime.de/provenance
        /MTCASInternalModel"/>
</rdf:Description>
…
```

Figure 10: Excerpt from the Provenance Metadata Stored by the TDMS.

To start with the actual analysis, the raw data stored in the NMEA0183 topic was compared to the representation of the scenario, that was utilized in the simulation. Figure 11 shows an excerpt from the raw data and visualizes all the recorded positions of the two ships approaching each other.

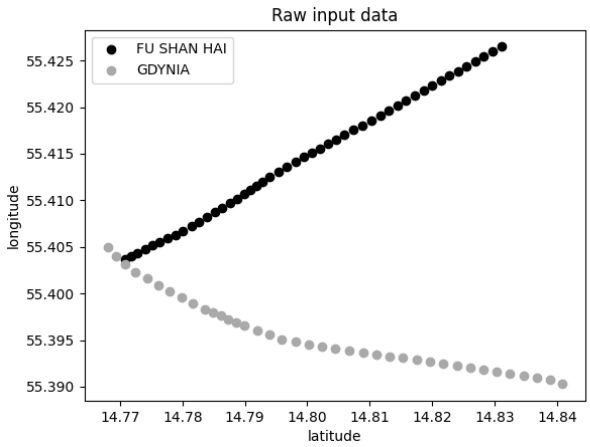

Figure 11: Raw position data, provided by the simulation.

After validating the correctness of the raw input data for MTCAS, internal data processing steps from MTCAS' behavior prediction component were analyzed: As collision risk is one of the most relevant variables to consider, when looking at the escalation states, and collision risk correlates with the predicted minimum distance (Closest Point of Approach distance / CPA distance) between two vessels, it was decided to further investigate the predicted CPA distance (cf. Figure 12) between the two vessels from the behavior prediction topic. This is a prediction of the minimal encounter distance of the vessels, based on their current state.

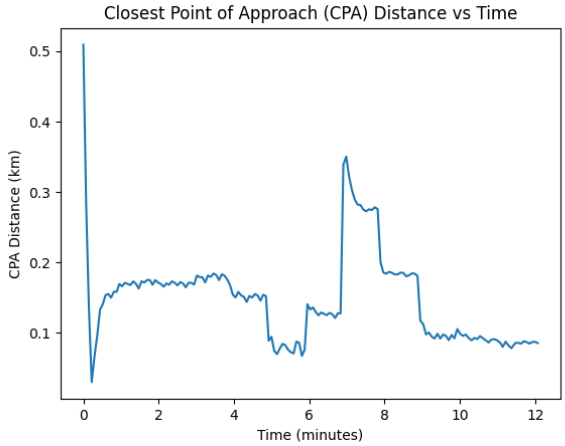

Figure 12: MTCAS internal predicted CPA distance.

Several variables are used by MTCAS to predict the CPA distance at runtime. After analyzing the data extracted from the traffic situation topic, it was found that the GDYNIA started a slight clockwise rotation maneuver at minute 7 (cf. Figure 13), leading to changes in the CPA distance prediction.

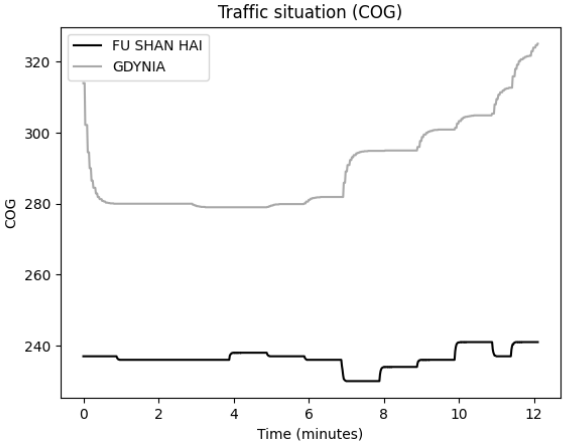

Figure 13: COG of the encountering ships, as perceived in the MTCAS Traffic Situation model.

MTCAS does not only calculate the CPA distance as such, but projects two encountering vessel poses to the future at the point of their closest approach. These CPA-projections were also extracted from the MTCAS behavior prediction topic and are visualized in Figure 14: The maneuver of the GDYNIA initially leads to a different projection, with a greater CPA distance.

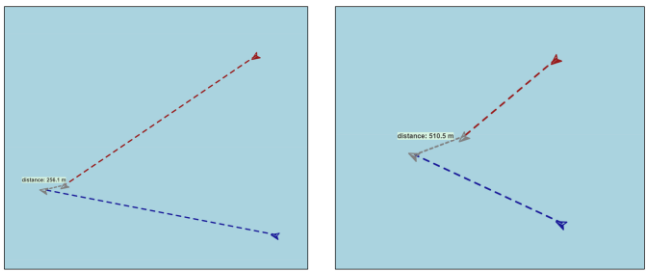

Figure 14: Update of predicted CPA (before and after minute 7).

F. Discussion

In summary, the case study showed the applicability of the proposed TDMS architecture in a service-oriented testbed with multiple stakeholders and fulfills the requirements defined in section III: With the help of the test-setup from section B, two decentrally organized data sources from two different data providers could be integrated, dynamically queried and merged as input data for a scenario-based testing approach. Metadata querying and query-mapping functionalities are provided by the Connectors and the Connector Access Layer of our architecture. Secondly, it is possible for the data providers to manage access to their data with a fine-granular access management. Likewise, users of the testbed can manage access to single parts of collaborative data processing workflows as it has been shown in sub-section C. Furthermore, data exchange is controlled by every party individually: Data providers may use custom connectors to enforce data usage policies, and users of the TDMS can utilize different workflow and scenario metadata management structures to organize and encapsulate their data (cf. sub-section D). Thirdly, data processing steps can easily be integrated (e.g., with the provided library) to the TDMS, documented from end-to-end in a standardized format, and processed data can be exported for further analyses as demonstrated in sub-section E. The complete system was secured with standard protocols for authentication and authorization (OAuth2 with Keycloak) and the attack surface of the architecture was minimized by only providing a single access point to the TDMS – the Data Distribution Layer, which distributes and monitors requests to the other layers. Finally, the case study also demonstrated that the proposed architecture is specifically applicable to the maritime domain, supporting efficient data exchange in areas with low network bandwidths by serializing data to the space-efficient Protobuf format.

VIII. Summary and Conclusion

In this paper, we discussed the need for a data management system, which can meet the requirements of a service-oriented maritime testbed. After identifying the gaps in existing work, a layer-based architecture was developed that incorporates different stakeholder needs and can support activities in the area of knowledge generation and verification and validation tasks from the perspective of data management by using a dedicated concept of managing distributed data workflows and scenarios. On the technical level, the concept of a Kappa architecture was integrated into a data space representation of the testbed to increase data availability and facilitate collaboration. With the help of a case study, the applicability of the system was demonstrated. The proposed architecture closes the gap of lacking data management concepts for service-oriented maritime testbeds with multiple stakeholders. Furthermore, it highly supports current trends in data protection and the need to preserve sovereignty over data. In summary, the proposed architecture provides a holistic approach to data management in service-oriented testbeds. However, when it comes to an extension of the TDMS with multiple new data sources, some limitations of our architecture need to be mentioned: The structure of a data space with many data sources can result in a large heterogeneity of data. This can lead to problems if users do not know exactly which data source they need for their tests. Depending on the testbed, it could make sense to enforce the usage of standardized data models or to provide a support system for accessing the data (cf. [34]). Similar limitations apply to data usage policies and their enforcement: As more data providers exist and different kind of usage policies exist, it makes sense to implement harmonized models for their enforcement. As described in [85], there already exist sophisticated models that could be utilized to deal with this problem, when scaling the TDMS architecture.

In future work, we envision to integrate the testbed data ecosystem into a greater context of data infrastructures, such as the European GAIA-X project for cross-eco-system data exchange. Furthermore, we plan to investigate the applicability of our architecture to the automotive domain and analyze related work in other transport domains, such as rail or aerospace.

References

[1] ISO/IEC/IEEE, 'ISO/IEC/IEEE International Standard - Systems and software engineering – Vocabulary', *ISO/IEC/IEEE 24765:2010(E)*, pp. 1–418, Dec. 2010, doi: 10.1109/IEEESTD.2010.5733835.

[2] S. Abbaspour Asadollah, R. Inam, and H. Hansson, 'A Survey on Testing for Cyber Physical System', in *Testing Software and Systems*, Cham, 2015, pp. 194–207. doi: 10.1007/978-3-319-25945-1_12.

[3] C. Tschirner, L. Kaiser, R. Dumitrescu, and J. Gausemeier, 'Collaboration in model-based systems engineering based on application scenarios', *DS 81: Proceedings of NordDesign 2014, Espoo, Finland 27-29th August 2014*, 2014.

[4] N. Rüssmeier, A. Lamm, and A. Hahn, 'A generic testbed for simulation and physical-based testing of maritime cyber-physical system of systems', *J. Phys.: Conf. Ser.*, vol. 1357, no. 1, p. 012025, Oct. 2019, doi: 10.1088/1742-6596/1357/1/012025.

[5] Y. Duan, Y. Cao, and X. Sun, 'Various "aaS" of everything as a service', in *2015 IEEE/ACIS 16th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, 2015, pp. 1–6.

[6] Y. Duan, G. Fu, N. Zhou, X. Sun, N. C. Narendra, and B. Hu, 'Everything as a service (XaaS) on the cloud: origins, current and future trends', 2015, pp. 621–628.

[7] P. Banerjee *et al.*, 'Everything as a service: Powering the new information economy', *Computer*, vol. 44, no. 3, pp. 36–43, 2011.

[8] M. Hossain, S. Noor, Y. Karim, and R. Hasan, 'IoTbed: A Generic Architecture for Testbed as a Service for Internet of Things-Based Systems', in *2017 IEEE International Congress on Internet of Things (ICIOT)*, Jun. 2017, pp. 42–49. doi: 10.1109/IEEE.ICIOT.2017.14.

[9] A. Hahn and T. Noack, 'eMaritime Integrated Reference Platform', Braunschweig, Oct. 2016. Accessed: Aug. 26, 2021. [Online]. Available: https://publikationen.dglr.de/?tx_dglrpublications_pi1[document_id]=420297

[10] H. Winner, *Handbuch Fahrerassistenzsysteme: Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort ; mit ... 45 Tabellen*. Wiesbaden: Vieweg + Teubner, 2009.

[11] H. Winner and W. Wachenfeld, 'Auswirkungen des autonomen Fahrens auf das Fahrzeugkonzept', in *Autonomes Fahren: Technische, rechtliche und gesellschaftliche Aspekte*, M. Maurer, J. C. Gerdes, B. Lenz, and H. Winner, Eds. Berlin, Heidelberg: Springer, 2015, pp. 265–285. doi: 10.1007/978-3-662-45854-9_13.

[12] J. Schäuffele and T. Zurawka, *Automotive Software Engineering*. Wiesbaden: Springer Fachmedien Wiesbaden, 2016. doi: 10.1007/978-3-658-11815-0.

[13] M. Brinkmann, B. Å. Hjøllo, A. Hahn, and V. Bertram, 'Physical Testbed for Highly Automated and Autonomous Vessels', May 2017.

[14] R. Martins, J. B. de Sousa, R. Caldas, C. Petrioli, and J. Potter, 'SUNRISE project: Porto university testbed', in *2014 Underwater Communications and Networking (UComms)*, 2014, pp. 1–5.

[15] A. J. Singh, A. Kumar, and H. C. Lau, 'Hierarchical multiagent reinforcement learning for maritime traffic management', 2020.

[16] B. J. Rhodes, N. A. Bomberger, and M. Zandipour, 'Probabilistic associative learning of vessel motion patterns at multiple spatial scales for maritime situation awareness', in *2007 10th International Conference on Information Fusion*, 2007, pp. 1–8.

[17] C. Gkerekos and I. Lazakis, 'A novel, data-driven heuristic framework for vessel weather routing', *Ocean Engineering*, vol. 197, p. 106887, 2020.

[18] M. D. Wilkinson *et al.*, 'The FAIR Guiding Principles for scientific data management and stewardship', *Scientific data*, vol. 3, no. 1, pp. 1–9, 2016.

[19] A. Engel, *Verification, Validation, and Testing of Engineered Systems*. 2010. doi: 10.1002/9780470618851.

[20] A. Lamm and A. Hahn, 'Towards Critical-Scenario Based Testing With Maritime Observation Data', in *Proceedings of the OTO'18*, Kobe, May 2018, pp. 1–10. doi: 10.1109/OCEANSKOBE.2018.8559045.

[21] C. Neurohr, L. Westhofen, T. Henning, T. de Graaff, E. Möhlmann, and E. Böde, 'Fundamental Considerations around Scenario-Based Testing for Automated Driving', *arXiv:2005.04045 [cs]*, May 2020, Accessed: Jan. 16, 2022. [Online]. Available: http://arxiv.org/abs/2005.04045

[22] DNV GL AS, 'DNVGL-CG-0264: Autonomous and remotely operated ships'. Sep. 2018. Accessed: Jul. 15, 2021. [Online]. Available: https://rules.dnv.com/docs/pdf/DNV/cg/2018-09/dnvgl-cg-0264.pdf

[23] C. S. Dixon and R. G. Morrison, 'A Pseudolite-Based Maritime Navigation System: Concept through to Demonstration', *Positioning*, vol. 1, no. 13, 2008.

[24] F. Liang, W. Yu, D. An, Q. Yang, X. Fu, and W. Zhao, 'A survey on big data market: Pricing, trading and protection', *IEEE Access*, vol. 6, pp. 15132–15154, 2018.

[25] A. Munoz-Arcentales, S. Lopez-Pernas, A. Pozo, A. Alonso, J. Salvachua, and G. Huecas, 'An architecture for providing data usage and access control in data sharing ecosystems', 2019, vol. 160, pp. 590–597. doi: 10.1016/j.procs.2019.11.042.

[26] M. Falkenthal, F. W. Baumann, G. Grünert, S. Hudert, F. Leymann, and M. Zimmermann, 'Requirements and Enforcement Points for Policies in Industrial Data Sharing Scenarios', 2017.

[27] M.-Å. Hugoson, 'Centralized versus Decentralized Information Systems', in *History of Nordic Computing 2*, Berlin, Heidelberg, 2009, pp. 106–115. doi: 10.1007/978-3-642-03757-3_11.

[28] J. Gelhaar and B. Otto, *Challenges in the Emergence of Data Ecosystems*. 2020.

[29] M. I. S. Oliveira and B. F. Lóscio, 'What is a data ecosystem?', in *Proceedings of the 19th Annual International Conference on Digital Government Research: Governance in the Data Age*, Delft The Netherlands, May 2018, pp. 1–9. doi: 10.1145/3209281.3209335.

[30] L. Xu, C. Jiang, J. Wang, J. Yuan, and Y. Ren, 'Information security in big data: privacy and data mining', *Ieee Access*, vol. 2, pp. 1149–1176, 2014.

[31] M. I. S. Oliveira, L. E. R. de Alencar Oliveira, G. F. A. Barros Lima, and B. F. Lóscio, 'A platform for supporting open data ecosystems', *Lecture Notes in Business Information Processing*, vol. 291, pp. 313–337, 2017, doi: 10.1007/978-3-319-62386-3_15.

[32] B. Otto, S. Steinbuß, and et al., 'International Data Space - Reference Architecture Model'. International Data Spaces Association, Apr. 2019. Accessed: Nov. 11, 2019. [Online]. Available: https://www.internationaldataspaces.org/ressource-hub/publications-ids/

[33] M. Franklin, A. Halevy, and D. Maier, 'From databases to dataspaces: a new abstraction for information management', *ACM Sigmod Record*, vol. 34, no. 4, pp. 27–33, 2005.

[34] J. Möller, D. Jankowski, and A. Hahn, *Towards an Architecture to Support Data Access in Research Data Spaces*. 2021, p. 317. doi: 10.1109/IRI51335.2021.00049.

[35] Y. Wang, S. Song, and L. Chen, 'A Survey on Accessing Dataspaces', *SIGMOD Rec.*, vol. 45, no. 2, pp. 33–44, Sep. 2016, doi: 10.1145/3003665.3003672.

[36] V. Raj and R. Sadam, 'Performance and complexity comparison of service oriented architecture and microservices architecture', *International Journal of Communication Networks and Distributed Systems*, vol. 27, no. 1, pp. 100–117, 2021.

[37] T. Cerny, M. J. Donahoo, and J. Pechanec, 'Disambiguation and comparison of soa, microservices and self-contained systems', in *Proceedings of the International Conference on research in adaptive and convergent systems*, 2017, pp. 228–235.

[38] L. A. Vayghan, M. A. Saied, M. Toeroe, and F. Khendek, 'Deploying microservice based applications with kubernetes: Experiments and lessons learned', in *2018 IEEE 11th international conference on cloud computing (CLOUD)*, 2018, pp. 970–973.

[39] C. Neurohr, L. Westhofen, T. Henning, T. de Graaff, E. Möhlmann, and E. Böde, *Fundamental Considerations around Scenario-Based Testing for Automated Driving*. 2020.

[40] P. W. H. Chung and Z. Liao, 'Cross-organisation dataspace (COD) - Architecture and implementation', 2008, vol. 6, pp. 448–451. doi: 10.1109/CSSE.2008.1638.

[41] J. Qin, A. Ball, and J. Greenberg, 'Functional and Architectural Requirements for Metadata: Supporting Discovery and Management of Scientific Data', *Proceedings*

of the International Conference on Dublin Core and Metadata Applications, Jan. 2012.

[42] M. Zhong, M. Liu, and Q. Chen, 'Modeling heterogeneous data in dataspace', 2008, pp. 404–409. doi: 10.1109/IRI.2008.4583065.

[43] M. Zichichi, M. Contu, S. Ferretti, and V. Rodríguez-Doncel, *Ensuring Personal Data Anonymity in Data Marketplaces through Sensing-as-a-Service and Distributed Ledger*. 2020.

[44] S. Cuno, L. Bruns, N. Tcholtchev, P. Lämmel, and I. Schieferdecker, 'Data governance and sovereignty in urban data spaces based on standardized ICT reference architectures', *Data*, vol. 4, no. 1, p. 16, 2019.

[45] Y. Demchenko, P. Grosso, C. De Laat, and P. Membrey, 'Addressing big data issues in scientific data infrastructure', in *2013 International Conference on Collaboration Technologies and Systems (CTS)*, 2013, pp. 48–55.

[46] D. Rousidis, E. Garoufallou, P. Balatsoukas, and M.-A. Sicilia, 'Metadata for Big Data: a preliminary investigation of metadata quality issues in research data repositories', *Information services & use*, vol. 34, no. 3–4, pp. 279–286, 2014.

[47] M. I. M. Saad, K. Abd Jalil, and M. Manaf, 'Achieving trust in cloud computing using secure data provenance', in *2014 IEEE Conference on Open Systems (ICOS)*, 2014, pp. 84–88.

[48] M. Mirović, M. Miličević, and I. Obradović, 'Big data in the maritime industry', *NAŠE MORE: znanstveni časopis za more i pomorstvo*, vol. 65, no. 1, pp. 56–62, 2018.

[49] Z. H. Munim, M. Dushenko, V. J. Jimenez, M. H. Shakil, and M. Imset, 'Big data and artificial intelligence in the maritime industry: a bibliometric review and future research directions', *Maritime Policy & Management*, vol. 47, no. 5, pp. 577–597, Jul. 2020, doi: 10.1080/03088839.2020.1788731.

[50] IALA, 'E-navigation Testbeds and FAQ', *IALA AISM*. https://www.iala-aism.org/technical/e-nav-testbeds/ (accessed Aug. 26, 2021).

[51] R. Amorim, J. Aguiar Castro, J. Rocha, and C. Ribeiro, 'A comparison of research data management platforms: architecture, flexible metadata and interoperability', *Universal Access in the Information Society*, vol. 16, Nov. 2017, doi: 10.1007/s10209-016-0475-y.

[52] I. Elsayed and P. Brezany, 'Dataspace Support Platform for e-Science', *csci*, vol. 13, no. 1, p. 49, 2012, doi: 10.7494/csci.2012.13.1.49.

[53] E. Curry, 'Fundamentals of Real-time Linked Dataspaces', in *Real-time Linked Dataspaces, Enabling Data Ecosystems for Intelligent Systems*, 2019, pp. 63–80. doi: 10.1007/978-3-030-29665-0_4.

[54] B. Otto and M. Jarke, 'Designing a multi-sided data platform: findings from the International Data Spaces case', *Electron Markets*, vol. 29, no. 4, pp. 561–580, Dec. 2019, doi: 10.1007/s12525-019-00362-x.

[55] H. Ando, 'Activities of smart ship application platform 2 project (SSAP2)', *International Marine Purchasing Association (IMPA), London, UK*, 2017.

[56] C. Claramunt *et al.*, *Maritime data integration and analysis: recent progress and research challenges*. 2017.

[57] G. A. Vouros, C. Doulkeridis, G. Santipantakis, and A. Vlachou, 'Taming Big Maritime Data to Support Analytics', in *Information Fusion and Intelligent Geographic Information Systems (IF&IGIS'17)*, Cham, 2018, pp. 15–27. doi: 10.1007/978-3-319-59539-9_2.

[58] B. Andreasson *et al.*, 'STM Validation Deliverables 2.6, 2.10 & 2.12 - Voyage Management Testbed Report'. Jun. 03, 2019. Accessed: Sep. 01, 2021. [Online]. Available: https://stm-stmvalidation.s3.eu-west-1.amazonaws.com/uploads/20200225090150/STMVal_D2.6-D2.10-D2.12-Voyage-management-testbed-report-1.pdf

[59] M. Lind *et al.*, 'STM Validation Deliverable 1.1 - Enabling port optimization by a digital collaborative platform'. Dec. 31, 2015. Accessed: Sep. 01, 2021. [Online]. Available: https://s3-eu-west-1.amazonaws.com/stm-stmvalidation/uploads/20190402152729/STMVal_D1.1-Enabling-port-optimization-by-a-digital-collaborative-platform.pdf

[60] A. Lamm and A. Hahn, 'Towards Critical-Scenario Based Testing With Maritime Observation Data', in *2018 OCEANS - MTS/IEEE Kobe Techno-Oceans (OTO)*, Kobe, May 2018, pp. 1–10. doi: 10.1109/OCEANSKOBE.2018.8559045.

[61] P. Lutterotti, G. Pau, D. Jiang, M. Gerla, and L. Delgrossi, 'C-vet, the UCLA vehicular testbed: An open platform for vehicular networking and urban sensing', in *International Conference on Wireless Access for Vehicular Environments (WAVE 2008)*, 2008, vol. 182.

[62] M. A. Hoque and R. Hasan, 'VFbed: An Architecture for Testbed-as-a-Service for Vehicular Fog-based Systems', New Orleans, Louisiana, USA, Apr. 2020. doi: 10.1109/WF-IoT48130.2020.9221384.

[63] L. Klitzke, C. Koch, A. Haja, and F. Köster, 'Real-world Test Drive Vehicle Data Management System for Validation of Automated Driving Systems':, in *Proceedings of the 5th International Conference on Vehicle Technology and Intelligent Transport Systems*, Heraklion, Crete, Greece, 2019, pp. 171–180. doi: 10.5220/0007720501710180.

[64] C. Ameixieira *et al.*, 'Harbornet: a real-world testbed for vehicular networks', *IEEE Commun. Mag.*, vol. 52, no. 9, pp. 108–114, Sep. 2014, doi: 10.1109/MCOM.2014.6894460.

[65] M. Richards, *Software architecture patterns: understanding common architecture patterns and when to use them*. Sebastopol, CA, USA: O'Reilly Media, Inc., 2015.

[66] J. Zrenner, F. O. Möller, C. Jung, A. Eitel, and B. Otto, 'Usage control architecture options for data sovereignty in business ecosystems', *Journal of Enterprise Information Management*, vol. 32, no. 3, pp. 477–495, Jan. 2019.

[67] G. K. Kalipe and R. K. Behera, 'Big Data Architectures: A detailed and application oriented review', *Int. Journal Innov. Technol. Explor. Eng*, vol. 8, pp. 2182–2190, 2019.

[68] H. Wu, Z. Shang, and K. Wolter, 'Performance prediction for the apache kafka messaging system', in *2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, 2019, pp. 154–161.

[69] S. Soiland-Reyes, *English: Overview of the W3C PROV model. CC-BY 4.0*. 2018. Accessed: Aug. 15, 2022. [Online]. Available: https://commons.wikimedia.org/wiki/File:W3C_PROV_Data_Model.svg

[70] K. Belhajjame *et al.*, 'PROV-DM: The prov data model', 04 2013, [Online]. Available: http://eprints.soton.ac.uk/id/eprint/356851

[71] D. Nalic, T. Mihalj, M. Bäumler, M. Lehmann, A. Eichberger, and S. Bernsteiner, 'Scenario based testing of automated driving systems: A literature survey', 2020.

[72] J. De Clercq, 'Single sign-on architectures', in *International Conference on Infrastructure Security*, 2002, pp. 40–58.

[73] R. S. Sandhu and P. Samarati, 'Access control: principle and practice', *IEEE Communications Magazine*, vol. 32, no. 9, pp. 40–48, Sep. 1994, doi: 10.1109/35.312842.

[74] D. Miller *et al.*, 'OpenAPI Specification v3.1.0 | Introduction, Definitions, & More', Feb. 15, 2021.

https://spec.openapis.org/oas/latest.html (accessed May 10, 2022).

[75] D. Hardt, 'The OAuth 2.0 authorization framework'. RFC 6749, October, 2012.

[76] G. Wang *et al.*, 'Building a replicated logging system with Apache Kafka', *Proceedings of the VLDB Endowment*, vol. 8, no. 12, pp. 1654–1655, 2015.

[77] 'Strimzi - Apache Kafka on Kubernetes'. https://strimzi.io/ (accessed May 10, 2022).

[78] D. N. Divyabharathi and N. G. Cholli, 'A review on identity and access management server (keycloak)', *International Journal of Security and Privacy in Pervasive Computing (IJSPPC)*, vol. 12, no. 3, pp. 46–53, 2020.

[79] 'MongoDB: The Application Data Platform', *MongoDB*. https://www.mongodb.com (accessed May 10, 2022).

[80] 'GraphQL'. https://spec.graphql.org/October2021/ (accessed May 10, 2022).

[81] M. Steidel and A. Hahn, 'MTCAS -An Assistance System for Collision Avoidance at Sea', Tullamore, Mar. 2019.

[82] R. B. Langley, 'NMEA 0183: A GPS receiver interface standard', *GPS world*, vol. 6, no. 7, 1995.

[83] International Maritime Organization, 'COLREGS - International Regulations for Preventing Collisions at Sea'. 1972.

[84] Danish Maritime Authority, 'Collision between Chinese bulk carrier FU SHAN HAI and Cypriot container vessel GDYNIA', MINISTRY OF ECONOMIC AND BUSINESS AFFAIRS, Copenhagen, Denmark, Aug. 2003. Accessed: Apr. 12, 2022. [Online]. Available: https://www.vragguiden.dk/FuShanHai.pdf

[85] C. Jung and J. Dörr, 'Data Usage Control', in *Designing Data Spaces : The Ecosystem Approach to Competitive Advantage*, B. Otto, M. ten Hompel, and S. Wrobel, Eds. Cham: Springer International Publishing, 2022, pp. 129–146. doi: 10.1007/978-3-030-93975-5_8.