The Semantic Container Approach

Techniques for ontology-based data description and discovery in a decentralized SWIM knowledge base

Eduard Gringinger, Christoph Fabianek
Frequentis AG
Vienna, Austria

Christoph G. Schuetz, Bernd Neumayr, Michael Schrefl
Johannes Kepler University Linz
Linz, Austria

Audun Vennesland
SINTEF
Trondheim, Norway

Scott Wilson
EUROCONTROL
Brussels, Belgium

***Abstract*— System Wide Information Management (SWIM) in Air Traffic Management (ATM) aims to facilitate access to ATM information via information services, thereby fostering common situational awareness among stakeholders. The development of information services and applications with added value in SWIM will comprise finding, selecting, filtering and composition of data/information from different sources, which is also referred to as 'data logic'. Semantic containers are a means to encapsulate the data logic and clearly separate it from business and presentation logic. The provisioning of semantic containers for a specific purpose encompasses the discovery of existing source containers and often further value-adding processing steps such as filtering and annotation. Common semantic web technologies may serve to implement the semantic container approach.**

Data Mediation; Data Aggregation; System Wide Information Management; Air Traffic Management

I. INTRODUCTION

Achieving the BEnefits of SWIM by making smart use of Semantic Technologies (BEST) was a SESAR Exploratory Research project (TRL 1) [1] that investigated the use of semantic technologies within a SWIM-enabled environment. BEST addressed research questions about the use of semantic technologies to handle metadata, achieving scalable solutions for data management, realizing automated compliance checking with the help of ontology matching techniques, optimizing data distribution, and using automated modularization – with implications for governance.

SWIM is one of the major results of the SESAR programme, and the adoption of SWIM by the ATM community will lead to dramatic changes in how ATM services are provided. Traditional ATM information management was based on point-to-point message transfer, meaning information producers had to decide in advance who the target recipients would be. SWIM will change all this because it is based on an information sharing approach where information producers do not need to know anything about who might use the information, and where information consumers can access information from different sources if they have permission.

Standardized exchange models such as the Aeronautical Information Exchange Model (AIXM) [2], the Flight Information Exchange Model (FIXM) [3], the ICAO Weather Information Exchange Model (IWXXM) [4], or semantic models such as the ATM Information Reference Model (AIRM) [5] already affect software architecture and software development in a positive manner.

Without a clear description of the information/data semantics, applications and service implementations will rely on hard-coded data logic – intertwined with business and presentation logic – that deals with the information, thereby hampering reuse of information between services and applications. In this regard, SWIM serves different providers for publishing information. Stakeholders then need to find the relevant information for a specific task. Developers will likely spend a significant amount of time with mastering the complexities of the data logic for handling all the information in SWIM, which holds developers back from developing innovative applications and value-added services. The separation of data logic from business and presentation logic is a commonly accepted principle in software engineering.

"Semantic technologies" is an umbrella term comprising modelling techniques, languages and tools that allow for the development of software that can process and "understand" information that was designed for human perception in the first place. The basic premise of BEST is that such semantic technologies could be used to complement what is provided in SWIM to enable truly effective information management. There is no tradition in ATM of using semantic technologies, and perhaps even scepticism from parts of the community concerning the usefulness of semantic technologies. The main objective of the BEST project [1] was therefore to determine how semantic technologies can be used effectively to maximize the benefits of adopting SWIM. One of the sub-objectives was to develop an ontology infrastructure for ATM, which is an important cornerstone of the semantic container approach. The outcome of these development efforts is described in this paper, which includes excerpts from Deliverables D1.1 [6],

D3.2 [7], and D5.2 [8] of the BEST project; the paper thus summarizes the project's most important results.

II. ATM Information Reference Ontology

The ontology infrastructure includes ontologies developed from the AIRM UML model and a set of ontologies, each representing different domains of ATM information exchange, namely AIXM [2] and IWXXM [4]. All ontologies are formalized in the Web Ontology Language (OWL) as standardized by the World Wide Web Consortium. The ontologies are used as the vocabulary for describing and supporting retrieval of relevant information by applications developed in the project. Furthermore, the ontologies form a baseline for the establishment of guidelines describing how semantic technologies can be applied to support information exchange in a SWIM environment.

The ontology development basically included three sub-processes:

1. Transformation from UML to OWL, since the AIRM is defined in UML
2. Semi-automated extraction of modules
3. Automated extraction of modules

Figure 1 serves to illustrate these processes. The development of the ontologies followed different paths, primarily due to different complexity in the UML structures of the different models, but also for the sake of experimenting with different techniques.

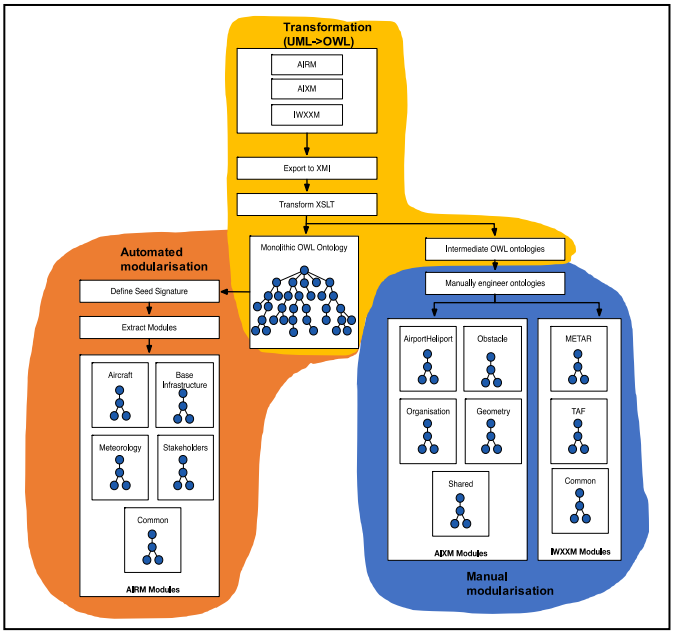

Figure 1. Ontology development approach [6]

A. Transformation from UML to OWL

The first step was to generate an XML Metadata Interchange (XMI) representation of the UML models. In the next step, we applied a set of transformation rules to transform from XMI to OWL. The transformation rules are developed with support from the (non-normative) guidelines for mapping between UML and OWL in the OMG ODM specification (see Appendix A of [6]). Table I provides an overview of the transformations performed and a more detailed explanation of each transformation and its resulting OWL entity is provided in [6]. The XSLT scripts used in the transformation are available online (http://project-best.eu/downloads/ontologies/xslt/xslt.zip).

TABLE I. Overview of Transformations between UML and OWL

UML Construct	OWL Construct
UML Class	OWL Class
UML Generalization	OWL SubClassOf
UML Boolean attribute	OWL Class
UML Attribute with complex data type	OWL Object Property
UML Association	OWL Object Property
UML Aggregation (AIRM only)	OWL Object Property
UML Composition (AIXM and IWXXM)	OWL Object Property
UML Attribute with simple data type	OWL Data Property

B. Semi-automated extraction of modules

The IWWXM and AIXM UML exchange models include many package interdependencies, intricate data typing, and other modelling conventions (e.g. XOR relationships and association classes) that makes it challenging to completely automate a transformation from UML to OWL. Therefore, the development of the ontology modules from IWXXM and AIXM has been performed semi-automatically in the sense that much of the class and property axioms in OWL are established automatically using XSLT and the same set of rules as for AIRM.

After this there was a need to enhance and structure the content manually in ontology editor Protégé [9]. In principle, the same XSLT transformation rules used for the AIRM could be applied for exchange models to get to a monolithic ontology but as already mentioned there are some differences in modelling techniques that prevent a completely generic approach among the models. Therefore, in the case of the exchange models the XSLT transformation results in an intermediate OWL representation. The intermediate ontologies consist of most entities present in the UML models, however quite a bit of manual engineering was still required to organize the proper relationship between classes, object properties, data properties and individuals, before the ontology modules are complete.

C. Automated extraction of modules

Monolithic ontologies can be characterized as large-sized and complex, often spanning several different topics and knowledge domains. Developing and maintaining such monolithic ontologies is a cumbersome and sometimes overwhelming task due to their size and complexity [10].

Advantages of ontology modules on the other hand include that they promote use, re-use, more efficient processing, and simple maintenance (to name a few).

The task of automatically decomposing a monolithic ontology into a set of sub parts (modules) is called *ontology modularization*. There is no single approach to ontology modularization that works for all situations, it depends on the application requirements. There are however two overall strategies, namely 1) ontology partitioning and 2) ontology module extraction. *Ontology partitioning* consists of decomposing the full set of axioms in an ontology into a set of modules (partitions) and the union of all modules should in principle be equivalent to the original ontology.

For example, Stuckenschmidt and Schlicht [11] applied structural characteristics such as target module size and number of target modules to determine suitable partitions of an input ontology. *Ontology Module Extraction* extracts modules from an ontology based on a definition of a sub-vocabulary, also called a seed signature. This signature consists of a set of entities (classes and/or properties and/or individuals) from which the technique recursively traverses through the ontology to gather related entities to be included in the module [10].

In BEST, we employed the latter strategy and more specifically a technique called Syntactic Locality Modularisation [12], [11] for extracting ontology modules from the AIRM, AIXM and IWXXM monolithic ontologies. The reason for this choice was primarily based on the use cases we had in the BEST project. First, all the three original models were structured according to topicality. For example, the AIRM model is organized into different subject fields, where each subject field is responsible for describing semantics about a certain topic, for example “Aircraft” or “Meteorology”. Secondly, the semantic containers are described in detail in section III.

The BEST project developed a set of prototype applications to support different steps of the modularization process. The module extraction functionality was implemented in Java using the OWL API library (version 4.1.2). It is important to realize that ontology modularization is not just about extracting isolated modules from a monolithic representation. To have a consistent set of modules in the end (i.e. a network of modules), one must capture and maintain dependencies among the extracted modules and resolve any redundancy that might exist. For this reason, a set of prototype ontology modularization applications were developed (https://github.com/sju-best-project/ontology-modules).

TABLE II. MODULES GENERATED FROM AIRM

Ontology Module	Classes	Object properties	Data properties	Individuals
Aircraft	71	84	32	182
AerodromeInfrastructure	117	345	69	0
NavigationInfrastructure	34	70	39	0
SurveillanceInfrastructure	34	21	17	0
Obstacle	12	27	8	0
BaseInfrastructureCodelists	100	0	0	1574
Meteorology	74	69	15	97
Stakeholders	148	131	40	316
Common	78	44	19	396

III. SEMANTIC CONTAINER: DEFINITION

We introduce semantic containers to encapsulate the data logic of SWIM services [13] and clearly separate it from business and presentation logic. A semantic container allows developers to organize and make sense of the provided ATM information. A semantic container provides a SWIM application or service with all the relevant ATM information, hiding the complexities of compiling the information package. Semantic containers come with ontology-based metadata that allow users, services, and applications to know what the content of the container is and assess the freshness as well as the quality of the data.

The provisioning of semantic containers for a specific purpose encompasses the discovery of existing source containers and often further value-adding processing steps such as filtering and annotating [14]. These tasks are supported by matching of information need and available data containers and services. Based on a formal ontology-based specification – employing the ontology infrastructure – of the information needed for an operational scenario, the semantic container management system should find existing data containers that most closely fulfil the specified information need and identify missing processing steps. Note that the implementation of the corresponding algorithms of identifying missing processing steps to obtain a full match is left to future work. More information about the definition of a semantic container can be found in [15] and [16].

IV. SEMANTIC CONTAINER DISTRIBUTION AND SWIM

Effective use of the semantic container approach developed in BEST depends on the existence of a Semantic Container Management System (SCMS) controlling the replication, distribution and consistency of containers. In the field of distributed databases, there are many existing techniques for distribution, replication and consistency management, mostly based on a single generic data model. In BEST, we refine existing techniques using different types of models for different kinds of information [17].

High availability of information and low network load are key goals for the success of the SWIM approach. Semantic containers, supported by an SCMS, can contribute significantly to these goals. The semantic container approach distinguishes between logical and physical containers to indicate which containers are allocated at which nodes. The semantic container approach also allows for the definition of different versions of containers, supporting consistency management and different forms of synchronization. Finally, semantic containers allow for traceability of data provenance, and definition of composite containers that gather data from lower-level elementary containers. We stress that the semantic container approach applies to various types of ATM information as well.

In SWIM, different applications require different types of ATM information at various degrees of freshness and availability. An aircraft pilot may, for example, request current weather data. For availability’s sake, consistency may be

sacrificed: Slightly outdated weather information is better for a pilot than none. With respect to notifications about runway closures, on the other hand, pilots require fresh data because wrong information would entail potentially disastrous consequences. Semantic containers allow us to make the inherent trade-off between freshness and high availability tangible for the consumer of ATM information: A semantic container packages ATM information and the resulting packages can be redundantly stored at multiple locations for high availability; administrative metadata indicate freshness and data quality.

Semantic containers also increase availability of the overall system by considering multiple sources of ATM information which semantic containers may be derived from. The semantic container metamodel [14] allows for the representation of multiple data sources for the same semantic container. An SCMS may switch dynamically and transparently between different sources. Different sources may provide the same data with different quality to ensure that the consumer is alert to any reduction in quality of service. A primary source is a source with the highest data quality among the sources of the container. Secondary sources of lesser quality are only used when no primary source is available at the expected freshness.

An advantage of packaging ATM information in semantic containers is the possibility to allocate relevant information directly in the aircraft that operates a specific flight. The semantic container can be created a couple of days prior to the date the actual flight takes place, being filled with relevant information in advance. Shortly before the flight, at the departure airport with high bandwidth, the container can be uploaded onto the plane, and during the flight updated with only the critical information or information that requires low bandwidth.

ATM information is inherently dynamic: Government authorities and authoritative sources, e.g., GroupEAD, push new data and updates to existing data. Hence, the semantic container approach requires a mechanism to keep the contained ATM information up to date. The proposed semantic container approach paves the way for both push- and pull-based handling of updates.

Multiple service consumers may request the same ATM information from a remote entity. Typically, each request for ATM information is processed individually, thereby putting stress on the available bandwidth. With a SCMS in place, SWIM services may cache frequently requested ATM information (e.g. weather data) as semantic containers. They can even store the semantic containers at locations where they are frequently needed, thereby reducing the bandwidth and computation effort. For example, a NOTAM filtering service may cache relevant NOTAMs for the most important flight routes as semantic containers. When concrete requests for specific flights come in, rather than sifting through the whole body of NOTAMs currently in place, the service may use the pre-filtered semantic containers as a starting point for further filtering.

V. Semantic Container Management System: Architecture

In this section we introduce a possible architecture for a SCMS. The semantic container approach as introduced in BEST Deliverable D2.1 [15], and the language for container management – basically a UML metamodel – may be implemented in many different forms; the semantic container approach is independent of a concrete software and data distribution architecture. Other and maybe more adequate architectures may be developed in the future based on the vast literature on distributed systems (e.g., [18], [19]). The proposed architecture as described in this paper serves two purposes:

- to give a more complete picture of a globally distributed SCMS, and
- to serve as a starting point for the development of more advanced software and data distribution architectures.

An SCMS is distributed over multiple server locations and multiple client locations. Locations are connected over the internet. Container content and metadata are allocated redundantly at multiple locations. Centrally-provided software is run independently at the different locations which cooperate to provide globally-distributed semantic container management. Container content and metadata are allocated redundantly at multiple locations. A semantic container consists of location-independent metadata (represented by the logical semantic container), location-dependent metadata (represented by the physical semantic container) and content (also referred to as data/information, e.g., a set of AIXM Digital NOTAMs).

The container metadata can be represented as RDF triples [7]. All container metadata can thus be collected into an RDF graph. This RDF graph of all semantic containers is fully replicated at every server location and partially replicated at client locations. Each location runs an RDF database management system (a.k.a. graph store) and SPARQL query engine for storing, modifying and querying (parts of) the RDF graph. Modifications of metadata at some location are replicated in an asynchronous manner to other locations to provide for redundancy of metadata in case of connection or network failures. Replica consistency of metadata is maintained by giving priority to most recent writes.

Container contents remain in their original form (XML documents according to AIXM, IWXXM, or FIXM). Each location runs an XML database management system (a.k.a. document store) for storing and querying the contents of its allocated containers.

Each server location independently runs a software package which makes available functionality for managing and querying data and metadata via RESTful web services. A client location (or sink), e.g., an electronic flight bag on board of an aircraft, may run a client variant of the software package which provides a subset of this functionality. The software package (in its server and client variants) is distributed from a central software repository.

A client location provides functionality for:

- Allocating an existing semantic container
- Provisioning of semantic containers including content and metadata
- Keeping data and metadata of allocated semantic containers up-to-date via push and/or pull from their primary sources
- Keeping semantic containers up-to-date from alternative sources in case of unavailability of primary sources

A server location additionally provides functionality for:

- Creating a semantic container, storing its primary copy, deriving locations for secondary copies
- Calling services to derive/update the contents of semantic containers
- Forwarding modifications of semantic containers to client containers via push and pull
- Creating, updating and deleting semantic containers
- Discovery of semantic containers

VI. SEMANTIC CONTAINER PROOF-OF-CONCEPT

In this section a proof-of-concept prototype is described in which the semantic container approach is integrated into a SWIM environment. Figure 2 gives an overview about the various systems involved in this scenario. The goal of the scenario is to give an idea how the TRL1 concept can be used in a complete SWIM lifecycle. For the scenario the Frequentis SWIM Registry was integrated to provide not only information about SWIM services but also about semantic containers via SWIM. The SCMS is used to define and create containers that are then visible through the SWIM registry. On an organizational level the Frequentis SWIM integration platform – called MosaiX – serves to configure organization internal the SWIM information for the specific SWIM applications. The information is ultimately accessed by a SWIM application. For the BEST integration we used an existing SESAR 1 prototype [20], namely the Integrated Digital Briefing from that project's WP13.2.2.

A. Integration: SWIM Registry

As a starting point to demonstrate setup and use of semantic containers, a SWIM service registry is introduced. This registry provides a list of available SWIM services and semantic containers. It also allows to query information about this service providers, e.g., source, content, freshness. For the current use case a dedicated Frequentis Semantic Container Service Registry was adapted. A list of available SWIM services and semantic containers is available under "Entities" > "Instances" together with the functionality to show details about the services or edit entries (see Figure 3).

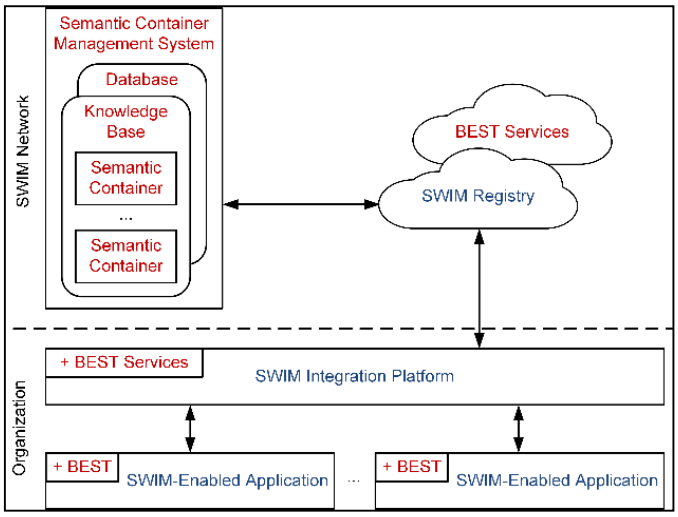

Figure 2. Proof-of concept overview [7]

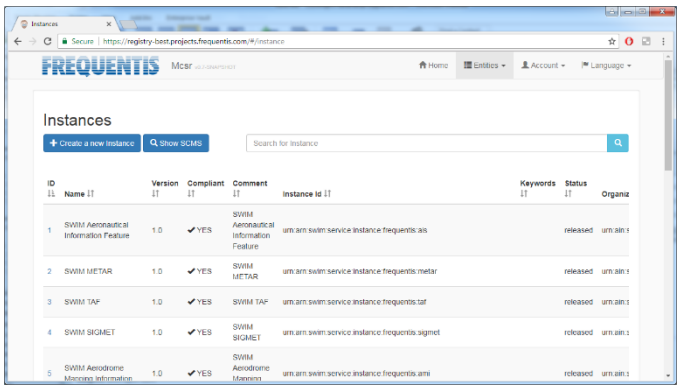

Figure 3. Instances in the Frequentis Service Registry [7]

Besides showing the content of existing semantic containers, it is also possible to create new containers in the "Containers" section of the SCMS. Opening the SCMS allows showing further details about the semantic containers.

B. Prototype: Semantic Container Management System

The SCMS is used to create and maintain semantic containers. In the section "Containers", a new container can be created. The user can either create a new container and select an XML file to be used as payload, or copy an existing container to create compound containers. It is also possible to already create a semantic container restricted to a specific aircraft type. Figure 4 is a screenshot showing the creation of a new container based on a tailored NOTAM container for the specific aircraft type. After creating the container, the container hierarchy can be explored in the SCMS (Figure 4).

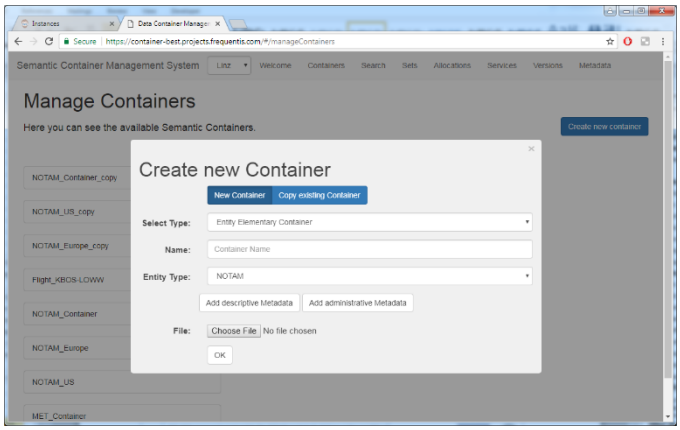

Figure 4. Creating a new Semantic Container [7]

C. *Integration: SWIM Integration Platform*

To configure available data sources for an organization the Frequentis MosaiX SWIM Management Console is used – see Figure 5. There it is possible to establish, manage and monitor relevant data sources for an organization to provide access for those entities with legal permission.

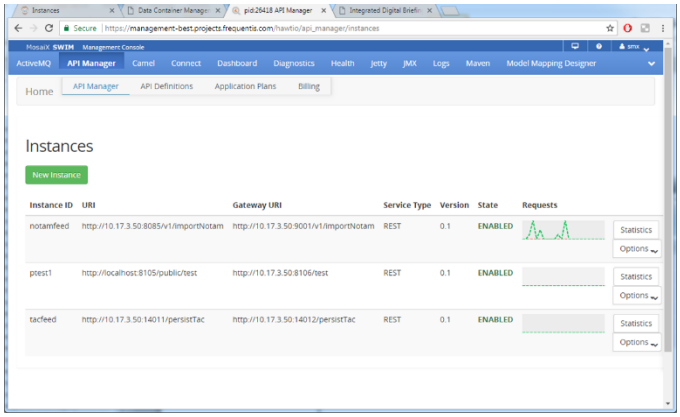

Figure 5. Frequentis MosaiX SWIM Management Console [7]

D. *Integration: SWIM Integrated Digital Briefing*

Based on the described use cases [16] Figure 2 shows the components of the SWIM application with integrated semantic containers. In red one can see the SWIM services that are used to fill the semantic containers needed for the digitally enhanced Pre-flight Information Bulletin (ePIB).

The SWIM application is managed by the organizational SWIM integration platform (see Figure 3), which is responsible for the service management, data mediation and other configuration options. Since the SWIM Integration Platform is also used as the access point to the SWIM Registry all registered SWIM services and semantic container services are available. For SWIM applications it is completely transparent to connect to either a SWIM service or a semantic container. However, to benefit from additional functionality provided in semantic containers (i.e., requesting a defined data quality like freshness or locality) also SWIM applications must be adapted for that purpose.

The prototype has been integrated to use the BEST semantic container concept and is able to retrieve containerized information to be used and save calculation time the application normally would need. The integration of the semantic container concept into an existing SWIM application showed that it can be used without any changes and only little integration is necessary to visualize the added value provided by the semantic containers.

VII. Information Analyses: Semantic Container

To demonstrate characteristics and benefits of semantic containers in the SWIM environment, an information analysis was performed for different usage scenarios. Storage and bandwidth requirements were compared between current settings and the envisioned use of semantic containers. The usage scenarios include a pilot briefing, information needs for a fuelling service at an airport, an airline managing its fleet, flight data for an international flight, as well as running the Network Manager operations centre. In these scenarios, three general benefits could be identified:

- Decoupling of services: Semantic containers decouple information consumers from information service providers and in this way, make it easier to replace and maintain SWIM components.
- Optimization for message distribution: Data provider in the SWIM context process many requests from different applications. With semantic containers providers can package and compress those usually small messages to a single response and deliver the necessary data in a more efficient way, improve reliability of the overall network, and increases response times for SWIM applications.
- Easier testing and monitoring of end-to-end workflows in SWIM networks: Semantic containers can act as black boxes in a SWIM network and allow shielding functionalities behind. When testing a new data provider or consumer a semantic container act as a single interface with defined behaviour and thus allow a wide range of tests in a realistic environment. It would also be possible to record data traffic over a time and then replay this traffic in a test scenario. Additionally, semantic containers occupy critical nodes in a SWIM network and allow therefore monitoring data traffic at the relevant points.

Conclusion and Future Work

The implementation of the SWIM concept enables direct ATM business benefits to be generated by assuring the provision of commonly understood quality information delivered to the right people at the right time [21]. Semantic containers as described in the BEST project build on this concept and establish additional patterns in such an information network. However, considering that as of today still only a limited number of SWIM services is operational we need to acknowledge that any service on top – like semantic containers – will require even more time before they become operational. Nevertheless, more and more SWIM services will become operational over time and it makes sense to already think now

about addressing foreseeable bottlenecks that can be solved with semantic containers.

A replication mechanism for the redundant storage of semantic containers promises higher availability of mission-critical data within SWIM while at the same time reducing the network load of SWIM. By packaging ATM information in semantic containers, SWIM information services may cache often used information and thus avoid frequent calls to other SWIM services. Furthermore, semantic containers are a mechanism to retain provenance information when packaging ATM information from different SWIM information services. Thus, when a composite SWIM information service returns a composite semantic container based upon information from various other SWIM services, provenance information about the semantic container's components is preserved, which is important for auditability purposes.

An SCMS providing mission-critical data and metadata requires special consideration of trustful communication to ensure authentication, integrity, and non-repudiation of data and metadata. In a decentralized system, trust can only be provided based on cryptographic protocols [22]. This was clearly out of scope of the BEST project. Future research needs to investigate how semantic containers can leverage cryptographic protocols (e.g., using blockchain technology) to provide trustful semantic container management and secure SWIM.

The proof-of-concept scenario has shown that the semantic container approach can extend the SWIM concept and add value to it by facilitating data discovery through semantic annotation, thus leveraging necessary benefits in SWIM networks. Since BEST was a TRL 1 project, future work will improve the semantic container concept and validate the SWIM integration in a comprehensive manner. This should include more scenarios, including data from an airline, an airport, and ANSPs and SWIM components such as the SESAR 2020 SWIM registry.

Further details, including the full text of project deliverables (and summaries thereof), information about how to access technical results of the project (software and ontologies), and a short video explaining some technical details of parts of the work, are available on the project website (https://project-best.eu/).

ACKNOWLEDGMENT

This research has received funding from the SESAR Joint Undertaking under grant agreement No 699298 under the European Union's Horizon 2020 research and innovation program. The views expressed in this paper are those of the authors.

REFERENCES

[1] A. Vennesland, J. Gorman, "BEST D6.3 Final Report," 2018. http://www.project-best.eu/publications.html

[2] Aeronautical Information Exchange Model 5.1.1, 2016 [online] Available: http://www.aixm.aero/.

[3] Flight Information Exchange Model 4.1.0 2017 [online] Available: http://www.fixm.aero.

[4] ICAO Weather Information Exchange Model 2.1.1, 2017 [online] Available: https://github.com/wmo-im/iwxxm.

[5] ATM Information Reference Model 4.1.0, 2017 [online] Available: http://www.airm.aero.

[6] A. Vennesland, B. Neumayr and C. Schuetz, "BEST D1.1 Experimental ontology modules formalising concept definition of ATM data," 2017. http://www.project-best.eu/publications.html

[7] E. Gringinger, C. Fabianek, C. G. Schuetz, "BEST D3.2 Prototype SWIM-enabled Applications," 2018. http://www.project-best.eu/publications.html

[8] A. Vennesland, E. Gringinger and A. Kocsis "BEST D5.2 Ontology Modularisation Guidelines," 2017.

[9] M. A. Musen, "Protégé Ontology Editor," 2015. [Online]. Available: http://protege.stanford.edu/

[10] M. D'Aquin, A. Schlicht, H. Stuckenschmidt, and M. Sabou, "Criteria and evaluation for ontology modularization techniques," in *Modular ontologies*, 2009, pp. 67–89.

[11] A. Schlicht and H. Stuckenschmidt, "Towards structural criteria for ontology modularization," in *Proceedings of the 1st International Conference on Modular Ontologies-Volume 232*, 2006, pp. 85–97.

[12] B. C. Grau, B. Parsia, E. Sirin, and A. Kalyanpur, "Modularity and Web Ontologies," in *KR*, 2006, pp. 198–209.

[13] B. Neumayr, E. Gringinger, C. G. Schuetz, M. Schrefl, S. Wilson and A. Vennesland, "Semantic data containers for realizing the full potential of system wide information management," IEEE/AIAA 36th Digital Avionics Systems Conference (DASC), St. Petersburg, FL, USA, 2017, pp. 1-10.

[14] E. Gringinger, C. G. Schuetz, B. Neumayr, M. Schrefl and S. Wilson, "Towards a value-added information layer for SWIM: The semantic container approach," IEEE 18th Integrated Communications Navigation Surveillance Conference (ICNS), Herndon, VA, USA, 2018, pp. 3G1-1-3G1-14.

[15] C. Schuetz, B. Neumayr, M. Schrefl and E. Gringinger, "D2.1 Techniques for ontology-based data description and discovery in a decentralized SWIM knowledge base," 2018. http://www.project-best.eu/publications.html

[16] E. Gringinger, C. Fabianek, B. Neumayr and A. Savulov, "BEST D3.1 Prototype Use Case Scenarios," 2018. http://www.project-best.eu/publications.html

[17] C. Schuetz, B. Neumayr, M. Schrefl and E. Gringinger, "BEST D2.2 Ontology-based techniques for data distribution and consistency management in a SWIM environment," 2018. http://www.project-best.eu/publications.html

[18] A. S. Tanenbaum and M. van Steen, Distributed systems: Principles and paradigms: Prentice-Hall, 2007.

[19] M. T. Özsu and P. Valduriez, Principles of distributed database systems: Springer Science & Business Media, 2011.

[20] C. G. Schuetz, B. Neumayr, M. Schrefl, E. Gringinger and S. Wilson, "Semantics-Based Summarization of ATM Information to Manage Information Overload in Pilot Briefings," 31st Congress of the International Council of Aeronautical Sciences (ICAS), Belo Horizonte, Brazil, 2018.

[21] A. Vennesland, J. Gorman, S. Wilson, B. Neumayr and C. G. Schuetz, "Automated Compliance Verification in ATM using Principles from Ontology Matching," 10th International Conference on Knowledge Engineering and Ontology Development (KEOD), Seville. Spain, 2018

[22] B. Schneier, *Applied Cryptography,* 2nd ed.: John Wiley and Sons, 1996.