HOW SEMANTIC TECHNOLOGIES ENRICH AERONAUTICAL INFORMATION MANAGEMENT FOR ONTOLOGY BASED SOFTWARE DEVELOPMENT

Eduard Gringinger*, Günter Graf*, Harald Milchrahm*, Dieter Merkl**

*Frequentis AG, **Vienna University of Technology

Keywords: Ontology, Semantic, ATM, Control Room Framework, Modular Architecture

Abstract

The Federal Aviation Administration’s (FAA) Next Generation Air Transportation System (NextGen) and EUROCONTROL’s Single European Sky ATM Research Program (SESAR) are transforming the global Air Traffic Management (ATM) as we know it today and will break up with existing roles predicated by 50 year old technology [1], [2]. Similar to SESAR, NextGen right now runs the risk to specify ATM systems based on architectures already out-of-date. While current functionality is based on historical grown technical restrictions, a performance-based and most efficient approach requires new paradigms and eventually has to lead to a balanced approach to prevent over-optimizing one area at the expense of others [3]. The new approach should be based on semantically enriched service models allowing easier development and modular applications for multiple domains. This paper describes an ontology based multi-domain software development approach called Ontology-Based-Control-Room-Framework (ONTOCOR) featuring high software code re-usage and rapid development. It focuses on improving efficiency and increasing the code reusability in order to achieve SESAR’s and NextGen’s claim for a performance-based and cost-efficient system [2]. The main goal is using semantic technologies to enhance software development in ATM and further, to define tasks with enough similarity to allow applicability in different domains. ONTOCOR uses semantic standards and tools, and seamless information interchange. As described in the European Air Traffic Management Master Plan, “The Information Management Work Package (...) defines the ATM Information Reference Model and the Information Service model (...) by establishing a framework, which defines seamless information interchange between all providers and users of shared ATM information” [2]. A specific example to implement inter domain, is the European ATM Information Reference Model (AIRM). In general, domain independent implementation of components is a future goal and EUROCONTROL defines AIRM as a model, which contains all of the ATM information to be shared in a semantic way [4].

The paper will discuss a first case study for ontology-based modeling and development, a service for digital notices to airmen (NOTAM). Fast information distribution and retrieval are key elements together with service modeling and definitions according to Enterprise Architecture (EA) and SOA principles. Aeronautical Information Management (AIM) using ontologies provides the benefit of fast ramp up of on-the-spot services, reduced development efforts and additional defined data as source for collaborative decision making within the System-Wide Information Management (SWIM) [5]. Such services deployed in NextGen and SESAR will optimize operations workflow, communication needs, and information sharing.
1 Context

Control rooms are typically found in the Security, Public Safety (PS), Public Transport (PT), and ATM domains. Today, each of these sectors uses domain specific concepts of operation, which result in different solutions for every targeted environment. This limits the potential for cost efficient software development and increases the time-to-market. Information management, like systems for the ATM or other domains as emphasized before, typically consist of many heterogeneous sub components. Those sub components are mostly implemented with diverse types and structures of data, which result from the circumstance that such complex information management systems are developed for specific business needs. But, when the business scope changes, for example to combine two existing parts, some sort of integration is needed [6]. To win the challenges of the data and system integration, a framework, which defines seamless information interchange, is needed. Exactly within these circumstances an ontology-based approach brings the breakthrough. Semantic structures improve the productivity and increase the reusability of software through a component based framework, which are both key goals of the Pan-European SESAR Joint Undertaking (SJU) [3].

Within ONTOCOR, an Ontology-based Control Room Framework has been setup in order to survey the potential benefits of such an approach over traditional software development. ONTOCOR enhances software development with semantic technologies and further, enables an interchange of different domains with similar types of tasks. An important aspect of a modular architecture is to gain control over accumulation and utilization of control room content. It is necessary to define analytic methods to describe the behavior of interfaces and to enrich the entire set of services semantically. Therefore, an exploration of existing ontology frameworks in the field of software-development in context to the ONTOCOR project was needed [7]. In addition reasoner and visualization tools have been investigated for ONTOCOR [8]. One main goal is to accentuate that ontology-based development has the potential to develop, from a qualitative point of view, better software for mission critical environments in less time and at less cost. This paper will give an overview about the term ontology in context with semantic based-AIM. Finally, the authors will describe a case study of how the ONTOCOR framework is used within software development lifecycle.

2 Ontology in Computer Science

A precise definition of an ontology is not a trivial task. The origin of the word ontology is the field of philosophy. Therefore, it is important to go back in time. The term itself is loan from the Greek word ὄν (being) and λογία (science, study, theory), which has a different meaning in the philosophical context, where it refers to the study of being [9]. Greek philosophers from the Platonic school stated that some categories of being are fundamental. Under the doctrine of Plato, Aristotle (384-322 B.C.) hypothesized four ontological dimensions in his Metaphysics book Theta [10]. In the Middle Ages, European academics used ontological arguments to explain the existence of god in a scientific manner. The argument examines the concept of God, and states that the greatest possible being is on the top in a scale of terms ranging from the bottom to an infinity form of being. These ontological arguments are controversial in philosophy since then [11]. From a modern perspective this argument could be described through an ontology language in a way that God is the overall Thing class, and all other beings are underlying subclasses of Thing.

Computer scientists became interested in ontologies in the 1970s within the research field of Artificial Intelligence (AI) [12]. They were tempted by the applicability to perform certain kinds of automated reasoning on ontologies as computational models, with mathematical logic [13]. Such ontology could for example define classes, relations, formal functions with a concept description and axioms that constrain the interpretation. The first definition of ontology in terms of computer science was created by Tom Gruber in the early 90’s. He
defined ontology as an explicit and formal specification of a shared conceptualization [14]. The word explicit implies that the type of concepts and their constraints are explicitly defined. Formal connotes that the ontology is readable by a machine. And a shared conceptualization is specified to state axioms that do include the possible interpretations for the defined terms, which contain the knowledge of a specific domain and were accepted by a group. This early definition has kicked up much dust, therefore Gruber described the essential points of an ontology in the Encyclopedia of Database Systems in 2009 as a definition of “concepts, relationships, and other distinctions that are relevant for modeling a domain” whereas “the specification takes the form of the definitions of representational vocabulary (classes, relations, and so forth), which provide meanings for the vocabulary and formal constraints on its coherent use” [15]. But there is no all-in-one terminology. Often ontology is defined by its use or in context of the Semantic Web, where the World Wide Web Consortium (W3C) specified ontologies as “formalized vocabularies of terms, often covering a specific domain and shared by a community of users. They specify the definitions of terms by describing their relationships with other terms in the ontology” [16].

Corresponding to Benjamin, Borst and Akkermans [17], first ontologies in technical domains were developed as reusable knowledge libraries. In the field of software engineering, ontologies are often used to refer to what exists in a system model [18]. Per default, all software applications have their own underlying paradigms in form of standardized libraries, components, documentation and files, which also acts like an ontology. However, often this is not enough or the description is poor for some reason, ontologies are precisely made to support under those specific purpose [19]. Through the initial work of Gruber and other computer scientists several markup ontology languages were developed. Most ontologies are based on Description Logics (DL), which are a conglomeration of knowledge representation formalisms [20]. Logical statements relating to roles in form of axioms are the fundamental of the modeling concept, which is the big difference to frame-based languages where a frame specification declares and completely defines a class. DLs are used in AI, information management and metadata integration. Within the context of the Semantic Web several languages based on DL were developed, like DARPA Agent Markup Language (DAML) [21], Ontology Inference Layer (OIL) [22], DAML+OIL [23], Simple HTML Ontology Extensions (SHOE) [24], Resource Description Framework (RDF) [25], Web Ontology Language (OWL) [26] et cetera. OWL for example, is still in a development phase, which means that the language is evolved by the W3C continuously. The first W3C recommendation of OWL came out in 2004 [27] and with the revision of OWL 1.1 in 2007 more expressiveness was added [28]. But OWL 1.1 was only another step to the further development which ended up in OWL 2, which was published by W3C in October 2009 [16] and obtains additional expressiveness through innovative ontological axioms to solve known problems that occur with OWL. Despite the new extensions, the main goal is to facilitate ontology development. The background logic of OWL is the DL SHOIQ [27], and SROIQ [29], which is used in OWL 2.

3 Semantic-based Aeronautical Information Management

Similar to object-oriented languages, a typical OWL ontology consists of instances to represent knowledge items, properties, and classes. But, thinking in object-oriented terms during development with OWL will almost always lead you off target. You have to keep in mind that both paradigms are developed among other circumstances and so have different semantic competence, but there are some parallels. A comparison with the Unified Modeling Language (UML) shows that meta-models are closely related to ontologies and both are model languages to describe and analyze the relations between concepts. However, UML and OWL use classes in a
significantly different way [30]. In UML a class describes a set of software objects which entails the same specifications of features, constraints and semantics. Instance objects share their behavior from the class definition, and all objects in UML are general instances of titled classes. Instances of a class also have run-time semantics in a way that there are notions of static values and variables [31]. In OWL terms, a class is a labeled set of domain related things. Resources (individuals in OWL terms) are simply identifiers, without run-time semantics, state or storage. If an individual fits a criterion of the class, then it will be within the membership of that class. Reasoning allows identifying individuals, from whom you don’t even know that they are within a class. As mentioned before, OWL has an ultimate class called Thing, whereas all other classes are subclasses of it and individuals can only be instances of Thing [16]. The real supremacy of an ontology-based approach lies in the capability to build relationships between instances and classes. The properties of those relationships allow reasoner to make suggestions about them. Consider a brief example (Figure 1), :RunwayClosure (concept) is a specific event of :Notam (property :AeronauticalInformationEvent), :RunwayClosure could be an :Issue (relation), an :Airmen gets alerted through an :Issue (assertion). The labeled relationships isSortOf and isAlertedBy infer the fact that an :Airmen is alerted by a :RunwayClosure, which is a specific :type of :Notam, which in turn is a subclass of :AeronauticalInformationEvent (reasoning).

This reasoning is possible because of the inverse property of isSortOf, which relates the two instances in the reverse direction. Several facts could be inferred from these relationships. Instances can either belong to a set of aeronautical information events or the set of risks, but only specific kinds of events are critical. In terms of ontology languages, classes are disjoint to each other. There are no instances that belong to both. In Figure 1, a :RunwayClosure is some sort of :Issue. However, with the knowledge of this example we could not conclude that some type of :Snowtam is an :Issue. That is possible because OWL follows the open world assumption, which defines that any assertion not stated is indistinguishable. Individuals need not necessarily have a unique name because OWL does not use the unique name assumption. But before starting to dive into the details of the case study, the following section will cover how AIMs evolved.

### 3.1 Modeling Aeronautical Information

Main goal of AIM is to provide operators the right information at the time and place they need it. To ensure this, the concept of net-centric ATM operations is used, supported by potent information filtering. It is important to split the essential data from the huge ATM information that is available before distributing it. Current aeronautical information requirements are declared within the International Civil Aviation Organization (ICAO) Standards and Recommended Practices (SARPs), especially in Annex 15 [32], which defines the aeronautical information services. With these requirements in combination with the digital age a model for the European AIS Database (EAD) called Aeronautical Information Exchange Model (AIXM) was created in Europe. In 2003 EUROCONTROL and FAA established a cooperation, which led to the current AIXM version 5.1. The actual version breaks up limitations of previous version 4.5, as it models more than only static data. In 2008, an ICAO Study Group has been formed which main objective is to globally harmonize aeronautical information data.
exchange. AIXM 5.1 also covers temporary data updates such as NOTAM contains. As described in the European ATM Master Plan [2], digital NOTAM is a primary objective within SESAR and will be operational ready within 2012. The European Commission delivers additional requirements through the aeronautical data quality interoperability regulation, which was developed for the single European sky phase 1 project. In a service-oriented approach, data exchange and its fundamental data models and the processing of data are decoupled, in order to prevent over-reliance by the end-user applications on the data models. To fulfill these Service-oriented architecture (SOA) requirements, known elements such as airspace, aerodrome, flight procedure as well as common definitions such as geometry and time are considered in the frame of the ATM Information Reference Model (AIRM).

![Figure 2: AIRM Semantic and Common Syntax](image)

Through harmonized conceptual and logical data models AIRM provides a definition of all ATM information. It is used as a common reference for the different models (Figure 2) that are developed as part of SESAR. A first version of the AIRM, so called initial load, was established in summer 2010 based on existing data models such as the Weather Information Exchange Model (WXXM), AIXM and industry standards like Geography Markup Language (GML) [33] or ISO 19103 the geographic information conceptual schema language, ISO 19107 the geographic information spatial schema and ISO 19108 the geographic information temporal schema¹. In fact GML is not part of the AIRM. AIRM uses only the conceptual schemas from the ISO 19100 series e.g. ISO 19108. GML appears at the physical modeling level. Further increments of the AIRM are planned every six months. The AIRM is defined using UML and used in the context of Service Modeling based on the Service oriented architecture Modeling Language (SoaML). To facilitate reusability and software development itself, ontologies are used during the prototype development. Semantic interoperability is significant for the ATM system development. In this regard the AIRM will be open to a change in its scope and will be capable of absorbing new requirements as they are identified. This implies that the AIRM will stand clear from constraints with respect to the ongoing ATM business, which may change over time. To implement successful prototypes around the AIRM, the ONTOCOR framework is used to gain reusable software components. One of the first prototypes is being developed for digital NOTAMs.

### 4 SWIM Vision

For the domain system and its users to operate at their full potential, pertinent information needs to be available when and where required. Indeed, the ATM community increasingly depends on the provision of timely, relevant, accurate, accredited and quality-assured information in order to collaborate and make informed decisions. Sharing the best possible integrated picture of the historical, real-time and planned or expected future state of the ATM situation on a system-wide basis will allow the ATM community to conduct its business and operations in a safer and more efficient manner. These requirements are supported by System Wide Information Management, through an interconnected set of domain systems providing or consuming information, including human users and aircraft. Through SWIM, information is made available and processed through services which need to conform to applicable standards and be registered so that they are accessible. In addition, SWIM improves the interconnectivity...

¹ [http://www.iso.org/iso/home.html](http://www.iso.org/iso/home.html)
of domain systems. SWIM promotes and contributes to open standards, and it also provides technology recommendations. The aim of this is to improve information management and therefore information sharing on a wide basis, providing support for permanent dialogue between the various partners. It also enables wider discoverability of the pertinent information, while making it easier and less costly to share.

Aircraft operators will have up-to-date, accurate and integrated information on which to base decisions about their flights, while ATM service providers, including aerodrome operators, will have a better knowledge of flight intentions for operational and planning purposes. Thereby, controllers, pilots, dispatchers and other flight operational personnel will share a common situational awareness with regard to the status and condition of the aeronautical infrastructure, the weather, the air traffic situation and other operationally significant information. On the basis of this shared situational awareness, the ATM actors will make better and faster decisions collaboratively for the purpose of orchestrating and conducting highly efficient operations. The definition of SWIM is as follows: “SWIM consists of standards, infrastructure and governance enabling the management of ATM information and its exchange between qualified parties via interoperable services” [34].

3.1 Concept of Operations Components

The operational concept of ATM (ICAO Doc 9854) foresees the implementation of IM and SWIM. Prior to explaining the scope of the Concept of Operations, the SWIM concept is encompassing all the processes and changes to assure that the full lifecycle of all ATM information is managed systematically and in a system wide manner throughout the ATM enterprise ensuring that the right information is available at the right time and at the right location.

- SWIM: System Wide Information Management is the gateway to access ATM information and to discover and subscribe to data services (It is the ATM implementation of Information Management and covers the SWIM information as well as the SWIM technical infrastructure. The information itself may come from datasets maintained and operated by different organizations).
  - SWIM information: SWIM information represents all ATM data exchanged through the SWIM technical infrastructure.
  - SWIM data management: SWIM data management comprises all definitions, standards and processes that are required to deliver ATM data, information and services in the right quality.
  - SWIM technical infrastructure: Technical capabilities that are, or need to be put, in place with ATM stakeholders to support having the right information available at the right time and at the right location. This covers both ‘design-time’ and ‘run-time’ capabilities.
  - (SW)IM functions: The functions addressing the broad range of processes that need to be put in place to deliver the (SW)IM concept.

The SWIM concept concerns and involves all participants who have a stake in, or a right to, the shared ATM information. The scope of the SWIM concept covers the collection and management of ATM information from one or more sources and the interoperable distribution of that information to one or more consumers. This implies SWIM activities which entail retrieving, acquiring, maintaining and exchanging the shared ATM information on a system wide basis. SWIM activities are performed with the support of a set of orthogonal SWIM functions which concern the collaborative management of organizing and controlling ATM information sharing and distributing. Furthermore, running SWIM relies on the availability of a SWIM Technical Infrastructure and its services for exchanging ATM information between the SWIM participants. The SWIM Technical
Infrastructure is the technical part and an enabler of SWIM.

As presented in the above figure, SWIM starts with the standardization of the information on the participants’ side, which is referred to as SWIM data management. It covers the interoperability, through the alignment of processes and procedures. The SWIM information itself may come from datasets maintained and operated by different organizations. The SWIM technical infrastructure covers the technical aspects, from the system interface standardization to the interaction, transport and the enabling technical services, such as registration, authentication, supervision, etc. The SWIM data management deals with:

- (SW)IM functions: The functions addressing the broad range of processes that need to be put in place to deliver the (SW)IM concept.
- General Principles, Rules and Recommendations regarding ATM Information
- Information Modeling Conventions and common meaning: AIRM (ATM Information Reference Model)
- Information Services Specifications: ISRM (Information Services Reference Model), registry services

The SWIM information deals with:

- All ATM data exchanged through the SWIM technical infrastructure (although this information itself may come from datasets maintained and operated by different organizations)

The SWIM technical infrastructure deals with:

- All ATM data exchanged through the SWIM technical infrastructure (although this information itself may come from datasets maintained and operated by different organizations)
- The technical implementation to exchange ATM Information, in support of SWIM.
- The technical services needed to exchange the ATM information.

The SWIM concept addresses IM, the technical infrastructure and the IM functions. In order to exchange information, the participants need to enable information sharing through services. These services are exposed on the SWIM technical infrastructure.

5 Case Study

This case study describes the implementation of applications upon aeronautical information models with a modern software development framework. Through the static growth of data, which is processed within a control room, overlapping software components for different domains have been developed. It is often the case that the descriptive name is different, but the functionality of such components is quite the same. The motivation of ONTOCOR is to show that the improvement of an ontology-based framework can be approved as a real business-case. Therefore, it is necessary to find the best fitting domains. AIM services in the domain of weather data, geographic information, briefing, tracking and tracing are all together systems which have approximately the same requirements for different domains. Nevertheless, such components are often developed twice for each domain for example the component for visual representation. The use of ontologies provides high flexibility for the future integration of new legacy
applications, systems and services. Unified and open standards can raise the reuse of components for different applications in different domains. Ontologies, semantic annotation of content and semantic search are technologies, addressing the problems outlined above. They open up new ways to benefit from already developed systems. A digital NOTAM service was implemented as a prototype by the methodology of the Ontology-based Control Room Framework. A significant point for the whole project is that it follows the idea of open standards. It was decided that ONTOCOR would rely on open source solutions only. The ONTOCOR framework is mainly supported by different state-of-the-art ontology languages as well as relevant semantic environments for ontology development. Several research have identified necessary tools and languages [7] like Frame Logic, RDF, RDF(S), OWL, OWL 2 and SPARQL, a standardized RDF query language. The knowledge in ONTOCOR is captured with OWL 2, due to the fact that it has the most complete set to express different concepts and relationships that occur within an ontology. As different ontology languages have different facilities, it was necessary to evaluate them. Of similar importance is the right choice of frameworks and tools.

Step 1 - Ontology Development

In Step 1 different ontology models have been developed according to the use case requirements, an information domain model and a software component model (Figure 4). In the case of the digital NOTAM prototype the requirement model was filled with data provided by the Air Navigation Service Providers (ANSPs). The necessary part of the AIRM was taken into account to build up an information domain model. The visualization of obstacles within a Geographic Information System (GIS) was used for the software component model. In Step 1, tools for ontology development are required. A relevant mix of open source and commercial environments like Protégé, NeOn toolkit, Jena framework, OntoStudio, TopBraid Composer and Altova SemanticWorks were already analyzed for ONTOCOR [7]. The ONTOCOR framework is mainly supported by Protégé 4.1 in combination with the NeOn Toolkit. The need of selecting two environments, Protégé and NeOn, is that both support OWL 2 and have alternate strengths and weaknesses because of their different purposes.

![Figure 4: ONTOCOR Framework Workflow](http://protege.stanford.edu/)

Figure 4: ONTOCOR Framework Workflow

Together they are completing each other and are used in Step 1 to model the three layers provided by the ONTOCOR framework (Figure 4): One for the domain information model, one for requirements and another for re-usable components.

Step 2 - Semantic Reasoning, Matching

In Step 2 these models are then merged into one single semantic description by reasoner tools. A serious problem with ontologies is to develop different types, which in the end cannot be merged because of semantic and structural incompatibility. A workaround is to interlink the different types not to match them together. To avoid this possible risk existing semantic methodologies and existing ontologies in the

---


environment has to be further evolved. ONTOCOR is structured in different phases to implement the goals described above. These areas reflect the usual approach of an integration project by first describing requirements, afterwards developing a solution model that meets the demands. Figure 4 gives an overview of the main components of ONTOCOR and shows the split into three different steps. The relationships between the three model layers are merged together [35], [36] in the design phase. Based on the match of different ontologies without interfering with the software system, a semantic description is developed. In Step 2 tools for the production of the semantic description are used, such as reasoners, alignment tools or visual representations. The fact that relationships in OWL 2 are formally defined, offers the possibility to use a reasoner. One main service that such reasoning system can determine is to test whether or not one class is a subclass of another class such as shown in Figure 1. AeronauticalInformationEvent has the subclass Notam. This relationship is called a necessary implication. So we can conclude that because :RunwayClosure is some sort of :Notam, and all types of :Notam are :AeronauticalInformationEvent, then a :RunwayClosure is also a type of :AeronauticalInformationEvent. A reasoner can show that the class of NOTAM is a valid subclass of aeronautical information events, and that it contains at least one member. Such a test allows a reasoner to compute the ontology’s inferred class hierarchy and could discover if a given class has any instances. Without any instance you can properly conclude that a class is inconsistent. Protégé 4.1 enables the opportunity to take advantage of different OWL 2 reasoners as a plug-in. This all sounds great in theory, but often semantic reasoners are incomplete in order to reach the required scalability, which means that they could not guarantee to provide only valid output. An excellent insight around that topic provides a paper [37] from Giorgos Stoilos et.al. published at the Oxford University Computing Laboratory. Within the ONTOCOR project the capabilities of some state-of-the-art reasoners, which support OWL 2 were also evaluated [8]. In addition, FACT++ and Pellet are selected for reasoning. The decision was based on the fact that both are compatible with OWL 2 and are well supported. OWL 2 is currently the best choice, especially for building complex ontologies. For the developer it is also necessary to select visualization tools, which support OWL 2. Visualization often deals with abstract data and offers a bundle of techniques to represent hierarchical or semi-structured data. There are several of studies where different ontology visualization tools are compared [38], [39]. Considering the variety of methods and approaches to visualize ontologies, such tools can be separated into two big groups. One category uses variations of simple lists, the other uses simple types of visualizations like two-dimensional trees, node-links or even offers 3D information. As Protégé and the NeOn Toolkit were picked to use within ONTOCOR, the visualization tools OWLViz\(^4\) in combination with OWLdiff\(^5\), Matrix\(^6\) and Cloud View\(^7\) complete the list of tools, which are used within the ONTOCOR project [8].

**Step 3 – Solution Model**

In Step 3 the model transformation algorithm (MTA) provides a key functionality by transferring the semantic description into a solution model Figure 4. The solution model is then able to provide the developer the right software components needed for further implementation. At this point are the main strengths of an ontology-based development come to light. Redundancies and overlaps are detected in earlier stages of the development. Already existing software components can be recycled. The full line of power of ONTOCOR comes to light, only when it is used in multiple domains, which overlaps in some way. The potential is even higher more areas are covered.

---

\(^5\) [http://clarkparsia.com/pellet](http://clarkparsia.com/pellet)
\(^7\) [http://krizik.felk.cvut.cz/km/owldiff/index.html](http://krizik.felk.cvut.cz/km/owldiff/index.html)
\(^8\) [http://clarkparsia.com/pellet](http://clarkparsia.com/pellet)
For future development within the ONTOCOR framework it is planned to integrate this functionality into an integrated development environment (IDE).

4.1 Digital NOTAM

The implementation of digital NOTAM in Europe requires applications, which support both, static and dynamic data. The event specification is done in parallel by EUROCONTROL and FAA. A digital NOTAM event could be parsed through automatic checks, which validate them against the ICAO standards to improved coherence and correctness. EUROCONTROL specifies a digital aeronautical information update as "a data set made available through digital services containing information concerning the establishment, condition or change in any aeronautical facility, service, procedure or hazard, the timely knowledge of which is essential to systems and automated equipment used by personnel concerned with flight operations"\(^{10}\).

But a digital NOTAM is not just the transformation into a machine readable, structured format. It is a significant change by which the information updates (both temporary and permanent) are merged with the information of longer duration, using the same data structures and distribution channels.

Within the EAD a digital NOTAM prototype was developed to provide an efficient and user-friendly way to handle Digital NOTAMs (Figure 5). Common encoding and data validation rules are the key for a successful implementation of the Digital NOTAM concept. A catalogue of events is stored in an ontology, for example a runway closure or a navaid unserviceable, in order to specify the rules for that type of event. ONTOCOR offers a primary version of a generic infrastructure for highly reliable information models between heterogeneous domains and has an integrated infrastructure for a domain-specific layer, which defines patterns and configurations in any specific domain. Dynamic information as a NOTAM can be linked to static data elements, making it possible to identify e.g. all NOTAM affecting a specific facility. Traditional NOTAM received by data originators or other ANSPs can be processed as ICAO NOTAM or digitized to digital AIXM 5 compatible NOTAM (xNOTAM). It provides facilities for processing, checking, and creating international and national NOTAM (including SNOWTAM, ASHTAM etc.) and other relevant dynamic aeronautical data.

Figure 5: Digital NOTAM Runway Closure

\(^{10}\) http://www.eurocontrol.int/aim/public/standard_page/xnotam.html

Figure 6: 3D view showing a Digital NOTAM
This innovation is now used within smartINMO a subsystem of the smartAIM\textsuperscript{11} product suite, a commercial off-the-shelf product, owned by Frequentis. The product consists of a number of modules, which can be selected freely. Individual modules can be combined as required and even be replaced by third party modules if desired.

There is also a smartAIM GIS visualization tool to graphically display the result of NOTAM (Figure 6), METEO data, etc. It offers interfaces based on the Java Message Service (JMS) standard using Extensible Markup Language (XML) based data exchange and Web services. It furthermore supports easy collaboration through many other versatile interfaces, which can be customized to client requirements easily. smartAIM is truly net-centric. It consists of server components, which are location transparent. I.e. the server components can be centralized or distributed with no difference for the client workstations.

The current study shows that, the approach of ONTOCOR goes in the right direction. In future’s environment, software development will more and more profit from the reuse of code. But additional research is necessary to prove the economic benefit in detail and to improve the time effort to use ontologies during the software development process.

5 Conclusion

In order to meet the challenges of future aviation, tools and languages for ontology-based software development were presented in this paper. In modern environment, software development profits from the reuse of code to improve the economic benefit. To achieve those benefits, specific information models for different ATM domains have been developed and the development will progress in the future. As shown in the case study, this requires more than simple UML-descriptions. Semantic logic is necessary to design and develop within a component-based architecture. Such a venture has far reaching effects on systems, elements, procedures and regulations, but is required to achieve the benefits of an Ontology-based Framework and fulfill the key goals of NextGen and SESAR. It is still a dream to generate directly out of a data model useable lines of code. But ontology-based software development in combination with SOA models has already come very close. Further research is needed to increase the efficiency of the development framework ONTOCOR and to prove the economic benefit in detail.

\textsuperscript{11}http://www.frequentis.com/Internet/AirTrafficManagement/Informatio\textsuperscript{p} Solutions/smart\textsuperscript{+}AIM/
References


Engineering Ontology, Fourth International Conference on Semantics, Knowledge and Grid.


[34] SESAR, 2011, SWIM Concept of Operations, Deliverable A01, Edition 00.02.00.


Copyright Statement
The authors confirm that they, and/or their company or organization, hold copyright on all of the original material included in this paper. The authors also confirm that they have obtained permission, from the copyright holder of any third party material included in this paper, to publish it as part of their paper. The authors confirm that they give permission, or have obtained permission from the copyright holder of this paper, for the publication and distribution of this paper as part of the ICAS2012 proceedings or as individual off-prints from the proceedings.

About the Authors

Eduard Gringinger is Project Manager, SESAR Project Member and Software Researcher at FREQUENTIS. He is Ph.D. Student at the Vienna University of Technology, Institute of Software Technology and Interactive Systems. He finished his master studies in media and computer science in October 2007 and his master studies in management and computer science in March 2008. In both studies he received distinction. His major research interests are in the areas of information processing, software development and aeronautical information management. Mr. Gringinger works actively to apply modern technologies to SESAR, in areas of information modeling and flight data management, especially in the meteorological domain.

Günter Graf is New Business Development Manager for FREQUENTIS. Dr. Graf has 6 years of experience in engineering analysis and product strategy in communications systems for mission critical applications. He received his master’s and doctoral degrees from the department of Production/Operations Management, Business Logistics and Environmental Management at the University of Klagenfurt. He is involved to bring new technologies in to the Air Traffic Management and Tower Automation Systems. He is also working in the field of the integrated controller working positions optimizing the future symbiosis between controllers and the ATM system.

Dieter Merkl is Associate Professor of Applied Computer Science at the Department of Software Technology and Interactive Systems, Vienna University of Technology. He received his master’s and doctoral degrees in Social and Economic Sciences from the University of Vienna. His major research interests are in the areas of multi-user virtual environments, data mining and machine learning. He has published more than 150 papers in conference proceedings and journals.

Reviewed by:

eduard.gringinger@frequentis.com

guenter.graf@frequentis.com

dieter.merkl@ec.tuwien.ac.at