



Ontology Modularisation Guidelines

Deliverable 5.2

BEST

Grant:

699298

Call:

H2020-SESAR-2015-1

Topic:

Sesar-03-2015

Information Management in ATM

Consortium coordinator:

SINTEF

Dissemination Level:

PU

Edition date:

[09 March 2018]

Edition:

[01.03.00]

Founding Members



Authoring & Approval

Authors of the document

| Name/Beneficiary | Position/Title | Date |
|----------------------------------|----------------|------------|
| Audun Vennesland (SINTEF) | Project Member | 12.02.2018 |
| Eduard Gringinger (FRQ) | Project Member | 23.01.2018 |
| Andrej Kocsis (SLOT) | Project Member | 19.01.2018 |

Reviewers internal to the project

| Name/Beneficiary | Position/Title | Date |
|-----------------------------|----------------|------------|
| Scott Wilson (ECTRL) | Project Member | 24.01.2018 |

Approved for submission to the SJU By — Representatives of beneficiaries involved in the project

| Name/Beneficiary | Position/Title | Date |
|---|----------------|------------|
| Approved by consortium, in accordance with procedures defined in Project Handbook | All partners | 09.03.2018 |

Rejected By - Representatives of beneficiaries involved in the project

| Name/Beneficiary | Position/Title | Date |
|------------------|----------------|------|
|------------------|----------------|------|

Document History

| Edition | Date | Status | Author | Justification |
|----------|------------|--|------------------|--|
| 00.00.01 | 23.03.2017 | Document created | Audun Vennesland | First draft |
| 00.00.02 | 02.10.2017 | PCOS Proposed | Audun Vennesland | Prepared document for internal review (Planned Content and Structure) |
| 00.00.03 | 01.11.2017 | PCOS Proposed revised from comments from internal review | Audun Vennesland | Added some material and placeholders for aspects related to forming and maintaining networks of ontology modules |
| 01.01.00 | 18.01.2018 | Final draft | Audun Vennesland | Version circulated for QA among co-authors |

| | | | | |
|----------|------------|-------------------|------------------|---|
| 01.02.00 | 23.01.2018 | Final draft | Audun Vennesland | Version sent to internal review (External Proposed) |
| 01.03.00 | 12.02.2018 | External Approved | Audun Vennesland | Final version addressing comments from internal review |
| 01.03.00 | 09.03.2018 | Released | Joe Gorman | No changes in content; updated dates and approval status. |



Achieving the **BE**nefits of **SWIM** by making smart use of Semantic Technologies

This deliverable is part of a project that has received funding from the SESAR Joint Undertaking under grant agreement No 699298 under the European Union's Horizon 2020 research and innovation programme.

Executive Summary

This document describes guidelines on how to extract ontology modules from monolithic ontologies in context SWIM environment. Monolithic ontologies are typically large in size and complex in structure and will, in many cases, include more concepts, properties and instances than needed for a particular use case. Ontology modules on the other hand represent subsets of a monolithic ontology that can be customised to encompass only the entities required for describing a single knowledge domain and/or a particular purpose. Ontology modularisation is a process whereby ontology modules are automatically obtained from monolithic ontologies using a variety of techniques. The rationale for operating with modules instead of their monolithic counterparts can be, for example, improved performance, usability and maintainability. Supported by a well-established theoretical framework, the document recommends a set of step-by-step guidelines combining theoretical principles for ontology modularisation with lessons learned from the modularisation efforts performed in the BEST project.

Table of Contents

| | |
|---|-----------|
| Executive Summary | 4 |
| 1 Introduction: About this document..... | 7 |
| 1.1 Purpose..... | 7 |
| 1.2 Intended Readership..... | 7 |
| 1.3 Relationship to other deliverables | 7 |
| 1.4 Acronyms and terminology..... | 8 |
| 2 Background knowledge | 10 |
| 2.1 Theoretical foundation on ontology modularisation..... | 10 |
| 2.1.1 Motivation behind ontology modularisation | 10 |
| 2.1.2 Challenges of ontology modularisation | 10 |
| 2.1.3 Ontology modularisation strategies | 11 |
| 2.2 Ontology modules developed in BEST..... | 11 |
| 2.2.1 Monolithic AIRM Ontology | 12 |
| 2.2.2 Ontology Modules in BEST | 12 |
| 3 Approach | 13 |
| 4 Guidelines for modularisation | 15 |
| 4.1 Identifying the purpose of modularisation..... | 15 |
| 4.2 Selecting a modularisation approach | 15 |
| 4.3 Defining modularisation criteria | 15 |
| 4.4 Selecting a base modularisation technique..... | 16 |
| 4.5 Parametrising the technique and applying it | 16 |
| 4.6 Combining results | 17 |
| 4.7 Evaluating the modularisation | 17 |
| 4.8 Finalising the modularisation..... | 17 |
| 5 Example of using the modularisation guidelines | 19 |
| 5.1 Identifying the purpose of modularisation..... | 20 |
| 5.2 Selecting a modularisation approach | 20 |
| 5.3 Defining modularisation criteria | 21 |
| 5.3.1 Criteria from an application context..... | 21 |
| 5.3.2 Criteria from a governance context..... | 21 |
| 5.4 Selecting a base modularisation technique..... | 22 |

| | | |
|------------|---|-----------|
| 5.5 | Parametrising the technique and applying it | 22 |
| 5.6 | Combining results | 23 |
| 5.6.1 | A second iteration of modularisation | 23 |
| 5.6.2 | Import dependencies..... | 23 |
| 5.7 | Evaluating the modularisation | 25 |
| 5.7.1 | Size | 25 |
| 5.7.2 | Topic-based scope | 25 |
| 5.7.3 | Redundancy | 26 |
| 5.8 | Finalising the modularisation..... | 27 |
| 6 | Conclusions | 30 |
| 7 | References | 31 |
| | Annex 1: Modularisation tools..... | 32 |
| | Annex 2: Description of the ontology modules | 34 |

1 Introduction: About this document¹

1.1 Purpose

The purpose of this document is to describe a set of guidelines for ontology modularisation in a SWIM environment. As the scope is limited to the SWIM environment of the document focuses on ontologies and software applications representing ATM knowledge using SWIM information models, in our case with an emphasis on the AIRM. Ontology modularisation is the task of decomposing a monolithic ontology to a set of sub parts. Reasons for doing this can be to improve performance, usability, maintainability, and re-usability, to name a few. The document combines theoretical principles on ontology modularisation with experiences drawn from the development of a set of ontology modules in the BEST project.

1.2 Intended Readership

This document is targeted towards people having an interest in:

- Aeronautical information exchange
- Application of semantic technologies in ATM
- SWIM (System Wide Information Management)

1.3 Relationship to other deliverables

Table 1. Relationship to other deliverables in BEST

| Deliverable | Relationship |
|---|---|
| D1.1 Experimental ontology modules formalising concept definition of ATM data | D1.1 delivers the ontology infrastructure in BEST that will be used for describing and supporting retrieval of relevant aeronautical data by applications developed in other work packages of the project. |
| D1.2 AIRM Compliance Validator | D1.2 delivers a prototype application called the AIRM Compliance Validator. This application enables automatic mapping between elements in different ontologies (monolithic ontologies and ontology modules). |

¹ The opinions expressed herein reflect the author's view only. Under no circumstances shall the SESAR Joint Undertaking be responsible for any use that may be made of the information contained herein.

| | |
|--|--|
| D2.1 Techniques for ontology-based data description and discovery in a decentralized SWIM knowledge base | D2.1 provides requirements with regards to how modules should be represented in order to fulfil the needs for vocabularies describing the data residing in the semantic containers. |
| D2.2 Ontology-based techniques for data distribution and consistency management in a SWIM environment | D2.2 provides requirements with regards to how modules should be represented in order to fulfil the needs for supporting data distribution and consistency management through ontology based techniques. |
| D3.1 Use Case Scenarios | This deliverable is delivering the use-cases and therefore is the baseline of the initial idea of BEST. This is also important for the modularisation as it defined the domains used as input for the ontology modules. |
| D3.2 Prototype SWIM-enabled applications | The prototype applications developed in D3.2 brings input that must be considered during the formulation of ontology modules. Such input comes both from the development of the applications and when experimenting with them once they are developed. |
| D4.3 Governance recommendations for the use of semantic technologies in SWIM | This deliverable describes an overall approach to governance that deals with the emergence and evolution of semantic technologies in ATM, including ontology modules developed following the guidelines in this deliverable. |
| D4.4 Tutorial for Software Developers | Guidelines developed in this deliverable should be included in the tutorial for software developers enabling the developers to make conscious decisions with regards to for example size (coverage) and complexity of ontology modules when they are used. |

1.4 Acronyms and terminology

Table 2. Acronyms and abbreviations used in this deliverable report

| Definition | Explanation |
|------------|---|
| AIRM | ATM Information Reference Model |
| AIXM | Aeronautical Information Exchange Model |
| API | Application Programming Interface |
| ATM | Air Traffic Management |

| | |
|-------|---|
| FIXM | Flight Information Exchange Model |
| GML | Geography Markup Language |
| ICAO | International Civil Aviation Organization |
| IWXXM | ICAO Meteorological Information Exchange Model |
| METAR | Aviation routine weather report (in aeronautical meteorological code) |
| OWL | Web Ontology Language |
| SWIM | System Wide Information Management |
| UML | Unified Modelling Language |
| W3C | World Wide Web Consortium |
| XMI | XML Metadata Interchange |
| XML | eXtensible Markup Language |
| XSLT | eXtensible Stylesheet Language Transformation |

2 Background knowledge

This chapter provides some foundation on the topic of ontology modularisation. This includes motivation for performing modularisation, some relevant modularisation strategies and challenges, and an overview of the modules developed in BEST.

2.1 Theoretical foundation on ontology modularisation

Monolithic ontologies are typically characterised as ontologies large in size and complexity, and often spanning several different topics and knowledge areas. Ontology modules on the other hand, aim to provide ontology users with the specific knowledge they require, reducing the scope as much as possible to what is strictly necessary [1]. An ontology consists of a set of axioms, i.e. logical statements, that holds some knowledge. An ontology module encapsulates a subset of the axioms compared to the “monolithic” ontology. For example, if we are interested in only the knowledge related to the concept Aircraft in AIRM, we can represent this knowledge in an Aircraft ontology module, while disregarding other axioms from the AIRM ontology that are not relevant for expressing knowledge about an Aircraft.

2.1.1 Motivation behind ontology modularisation

Developing and maintaining large ontologies can be a cumbersome and sometimes overwhelming task due to their size and complexity [2]. Ontology modules, on the other hand, promote use, reuse, simpler maintenance, enable distributed engineering over different geographical locations and different areas of expertise, enable effective management and navigability, and will (in most cases) result in faster processing of reasoning operations.

In the BEST project, there are several arguments for dealing with modules rather than monolithic versions of the ontology infrastructure. At the application level the techniques developed in work package 2 perform reasoning operations to offer the most relevant ATM information to the end user. These techniques will not scale if such reasoning operations are run against large monolithic ontologies such as the AIRM ontology. From a governance point of view, maintenance of large models such as the AIRM needs to be performed by a team of experts. By distributing the maintenance effort so that an individual or a team of individuals is responsible for a single module or set of modules focusing on a topic-specific scope (e.g. meteorology or airport infrastructure related ontology structures) would bring several advantages, such as a better overview of the entities within a defined scope, clearly defined dependencies, and improved version management.

2.1.2 Challenges of ontology modularisation

There are, however, several challenges related to modularisation. One challenge is finding exactly the set of axioms describing properties related to the knowledge domain in question as an *a priori* activity before performing the actual modularisation. Another challenge is defining an adequate size of the modules. If the modules become too large, many of the issues with monolithic ontologies still remain. If they become too small, there might be too many modules to manage and maintaining an overview and keeping the ontology network consistent can become challenging. There are also challenges related to restricting the boundaries of modules, that is, ensuring that the modules are self-contained

and that they interlink with other relevant modules in order to establish an ontology network without inconsistencies such as cyclic and transitive dependencies.

2.1.3 Ontology modularisation strategies

There is no universal approach to ontology modularisation, it depends on the application requirements and context in which the modules should be developed and maintained. There are however two main strategies for splitting up a monolithic ontology into ontology modules, 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 [3] applied structural characteristics such as target module size and number of target modules to determine suitable partitions of an input 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 [4].

In BEST, we have used the ontology module extraction strategy and more specifically a technique called Syntactic Locality Modularisation [5] for extracting ontology modules from 3 ATM ontologies developed in task 1.1 [6]. These ontologies and the extracted modules are described in the next chapter.

2.2 Ontology modules developed in BEST

The ontology development in BEST is based on transforming information models in UML to OWL ontologies and then obtaining a set of ontology modules for each monolithic OWL ontology using different strategies. The information models that have been transformed to OWL ontologies in BEST are:

- ATM Information Reference Model (AIRM)
- Aeronautical Information Exchange Model (AIXM)
- ICAO Meteorological Information Exchange Model (IWXXM)

In this document, we focus on the modules obtained from the monolithic AIRM ontology. Furthermore, the scope is restricted to encompass only a subset of the AIRM. This scope includes information structures relating to the airport, the aircraft and weather-related information². It was not within the

² This scope was decided during the kick-off of the BEST project in June 2016.

scope of this project to transform Flight Information Exchange Model (FIXM), but this should be a future task to complete the picture.

2.2.1 Monolithic AIRM Ontology

AIRM is a reference model that addresses semantic interoperability through harmonised and agreed definitions of the information being exchanged in ATM [7]. In D1.1 [6] of the BEST project a monolithic AIRM OWL ontology was developed using XSLT to transform from the original AIRM UML model to an OWL representation of AIRM.

2.2.2 Ontology Modules in BEST

Table 3 lists the ontology modules obtained from AIRM and some statistics associated with them.

Table 3. Ontology modules in BEST

| Ontology Module | Classes | Object properties | Data properties | Individuals |
|-------------------------|---------|-------------------|-----------------|-------------|
| AIRM-Aircraft | 71 | 84 | 33 | 182 |
| AIRM-BaseInfrastructure | 298 | 463 | 133 | 1574 |
| AIRM-Meteorology | 58 | 69 | 15 | 97 |
| AIRM-Stakeholders | 106 | 131 | 40 | 316 |
| AIRM-Common | 58 | 44 | 19 | 396 |
| AIRM-Flight | 177 | 265 | 48 | 264 |

3 Approach

This chapter elaborates on the research question relevant for this work before we describe the overall approach followed to address it.

The research question addressed in this work is the following:

In order to (ultimately) provide ontology-based modelling of the full breadth of all ATM information, how can a practical modular approach be designed that balances factors such as scope, scale, overlap and data distribution?

In order to address the above research question there are some clarifications that needs to be made.

By *scope* we mean that a module should ideally focus on a particular topic-specific scope and not (as the monolithic ontologies) include entities (axioms) that span several knowledge domains. This is a particularly important criterion with respect to maintenance. Furthermore, scope also means that the modules should as much as possible be self-contained. For example, when performing reasoning on a module about a certain concept this would result in the same conclusion as if the reasoning was performed on the original (monolithic) ontology from which the module was extracted.

By *scale* we mean that when applications interact (e.g. perform reasoning) with the modules their scalability constraints should be acceptable.

By *overlap* we mean that ontology entities (classes, properties and individuals) for describing a certain piece of knowledge should be kept within a single module and that there are no overlapping entities in modules within the same ontology network. It could very well be that a knowledge request cannot be fulfilled by a single module, but then the appropriate import declarations should be defined so as to utilise a network of ontology modules.

Data distribution builds on this and refers to the ability of a network of ontology modules to ensure that the data which the ontology structures describe can be accessed and reasoned upon as if there was a single monolithic ontology.

The goal of ontology modularisation is to obtain one or more modules from a monolithic ontology that fit with a particular application or a particular scenario [4]. The motivation for performing modularisation could for instance be to support re-use of modules for software development, or to ease the task of governance. There is no universal approach that works for all applications, so the guidelines will suggest different aspects and alternative techniques that should be considered rather than absolute solutions. We employ an ontology modularisation framework suggested by d'Aquin [4] in order to frame the guidelines in chapter 4 and further elaborate on how the guidelines can be

applied in Chapter 5 where we use concrete examples from the ontology developments in the BEST project. The framework by d'Aquin is illustrated in Figure 1.

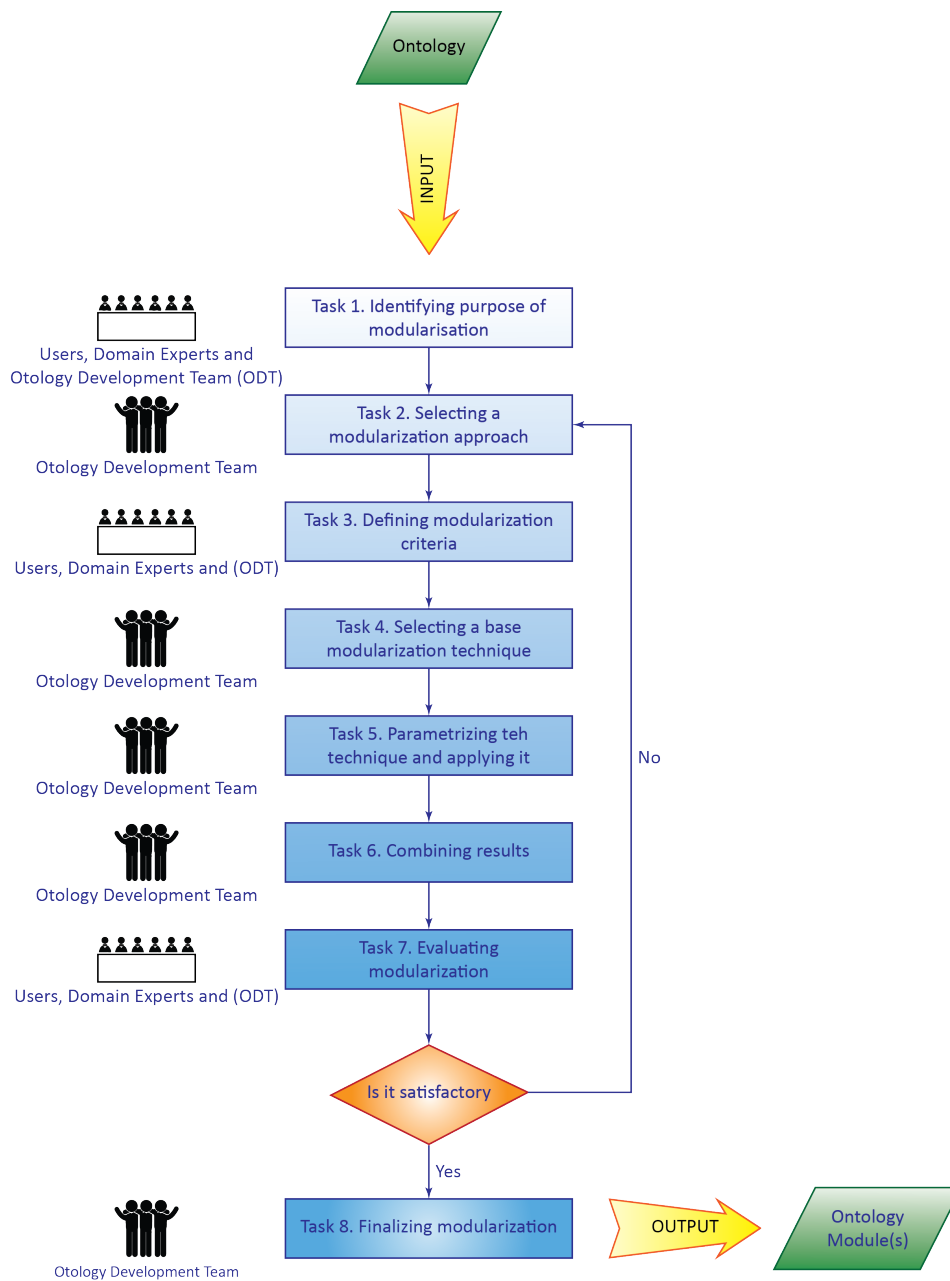


Figure 1. Framework for ontology modularisation

4 Guidelines for modularisation

This chapter describes the theoretical framework for ontology modularisation, as illustrated in Figure 1 in the previous chapter. The framework consists of eight consecutive steps, and in each of these we provide some guidelines and highlight key considerations that should be made in each step.

4.1 Identifying the purpose of modularisation

The modularisation of an ontology highly depends on the application requirements and the context in which the ontology is developed and maintained. It is therefore important that the purpose of modularisation is defined before the actual modularisation is performed. For example, if the purpose of modularisation is to improve reasoning run-time by distributing the reasoning across several modules or by simply omitting parts of a large ontology, then perhaps the size of resulting modules is more important than selecting modules based on a topic-specific scope. On the other hand, if the maintenance of modules is performed by a distributed team of experts where different expert groups are responsible for their knowledge domain, extracting modules based on topicality is more important than restricting on module size.

4.2 Selecting a modularisation approach

According to d'Aquin there are basically two main approaches to ontology modularisation:

1. Ontology Partitioning, which should be considered when an entire ontology should be partitioned into a set of modules that together fulfil the whole ontology.
2. Ontology Module Extraction, which is more appropriate whenever specific parts of an ontology are to be extracted, e.g. for the purpose of customisation, maintenance or reuse.

Of course, a combination of these two approaches might also be feasible. For example, assume that a module is extracted based on a topic-specific scope (e.g. meteorology) but due to the large size of the extracted module, the maintenance of it needs to be distributed across a team of experts. In such a scenario, it could make sense to perform the Ontology Module Extraction first in order to extract only axioms defining meteorological entities, and then in a second iteration either re-run this technique or performing partitioning to establish modules more appropriately sized for each expert within the team.

4.3 Defining modularisation criteria

Modularisation criteria refer to the required or desired outcome of the modularisation approach. The criteria have a strong relationship with the purpose of the modularisation (step 1), and requirements expressed in the environment where the modules will be applied. If possible, the criteria should be expressed in a way that allows for an evaluation with quantitative measurements determining if the criteria are satisfied. Usually there are trade-offs to be made, such as between maintainability and efficiency.

In D’Aquin et al [8], the authors separate logical modularisation criteria from structural modularisation criteria.

Logical modularisation criteria consider the logical properties of ontologies, i.e. focusing on the inferences (entailments) that can be deduced from ontologies and relates to the “self-containment” of modules. Grau et al. [9] states that when a logical module is extracted from its original context, no consequences in the signature of the module are lost and no new consequences should be obtained. From this two criteria emerges: *Local Correctness*, which means that any sentence provable in the module should also be provable in the source ontology; and *Local Completeness*, which means that every sentence in the signature (i.e. the vocabulary of the ontology comprised of classes, properties or individuals) of the module, that is provable in the source ontology, should also be provable in the module.

Structural modularisation criteria are criteria that can be computed from the structure of the modularised ontology. Schlicht and Stuckenschmidt [10] suggest that the following structural modularisation criteria should be considered:

- Size, which is concerned with the size of the modules created. The naïve solution to what is an appropriate module size is that the module should be as small as possible while still guaranteeing that the meaning of the terms used in the module is captured [5], [11].
- Redundancy, which refers to duplication of axioms across ontology modules interacting in a distributed network of modules. Although some duplication can increase efficiency, the maintainability of the modules and the network they operate in decreases.
- Connectedness is concerned with the independence of a set of modules. A module can be represented as a graph structure where axioms are nodes and edges connect every two axioms that share a symbol (classes and property names). The connectedness of a module is then evaluated by the number of edges it shares with the other modules.

4.4 Selecting a base modularisation technique

Having been subject to a lot of research the last decade, a number of techniques and tools for ontology modularisation has been suggested, both for ontology partitioning and for ontology module extraction. As described in chapter 4.2 it could be relevant to include both these two approaches, and in such a case the issue is which approach to apply first. It is outside of scope to perform a comprehensive survey of the different state of the art techniques in this report and instead we refer to [8] where two partitioning tools and two module extraction tools were tested. We also refer to chapter 5 where the techniques used for modularising ATM ontologies are described.

4.5 Parametrising the technique and applying it

The parameters used when performing the modularisation techniques depend on whether a partitioning or a module extraction approach is applied. When performing partitioning the parameters are typically set according to wanted structural characteristics of the resulting modules, such as minimal and/or maximal size of the modules. When performing ontology module extraction, a seed signature defining the area of interest and the module scope must be formulated. A seed signature

represents a sub-vocabulary of the original ontology and can consist of one or more classes, one or more properties, etc. from the original ontology.

4.6 Combining results

d'Aquin [4] recommends an iterative approach where modules are obtained by refining and combining the results in each iteration with different parameters, techniques and approaches. Depending on the evaluation results in each iteration, there are different refinements that can be performed to ensure that modules obtained in new iterations can be integrated with each other and modules obtained in previous iterations. If, for example, the obtained modules turn out too small and are complementary, they should be merged by a union operation. If the modules turn out too large and contain complementary axioms their common parts should be identified in an intersection operation. If two or more modules have overlapping axioms, a difference operation should be performed to avoid the overlap.

4.7 Evaluating the modularisation

The evaluation of the modularisation determines if the modules are satisfactory or if additional modularisation iterations are required. The modules are evaluated by checking if the criteria defined according to chapter 4.3 are satisfied and/or testing against the purpose of the modularisation defined in chapter 4.1. The evaluation could be performed using quantitative and qualitative methods.

Quantitative measures could include running a suite of reasoning operations in order to see if the inferences have the same quality and how interaction with the modules compare timewise against running the same operations on the original ontology. Other metrics could evaluate structural criteria such as size and number of modules created. With respect to size there is no universal requirement or best practices on what constitutes an optimal size, this depends on the use case. However, in [12] they used 250 class descriptions as threshold for an appropriate module size and based this number on experiences from modularisation in traditional software engineering where the optimal size of a software module allegedly is about 200-300 lines of code. In order to evaluate a set of modules the standard deviation from such a number can represent a quantitative figure. Ontology matching tools can be employed in order to discover redundancies and duplicate module parts. These tools can help identify equivalent or otherwise related (e.g. specialised or generalised) classes and properties and ensure that dependencies between modules are treated properly.

Qualitative evaluation could include a panel of ontology engineers and domain experts evaluating the maintainability and usability of the resulting modules.

4.8 Finalising the modularisation

Once the evaluation of the ontology modules is completed and the results are satisfactory, the modules are ready for deployment. Some preparatory steps for the deployment usually include implementing appropriate identifiers for each module, re-establishing links and importing declarations

and publishing the ontology modules so that they are available for the applications that will utilise them.

5 Example of using the modularisation guidelines

This chapter describes how the ontology module development in BEST was performed in relation to each step of the framework described in the previous chapter.

The goal of the modularisation process may be simply to extract one or more independent modules from a larger monolithic ontology and leave it with that. This could be the case if an application only needs to use a subset of a larger ontology for its purpose. But it may also be the case that the goal is to decompose such a monolithic ontology into a network of ontology modules. In the latter case, it is important to re-establish the relationships or dependencies between the modules, by forming an ontology network. Furthermore, it could also be relevant to extend the produced modules by re-using other modules developed elsewhere (for example a GML (Geography Markup Language) ontology). When doing this, the modules have to import the ontologies they depend on or use as extensions. An example of this is shown in Figure 2 where the METAR module imports the W3C Time ontology for describing temporal aspects.

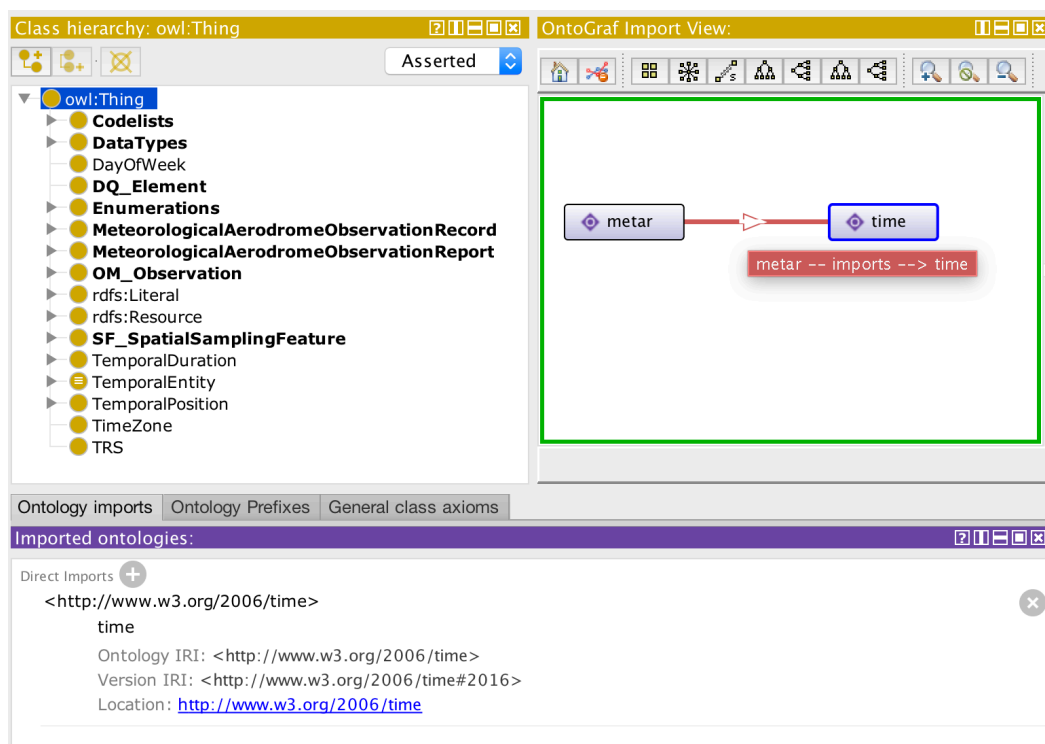


Figure 2. Illustration from the Protégé ontology editor showing how the METAR module imports the W3C Time ontology

5.1 Identifying the purpose of modularisation

The modularisation in BEST is performed based on two purposes:

1. From an application level perspective. In work package 2 techniques for ontology-based data description and discovery (task 2.1) and data distribution and consistency management (task 2.2) are being developed. These techniques, which employ reasoning in their operation, use ontologies in order to filter relevant information and ensure optimal data distribution and consistency. As described earlier, the use of modules rather than monolithic ontologies brings advantages with respect to performance and maintainability. In D2.1 [13], experiments were performed to measure the performance of semantic reasoning when working with the whole AIRM ontology compared to working with modules of the same ontology. These experiments showed that the time required for running the reasoner on modules rather than on the full ontology can be reduced by orders of magnitude.

Other use cases are to extend existing ontologies (or ontology modules) by importing modules, either with more elaborate semantics or modules that act an intermediate ontology level towards other ontologies or ontology networks. One example of the first use case is if you have lightweight ontologies, such as the ontologies in BEST that have been automatically transformed from the UML models, the imported ontology modules can specify concept definitions that can facilitate more powerful reasoning. In the second case, imported ontology modules can act as a bridge towards upper-level ontologies, which again can establish a relation towards ontologies representing other domains.

2. From a governance perspective. The BEST project investigates benefits of using semantic technologies in ATM. This includes recommendations on how these technologies, including ontologies, should be governed. Governance of ontologies relies on having a well organised distribution of work spanning several expert areas, especially for large reference models such as the AIRM. By following principles for ontology modularisation as a means for facilitating such work distribution in large monolithic reference models, whether they are represented as OWL ontologies or UML models, this could help make the task of model governance easier. Taking an example from the AIRM, which represents a good basis for a topic-based modularisation approach, each subject field could be represented as a separate ontology module.

5.2 Selecting a modularisation approach

From the identification of purpose in the previous step, it makes sense to establish modules based on extracting relevant topic-specific modules rather than partitioning the whole ontology into partitions based on structural criteria. This would be a natural approach both from an application-level perspective and from a governance perspective.

The ontology development in work package 1 transformed 3 different information models in UML to OWL representation as described in chapter 2.2. The AIRM was transformed automatically from UML to OWL using XSLT, and then, from this monolithic OWL ontology, modules were extracted automatically based on seed signatures expressing the wanted topic-specific scope. A monolithic OWL ontology of AIXM was generated using XSLT transformation, but the modules were crafted manually after that. The IWXXM ontology modules were all developed completely manually. We chose to perform the modularisation differently for these 3 information models for different reasons. For one,

the structural characteristics of the original UML models influenced the choice of modularisation strategy. The structure of AIRM was suited for a module extraction approach since the AIRM Subject Fields already were organised into different themes that naturally could be represented as modules. This was also partly true for the AIXM, even if AIXM is developed as a UML model designed to facilitate transformation to XML schemas. IWXXM is, as AIXM, targeted for XML schema development and does not lend itself well to automatic module extraction. The size of the original UML models did also matter. Manually developing an OWL representation of a large model like AIRM would require too much effort, and it was considered more sensible to invest resources in developing a re-usable transformation script. In the other end, the IWXXM is a relatively small model and a manual module development required fewer resources than either creating a new transformation script or amending the AIRM or AIXM transformation scripts.

5.3 Defining modularisation criteria

There are different sets of criteria depending on the context in which the modularisation will be applied. The BEST project develops semantic applications and there are criteria emerging from these developments. In addition, the project will formulate a set of guidelines related to governance of semantic technologies and in this context, another set of criteria apply.

5.3.1 Criteria from an application context

In this context, the modules will be used by applications of semantic technology, such as the Semantic Container. Ideally, logical criteria (local correctness and local completeness) should apply also here, but some flexibility should be allowed when it comes to module size and redundancy since performance typically is of high importance for such applications. It should be allowed to establish modules with finer granularity. A Semantic Container might only need a part of a “Meteorology” module for providing a facet ontology that describes the data items relevant for that particular container. Hence, the techniques applied should allow for a flexible modularisation approach that enables the extraction of for example only “Airspace Conditions” in order to describe data about meteorological conditions for a particular airspace. Ensuring the modules used for semantic applications are compatible with “standardised” modules is primarily the responsibility of the SWIM governance bodies. However, the modules should be created in a methodologically rigorous manner, for example using the locality-based modularisation techniques used in BEST (see chapter 5.4) to avoid that the automated modularisation process compromises compatibility.

5.3.2 Criteria from a governance context

In a governance context, the focus is on maintainability rather than performance. One scenario is that for example the AIRM is decomposed into a set of topic-specifically scoped modules for distributed maintenance. In such a scenario, it is important that the modularisation process preserves the topic-specific scopes or subject fields (for example “Aircraft” from the AIRM ontology) as they are already organised in the original model. Delimiting the size of the obtained modules is an important criterion, both the maximum and minimum size. Too large modules are difficult to navigate and maintain, and if the modules are too small this may reduce the ability to maintain a good overview and consistency. It

is difficult to establish an absolute figure here, so we follow the recommendation from d'Aquin of 250 class definitions. From a governance perspective redundancy is a problem because this means that the same module parts would have to be maintained and synchronised several places.

5.4 Selecting a base modularisation technique

Our base modularisation technique was module extraction based on locality-based modules. Locality-based modules combines strong logical guarantees and the computation involved for producing them is quite efficient. In short, a locality-based module M is a subset of the axioms in an ontology O , and is extracted from O for a set of terms S . This set S is called the seed signature of M (formal descriptions are available in [5]).

5.5 Parametrising the technique and applying it

Following from the locality-based module extraction technique described above, we created a simple Java program we call the **Module Extractor** that enables the extraction of modules based on a seed signature. This tool is based on an OWL API [14] implementation of locality-based module extraction developed by the University in Manchester [15]. This java program takes three input parameters:

1. The original ontology to extract modules from
2. The name of the ontology module to be created
3. A seed signature consisting of classes or properties the module should represent

The below illustration shows how this java program is executed and the results returned. The green text is user input.

```
Enter path to (monolithic) ontology file: ./files/modules/v2Sept2017/airm_mono.owl
Enter name of ontology module to create: Meteorology_Module
Enter signature to create module from: #_Meteorology_

Ontology created!
Number of classes: 74
Number of object properties: 69
Number of data properties: 15
Number of individuals: 97
Number of axioms: 746
```

Figure 3. An example of how to use the modularisation tool developed in BEST to extract a Meteorology module from the AIRM ontology³

In this example, we wanted to create a module consisting of axioms describing the meteorological structures in the AIRM. Therefore, we used ‘#_Meteorology_’ as seed signature since this is how the Meteorology class is represented in the AIRM ontology. One curiosity is that when testing this functionality in the OWL API, we discovered that the resulting modules only contained classes and

³ The modularisation tool is available from GitHub at: <https://github.com/sju-best-project/ontology-modules>

individuals, all properties were omitted in the extraction. Therefore, we extended the OWL API implementation with functionality that also extracted object properties and data properties for a given ontology module from the AIRM ontology.

5.6 Combining results

5.6.1 A second iteration of modularisation

From the first iteration of modularisation in BEST, one of the modules, the AIRM-BaseInfrastructure was larger than our threshold of 250 classes (see the listing in Table 3). In order to reach the desired module size, another iteration of modularisation was required. Looking at the structure of the AIRM-BaseInfrastructure ontology, we decided to decompose this ontology into four smaller modules: AerodromeInfrastructure, NavigationInfrastructure, SurveillanceInfrastructure and Obstacle. See the original package structure of the BaseInfrastructure subject field in AIRM in Figure 4.

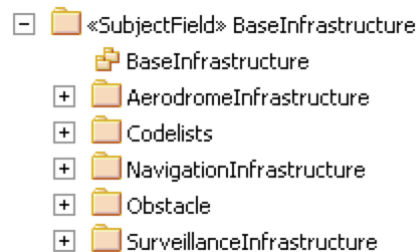


Figure 4. How the subject field BaseInfrastructure is represented in AIRM

We used the same technique as in the previous modularisation, the locality-based modularisation. The result of this operation is shown in Table 4.

Table 4. Modules created from modularising AIRM-BaseInfrastructure

| Ontology Module | Classes | Object properties | Data properties | Individuals |
|---------------------------------|---------|-------------------|-----------------|-------------|
| AIRM-AerodromeInfrastructure | 117 | 345 | 69 | 0 |
| AIRM-NavigationInfrastructure | 34 | 70 | 39 | 0 |
| AIRM-SurveillanceInfrastructure | 34 | 21 | 17 | 0 |
| AIRM-Obstacle | 12 | 27 | 8 | 0 |

5.6.2 Import dependencies

The iteration described in the previous section turned out to be problematic as the relations to the code list values were then lost. In the BEST ontologies, code lists are represented as classes and their values are represented as individuals (see [6] for additional details). By extracting the four above modules using the locality-based module extraction technique, this de-referenced the relations to the code lists (and their values) from the `_Codelists_BaseInfrastructure` class hierarchy as it was represented in the original AIRM-BaseInfrastructure module (see Figure 5). Hence, we also needed to extract the `_Codelists_BaseInfrastructure_` as a separate module in order to maintain the relationship

between a class and a code list via an object property. This module should then be imported by the other four modules to ensure that the relationship between the classes and the codelists and their values is maintained.

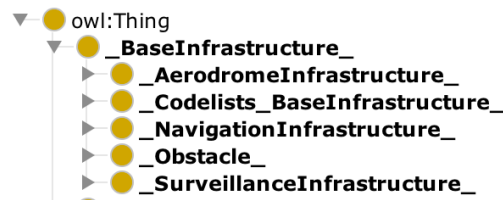


Figure 5. The BaseInfrastructure module

By having a separate module for codelists, this would solve the issue of missing relationship between classes and codelists and their values.

| Ontology Module | Classes | Object properties | Data properties | Individuals |
|----------------------------------|---------|-------------------|-----------------|-------------|
| AIRM-AerodromeInfrastructure | 117 | 345 | 69 | 0 |
| AIRM-NavigationInfrastructure | 34 | 70 | 39 | 0 |
| AIRM-SurveillanceInfrastructure | 34 | 21 | 17 | 0 |
| AIRM-Obstacle | 12 | 27 | 8 | 0 |
| AIRM-BaseInfrastructureCodelists | 100 | 0 | 0 | 1574 |

An ontology network is effectively formed by establishing dependencies through module import statements. If one module has entities that depend on entities from other modules, these other modules must be imported.

To identify module dependencies and support the establishment of an ontology module network, we developed a tool called Ontology Module Network Report. This tool analyses an ontology module and reports all missing dependencies. These missing dependencies are identified on the basis of outlier classes. These outlier classes occur when a range class referred to in the object properties belong to another module. For example, Figure 6 shows a Protégé excerpt from the AIRM-AerodromeInfrastructure module. Here, there is an object property Aerodrome-servedCity that relates the Aerodrome class (the domain of the property) with the City class (the range of the property).

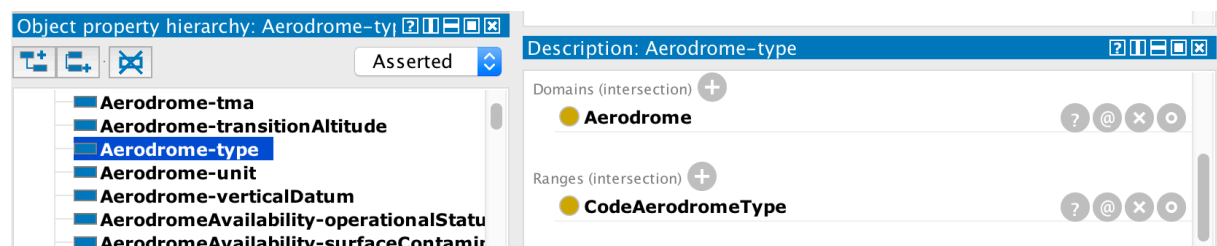


Figure 6. ObjectProperty relating Aerodrome to City

Looking further at the list of entities of the AIRM-AerodromeInfrastructure ontology module in Figure 7, we see that the class City is listed as one of the classes outside the AerodromeInfrastructure subclass hierarchy. The task is then to identify in which module the City class is represented and then import

this module in the AIRM-AerodromeInfrastructure module. In this case, City belongs to the AIRM-Common ontology module, so AerodromeInfrastructure needs to import the AIRM-Common ontology. The same operation is required for the other “outlier” classes shown in Figure 7 in order to establish a consistent ontology module network.



Figure 7. List of external classes in an ontology module

5.7 Evaluating the modularisation

This step involves evaluating the obtained ontology modules with respect to the purpose of modularisation and the modularisation criteria formulated in chapters 5.1 and 5.3 respectively. From these two chapters, we learned that the maximum size, a topic-based scope and to avoid redundancy were important criteria. With regards to ensuring that there is no information loss as a result of the modularisation the local completeness (see chapter 4.3) criterion is important. Logic-based modularisation techniques, such as the locality-based module extraction performed in BEST, usually result in modules where completeness is ensured, but partitioning-based techniques do not guarantee completeness [16].

5.7.1 Size

The modules should ideally be less than 250 class descriptions for the sake of usability, maintainability and performance among others. By further modularising the AIRM-BaseInfrastructure ontology module, all modules contained less than 250 class descriptions as shown in Table 5.

5.7.2 Topic-based scope

The modules should have a topic-based scope, both to support the (faceted) ontologies used for semantically describing the content of a semantic container and to support a distributed governance scheme of modules. Since the original AIRM-BaseInfrastructure module was considered too large according to the size criterion (see next chapter), a second iteration of modularisation was performed

resulting in 5 new modules. The resulting modules still maintain the requirement of having a topic-based scope of the resulting modules.

Table 5 presents the final ontology modules in BEST. White colour indicates modules obtained from the first iteration of modularisation, red colour indicates the module that was further modularised in a second iteration and green indicates the new modules obtained in the second iteration.

Table 5. Final ontology modules in BEST

| Ontology Module | Classes | Object properties | Data properties | Individuals |
|-----------------------------|---------|-------------------|-----------------|-------------|
| Aircraft | 71 | 84 | 32 | 182 |
| BaseInfrastructure | 298 | 463 | 133 | 1574 |
| 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 |

5.7.3 Redundancy

As described earlier redundancy in the sense of duplicate descriptions in the obtained modules is problematic with respect to a consistent governance scheme. If such redundancy is to be resolved, we need to identify in which modules the duplicate entries reside. An ontology matching system can identify duplicate ontology entities as long as the entities are identical or similar⁴. In order to check for redundancy, we have developed a simple tool called **Redundancy Report Generator** that performs a pairwise ontology matching operation of a set of modules in order to identify duplicate classes. A screenshot illustrating how the Redundancy Report Generator operates is shown in Figure 8. The Redundancy Report Generator re-uses functionality provided by the Alignment API [17].

⁴ Typically, ontology matching systems focus on identifying equivalent class entities. Less emphasis is put on identifying equivalent properties and subsumption (that a class or property is a sub- or superclass of another)

```

Enter path to folder holding the ontology modules: ./test-files/modules/output-modules
Enter path to folder where the alignments holding duplicate classes will be stored: ./test-files/modules/output-alignments
Running Redundancy Report Generator...

stakeholders and meteorology contain 3 duplicates, and the duplicates are:
<http://www.project-best.eu/owl/airm-mod/stakeholders.owl#Aerodrome> - <http://www.project-best.eu/owl/airm-mod/meteorology.owl#Aerodrome>
<http://www.project-best.eu/owl/airm-mod/stakeholders.owl#Airspace> - <http://www.project-best.eu/owl/airm-mod/meteorology.owl#Airspace>
<http://www.project-best.eu/owl/airm-mod/stakeholders.owl#ValDistanceType> - <http://www.project-best.eu/owl/airm-mod/meteorology.owl#ValDistanceType>

stakeholders and navigationinfrastructure contain 11 duplicates, and the duplicates are:
<http://www.project-best.eu/owl/airm-mod/stakeholders.owl#CodeAuthorityRoleType> - <http://www.project-best.eu/owl/airm-mod/navigationinfrastructure.owl#CodeAuthorityRoleType>
<http://www.project-best.eu/owl/airm-mod/stakeholders.owl#ValFrequencyType> - <http://www.project-best.eu/owl/airm-mod/navigationinfrastructure.owl#ValFrequencyType>
<http://www.project-best.eu/owl/airm-mod/stakeholders.owl#RadioNavigationService> - <http://www.project-best.eu/owl/airm-mod/navigationinfrastructure.owl#RadioNavigationService>
<http://www.project-best.eu/owl/airm-mod/stakeholders.owl#Organisation> - <http://www.project-best.eu/owl/airm-mod/navigationinfrastructure.owl#Organisation>
<http://www.project-best.eu/owl/airm-mod/stakeholders.owl#Aerodrome> - <http://www.project-best.eu/owl/airm-mod/navigationinfrastructure.owl#Aerodrome>
<http://www.project-best.eu/owl/airm-mod/stakeholders.owl#CodeRadioEmissionType> - <http://www.project-best.eu/owl/airm-mod/navigationinfrastructure.owl#CodeRadioEmissionType>
<http://www.project-best.eu/owl/airm-mod/stakeholders.owl#SpecialNavigationSystem> - <http://www.project-best.eu/owl/airm-mod/navigationinfrastructure.owl#SpecialNavigationSystem>
<http://www.project-best.eu/owl/airm-mod/stakeholders.owl#ValDistanceType> - <http://www.project-best.eu/owl/airm-mod/navigationinfrastructure.owl#ValDistanceType>
<http://www.project-best.eu/owl/airm-mod/stakeholders.owl#RadioFrequencyArea> - <http://www.project-best.eu/owl/airm-mod/navigationinfrastructure.owl#RadioFrequencyArea>
<http://www.project-best.eu/owl/airm-mod/stakeholders.owl#DirectionFinder> - <http://www.project-best.eu/owl/airm-mod/navigationinfrastructure.owl#DirectionFinder>
<http://www.project-best.eu/owl/airm-mod/stakeholders.owl#InformationService> - <http://www.project-best.eu/owl/airm-mod/navigationinfrastructure.owl#InformationService>

```

Figure 8. Redundancy Report Generator in action

5.8 Finalising the modularisation

The final ontology module network representing our defined scope of the AIRM is illustrated in Figure 10. As can be seen there are quite many dependencies between the modules, and without further analysis we see that some of the modules have more dependency towards them than others. Since we only included a subset of the AIRM in this work, there are also some dependencies to classes outside of our defined network. The tool **Ontology Module Network Report Generator** reports these dependencies so that appropriate measures can be taken if the network is to be extended at a later stage. See Figure 9 for an example where a report of outlier classes, required imports and class dependencies outside our defined AIRM scope is provided. Once the list of dependencies between the modules is identified, the **Module Network Dependency Manager** automatically resolves these dependencies by declaring the appropriate import statements in the modules.

```

***Outlier classes (22)***

Flight
CodeEquipmentOperationalType
Unit
AircraftState
AircraftStand
AircraftOperator
AerospaceManufacturer
TaxiwayElement
ValMassType
ValAltitudeType
Trajectory
MatterEmission
SpeedRangeType
TrajectoryPoint
CrewMember
RunwayDirection
CodeFlightPhaseType
CodeValueInterpretationType
ValSpeedType
ValDistanceType
ValPressureType
Organisation

***This ontology module should import***

http://project-best.eu/owl/airm-mod/flight
http://project-best.eu/owl/airm-mod/stakeholders
http://project-best.eu/owl/airm-mod/common
http://project-best.eu/owl/airm-mod/aerodromeinfrastructure
http://project-best.eu/owl/airm-mod/datatypes

***Classes belonging to ontology modules outside our scope***

MatterEmission

```

Figure 9. An example report from the Ontology Module Network Report Generator tool

With regards to identifiers each module has its unique URI (IRI) and all entities belonging to this particular module inherit this URI.

The URI scheme applied for the modules is:

[http://project-best.eu/owl/airm-mod/\[module name\]](http://project-best.eu/owl/airm-mod/[module name])

and each entity is identified by:

[http://project-best.eu/owl/airm-mod/\[module name\]#\[entity name\]](http://project-best.eu/owl/airm-mod/[module name]#[entity name])

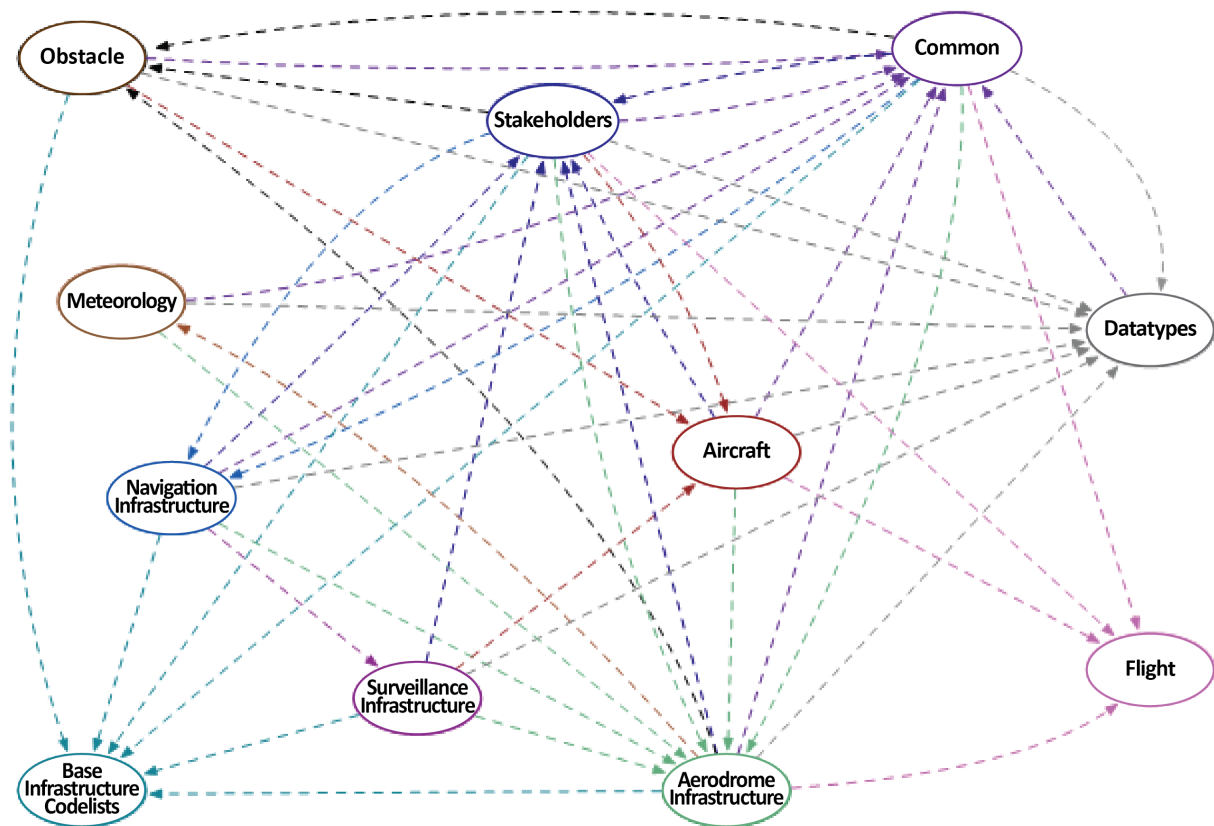


Figure 10. The ontology module network of the AIRM

Annex 2 includes a description of each of the modules extracted from the AIRM monolithic ontology.

6 Conclusions

Operating with ontology modules instead of large monolithic ontologies has advantages for application performance, usability, maintainability, to name but a few. This document has presented a set of guidelines on how such ontology modules can be extracted from monolithic ontologies, using ATM as the scenario. In order to frame these guidelines, we have applied a theoretical framework for modularisation that prescribes an 8-step process. Each of these steps emphasises different aspects that should be considered when modularising an ontology. Step by step we have described experiences from the ontology development in BEST, with an emphasis on how the AIRM ontology was decomposed into a set of ontology modules. In each step, we have described the considerations that were made, any challenges we came across and how these were addressed, as well as some tools that were developed in order to support the development, refinement and evaluation of the modules.

The ontology modularisation approach we adopted was ontology module extraction using a technique called locality-based module extraction. The benefit of this approach over an alternative approach, ontology partitioning, is that the modules can be extracted on the basis of a topic-based query (a.k.a. seed signature). This approach also ensures local completeness of the resulting modules, contrary to partitioning-based approaches. Such an approach was appropriate for the purposes of modularisation in BEST, both from an application development and governance perspective, where it was important that the modules were organised according to topic (e.g. meteorology) rather than purely structural criteria such as size or number of hierarchical levels (as the ontology partitioning approach).

The prototype modularisation tools developed as part of this work are available from GitHub at <https://github.com/sju-best-project/ontology-modules>.

7 References

- [1] S. B. Abbes, T. Meilender, and M. D'Aquin, "Characterizing modular ontologies," in *Conference on Formal Ontologies in Information Systems-FOIS*, 2012.
- [2] Z. C. Khan and C. M. Keet, "An empirically-based framework for ontology modularisation," *Appl. Ontol.*, vol. 10, no. 3–4, pp. 171–195, 2015.
- [3] H. Stuckenschmidt and A. Schlicht, "Structure-Based Partitioning of Large Ontologies," in *Modular ontologies*, Springer, 2009, pp. 187–210.
- [4] M. D'Aquin, "Modularizing Ontologies," in *Ontology Engineering in a Networked World*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 213–233.
- [5] B. C. Grau, I. Horrocks, Y. Kazakov, and U. Sattler, "Modular Reuse of Ontologies: Theory and Practice," *J. Artif. Intell. Res.*, pp. 272–318, 2008.
- [6] A. Vennesland, B. Neumayr, C. Schuetz, and A. Savulov, "D1.1 Experimental ontology modules formalising concept definition of ATM data," 2017.
- [7] S. Wilson, R. Suzic, and S. Van der Stricht, "The SESAR ATM information reference model within the new ATM system," in *2014 Integrated Communications, Navigation and Surveillance Conference (ICNS) Conference Proceedings*, 2014.
- [8] M. D'Aquin, A. Schlicht, H. Stuckenschmidt, and M. Sabou, "Criteria and evaluation for ontology modularization techniques," in *Modular ontologies*, 2009, pp. 67–89.
- [9] B. C. Grau, B. Parsia, E. Sirin, and A. Kalyanpur, "Modularity and Web Ontologies," in *KR*, 2006, pp. 198–209.
- [10] 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.
- [11] J. Pathak, T. M. Johnson, and C. G. Chute, "Survey of modular ontology techniques and their applications in the biomedical domain," *Integr. Comput. Aided. Eng.*, vol. 16, no. 3, pp. 225–242, 2009.
- [12] H. Stuckenschmidt and A. Schlicht, *Structure-based Partitioning of Large Ontologies*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.
- [13] C. Schuetz, B. Neumayr, and M. Schrefl, "D2.1 Techniques for ontology-based data description and discovery in a decentralized SWIM knowledge base," 2017.
- [14] M. Horridge and S. Bechhofer, "The OWL API: A Java API for OWL Ontologies," *Semant. Web J.*, vol. 2, no. 1, pp. 11–21, 2011.
- [15] C. Del Vescovo, R. Gonçalves, B. Parsia, and U. Sattler, "OWL @ Manchester: Modularity." .
- [16] Z. C. Khan, "Evaluation Metrics in Ontology Modules," in *Description Logics*, 2016.
- [17] J. David, J. Euzenat, F. Scharffe, and C. T. Dos Santos, "The alignment API 4.0," *Semant. Web*, vol. 2, no. 1, pp. 3–10, 2011.

Annex 1: Modularisation tools

During this work, a set of prototype tools to support the modularisation has been developed. These are available from GitHub at <https://github.com/sju-best-project/ontology-modules>

Module Extractor

Creates a (locality-based) module according to a seed signature from the AIRM ontology. This tool is based on an OWL API implementation of locality-based module extraction developed by the University in Manchester. However, when testing this functionality in the OWL API, we discovered that the resulting modules only contained classes and individuals, all properties were omitted in the extraction. Therefore, we extended the OWL API implementation with functionality that also extracted object properties and data properties for a given ontology module from the AIRM ontology. One consequence of including the object properties is that the resulting module includes outlier classes. This happens because some of the range classes referred to in the object properties belong to other modules extracted from the AIRM ontology.

Ontology Module Network Report Generator

Analyses an ontology module and reports missing dependencies. Missing dependencies are discovered by searching for an ontology module that has an outlier class in its signature. From this the Ontology Module Network Report tool suggests which ontologies that the module should import and if there are any classes for which there is no relevant module, the names of these classes are presented to the user for further manual analysis of which ontology this class belongs to and consequently which ontology should be imported.

Module Network Dependency Manager

Identifies relevant ontology modules to import, declares the import statements so that the ontology module actually imports these modules, removes outlier classes, that is, those classes that previously missed a dependent ontology module.

Redundancy Report Generator

This tool checks for duplicate classes in modules. By pairwise matching of ontology modules using string similarity matching it identifies duplicates and creates a report that lists all duplicates among the ontology modules used as input parameters.

These tools take part in the following workflow:

Prerequisite:

A monolithic AIRM ontology is created by an XSLT transformation from UML (via XMI).

1. The Module Extractor extracts a module from the AIRM ontology given a seed signature as parameter.
2. The Ontology Module Network Report Generator checks if there are any classes in the resulting module for which a dependency is not declared. This tool also suggests which ontology modules should be imported to resolve the missing dependencies.
3. The Module Network Dependency Manager acts on the analyses performed in the previous step and automatically declares the relevant import statements in the ontology module and removes the outlier classes so that there are no duplicate entries in the ontology module.
4. The Redundancy Report Generator analyses the ontology modules for duplicate classes and presents a list of (potential) duplicates. Resolving the redundancy is a manual operation.

Annex 2: Description of the ontology modules

Aircraft

Dependencies with other modules:

- <http://project-best.eu/owl/airm-mod/flight>
- <http://project-best.eu/owl/airm-mod/stakeholders>
- <http://project-best.eu/owl/airm-mod/common>
- <http://project-best.eu/owl/airm-mod/aerodromeinfrastructure>
- <http://project-best.eu/owl/airm-mod/datatypes>

<http://project-best.eu/owl/airm-mod/aircraft.owl>

Aerodrome Infrastructure

Dependencies with other modules:

- <http://project-best.eu/owl/airm-mod/flight>
- <http://project-best.eu/owl/airm-mod/stakeholders>
- <http://project-best.eu/owl/airm-mod/common>
- <http://project-best.eu/owl/airm-mod/obstacle>
- <http://project-best.eu/owl/airm-mod/meteorology>
- <http://project-best.eu/owl/airm-mod/baseinfrastructurecodelists>
- <http://project-best.eu/owl/airm-mod/datatypes>

Dependencies to classes outside of the defined scope:

- FlightRoutingElement
- TouchDownLiftOffContamination
- CodeStatusAirportType
- RunwaySectionContamination
- FlightConditionElement
- CodeInstrumentRunwayType
- CodeMilitaryOperationsType
- ApronContamination
- SignificantPoint
- TakeOffSequence
- Airspace
- RunwayContamination
- Deicing
- AircraftStandContamination

- AerodromeContamination
- LandingSequence
- TaxiwayContamination
- CodeInstrumentApproachCategoryType
- AirportSlot
- SurfaceContamination

<http://project-best.eu/owl/airm-mod/aerodromeinfrastructure.owl>

Navigation Infrastructure

Dependencies with other modules:

- <http://project-best.eu/owl/airm-mod/stakeholders>
- <http://project-best.eu/owl/airm-mod/common>
- <http://project-best.eu/owl/airm-mod/aerodromeinfrastructure>
- <http://project-best.eu/owl/airm-mod/surveillanceinfrastructure>
- <http://project-best.eu/owl/airm-mod/baseinfrastructurecodelists>
- <http://project-best.eu/owl/airm-mod/datatypes>

Dependencies to classes outside of the defined scope:

- CircleSector

<http://project-best.eu/owl/airm-mod/navigationinfrastructure.owl>

Surveillance Infrastructure

Dependencies with other modules:

- <http://project-best.eu/owl/airm-mod/stakeholders>
- <http://project-best.eu/owl/airm-mod/aerodromeinfrastructure>
- <http://project-best.eu/owl/airm-mod/aircraft.owl>
- <http://project-best.eu/owl/airm-mod/baseinfrastructurecodelists>
- <http://project-best.eu/owl/airm-mod/datatypes>

<http://project-best.eu/owl/airm-mod/surveillanceinfrastructure.owl>

Base Infrastructure Codelists

Dependencies with other modules:

- None

Founding Members



<http://project-best.eu/owl/airm-mod/baseinfrastructurecodelists.owl>

Datatypes

Dependencies with other modules:

- <http://project-best.eu/owl/airm-mod/common>

<http://project-best.eu/owl/airm-mod/datatypes.owl>

Dependencies to classes outside of the defined scope:

- CodeSpecialHeightValueType

Meteorology

Dependencies with other modules:

- <http://project-best.eu/owl/airm-mod/common>
- <http://project-best.eu/owl/airm-mod/aerodromeinfrastructure>
- <http://project-best.eu/owl/airm-mod/datatypes>

Dependencies to classes outside of the defined scope:

- Airspace
- AirspaceVolume
- Route

<http://project-best.eu/owl/airm-mod/meteorology.owl>

Obstacle

Dependencies with other modules:

- <http://project-best.eu/owl/airm-mod/common>
- <http://project-best.eu/owl/airm-mod/aircraft.owl>
- <http://project-best.eu/owl/airm-mod/baseinfrastructurecodelists>
- <http://project-best.eu/owl/airm-mod/datatypes>

Dependencies to classes outside of the defined scope:

- HoldingProcedure
- UnplannedHolding
- SignificantPoint

<http://project-best.eu/owl/airm-mod/obstacle.owl>

Common

Dependencies with other modules:

- <http://project-best.eu/owl/airm-mod/flight>
- <http://project-best.eu/owl/airm-mod/stakeholders>
- <http://project-best.eu/owl/airm-mod/navigationinfrastructure>
- <http://project-best.eu/owl/airm-mod/aerodromeinfrastructure>
- <http://project-best.eu/owl/airm-mod/obstacle>
- <http://project-best.eu/owl/airm-mod/baseinfrastructurecodelists>
- <http://project-best.eu/owl/airm-mod/datatypes>






<http://project-best.eu/owl/airm-mod/common.owl>

Stakeholders

Dependencies with other modules:

- <http://project-best.eu/owl/airm-mod/flight>
- <http://project-best.eu/owl/airm-mod/common>
- <http://project-best.eu/owl/airm-mod/navigationinfrastructure>
- <http://project-best.eu/owl/airm-mod/aerodromeinfrastructure>
- <http://project-best.eu/owl/airm-mod/obstacle>
- <http://project-best.eu/owl/airm-mod/aircraft.owl>
- <http://project-best.eu/owl/airm-mod/baseinfrastructurecodelists>
- <http://project-best.eu/owl/airm-mod/datatypes>

The BEST consortium:

| | |
|--|---|
| SINTEF |  |
| Frequentis AG |  |
| Johannes Kepler Universität (JKU) Linz |  |
| SLOT Consulting |  |
| EUROCONTROL |  |