CREATING AN ONTOLOGY FOR AIRCRAFT DESIGN

An Experience Report About Development Process and the Resulting Ontology

Markus Ast
ast@in.tum.de

Martin Glas
Bauhaus Luftfahrt e.V.
martin.glas@bauhaus-luftfahrt.net

Tobias Roehm
Technische Universität München
lastname@in.tum.de

Abstract

Experts from various aircraft sub-domains have to collaborate in order to design an aircraft system. Each sub-domain thereby focuses on a particular aspect of the system and uses special software tools to model those aspects and run various calculations. As those models partially overlap, inconsistencies can occur that might lead to delays in the design process or malfunctions of the aircraft system. Hence, different models need to be checked for consistency. Furthermore, exchangeability of model data between tools enables the reuse information from an already existing model.
To ensure that model data is compared and transformed appropriately to its meaning, a common, semantic representation of aircraft knowledge can be used to establish a reference. As ontologies are a formal, human and machine readable knowledge representation mechanism that explicitly captures semantics, they can serve as such a common semantic reference.
In this paper we describe the development of the AIRCRAFT ontology following the NEON process model. In particular, we describe our experiences from applying the NEON methodology and the resulting AIRCRAFT ontology. The AIRCRAFT ontology is an OWL ontology that covers system decomposition and component parameters of a single aisle civil transport aircraft. It can be used as a common semantic reference during model comparison and transformation. Furthermore, the AIRCRAFT ontology can be extended using the NEON process model to serve other purposes that require a semantic representation of aircraft knowledge.

1 INTRODUCTION

During the design of a new aircraft, designers and analysts create and refine several aircraft models using different software tools. Each model covers parts of the whole aircraft and usually focuses on one aspect of the aircraft. Especially during conceptual aircraft design the degree of diversity and content overlap are high compared to later design phases. Inconsistencies between those models may arise during the design process as those models are often developed in parallel, describe the same aircraft, and partially overlap. Some of those inconsistencies, if detected during the design process, might lead to delays in the design process when components have to be redesigned in order to react to a detected inconsistency. If inconsistencies between models are not detected during the design process, some of them bear the risk of aircraft malfunctions, e.g. if two components do not fit together as expected. Usually, the earlier an inconsistency is detected, the less expensive it is to react on it. Hence, inconsistency checking is desirable already at an early design phase.

Model integration deals with the problem to integrate different models into an overall, consistent model. It not only keeps overlapping content consistent but also enables designers from different disciplines to exchange content easily between models in order to complement one another. Because of model heterogeneity, a big challenge of model integration is the identification of model elements which represent the same thing across different representations in different models. One approach to tackle this challenge is to link model elements to the appropriate domain concepts in a common reference ontology which represents these concepts by formal semantics. This methodology has been demonstrated in the integration of different generic modeling languages by Kappel et al. [1], and between different business process modeling languages by Roser [2]. A framework for ontology-based integration of aircraft design models was shown by Glas [29] which also uses a reference ontology. If this framework, however, cannot use a publicly available reference ontology, its users have to derive it from the integrated design models. This derivation process not only diminishes the efficiency of the overall integration process but can also lead to a reference ontology which is tied to a specific set of models and is difficult to reuse in other integration scenarios. Even now – to the best of our knowledge – there is no ontology publicly available which represents concepts for aircraft design and that can be used as common representation.
Therefore, we decided to develop such an ontology representing aircraft design knowledge and make it

publicly available. We used the NEON process model for ontology development in order to create this ontology in an efficient way. As initial scope we chose the description of the static aircraft system structure consisting of a system decomposition and component parameters. Obviously, there are many other knowledge areas that are interesting for aircraft design such as system behavior during missions. Hence, we see our ontology as a foundation for further extensions.

The contributions of this paper are the following: First, we describe our experiences and lessons learned while applying the NEON process model to develop the AIRCRAFT ontology. Second, we report the design and results of an ontology evaluation method using expert interviews, instantiating and detailing guidelines for the evaluation of ontologies from the NEON methodology. Third, we describe the resulting AIRCRAFT ontology. It is an OWL ontology that is publicly available and covers system decomposition and component parameters of a single aisle civil transport aircraft. With both parts we hope to enable colleagues to use the AIRCRAFT ontology and extend it to fit their needs.

This paper is structured as follows. In Section 2, we give some background knowledge about ontologies. In Section 3 we briefly describe several ontology development methodologies and our rationale for choosing the NEON methodology. In Section 4 we describe how we put the NEON methodology in action. We also describe how we evaluate the resulting ontology using interviews with domain experts. In Section 5 we describe the resulting AIRCRAFT ONTOLOGY. Finally we discuss interesting observations and lessons learned in Section 6 and conclude in Section 8.

2 ONTOLOGY BASICS

Ontology originates from Greek philosophy, namely the study of being and existence, dealing with the questions what kinds of things exist and how they relate to one another. This concept has been adapted for use in computer science. Studer et al. [3] define an ontology as "a formal explicit specification of a shared conceptualization of a domain of interest", emphasizing formality which is needed for automated processing, a consensus about the contents, and the focus on a specific domain whereas the view on that domain is influenced by a certain interest for the ontology in mind.

2.1 Types of Ontologies

Depending on their purpose, ontologies can be categorized into the following types [4]:

Top-level ontologies cover general and abstract concepts, e.g notions of time and space that can be reused and refined in other ontologies.

Domain or task ontologies cover knowledge about a specific domain (e.g. aircraft) or a specific task (e.g. cooking); since this distinction is somewhat imprecise, both are normally referred to as domain ontology.

Application ontologies are typically developed in complement to an application and with certain usage scenarios in mind. They cover and refine specific aspects of domain ontologies for use in that specific application.

The ontology developed in the context of this paper can be categorized as domain ontology.

2.2 The Ontology Language OWL

As a language for describing ontologies, the World Wide Web Consortium W3C[1] recommends the Web Ontology Language (OWL) [5], respectively its revision OWL 2 [6]. The foundations for defining semantics between concepts in OWL are logical declarations which can be evaluated by *reasoners*. Reasoners are programs which provide services such as checking the consistency of logical declarations in an ontology and inference of new knowledge from explicitly declared knowledge.
In general, ontologies consist of concepts and roles. The concepts are organized in a hierarchical structure formed by *is-a* relations between these concepts. In OWL, these concepts are called *classes*, e.g. an `Airbus A320` is a specialized sub-class of its superclass `Aircraft`.
With the use of roles, more context can be added to classes in form of semantic relations. OWL expresses roles by properties which represent relations between two concepts. Possible sources and targets of these relations can be defined by the specification of appropriate domains and ranges of properties. OWL stipulates two kinds of *properties*: *object properties* relate two classes, whereas *data type properties* relate classes to data types. For example, an object property `hasWing` defines a relation between a class `Aircraft` and a class `Wing`. Object properties can be defined further by logical characteristics. For instance, `isPartof` can be declared *transitive* to express, that if `Cabin isPartOf Fuselage` and `Fuselage isPartOf Aircraft`, then `Cabin isPartOf Aircraft`. In this example declaring `isPartOf` *antisymmetric* can forbid that `Fuselage isPartOf Cabin`. As an example for an data type property, `numericalValue` defines a relation between a class `SingleAircraftParameter` and the `double` data type.
Finally, to fill an ontology with concrete concepts and values, classes and data types can be instantiated. The instance of a class, also known as class-member, is called *individual* in OWL. For example the individual `A320_MSN1471` is an instance of the class `Airbus A320`. The instance of a data type is a data value. For example, `0.5424` is an instance of `double`.

[1] `http://www.w3.org`

2.3 The Open World Assumption

A peculiarity of ontologies is the *open world assumption*. Essentially, it states that all knowledge, that is not explicitly or implicitly stated in an ontology, has to be regarded as unknown. In contrast, according to the closed world assumption, which is normally used with traditional data models such as database systems, missing knowledge would be regarded as non-existent. Take for example a knowledge base that consists of the single statement "Airbus is an aircraft manufacturer", and the question "Is Boeing an aircraft manufacturer?". According to the closed world assumption, the answer to the question would be "No", whereas the open world assumption would result in "Unknown". According to the open world assumption, no conclusions are made until more knowledge is available which might result in an unambiguous answer. Practically, it means that in principle it always possible to add new logically consistent knowledge to an ontology without invalidating its conceptualization or content, whereas with traditional data models in a closed world this might possibly require a complete overhaul of the model or its content.

3 PROCESS MODEL SELECTION

Developing an ontology is no trivial task. However, compared to software engineering, ontologies and their development are a rather young discipline which emerged only in the early 1990s. Thus, relevant development processes are not as widespread and well known as those in software engineering. Hence, we needed to get an overview about available ontology development processes for selecting an appropriate process for our project.
We focused our literature research on ontology development methodologies which exhibit the essential characteristics of a software engineering process. The following process models conform to this requirement:

METHONTOLOGY [7] (1997/1999): The goal of this process model was to establish a proper engineering approach and reproducible process for ontology development in contrast to “construction guidelines” from earlier publications on ontology creation. This resulted in a waterfall-like model of process activities, with defined products for each process phase that also serve as documentation.

OTK [8] (On-To-Knowledge, 2002): This is a detailed process model that discriminates between more project phases and more iterative steps. It incorporates “competency questions” as introduced by Grüninger and Fox [9] for specification purposes. The development of the ontology is embedded in the context of developing and using a "‘knowledge application"’.

DILIGENT [10] (Methodology for DIstributed, Loosely-controlled and evolvInG Engineering of onTologies, 2005): This process model is designed for distributed development and evolution of ontologies. While users of ontologies take an active role in this development, making suggestions and adapting the ontology to their needs, the latter leads to various versions of this ontology which have to be consolidated in an iterative development of the root-ontology organized via a board of experts.

HCOME [11] (Human-Centered Ontology Engineering Methodology, 2005): Similar to DILIGENT this process model stipulates a distributed development of an ontology. However, this development is solely made by the ontology users, with only domain experts being available as counsel in case of questions. Thus the development is dependent on a special tool framework that supports the users in this process.

UPON [12] (Unified Process for ONtology building, 2005/2008): This process model is an adaption of the Unified Process [13] for ontology development. The development is embedded in the context of a specific application. It is use-case driven and makes intensive use of the Unified Modeling Language (UML[2]).

NeOn [14] (NeOn Methodology for Building Ontology Networks)[3], 2006/2010): This process model addresses scalability in the context of an environment of interconnected ontologies. Its focus lies on integrating existing ontological resources into the development of an ontology and allows for a flexible process by describing a set of development scenarios which can be combined. Although a specific application context is recommended, it is not necessary.

As for selection criteria, **recency** was an important factor as most process models incorporated experience from previously proposed models. We further assessed the process models regarding **applicability** for our development objective. **Level of detail** was considered helpful for our objective as we had no experience from prior development projects, thus requiring more detailed guidance. Accordingly, we wanted to select one process model we could use for our first development cycle without major adaptions, rather than creating a new process model as combination of different process models.
Applying these criteria, we were able to select an appropriate process model. Distributed development, being the focal point of both DILIGENT and HCOME, was not an issue in our project, so both these process models were not applicable. As well, both OTK and UPON require the ontology development to be put into context with an accompanying application. As mentioned in section 2, the ontology development we had in mind did not correspond to this requirement, so these two process models were also not applicable. From the remaining two process models, NeOn was more up to date, provided more details and covered a broad spectrum of development scenarios, as compared to METHONTOL-

[2] http://www.uml.org
[3] http://www.neon-project.org

OGY. Concluding, we selected NEON as the process model for our development.

4 ONTOLOGY DEVELOPMENT

In the following, we provide a short introduction into the NEON methodology, show its application in our project, and conclude with the evaluation of the developed ontology.

4.1 NEON Methodology Overview

NEON addresses developing ontologies in the context of a networked environment where several different ontologies or semantic knowledge bases exist, possibly with various interconnections [14]. NEON approaches the ontology development by the use of *scenarios*. Each scenario covers certain aspects within the ontology development, for example the reuse or re-engineering of non-ontological resources, the application of ontology design patterns, etc. A scenario is described by a set of activities that have to be executed in a given order. Scenarios can also be combined with one another. All scenarios are also accompanied by several support activities, e.g. *documentation*.

NEON describes the following nine scenarios [15], which are also visualized in Figure 1:

1. From specification to implementation.
2. Reusing and re-engineering non-ontological resources.
3. Reusing ontological resources.
4. Reusing and re-engineering ontological resources.
5. Reusing and merging ontological resources.
6. Reusing, merging and re-engineering ontological resources.
7. Reusing ontology design patterns.
8. Restructuring ontological resources.
9. Localizing ontological resources.

The NEON process lifecycle for one iteration consists of a maximum of seven phases. The actual number of phases depends on the applicability of the NEON scenarios. These scenarios also define the process activities which are assigned to these phases when creating the project plan. The possible seven phases are:

1. **Initiation phase**: This phase deals with pre-project studies, project preliminaries, specification and instantiation of NEON for a concrete project plan.
2. **Reuse phase**: The second project phase is about searching for and selecting appropriate ontological and non-ontological resources for reuse in the project, which is supported in NEON by (basic and specific) activity flows [16, 17].

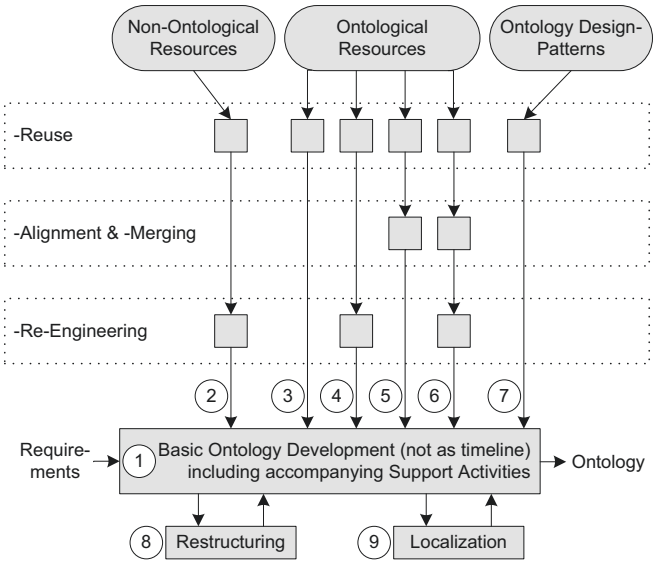

Figure 1: Overview of the NEON Scenarios, adapted from Suárez-Figueroa et al. [15]

3. **Re-engineering phase**: This third project phase covers the re- engineering of the identified ontological or non-ontological resources [16, 18].
4. **Merging phase**: This phase deals with aligning and merging two or more ontologies that cover similar concepts.
5. **Design (Modelling) phase**: This phase is about fully conceptualizing and formalizing the ontology.
6. **Implementation phase**: In this project phase the fully conceptualized and formalized model is implemented in the target ontology language.
7. **Maintenance phase**: This final phase is about maintaining the ontology and discovering possible improvements or incentives for further development, possibly leading to further development cycles.

To get a concrete instance of NEON for an ontology development project, the applicable scenarios for that project have to be selected according to the project requirements. The development can be divided into several iterations, with each iteration only covering certain scenarios. These decisions lead to the project plan for the ontology development. This selection process and acquiring the project plan is supported by the NEON Toolkit[4] in combination with the GONTT[5] plug-in.

4.2 Applying the NEON Methodology

In the following we describe our concrete application of NEON. We required about ten man weeks to perform the development process, roughly distributed in 10% for the initiation phase, 10% for the reuse phase, 20% for the re-engineering phase, 50% for the design and implementation, and 10% for the evaluation and last corrections.

[4] http://neon-toolkit.org/
[5] http://neon-toolkit.org/wiki/Gontt

Initiation Phase

At first, several preliminary decisions had to be made. Some of these decisions were important for the ontology specification, which itself is necessary for deriving the project plan. Regarding the language for the ontology, it was decided to use the OWL 2 Web Ontology Language published by the W3C, specifically OWL 2 DL, due to the availability of application programming interfaces (APIs), development tools, and reasoners. PROTÉGÉ[6] was chosen as development tool due to its open architecture, its large community of users and developers, and its practical user interface. From the NEON Toolkit, which has been developed together with the NEON methodology, only the GONTT tool seemed sufficiently stable to be used for project plan generation.

NEON requires the formulation of the Ontology Requirements Specification Document (ORSD). NEON provides a template for the ORSD which is later used for the generation of a project plan. It captures the ontology's purpose, scope, formality level, intended users and uses as well as *competency questions*. These questions are formulated in natural language and should be able to be answered using the ontology; they also serve as a benchmark for evaluation of the built ontology. Our main focus lay on structural aspects of a standard passenger aircraft, e.g. an A320, with the intention to also provide standard units and measurement dimensions.

Following the ORSD, NEON scenarios 1, 2, and 3 were applicable to our project. Due to limited time for the project we only planned for one development iteration. GONTT was used to generate a project plan conforming to our project conditions. This plan was manually reproduced in MICROSOFT PROJECT for more refinements.

We further adapted the project plan recommended by NEON. In particular, the pre-development studies in the *initiation* phase were not necessary in our project due to its explorative character and limited scope. The *merging* phase was also not applicable since we did not intend to merge several different ontologies in this development cycle. Due to the tight connection between the *design* and *implementation* phases, and the usage of the OWL 2 DL, it was deemed practical to consolidate these two phases into one. Although evaluation accompanies the whole project, we felt the need to add a project phase dedicated to the evaluation of the whole built ontology. Finally, the maintenance phase was not applicable to this first iteration.

Reuse Phase

Special requirements regarding the ontological and non-ontological resources can be derived from the ORSD, which is relevant for the assessment, comparison and selection of the identified resources for possible reuse.

[6] http://protege.stanford.edu

Regarding non-ontological resources, reference models from the two aircraft design tools SIMCAD and PACE APD were provided. SIMCAD is a tool which has been developed and operated at Airbus Future Project Office in SCILAB[7] for conceptual aircraft design. It contains 173 aircraft parameters with units which are saved in a tree data structure in 49 container objects. PACE APD is a framework based on the PACELAB SUITE[8]. We focused on 387 scalar aircraft parameters with units which are organized in a tree data structure composed by 71 container objects.

For this development iteration, we wanted to avoid additional overhead and thus relinquished to derive knowledge from textual resources such as Federal Aircraft Association Standards via natural language processing. Finally, on-site experts from Bauhaus Luftfahrt were available for interviews. However, due to having a high workload and tight schedules of their own, we were not able to involve these experts into the ontology development as much as we originally intended to.

With respect to the ORSD and possible ontological resources, it was planned to search for ontologies that comprise concepts of units, measures and quantities. Selection criteria were the proper ontology language and dialect with respect to the ORSD (OWL2-DL), semantic richness, useful content (e.g. also unit symbols, dimensions or conversion factors), and the usability with typical OWL reasoners, such as FACT++, HermiT, Pellet, Racer. The ontology that was finally selected is the Library for Quantity Kinds and Units (QU)[9].

Re-Engineering Phase

The selected unit ontology uses OWL, thus it could be integrated without modifications. Our non-ontological reference models were given in an XML format for models (XMI). These models were manually transformed into mind maps to better grasp and compare the underlying concepts. The content of those mind maps was then pruned regarding our requirements in the ORSD and consolidated to be used as the reference for the following project phase.

Design and Implementation Phase

With respect to the design and implementation, NEON refers to other established ontology development methodologies [19]. For this project it was decided to use the relevant aspects of OTK [8, 20, 21], especially since it also recommends the use of competency questions and encourages the use of mind maps for the conceptualization.
Due to our manual approach for creating the ontology, a top down approach was applied for the conceptualiza-

[7] www.scilab.org
[8] www.pace.de
[9] http://www.w3.org/2005/Incubator/ssn/XGR-ssn-20110628/#Units_of_measurement_and_quantity_ontologies

tion, meaning to start at a generic level and then extending into more detailed levels as needed. This is opposed to a bottom-up approach when using automated techniques, starting at the most detailed level and then integrating concepts into more generic levels, or a middle-out approach which is a combination of these two. Concepts from the mind maps were iteratively added to the ontology and enriched by relational concepts that were required according to the ORSD. To ensure a minimum level of quality and readability, certain naming conventions were applied, as well as some established best-practices (i.e. for model composition), and descriptive annotation comments where needed.

4.3 Evaluation of the AIRCRAFT Ontology

NEON regards evaluation as an accompanying activity that is performed for each activity and phase and not as an explicit project phase. However, we introduced a designated evaluation phase at the end of the development cycle. One reason was that we wanted to involve domain experts not before our first implementation had reached sufficient stability.
NEON itself differentiates between three dimensions for evaluation [22]: structural, functional, and usability-related. However, NEON focuses on metrics, gold standards such as implementation benchmarks, which were not available for our case, and formal methods, which are not suitable for this project. Thus, we established our own evaluation criteria for this phase in our project: first, expert interviews to find faults in the ontology, and second, the coverage of the competency questions from the ORSD.

For the interviews, two experts were available for an one-hour interview each. The focus lay on the correctness of the concepts in the ontology, with a conventional civil passenger aircraft in mind.
The interviews were structured as follows: First a short introduction about the ontology and the objective for the interview was presented. Second, ontology statements about the structural composition were discussed. The most complex and thus potentially most controversial concept definitions were discussed first in order to ensure that less controversial concept definitions had to be skipped if interview time ran out. Finally, parameter sets and their associations with component concepts were discussed. Again, the focus was on the most complex and thus most error prone parameter set definitions.

The interviews provided some general insights. First, some concepts are named differently in American English and British English, which can be misleading if not properly declared. It is also very important to clearly define and delimit concepts to avoid ambiguities. As concept boundaries become fuzzy, generalization becomes increasingly difficult. Second, with a growing level of detail, interdependencies increase, making exact distinctions between concepts difficult. For example, the question was raised whether the landing gear compartment has a stronger relation to the fuselage or to the landing gear. During the interviews not all such questions could be decided by the domain experts. In these cases the ontology developer resolved the issue by declaring both associations valid. Finally there exist some terms that have an established meaning, which were not properly used. A `Group`, for example, denotes a loosely connected set of components pertaining to one specific aspect like e.g. propulsion, whereas a `System` denotes an integrated component with all its parts.

Regarding the specific results of the interviews, out of 343 class definition statements , 314 were covered and 29 had to be omitted as time ran out. Effectively this means that only less than 10% of the least important parts of the ontology were not evaluated by domain experts, while more than the most important 90% were.
We defined five different types of issues in Table 1 as follows:

- W: The `horizontal stabilizer` was called `tailplane`, which is in fact a more general term that describes any stabilizing plane at the tail of an aircraft
- A: The parameter `isDescribedByRatioVolume some DimensionlessParameter` is somewhat vague regarding its name, since it is a ratio, but is calculated by multiplying an area with a length, which one would assume to be a volume.
- S: Some components had a parameter “centerline chord", as well as a “mean aerodynamic chord”, which is essentially the same. Ontologies allow a formal definition of synonyms. However, we decided to implement a proper handling of synonyms in later development cycles and, for now, to delete one of the synonyms.
- R: The parameter `isDescribedByYKink some DistanceParameter` for a wing is not clearly defined, since it is not clear from where (and in what direction) this distance respectively length is measured.
- M: Due to the misconception of a parameter “lever arm” for the vertical stabilizer, its “span” was omitted

All issues identified during the interviews (see Table 1) were thereafter corrected in the ontology.

Accordingly, by interviewing domain experts the correctness of most statements in the ontology could be validated. However, due to the open world assumption it is impossible to validate the completeness of an ontology on a general level. However, limited completeness of the ontology with respect to its requirements from the ORSD can be tested by checking whether the ontology can answer the predefined competency questions correctly. As shown in Table 2, all of the five questions categorized as priority 1 (essential) can be answered. Out of the five priority 2 questions (wanted, but not essentially needed for this first iteration), three can be answered. None of the four priority 3 questions (nice to have, but not expected for this first iteration) can be answered.

Type	Code	Count
Name is Wrong	W	4
Name or concept is Ambiguous	A	11
Undeclared Synonym	S	8
Reference for describing parameter is missing	R	3
Missing parameter due to wrong assumptions	M	2
Total		28

Table 1: Qualitative excerpt of the survey results among stakeholders

5 THE AIRCRAFT ONTOLOGY

The AIRCRAFT ontology is available at a GITHUB repository[10]. This repository also contains the thesis that provides an in-depth description of the ontology.

The AIRCRAFT ontology itself contains 96 classes and 224 object properties. For demonstration purpose the ontology also contains 22 individuals. The ontology imports the public `qu-rec20.owl` ontology to reuse it for defining appropriate quantity kinds and units of parameters. This imported ontology contains 275 classes, 22 object properties and 12 data type properties.

In the following we describe how the ontology represents knowledge about system decomposition and parametrization of a civil passenger transport aircraft like an Airbus A320, and give an example of how this knowledge can be applied.

5.1 Representation of System Decomposition

Figure 2 shows how system components and their subcomponents are represented in the ontology, following a clear naming and structure convention. In particular, class names start with upper case, while properties start with lower case. A class representing a component is accompanied with a subcomponent class. For instance, `Engine` is modeled as a subclass of `AircraftSubComponent` which is the subcomponent class of `Aircraft`. To assemble a component by its subcomponents, a part-of relation is modeled using appropriate object properties shown in Figure 3 and Figure 4. Accordingly, `Aircraft` has at least one `hasEngine` property to `Engine` and inversely `Engine` has at least one `isEngineOf` property to an `Aircraft`). Component classes on the same level of system decomposition can also be related to other component classes by the `connectedTo` object property. Accordingly, `Engine` can have a `connectedTo` relation to `Wing` or to the `Fuselage`. Following the open world assumption, this example does not specify a particular propulsion system configuration but represents knowledge on known component arrangements.

[10] published at `https://github.com/astbhltum/Aircraft-Ontology` under the Eclipse Public License, Version 1.0 (EPL-1.0)

Prio		Result
	Structure	
1	*What is the structure of a model component (from the given models)?*	✓
1	*Which component is component X part of?*	✓
2	*What is the wing configuration of the airplane?*	×
2	*What is the undercarriage configuration of the airplane?*	✓
2	*What is the engine configuration of the airplane?*	✓
1	*Which structural parameters define component X?*	✓
1	*Which component(s) is component X attached to?*	✓
2	*What parameters are relevant for aerodynamics?*	×
	Units	
1	*Which unit is a parameter measured in?*	✓
2	*Is a certain length X in foot equal to another length Y in meter?*	✓
3	*Which quantities does the speed of sound depend on?*	×
	Behaviour	
3	*What is the operational radius of the airplane?*	×
3	*Which parameters does the operational radius depend on?*	×
3	*Can the airplane go directly from take-off to cruise?*	×

Table 2: Evaluation results of the AIRCRAFT ontology by competency questions defined in the ORSD

5.2 Representation of Component Parameters

Besides knowledge about the structural interrelations of system components, the ontology also contains knowledge about relations of components to their defining parameters. However, while developing the ontology in PROTÉGÉ we encountered that combining structural aspects with parameter relations resulted in very large and cluttered class definitions. Therefore, we decided to introduce a proxy class between each component and its defining parameters. For example the class `Engine isDescribedByEngineDescribingParameter` called `EngineDescribingParameter`. This latter class is the proxy and effectively forms the set of all parameters associated to the component. The proxy is connected to all these parameters via specialized object properties. So, among others, `EngineDescribingParameter is-`

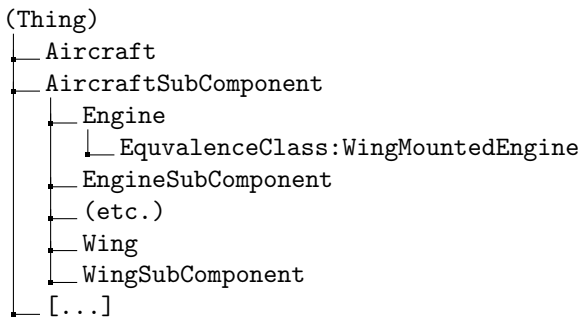

Figure 2: Class structure composition excerpt

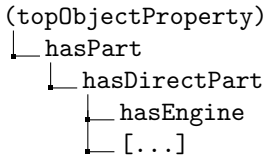

Figure 3: Object property excerpt of `hasPart`-relation

`DescribedByReferenceThrust` with a `ForceParameter`. Such explicitly named properties allow the restriction to parameters with units of appropriate dimensions. This is where the external ontology for units and dimensions, QU, comes to use. By assigning a unit type to a parameter, its dimension (distance, area, etc.) is automatically given, enabling the modeling of appropriate parameters. QU furthermore provides a property to assign a numerical value. So, following the example, `ForceParameter` has a `unit` of type `ForceUnit`, and a `numericalValue` as data type `double`.

5.3 Reasoning

Regarding reasoning, the two most relevant benefits are consistency checks and automated classification. By using the restrictive capabilities provided by OWL, we were able to enforce rather rigid consistency constraints. Trying to assign an `AreaUnit` to a `ForceParameter`, or to assign a `DistanceParameter` to `Reference Thrust`, will result in an inconsistency error. Furthermore, an inconsistency will be detected when a `LandingGear isFuselageOf Wing`. The subsumption service of reasoners can classify an `Engine` that `isConnectedTo` a `Wing` as `WingMountedEngine`. Given an individual of an `Engine` that was accidentally created as a direct member of the most general class `Thing`. When this individual `isEngineOf` an `Aircraft` individual, the reasoner can classify it as `Engine`.

6 DISCUSSION

In this section, we discuss our observations and lessons learned from the development of the AIRCRAFT ontology.

Advantages of the NEON Methodology We found the NEON methodology appropriate for our purpose. The

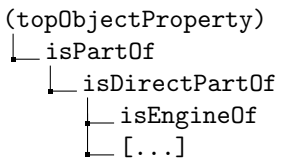

Figure 4: Object property excerpt of `isPartOf`-relation

scenarios defined by NEON allowed us to customize our ontology development process to our specific situation that was determined by the type of development and the data sources provided.

Disdvantages of the NEON Methodology We experienced some weaknesses of NEON regarding the initiation and the evaluation phases. In particular, guidelines for preliminary planning are missing in the process model. Furthermore, NEON proposes evaluation techniques which require the existence of comparable ontologies to collect metrics data and compare them for different ontologies. In bootstrapping development projects, comparable ontologies are not available and this approach cannot be applied.

Avantages of Ontologies We found several advantages for using ontologies as knowledge representation mechanism. First, the ontology infrastructure, in particular the open world assumption as well as the possibility to import one ontology into another one, was helpful to integrate knowledge from different sources. Using this infrastructure, we could integrate the existing OWL ontology for measures and units without complications. According to this experience, we recommend ontology reuse efforts from the beginning of ontology development as well as the use of standardized ontology languages. Second, we applied reasoners to handle the growing size and complexity of our ontology by checking logical consistency. We could use off-the-shelf reasoners as we employed a standard ontology language and not a modeling language such as UML with no formal semantics.

Disadvantages of Ontologies Besides advantages, we also experienced disadvantages by using ontologies. First, a full consistency check of our final ontology using a state of the art reasoner on a standard laptop required more than 20 seconds. Accordingly, an interactive ontology development - checking consistency after every modification of the ontology - could not be performed. Instead, we checked consistency only after adding several ontology elements which increased the difficulty of pinpointing a malicious element in case of an inconsistency error. Second, we felt that the complexity of an OWL class that represented structure decomposition and component parameter aspects was already quite big. Hence, we moved the component parameter aspects into special classes. While it is debatable when a class becomes too complex, the problem of too complex classes is independent of our specific case.

Lessons Learned from Ontology Evaluation We learned three things while evaluating our AIRCRAFT ontology by expert interviews. First, expert involvement is

crucial. Checking the correctness and completeness of an ontology about a special domain can only be done by involvement of domain experts. But those experts are usually busy people and it is difficult to get their time. We think we found a good compromise by evaluating a previously created ontology and revise it based on expert input. Second, it cannot be guaranteed that all semantic errors can be found by our evaluation method. Third, our AIRCRAFT ontology was developed independently from any design tool or application used by domain experts. A negative consequence of this strategy was that the developed ontology had no direct relevance to the daily work of the domain experts and it was difficult for them to assess the consequences of different solution alternatives or to evaluate design decisions in a concrete context. Therefore, we concur with the NEON recommendation to involve domain experts and to develop the ontology in the context of a concrete application as soon as possible.

7 RELATED WORK

There has been extensive work on the benefits of applying semantic technologies for the efficiency of model-driven systems engineering, which has been the motivational background for our ontology development project.

Broy et al. [26] studied model-based systems engineering environments. They found that formal semantics of system and process modeling languages are essential prerequisites for seamless tool integration. We consider our AIRCRAFT ontology as a contribution to future instances of their theoretical framework in the aircraft design domain.

Reiss et al. [23] give a general overview on the application of ontologies in the aeronautic domain. In particular, they address the differences and advantages of ontologies to other domain modeling techniques, such as metamodels. There are other aeronautic related ontology development projects. Different from our AIRCRAFT ontology which is supposed to be applied in the context of aircraft design, the ontology developing project by Valente et al. [24] is supposed to be applied in the context of flight operations where system behavior is more relevant than system structure. In their study they describe their experience in developing an ontology for air campaign planning. For example, they encountered that ontologies made for different use cases can have different structure which makes them less compatible. With our AIRCRAFT ontology development process we aimed for avoiding this “use-bias”.

Ontology-model integration of conceptual aircraft design model is shown by Glas [29]. The methodology uses a reference ontology which is derived form the integrated models in an interactive process. As a result, the structure of the final reference ontology is closely aligned with the structure of the source models. In contrast, the goal of our development process described in this paper was to design an ontology which is independent from a particular aircraft model.

8 CONCLUSION

Our motivation for developing the AIRCRAFT ontology is the lack of a publicly available ontology for aircraft design which can be used as a semantic reference during consistency checking and integration of different aircraft design models. We selected the NEON methodology and followed its guidelines to develop the AIRCRAFT ontology. We found the guidelines of the NEON methodology appropriate for our context with a few exceptions and describe our instantiation of the NEON methodology in detail. The resulting AIRCRAFT ontology is an OWL ontology which covers structural system decomposition and component parameters of an aircraft. It is publicly available from a GITHUB repository under the EPL license. The AIRCRAFT ontology was evaluated regarding correctness in interviews with two domain experts and regarding completeness by testing it with predefined competency questions. By presenting our ontology development process based on the NEON methodology as well as the resulting AIRCRAFT ontology, we hope to enable colleagues to use the AIRCRAFT ontology and extend or modify it to fit their needs.

Our next step will be to apply and exploit the AIRCRAFT ontology in consistency checking and integration tools for aircraft design models and evaluate its appropriateness. This exploitation is supposed to improve the efficiency of services such as model integration as proposed by Glas [29]. We also intend to exploit the AIRCRAFT ontology in existing aircraft design tools beyond model consistency checking and integration. Our long term goal is to contribute to the integration of semantic technologies into system design tools as proposed by Bauer and Roser [30], and to the establishment of knowledge engineering as a natural part of systems design.

Acknowledgements

We thank Arne Seitz and Patrick Vratny for their knowledgeable support during the evaluation of the AIRCRAFT ontology. We also thank Sven Ziemer for his constructive comments on earlier versions of this paper.

References

[1] Kappel, G., Kargl, H., Kramler, G., Schauerhuber, A., Seidl, M., Strommer, M., and Wimmer, M. (2007). “Matching Metamodels with Semantic Systems - An Experience Report”. In: *BTW Workshops*, pp. 38–52.

[2] Roser, S. (2008). “Designing and Enacting Cross-organisational Business Processes: A Model-driven, Ontology-based Approach”. eng. PhD thesis. Universitätsstr. 22, 86159 Augsburg: University of Augsburg.

[3] Studer, R., Benjamins, V., and Fensel, D. (1998). “Knowledge engineering: principles and methods”. In: *IEEE Trans. Knowl. Data Eng.* 25 (1-2), pp. 161–197.

[4] Studer, R., Grimm, S., and Abecker, A. (2007). *Semantic Web Services: Concepts, Technologies, and Applications*. Springer-Verlag New York, Inc.

[5] W3C (Feb. 2004). *OWL Web Ontology Language Overview*. Internet. W3C Recommendation. URL: http://www.w3.org/TR/owl-features/.

[6] W3C OWL Working Group (Mar. 2009 onward). *OWL 2 Web Ontology Language Document Overview (Second Edition)*. Internet. W3C Recommendation. URL: http://www.w3.org/TR/owl2-overview/.

[7] Fernández-López, M., Gómez-Pérez, A., and Juristo, N. (Mar. 1997). “Methontology: from ontological art towards ontological engineering”. In: *Proceedings of the Ontological Engineering AAAI-97 Spring Symposium Series*. Stanford University. Palo Alto, California: American Asociation for Artificial Intelligence. URL: http://www.aaai.org/Papers/Symposia/Spring/1997/SS-97-06/SS97-06-005.pdf.

[8] Sure, Y., Staab, S., and Studer, R. (2003). “On-To-Knowledge Methodology”. In: *Handbook on Ontologies*. Ed. by S. Staab and R. S. (eds.) Series on Handbooks in Information Systems. Springer, pp. 117–132.

[9] Grüninger, M. and Fox, M. (Apr. 1995). “Methodology for the Design and Evaluation of Ontologies”. In: *Workshop on Basic Ontological Issues in Knowledge Sharing, IJCAI-95*. University of Toronto. Montreal. URL: http://www.eil.utoronto.ca/enterprise-modelling/papers/gruninger-ijcai95.pdf.

[10] Pinto, H. S., Tempich, C., and Staab, S. (Aug. 2004). “DILIGENT: Towards a fine-grained methodology for DIstributed, Loosely-controlled and evolvInG Engingeering of oNTologies”. In: *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 2004), August 22nd - 27th*. Ed. by R. L. de Mantaras and L. Saitta. Valencia, Spain: IOS Press, pp. 393–397.

[11] Kotis, K. and Vouros, G. (2006). “Human-centered ontology engineering: The HCOME methodology”. In: *Knowledge and Information Systems* 10 (1). 10.1007/s10115-005-0227-4, pp. 109–131. URL: http://dx.doi.org/10.1007/s10115-005-0227-4.

[12] De Nicola, A., Missikoff, M., and Navigli, R. (2009). “A software engineering approach to ontology building”. In: *Information Systems* 34.2, pp. 258–275.

[13] Jacobson, I., Booch, G., and Rumbaugh, J. (1999). *The Unified Software Development Process*. Addison-Wesley Professional.

[14] Suárez-Figueroa, M.-C., Gómez-Pérez, A., Motta, E., and Gangemi, A. (2012). *Ontology Engineering in a Networked World*. Ed. by M.-C. Suárez-Figueroa, A. Gómez-Pérez, E. Motta, and A. Gangemi. Berlin: Springer.

[15] Suárez-Figueroa, M. C., Fernéndez-López, M., Gómez-Pérez, A., Dellschaft, K., Lewen, H., and Dzbor, M. (Nov. 2008a). *D5.3.2 Revision and Extension of the NeOn Development Process and Ontology Life Cycle*. Internet. Deliverable 5.3.2 in the NeOn Project (www.neon-project.org). URL: http://www.neon-project.org/web-content/images/Publications/neon_2009_d532.pdf.

[16] Suárez-Figueroa, M. C., Cea, G. A. de, Buil, C., Dellschaft, K., Fernéndez-López, M., García, A., Gómez-Pérez, A., Herrero, G., Montiel-Ponsoda, E., Sabou, M., Villazon-Terrazas, B., and Yufei, Z. (Feb. 2008b). *D5.4.1. NeOn Methodology for Building Contextualized Ontology Networks*. Internet. Deliverable 5.4.1 in the NeOn Project (www.neon-project.org). URL: http://www.neon-project.org/web-content/images/Publications/neon_2008_d5.4.1.pdf.

[17] Suárez-Figueroa, M. C., Gómez-Pérez, A., Poveda, M., Ramos, J. A., Euzenat, J., and Duc, C. L. (Jan. 2010). *D5.4.3. Revision and Extension of the NeOn Methodology for Building Contextualized Ontology Networks*. Internet. Deliverable 5.4.3 in the NeOn Project (www.neon-project.org). URL: http://www.neon-project.org/nw/images/c/c6/NeOn_2010_D543.pdf.

[18] Suárez-Figueroa, M. C., Blomqvist, E., D’Aquin, M., Espinoza, M., Gómez-Pérez, A., Lewen, H., Mozetic, I., Palma, R., Poveda, M., Sini, M., Villazón-Terrazas, B., Zablith, F., and Dzbor, M. (Feb. 2009). *D5.4.2. Revision and Extension of the NeOn Methodology for Building Contextualized Ontology Networks*. Internet. Deliverable 5.4.2 in the NeOn Project (www.neo-project.org). URL: http://www.neon-project.org/web-content/images/Publications/neon_2009_d542.pdf.

[19] Suárez-Figueroa, M. C., Cea, G. A. de, Buil, C., Caracciolo, C., Dzbor, M., Gómez-Pérez, A., Herrero, G., Lewen, H., Montiel-Ponsoda, E., and Presutti, V. (Aug. 2007). *D5.3.1 NeOn Development Process and Ontology Life Cycle*. Internet. Deliverable 5.3.1 in the NeOn Project (www.neon-project.org). URL: http://www.neon-project.org/web-content/images/Publications/neon_2007_d5.3.1.pdf.

[20] Sure, Y. (2003). “Methodology, Tools and Case Studies for Ontology based Knowledge Management”. Phdthesis. PhD thesis at the Universität Karlsruhe (TH), Fakultät für Wirtschaftswissenschaften.

[21] Staab, S. and Studer, R. (2009). *Handbook on Ontologies*. Springer-Verlag Berlin Heidelberg.

[22] Sabou, M., Angeletou, S., d’Aquin, M., Barrasa, J., Dellschaft, K., Gangemi, A., Lehmann, J., Lewen, H., Maynard, D., Mladenic, D., Nissim, M., Peters, W., Presutti, V., and Villazón, B. (May 2007). *D2.2.1 Methods for Selection and Integration of Reusable Components from Formal or Informal*

User Specifications. Internet. Deliverable 2.2.1 in the NeOn Project (www.neon-project.org).

[23] Reiss, M., Moal, M., Barnard, Y., Ramu, J.-P., and Froger, A. (2006). “Using Ontologies to Conceptualize the Aeronautic Domain”. In: *Proceedings of the International Conference on Human-Computer Interaction in Aeronautics*. Cépaduès-Editions, Toulouse, France, pp. 56–63.

[24] Valente, A., Russ, T., MacGregor, R., and Swartout, W. (1999). “Building and (re)using an ontology of air campaign planning”. In: *Intelligent Systems and their Applications, IEEE* 14.1, pp. 27–36.

[25] Liu, J., Tang, Li, Q., and Wang, W.-p. (2009). “An Ontology for Aircraft Route Planning”. In: *Advances in System Simulation, 2009. SIMUL ’09. First International Conference on*, pp. 68–72.

[26] Broy, M., Feilkas, M., Herrmannsdörfer, M., Merenda, S., and Ratiu, D. (2010). “Seamless Model-Based Development: From Isolated Tools to Integrated Model Engineering Environments”. In: *Proceedings of the IEEE* 98.4, pp. 526–545.

[27] Fabro, M. D. D. and Valduriez, P. (2009). “Towards the efficient development of model transformations using model weaving and matching transformations”. In: *Software and Systems Modeling* 8 (3). 10.1007/s10270-008-0094-z, pp. 305–324. URL: http://dx.doi.org/10.1007/s10270-008-0094-z.

[28] Simon Zayas, D., Monceaux, A., and Ait-Ameur, Y. (2011). “Using Knowledge and Expressions to Validate Inter-Model Constraints”. In: *World Congress*. Vol. 18. 1, pp. 2737–2742.

[29] Glas, M. (2013). “Ontology-based Model Integration for the Conceptual Design of Aircraft”. PhD thesis. Technische Universität München.

[30] Bauer, B. and Roser, S. (2006). “Semantic-enabled Software Engineering and Development.” In: *GI Jahrestagung (2)*, pp. 293–296.