Towards Integration and Coverage Assessment of Ontologies for Knowledge Reuse in the Aviation Sector

Jos Lehmann
Bauhaus Luftfahrt e.V.
Willy-Messerschmitt-Straße 1
82024 Taufkirchen, Germany
jos.lehmann@
bauhaus-luftfahrt.net

Michael Shamiyeh
Bauhaus Luftfahrt e.V.
Willy-Messerschmitt-Straße 1
82024 Taufkirchen, Germany
michael.shamiyeh@
bauhaus-luftfahrt.net

Sven Ziemer
Bauhaus Luftfahrt e.V.
Willy-Messerschmitt-Straße 1
82024 Taufkirchen, Germany
sven.ziemer@
bauhaus-luftfahrt.net

ABSTRACT

Ontology-based applications can support the reuse of corporate-knowledge in engineering environments. In order to streamline the harnessing of semantics, problems of integration and of coverage assessment of ontologies need to be addressed. This paper provides examples of such problems when applying Semantic Technology in the aviation sector and outlines a strategy towards their solution. The paper also provides a preliminary discussion of how knowledge management architectures such as the one presented may be positioned in the wider research area of *Industrie 4.0*.

Keywords

Knowledge Integration and Language Technologies; Terminology, Thesaurus and Ontology Management; Industry and Engineering; Web Ontology Language; Aviation; *Industrie 4.0*; Linked Data

1. INTRODUCTION

In the context of the German high-tech strategic program *Industrie 4.0*, which promotes research on the ongoing fourth industrial revolution yielded by the digitization of products, processes and organizations, we are investigating the application of Semantic Technology to the reuse of corporate-knowledge in the aviation sector.

This research focuses on how semantics could be harnessed by the information systems employed during the conceptual design of aeronautical components. The key idea, akin to examples of Knowledge-based Engineering such as [9] or [11], is that engineering-projects in their early stages would benefit from semantic search, as this would expand access to existing corporate-knowledge (e.g. legacy-data from previous projects) as well as make search-results more relevant. The types of corporate-knowledge being considered include two main categories of information sources: textual data sources and non-textual data sources.

SEMANTiCS 2017 workshops proceedings: LIDARI
September 11-14, 2017, Amsterdam, Netherlands

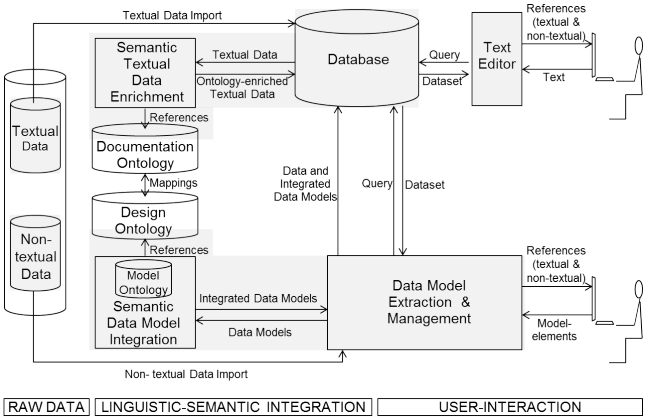


Figure 1: Architecture supporting Knowledge Reuse

The core Semantic Technology applied to increase access and relevance is ontology. This makes it possible, on the one hand, to relate non-matching information that conceptually belongs together and, on the other hand, to filter matching information that is conceptually unrelated.

Section 2 of this paper describes an ontology-based software architecture for the support of knowledge reuse in the aviation sector. Section 3 provides examples of the problems raised by the harnessing of the needed semantics. Section 4 outlines a strategy towards the solution of such problems. Section 5 provides a preliminary discussion of how knowledge management architectures such as the one presented should be positioned with respect to the wider research area of *Industrie 4.0*, in particular with respect to subareas that involve Linked Data and Robotics applications.

2. ARCHITECTURE

Figure 1 shows the software architecture being researched and developed to support knowledge reuse. The architecture attempts to combine two pre-existing architectures and legacy-data (all in gray areas). The pre-existing architectures are based on divergent ontological commitments, especially different *representational* choices (regarding, for instance, what to model as a class and what as an individual, how many different properties to use in the ontology, whether to represent constraints, etc.). Such pre-existing representational commitments are conveyed by the two main ontological modules being only partially in a gray area of

Figure 1. Moreover, textual and non-textual data too are pre-existing. Otherwise, all white parts of Figure 1, including the proper ontological content of the ontological modules, the criteria of mutual relevance of textual and non-textual data (conveyed by the larger data-cylinder) as well as import and query procedures are being researched and developed as part of the project discussed in this paper.

On the left of Figure 1, two datasets of textual and non-textual data are selected for integration. The present particular case study aims at working on a dataset of textual data that contains a varied corpus of documents ranging from design-descriptions, to performance-reports, to internal standards, to inspection-reports, to lessons-learned. The dataset of non-textual data is intended to contain a selection of component models ranging from Computer-Aided Design (CAD) models, to calculation (MatLab) models, to simulation (Simulink) models, to models for in-house tools used in preliminary design. It is assumed that the textual and the non-textual datasets contain information sources that are interrelated or relevant to one another. For instance, the textual data may contain indirect references to the non-textual data, in the form of figures of components generated from the non-textual data. It may also contain knowledge that helps recovering the original intended meaning of legacy non-textual data that is poorly documented.

The right side of Figure 1 shows the intended end-users. The user at the top types text into a text document through a text editor, and in the process receives references to relevant textual as well as non-textual data. The user at the bottom works on tagging and/or on modifying a model in a model-management tool (possibly embedding a design software, from which the model is accessed), and in the process receives references to relevant non-textual as well as textual data.

The central part of Figure 1 shows the part of the architecture being developed to achieve the linguistic-semantic integration within and across the two datasets. This part of the architecture relies on three ontologies: the Documentation Ontology (DocO), the Model Ontology (ModO), the Design Ontology (DesO). In the case-study under consideration, these ontologies have been developed independently of one another and they make disparate ontological commitments in order to meet the representational and computational requirements of the software components that rely on them (the boxes in Figure 1). Their differences should be resolved in such a way that each component can access knowledge available in all three ontologies without changing its representational requirements.

Figure 2 provides an overview of the main groups of concepts (white modules), which the three ontologies under consideration represent. The gray bars at the bottom show the extent to which each ontology covers such concepts, as well as the extent of their overlap. DocO, DesO and ModO contain specifications of upper, core and domain classes essential for aviation.

2.1 Documentation Ontology

The top part of Figure 1 illustrates a proprietary ontology-based Natural Language Processing software that classifies word and phrase occurrences in text using DocO, which contains aliases as well as disambiguation terms in multiple languages. As exemplified in Figure 3[1], DocO's terminology box ($\mathcal{T}_{DocO}$) is simple, as it includes a limited number of owl classes of things likely to be mentioned in aviation-related texts, e.g. names of aviation companies, of company management, of aircraft families, models or components. For the most part, DocO contains individuals in its assertion box ($\mathcal{A}_{DocO}$), which are related to one another by assertions of a single type of object-property akin to the *narrower-than* relation between a hyponym and its hypernym in linguistics. For instance, the individual representing a given aircraft model (e.g. *A320-100*) has a narrower scope than the individual representing such model's aircraft family (i.e. *A320*). Similarly, both individuals representing the model and the family have a narrower scope than the individual representing the airframer (i.e. *Airbus*). As explained in Section 3.1.2, this design choice allows greater representational freedom, by supporting for instance meta-classifications. Also note that at present this ontology is not publicly available.

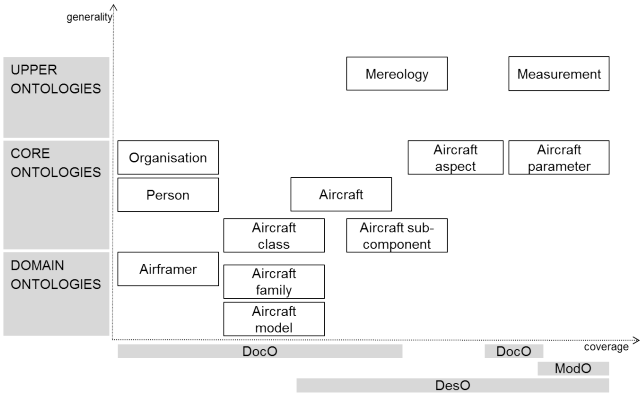


Figure 2: Ontological modules in Architecture

$$
\begin{aligned}
&\mathcal{T}_{DocO} ::= \{ \\
&\quad AicraftModel \sqsubseteq Thing, && (1)\\
&\quad AircraftFamily \sqsubseteq Thing, && (2)\\
&\quad Airframer \sqsubseteq Thing, && (3)\\
&\quad AircraftClass \sqsubseteq Thing, && (4)\\
&\quad AircraftComponent \sqsubseteq Thing, && (5)\\
&\quad narrower\text{-}than \sqsubseteq topObjectProperty\} && (6)\\
&\mathcal{A}_{DocO} ::= \{ \\
&\quad A320 : AircraftFamily, && (7)\\
&\quad A320\text{-}100 : AicraftModel, && (8)\\
&\quad SingleAisle : AircraftClass, && (9)\\
&\quad Airbus : Airframer, && (10)\\
&\quad narrower\text{-}than(A320\text{-}100, A320), && (11)\\
&\quad narrower\text{-}than(A320\text{-}100, SingleAisle), && (12)\\
&\quad narrower\text{-}than(A320, Airbus) && (13)\\
&\quad narrower\text{-}than(A320\text{-}100, Airbus)\} && (14)
\end{aligned}
$$

Figure 3: DocO on A320-100

[1]As in the Web Ontology Language (owl), in this paper the description logic (dl) notion of concept is called class. dl's logical constants have their standard meaning, i.e.: C1 $\sqsubseteq$ C2, C1 is a subclass of C2; i : C, i is an individual of class C; $\exists$ R.C, all individuals of a given class are in a relation R with individuals of class C.

2.2 Model Ontology

The central bottom part of Figure 1, shows the model extraction and management software that supports the integration of technical model data, based on a system proposed in [5] and similar to the framework discussed in [11]. That is achieved by first transforming a technical model into an ontology (a transformation that takes place in the Semantic Data Model Integration module). Consider for instance the decomposition for an *A320-100*'s fuselage height modeled in the model-excerpt shown in Figure 4. Such decomposition is transformed into a model ontology by creating an individual of class *MeasuredValue* with data properties for name, value and unit, as shown in Figure 5. Note that $\mathcal{T}_{ModO}$ in Figure 5 contains all of MODO's class hierarchy. An interface between such hierarchy and a particular tool's data model allows for the automatic transformation of a tecnical model's parameters into individuals of MODO's classes.

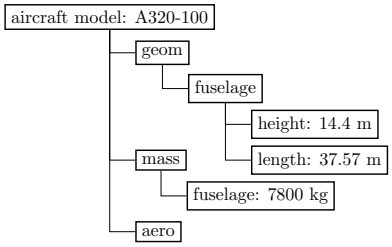


Figure 4: Excerpt of model for sizing code

$$\mathcal{T}_{ModO} ::= \{$$
$$< java : javax.measure.unit.Unit > \sqsubseteq Thing, \quad (15)$$
$$NamedElement \sqsubseteq Thing, \quad (16)$$
$$DataType \sqsubseteq NamedElement, \quad (17)$$
$$CompositeValues \sqsubseteq DataType, \quad (18)$$
$$LeafValue \sqsubseteq DataType, \quad (19)$$
$$Scalar \sqsubseteq LeafValue, \quad (20)$$
$$FloatPointValue \sqsubseteq Scalar, \quad (21)$$
$$MeasuredValue \sqsubseteq FloatPointValue, \quad (22)$$
$$FloatPointValue.value \sqsubseteq topDataProperty, \quad (23)$$
$$NamedElement.name \sqsubseteq topDataProperty, \quad (24)$$
$$MeasuredValue.unit \sqsubseteq topDataProperty\} \quad (25)$$
$$\mathcal{A}_{ModO} ::= \{$$
$$geom.fuselage.height : MeasuredValue, \quad (26)$$
$$NamedElement.name(geom.fuselage.height, \quad ''height''\hat{}\hat{}xsd : string), \quad (27)$$
$$MeasuredValue.unit(geom.fuselage.height, \quad ''m''\hat{}\hat{}(java : javax.measure.unit.BaseUnit)), \quad (28)$$
$$FloatPointValue.value(geom.fuselage.height, \quad ''4.14''\hat{}\hat{}xsd : double)\} \quad (29)$$

Figure 5: ModO on A320-100

2.3 Design Ontology

MODO is mapped onto a reference design ontology DESO. In the present case-study DESO builds on the AIRCRAFT ONTOLOGY[2] proposed in [1]. As exemplified in Figure 6, DESO contains a rich terminology for aircraft design. The emphasis is on providing conceptual descriptors: DESO contains upper and core modules for quantities, dimensions, units and parameters, imported from the ontologies QU[3] and QU-REC-20[4]. It also provides a mereological and connectedness structure specified between and across aircraft components, with a module for so-called aircraft aspects, i.e. functional combinations of physically separate subcomponents (e.g. the undercarriage group). As mentioned, the model ontology is combined with the design ontology as in Figure 7, i.e. by establishing an updated version of MODO, MODO*, which imports DOCO and allows to classify the individual introduced in Axiom (26).

$$\mathcal{T}_{DesO} ::= \{$$
$$Aircraft \sqsubseteq Thing, \quad (30)$$
$$Fuselage \sqsubseteq AircraftSubComponent, \quad (31)$$
$$FuselageDescribingParameter \sqsubseteq SubComponentDescribingParameter, \quad (32)$$
$$DistanceParameter \sqsubseteq SingleAircraftParameter, \quad (33)$$
$$hasFuselage \sqsubseteq \cdots \sqsubseteq hasPart, \quad (34)$$
$$isDescribedByHeight \sqsubseteq \cdots \sqsubseteq isDescribedByParameter, \quad (35)$$
$$Aircraft \sqsubseteq \exists hasFuselage.Fuselage, \quad (36)$$
$$Fuselage \sqsubseteq \exists isDescribedByFuselageDescribingParameter. FuselageDescribingParameter, \quad (37)$$
$$FuselageDescribingParameter \sqsubseteq \exists isDescribedByHeight.DistanceParameter \quad (38)$$
$$DistanceParameter \sqsubseteq \exists unit.DistanceUnit, \quad (39)$$
$$DistanceUnit \sqsubseteq Unit, \quad (40)$$
$$Unit \sqsubseteq \exists name.xsd : string \sqcap \exists symbol.xsd : string \quad (41)$$
$$SingleAircraftParameter \sqsubseteq \exists numericalValue.xsd : double\} \quad (42)$$
$$\mathcal{A}_{DesO} ::= \{$$
$$A320\text{-}100 : Aircraft\} \quad (43)$$

Figure 6: DesO on A320-100

$$\mathcal{T}_{ModO^*} ::= \mathcal{T}_{ModO} \cup \mathcal{T}_{DesO} \quad (44)$$
$$\mathcal{A}_{ModO^*} ::= \mathcal{A}_{ModO} \cup \{$$
$$geom.fuselage.height : DistanceParameter\} \quad (45)$$

Figure 7: Combined ModO and DesO on A320-100

3. THE CHALLENGES

There are two main types of challenges in combining DOCO, MODO and DESO. On the one hand, there are challenges of

[2]https://github.com/astbhltum/Aircraft-Ontology
[3]http://www.w3.org/2005/Incubator/ssn/ssnx/qu/qu
[4]http://www.w3.org/2005/Incubator/ssn/ssnx/qu/qu-rec20

ontology integration: given ontologies, each separately representing knowledge relevant to the aviation sector, how can they be combined? As mentioned, their differences should be resolved in a way that preserves each component's representational requirements. On the other hand, there are challenges of coverage: do the integrated ontologies adequately represent the aviation sector or should their content be enriched?

3.1 Ontology Integration

DocO, ModO and DesO need to be integrated in two ways: terminology alignments should be found by means of ontology matching techniques [8], difference in abstraction levels should be resolved by meta-modeling [7, 6].

3.1.1 Matching

DocO contains individuals for measurement-related notions, although not organized in any structure. On the other hand, the measurement-related modules of ModO and DesO largely overlap as apparent in the similarities between Axioms (27) and (38), or Axioms (28) and (39), or Axioms (29) and (42), all of which make Axiom (45) plausible. The challenge is to find a general approach to resolve the modeling differences between ModO and DesO. Without such alignment between $\mathcal{A}_{ModO}$ and $\mathcal{T}_{DesO}$, the consequences of classifying *geom.fuselage.height* as a *DistanceParameter* cannot be tested by a reasoner based on individual's and class' properties, thereby limiting the main feature of ontological modeling.

3.1.2 Meta-modeling

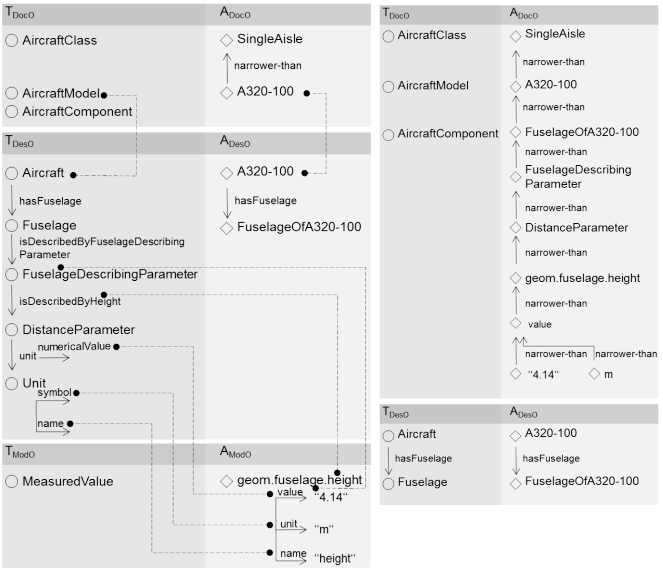
DocO contains meta-classifications. Axiom (8) asserts an individual aircraft model (class introduced in Axiom (1)). Axiom (9) asserts an individual aircraft class (class introduced in Axiom (4)). Yet, an A320-100 is often classified as an instance of (in owl: an individual of class) single-aisle. This would require to assert in dl a higher-order axiom like the following: $A320\text{-}100 : SingleAisle : AircraftClass$.

Of course, that is not possible, as an individual in dl (a fragment of first-order logic) cannot on its turn classify other individuals. DocO mimics such encapsulated classification between individuals by asserting a *narrower-than* relation, as in combined Axioms (9) and (12).

3.2 Ontology Coverage Assessment

While the ontology resulting from the integration of the ontologies shown in Figure 2 would include many notions relevant to aviation, they would miss modules usually included in multi-disciplinary engineering ontologies.

One way of assessing ontology coverage is to compare a given group of ontological modules with benchmarks proposed in the relevant ontological literature. For instance, [10] discusses the range of notions comprised in such multidisciplinary engineering domains. According to this proposal, DesO misses conceptualizations for: physical objects (though implied by part-of relationships between components), functionality (partly implicit in aircraft aspects), processes and materials. Also, DocO contains many individuals representing agents (e.g., persons, organizations). DesO does not provide any conceptualization of agents, given its focus on preliminary design. For a wider scope, though, at least one agent may become relevant: the pilot.



(a) Matching **(b) Meta-modeling**

Figure 8: Target results of integration

Automatic or human, the pilot has control functions that require agent-like properties. Finally, as described in [4], automated alternatives to literature-surveys have been proposed. Such techniques measure semantic similarity against a standard ontology, or against a relevant corpus or a thesaurus, or measure the fitness of an application that embeds the to-be-evaluated ontology to accomplish a certain task (e.g. answering competency questions relevant to the goals of the ontology development).

4. FUTURE WORK

The system being developed based on the proposed architecture attempts to tackle the challenges described above by providing operational definitions of the arrows References and Mappings in Figure 1. This should result in the following operations.

Match: To integrate taxonomical structures and parametric data, alignments between DocO, DesO and ModO need to be established. Recommender systems are being tested to support this operation, which present complexities, for instance when matching ModO with DesO. As shown in Figure 8a, the former entangles in a single individual the notions of: distance parameter, measure of height, fuselage parameter; the latter, on the other hand, separates these notions. The matching mechanism may need to be complemented with the creation of individuals or property assertions (e.g. between *FuselageDescribingParameter* and *geom.fuselage.height*).

Meta-model: To make available for DocO the result of the match operation at the appropriate level of abstraction, relevant classes in DesO need to be modeled as individuals, i.e. meta-modeled or, more specifically, reified in DocO. As shown in Figure 8b, the resulting version of $\mathcal{A}_{DesO}$ would contain the same knowledge as the three original matched ontologies, although extra classes would be added to classify the meta-modeled (reified) classes.

Assess Coverage: To estimate to which extent the matched DocO, DesO, ModO contain the terminology found in a corpus and point out missing notions, coverage assessment technique will be tested focusing on the automated testing of mereological and functional properties.

5. DISCUSSION

This section provides a preliminary methodological discussion of how knowledge management architectures, such as the one presented above, should be positioned with respect to the wider research area of *Industrie 4.0*, in particular with respect to subareas that involve Linked Data and Robotics applications.

In our present working definition of the relationships between the different research-areas that contribute to the vision of *Industrie 4.0* we are assuming a fairly rigid partition between:

Pre-production processes (or work phases) which, for the most part, are based on intellectual or experimental activities (i.e. the part of the process chain from Conceptual Design to Prototyping).

Production processes (or work phases) which have at their core physical activities or transformations (i.e. the part of the process chain from Mass Production and Assembly to Quality Assurance).

Questions about production processes are investigated in the subarea of *Industrie 4.0* usually referred to as Smart Factory. Here Robotics plays a central role, as a means to reduce production costs by more efficient and effective adjustment of production lines. Alongside Robotics, Cyber-physical Systems and the Internet of Things are key-ingredients in achieving interoperability and decentralization on the floor of the Smart Factory.

On the other hand, questions about pre-production processes are investigated in a subarea of *Industrie 4.0* that, by analogy, could be called Smart Studio. Here Robotics plays, if any, a less important role, whereas Knowledge Management and Artificial Intelligence are more prominent. The architecture presented in Figure 1, contributes to achieving interoperability in the Smart Studio.

In this context Linked Data, i.e. the result of interlinking structured data coming from different sources (as proposed in 2006 by Tim Berners-Lee[5] or in [3]) play an important role in achieving research goals either within the Smart Factory or within to Smart Studio separately, because requirements and models within each of these research areas are sufficiently homogeneous.

What still needs to be clarified, though, is the extent to which Linked Data (or any other integration approach) can deliver results across the Smart Factory and the Smart Studio. Is it possible to blur the distinction between data used or generated during pre-production and data used or generated during production?

Ideally, (i) knowledge that is gained during the production of a product would be fed back to previous phases (e.g. the design of a new version of that same product): such feedback loop would allow to modify the design of a product based on data generated during its production or its quality assessment; (ii) conversely, knowledge that is gained during the pre-production of a product may be fed forward to later phases (e.g. scheduling or configuration): such feed–forward loop would directly impact the production line and the robots operating in it.

Also, the increasing role of virtualization is an additional motivation for researching if it is possible to blur the distinction between data generated in the Smart Factory and in the Smart Studio. On the one hand, the Smart Factory needs virtual representations of physical products for new production techniques (e.g. 3D Printing). On the other hand, the Smart Studio applies virtual design approaches (e.g. virtual testing or hardware-in-the-loop testing). As both the Smart Factory and the Smart Studio will increasingly be working on virtual representations of the same final physical product, linking the data underlying those representations would translate into increased agility throughout the product lifecycle.

As a final (counter)point on what discussed in this Section, it should be noted that the possibility of blurring the distinction between data generated in the Smart Factory and in the Smart Studio, even if eventually viable, may not be unconditionally welcome by practitioners. The aviation industry, for instance, is subject to the strictest design certification requirements, which entail long and costly certification procedures. As a result, designs in Aviation are rather stable in time. Therefore, the advantages of a feedback loop, such as (i) above, may not be obvious, because putting effort into an automated Knowledge Management infrastructure to achieve fine-grained changes to product designs based on feedback from the production line may not be considered cost-effective.

[5]W3C recommendation https://www.w3.org/DesignIssues/LinkedData.html

6. ACKNOWLEDGMENTS

Research supported by German program LUFOV2, project EFFPRO 4.0, grant no. 20Y1509E.

7. REFERENCES

[1] M. Ast, M. Glas, and T. Roehm. Creating an ontology for aircraft design. In *Deutscher Luft- und Raumfahrtkongress 2013, Stuttgart*, 2013.

[2] S. Biffl and M. Sabou, editors. *Semantic Web Technologies for Intelligent Engineering Applications*. Springer, 2016.

[3] C. Bizer. The emerging web of linked data. *IEEE Intelligent Systems*, 24(5):87–92, 2009.

[4] N. DiGiuseppe, L. C. Pouchard, and N. F. Noy. SWEET ontology coverage for earth system sciences. *Earth Science Informatics*, 7(4):249–264, 2014.

[5] M. Glas. *Ontology-based Model Integration for the Conceptual Design of Aircraft*. Dissertation, Technische Universität München, München, 2013.

[6] B. Glimm, S. Rudolph, and J. Völker. Integrated metamodeling and diagnosis in OWL 2. In P. F. Patel-Schneider, Y. Pan, P. Hitzler, P. Mika, L. Zhang, J. Z. Pan, I. Horrocks, and B. Glimm, editors, *International Semantic Web Conference, Revised Selected Papers, Part I*, volume 6496 of *Lecture Notes in Computer Science*, pages 257–272. Springer, 2010.

[7] N. Jekjantuk, J. Z. Pan, and Y. Qu. Diagnosis of software models with multiple levels of abstraction using ontological metamodeling. In *Proceedings of the*

International Computer Software and Applications Conference, pages 239–244. IEEE Computer Society, 2011.

[8] O. Kovalenko and J. Euzenat. Semantic matching of engineering data structures. In Biffl and Sabou [2], pages 137–157.

[9] G. La Rocca. Knowledge based engineering: Between AI and CAD. review of a language based technology to support engineering design. *Advanced Engineering Informatics*, 26(2):159–179, 2012.

[10] C. Legat, C. Seitz, S. Lamparter, and S. Feldmann. Semantics to the shop floor: towards ontology modularization and reuse in the automation domain. *IFAC Proceedings Volumes*, 47(3):3444–3449, 2014.

[11] T. Moser. The engineering knowledge base approach. In Biffl and Sabou [2], pages 85–103.