Prognostic Health Ontology

Richard de Rozario [1], Paul Marsden [2]

[1] *The Hunt Laboratory for Intelligence Research, University of Melbourne, Parkville, Victoria, 3010, Australia*

[2] *Defence Science and Technology Group, 506 Lorimer Street, Fishermans Bend, Victoria, 3207, Australia*

Abstract

Aircraft prognostic health maintenance is underpinned by a complex regime of data collection and analytics across heterogeneous systems that perform at very different spatiotemporal scales. As such, aircraft health maintenance presents a significant challenge of analytical integration. We present a case study of the development of an ontology for aircraft health. A simple view of an ontology is as a comprehensive and coherent collection of meta-data. This meta-data describes the entities and relations that exist in a particular domain of analysis. An ontology supports questions such as "What data is available about X?", "Is the information in system X comparable to system Y?", "What is the precise meaning of result Z?", etc. An ontology provides a shared understanding of a domain and supports analytical integration. The case study results provide a foundation for information integration in aircraft maintenance, as well as insights into the practice and tools of information integration.

Keywords: prognostic health, ontology, system integration, aircraft maintenance

Introduction

Aircraft prognostic health maintenance is underpinned by a complex regime of data collection and analytics across heterogeneous systems. These systems also perform at very different spatiotemporal scales. As such, aircraft health maintenance presents a significant challenge of analytical integration. In this paper, we present a case study of the development of an *ontology* for aircraft health.

Applied ontology (as a practice) has been an active field of both research and industrial practice since the 1990s. An early example of this work is the Enterprise Ontology [1], a document and software model, which provided terms and definitions for enterprise modelling. Another prominent example is the work started in the early 2000s by Barry Smith's research group to create an ontology for biomedical research [2].

A simple view of an ontology is as a comprehensive and coherent collection of meta-data. This meta-data describes the entities and relations that exist in a particular domain of analysis. Using an ontology makes it easier to questions such as "What data is available about X?", "Is the information in system X comparable to system Y?", "What is the precise meaning of result Z?", etc.

An ontology typically takes the form of a document or electronic file with a structure that is readable by both humans and computers. The file mostly comprises definitions of "entities" and properties of entities (including relational properties between entities). An "entity" is a very general idea and can be concrete objects, or processes, or even an abstract concept. For example, in an ontology we might define that some number in a file is the *value* of a Compressor Temperature Variable, which *measures* the Temperature *aspect of* a Compressor and *uses units* Degrees Centigrade, where a Compressor *is part of* a Jet Engine, which is a *part of* an Aircraft of some *type*, which in turn is *part of* a Military Command, and so forth[1]. The consistent, logically coherent, and comprehensive "tagging" of information in this way enables several tasks:

- Finding information, both interactively as well as automated, even if we don't know the terminology (e.g. field headings) used in a particular source of data. For example, we could formulate a search for "*sources that contain data about the temperature of parts of an aircraft*" – without knowing the particulars of what the parts might be called in the source system, or what kind of data there is, or how the data is structured. Such "relational" searches are difficult to do with general language search engines or structured database queries unless the appropriate meta-data is available.
- Formulating Information Requirements. By querying the ontology, we may find that sources for particular information are not available. Conversely, we may be able to infer from the ontology that information in some system can be re-used for other purposes. The ontology effectively provides a *model* of information and systems, which supports requirements analysis. As models, ontologies also support testing whether systems are consistent with underlying intended concepts.
- Documentation and integration at "human scale". From the beginning, applied ontology work has incorporated "human readability" into its design. The idea was that a vital part of systems integration was the ability of people to understand diverse and complex systems – especially when trying to integrate into systems that they might not be familiar with.
- Data analytics and modelling. Although an ontology is not typically "executable software", unlike, say, an app or computational model, the typical ontology format is flexible enough that computational elements can be included. Some computational elements are already routinely included, such as logic rules for properties. This means that ontologies can support analytics and modelling of information systems. For example, one could build in transformation rules that enable queries to transform results from one structure or dimension into another. A simple example would be converting units like Fahrenheit to Celsius. A more complex example would be the use of ontologies to support analysis of modularization (e.g. "factoring") of systems.

An advantage of an ontology over other meta-information efforts of the past (e.g. "data dictionaries" or "taxonomies") is that the ontology aims at coherence starting from a deep foundational level – i.e. the level of fundamental elements and properties that are studied in science and philosophy. As such, ontologies typically include more detailed properties than just the "list" and "sub-categories" in, say, taxonomies. This foundational layer of ontologies bolsters the coherence and consistency in the data structures across domains. Insofar as the

[1] In the example, we have underlined Entities and italicized (relational) properties

foundational elements of an ontology are truly foundational, an ontology can also be expected to be more stable (i.e. have longevity) as a means of categorizing information.

The research presented here reflects the effort to build an ontology for the domain of (military) aircraft health maintenance. Although the ontology is limited to a specific domain of practice, the work is situated in the context of other ontology work — in particular, the Basic Formal Ontology [3] and the Common Core Ontology [4] and its extension into the military domain via the Joint Doctrine Ontology [5].

Methodology

Our methodology for developing the core ontology, broadly similar to Grüninger and Fox [6], Noy and McGuinness [7] and Arp *et al* [3, Ch. 3–4], is driven by questions of scope and usage, metrics and tests of performance, and ultimately the iterative definition of terms.

Broadly speaking, the methodology proceeds through interviews and document reviews as follows:

1. Determine the scope and use of the ontology. Here we establish the intended domain of the ontology and the priority of types of uses (e.g. "knowledge sharing among people"). We also identify technical constraints (e.g. desired representation format) and sources of domain terminology. Also identified here are the general ontological principles (e.g. representative realism) and links to other ontologies.
2. Next, we identify ways to measure the quality of the ontology. Here we borrow the idea of "competency questions" [8] – that is, questions and propositional sentences that should be retrievable or answerable with the ontology. Other possible metrics include the extent of terms from sources that are defined in the ontology, usage metrics, coherence testing, and compliance with technical restrictions.
3. The next major step starts with the collection of domain terminology, establishing the class hierarchy, linking to existing ontological terms where possible, and defining properties and axioms. We follow as much as possible conventions described in Arp *et al* [3, Ch. 3–4] such as use of singular nouns, rather than mass nouns. Also, where possible, testing happens in tandem with ontology development. That is, sets of instances (i.e. example items that fit the ontology) for testing purposes are identified or developed together with the definitions and structures of the ontology itself, as well as any scripts or procedures for testing.
4. Lastly, we iteratively deepen the level of formality, starting from informal definitions, questions and examples to formal definitions, queries and data.

The target format of the ontology file(s) was OWL 2 Web Ontology Language [9] – in particular, using the Turtle syntax [10] for simplicity and readability. However, during the project it became apparent that even the Turtle syntax was too complicated for human readability and efficient authoring, and the project switched to a simple outline structure for the ontology, based on the YAML [11] syntax. This syntax is another widely used format for information interchange, designed specifically to be more readable than the XML based syntax (which OWL2 uses). The YAML based structure was used to edit and maintain the ontology and was transformed into OWL2 for delivery. The transformation into OWL2 was also used as a structural validity test for the ontology.

The following tools were used for constructing the ontology:

- The Protégé ontology editor [12], desktop version – a freely available, open source editor developed at Stanford University. This is the most common tool for editing and basic querying of an ontology. It also has various add-on modules, such as OntoGraph to generate basic visualisations of the ontology.

- Standard desktop tools such as text editors (e.g. Microsoft VS Code) and spreadsheets (MS Excel). The Protégé editor is not very efficient for dealing with bulk changes to the ontology, such as transforming information from sources such as tables, or restructuring entire sections of an ontology. For this, it is often more efficient to either edit or construct the ontology file in a text editor using sophisticated search and replace features.

- The Prolog programming language for queries and utilities. Prolog is a commonly used language for AI and knowledge-based systems and has many libraries that support ontology work. Prolog is essentially a sophisticated query language. The version used for the project was SWI-Prolog [13], a widely used, open-source version developed at the University of Amsterdam.

Results and Discussion

The basic ontology

The initial discussions and exploration into the task of building a prognostic health ontology (PHO) for aircraft maintenance focused on dealing with the basic information and data that flows through the systems and processes. To start with, the ontology needed to be able to answer the simplest competency question: "what is the meaning of X?", where X is any basic entity in the domain of aircraft maintenance.

To this end, the team started with the most obvious terms, like "Aircraft", "Engine", "Helicopter", etc., as well as lists of data types that are typically processed in various parts of the aircraft and maintenance systems. We also reviewed literature on other ontology projects that were related to our domain. Of particular relevance in this regard were the Common Core Ontology [4] and the Basic Formal Ontology [11] on which it is built. The creators of the Common Core Ontology had also done partial work on Aircraft Maintenance for the United States Air Force (USAF), which they shared with us.

The initial terms focused on the structure of Jet Engine(s) and various flight performance data that related to engines. In the ontology, a term like "Compressor" represents the *type* (a "universal" in BFO terminology) of object that exists in the world. The meaning of "Compressor" is given in various ways in the ontology. Firstly, there is a definition given in ordinary language: "A jet engine component that increases the pressure of a gas by reducing its volume". The definition uses the pattern recommended in common ontology guidance[2]: an entity is defined as being of a certain *type*, with some detail (some essential aspect) that distinguishes it from other entities of that type. We also annotated the terms in the ontology with references to other ontologies (mainly BFO and the Common Core Ontology), as well as

[2] This format of definitions (genus-differentia) goes at least as far back as Aristotle's ontology work.

references to the Defence Aviation Safety Requirements Standard Numbering (DASRSN). In addition, the glossary of the DASRSN was itself extracted and formatted as an ontology to support text based searches for definitions and descriptions.

The common language definition enables the use of the ontology as a glossary – a simple but often effective information use. Moreover, the particular structure of the definitions is also reflected in the machine readable structure. That is, the ontology includes the following structure:

```
- Compressor:
   - type of:
      - Jet Engine Component
```

Or in OWL2 Turtle syntax:

```
<Compressor> rdfs:subClassOf <Jet_Engine_Component>
```

This machine readable structure of the definitions in the ontology enables software to make use of relationships between entities. In effect, the software *knows* that Compressor is a *type of* Jet Engine Component. As far as software systems are concerned, an entity is defined in terms of its relational structure to other entities. These relational definitions enable a very useful type of searching for information, whereby we can make use of generalisations. For example, we may want to search for parts of an aircraft that are monitored for temperature. The following query code would be one form of such a search (ignore the technical syntax, which is only displayed to convey a sense of length and technical simplicity).

```
sub(pho,'Aircraft',types,Aircraft),
full_content(pho,Aircraft,Prop,Part),
sub(pho,Temp,'aspect of',Part),
sub(pho,'Temperature',types,Temp).
```

The query simply asks for any type of Aircraft, any parts or inclusions of the aircraft, and any aspect of such a part that is a type of temperature. The results include for instance, the temperature of Static Air, the temperature of Gas in the Compressor, and the temperature of the Oil Cooler Outlet. Note that the particular query example makes use of some pre-defined helper functions such as **`sub`**, which searches for sub-entities and full content, which includes not only parts but also material that is not a part as such, but is contained within a part (e.g. gas in a compressor).

The ontology includes information about the relational properties that connect entities. For example, it is specified that the relational properties, like *type of*, have certain characteristics like *transitivity* (meaning, if A is a type of B, and B is a type of C, then A is also a type of C). This enables the software to find parts of an aircraft, even if they are several levels down in terms of sub-types. In other words, it enables us to ask questions in very general terms, and still get all the detail. Of course, it does mean that the user needs to know (or needs to be supported by the query software) about the relational properties that are typically used in an ontology. Some of the most prominent properties are:

- *type of* (also called ‘class’ in OWL and ‘universal’ in BFO) where, for example, Jet Fighter is a type of Aircraft.

- *part of* – for example, a Compressor is part of a Jet Engine
- *includes* – for example, an Oil Tank includes Oil
- *records* – used to convey the relationship between data and the types of real-world entities they represent. For example, a data field like trans_oil_temp_degC records the Temperature of Oil in Gearbox.
- *aspect of* – for example, Temperature is an aspect of the Oil in Gearbox

At last count, the PHO ontology contains close to 3,000 relational entries that describe close to 1,300 entities. Figure 1 shows the high level outline of the ontology.

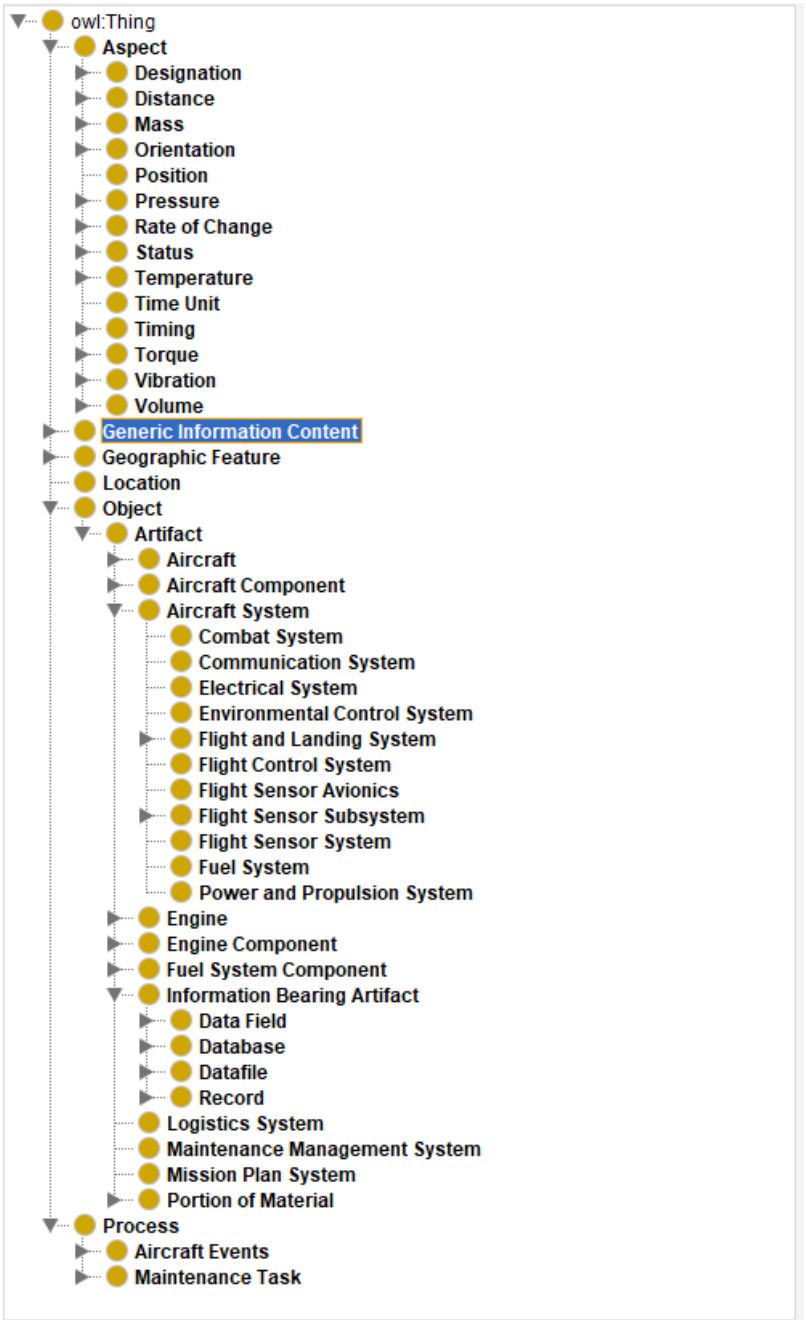

Figure 1-- high level outline of the Prognostic Health Ontology

Competency of the Ontology

The ontology developed so far, functions well as a meta-database. That is, one can answer (in the query language) general systems integration questions such as:

1. What is the meaning of X?
2. What sources of information (data) are available about X?
3. Does X have the same properties as Y?
4. What, if anything, is the relational connection between X and Y?

A caveat in the competency of the ontology is that the sources of information need to be "tagged" with meta data. For example, in order for the ontology to know about, say, files of flight data, we need to describe the fields of those files in the ontology. This may be a laborious task when done in bulk, but is reasonably efficient when done as an integral part of system design and development. For instance, most tabular data files of measurement data follow a similar structure, where each field can be described in terms of some aspect of a system component, with measurement units. For example, we might provide a definition of the number of ground starts of an aircraft as follows:

```
- Number of Ground Starts:
   - type of:
     - Flight Data Field
   - uses unit:
     - Ground Starts
   - records:
     - Aircraft Ground Starts:
       - type of:
         - Status
       - aspect of:
         - Jet Engine
```

The advantage of systematically curating data sources in this ontological way is that it becomes possible to assess how one piece of data compares to another. The upfront cost of this effort needs to be weighed against the longer term benefit of effectiveness of systems integration.

With regards to system integration, we also examined how the ontology might support conceptually broader questions. Of particular interest was a question that seemed to underpin most prognostic health user requirements, namely:

- Will aircraft X be mission ready in time?

With regards to maintenance, the mission-readiness question boils down to questions about maintenance planning, such as "what maintenance is scheduled (or planned, or predicted)?". Insofar as maintenance management systems are available with ready-made data about plans and schedules, it is a relatively simple task to extend the ontology across these systems such that the mission-readiness question can be shown to be answerable. However, in this context, it becomes clear that the linchpins for prognostic health maintenance are the prognostic analytical devices such as "remaining useful life" (RUL) models. In other words, if we want the ontology to determine if questions are answerable about the optimal time to perform maintenance tasks,

then we need to incorporate descriptions of prognostic models (and how they relate to maintenance plans).

As a first foray into an ontology of prognostic models, we extended the ontology with an example of a model of RUL of lithium batteries [14]. Much of this ontology development was equivalent to the work described above. The salient differences for prognostic models were the emphasis on the ontology of the *indicators* (i.e. input variables) and output variables that such models are based on. For example, for the particular prognostic model of lithium batteries we looked at, the following aspects were needed:

```
- Capacity:
  - {definition: "An aspect that is the maximum amount
         of something that an object can receive or contain"}
- Battery Health Indicator:
  - {definition: "An aspect that is some performance
               aspect of a battery}
  - H1:
    - {definition: "A battery health indicator that is the charge
         time interval of voltage varying from 3.9V to 4.2V"}
  - H2:
    - {definition: "A battery health indicator that is the charge
         voltage varying from 3.9V to the voltage after 500sec"}
  - H3:
    - {definition: "A battery health indicator that is the charge
         current drop between 1.5A and the current after 1000s"}
- Remaining Useful Life:
  - {definition: "An aspect that is the duration (cyclic or
         calendar) that an artifact is expected to be able to
         perform its function"}
```

These aspects then needed further definitions of measurement units (e.g. “Charge Discharge Cycles”). Additionally, the concept of a Model itself needed to be defined, as well as some type of entity that *uses* the model to create maintenance tasks, in order to provide a logical connection between prognostics and maintenance planning. In our example, we stipulated a hypothetical Prognostic Monitor that compares aircraft performance data with a predictive model, in order to suggest planned maintenance.

Our current work in the ontology is to generalise the example of the prognostic model ontology, to accommodate a broad variety of prognostic devices and integrate their characteristics into the ontology.

Conclusion

Overall, the development of the Prognostic Health Ontology was a worthwhile research exercise that revealed how such an ontology could be a useful tool for integrating systems and data across the domain of aircraft maintenance. In a sense, such an ontology is never really “finished”, because more detail can always be added and the scope of the ontology can always be broadened. In particular, generalization of the links between prognostic models and maintenance planning is needed to “complete” the prognostic health ontology. This work is currently ongoing.

Acknowledgements

This project was conducted in conjunction with the Aerospace Division of the Defence Science and Technology Group.

References

[1] M. Uschold, M. King, S. Moralee, and Y. Zorgios, "The enterprise ontology," *Knowl. Eng. Rev.*, vol. 13, no. 1, pp. 31–89, 1998.

[2] P. Grenon, B. Smith, and L. Goldberg, "Biodynamic ontology: applying BFO in the biomedical domain," *Stud. Health Technol. Inform.*, pp. 20–38, 2004.

[3] R. Arp, B. Smith, and A. D. Spear, *Building ontologies with basic formal ontology*. MIT Press, 2015.

[4] Rudnicki, Ron, "Modeling Information with the Common Core Ontologies," CUBRC Inc., Buffalo, NY, USA, 2017.

[5] P. Morosoff, R. Rudnicki, J. Bryant, R. Farrell, and B. Smith, "Joint Doctrine Ontology: A Benchmark for Military Information Systems Interoperability," in *Semantic Technology for Intelligence, Defense and Security (STIDS)*, CEUR vol. 1325, 2015, pp. 2–9.

[6] M. Grüninger and M. S. Fox, "Methodology for the design and evaluation of ontologies," 1995.

[7] N. F. Noy, D. L. McGuinness, and others, *Ontology development 101: A guide to creating your first ontology*. Stanford knowledge systems laboratory technical report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880, March 2001.

[8] N. F. Noy and C. D. Hafner, "The state of the art in ontology design: A survey and comparative review," *AI Mag.*, vol. 18, no. 3, pp. 53–53, 1997.

[9] B. Motik *et al.*, "OWL 2 web ontology language: Structural specification and functional-style syntax," *W3C Recomm.*, vol. 27, no. 65, p. 159, 2009.

[10] Jie Bao, E. Kendall, D. L. McGuinness, and P. F. Patel-Schneider, "OWL 2 Web Ontology Language -- Quick Reference Guide (Second Edition)." W3C, 2009. Accessed: Dec. 30, 2020. [Online]. Available: https://www.w3.org/2007/OWL/wiki/Quick_Reference_Guide

[11] Ruttenberg, Alan, "BFO - Basic Formal Ontology." 2019. Accessed: Mar. 19, 2019. [Online]. Available: http://basic-formal-ontology.org/

[12] "Protege Desktop."

[13] J. Wielemaker, T. Schrijvers, M. Triska, and T. Lager, "SWI-Prolog," *Theory Pract. Log. Program.*, vol. 12, no. 1–2, pp. 67–96, 2012.

[14] J. Liu and Z. Chen, “Remaining useful life prediction of lithium-ion batteries based on health indicator and Gaussian process regression model,” *Ieee Access*, vol. 7, pp. 39474–39484, 2019.