Applied Ontology 18 (2023) 169–206
DOI 10.3233/AO-230279
IOS Press

An ontology for maintenance procedure documentation

Caitlin Woods *, Tim French, Melinda Hodkiewicz and Tyler Bikaun
Department of Computer Science and Software Engineering, The University of Western Australia, Western Australia, Australia
E-mail: caitlin.woods@uwa.edu.au

Abstract. In mining, manufacturing and industrial process industries, maintenance procedures are used as an aid to guide technicians through complex manual tasks. These procedures are not machine-readable, and cannot support reasoning in digitally integrated manufacturing systems. Procedure documents contain unstructured text and are stored in a variety of formats. The aim of this work is to query information held in real industrial maintenance procedures. To achieve this, we develop an ontology for maintenance procedures using the OWL 2 description language. We leverage classes and object properties from the ISO 15926 Part 14 Upper Ontology and create a domain ontology. The key contribution of this paper is a demonstration of trade-offs required when modelling an existing engineering artifact, where an abstraction of its contents is given a-priori. We provide an ontologically rigorous abstraction of notions captured in procedure documentation to a set of classes, relations and axioms that allow reasoning over the contents. Validation of the ontology is performed via a series of competency questions based on queries relevant to technicians, engineers and schedulers in industry. The ontology is applied to real world maintenance procedures from two industrial organisations.

Keywords: Industrial ontology, maintenance, procedure, Industrial use case

Accepted by: Emilio Sanfilippo

1. Introduction

Engineers in asset-intensive organisations create maintenance procedure documentation to assist technicians in complex manual tasks such as equipment inspections and servicing. These procedures are critical in ensuring the safe and effective execution of maintenance work. Kanse et al. (2018) reported that procedures perceived as logical, at the right technical level, and easy for technicians to read can increase procedure compliance. However, procedures that are out of date, arduous, or contain unnecessary steps can lead to a failure to comply, contributing to workplace accidents. For these reasons, organisations must be attuned to the content and presentation of maintenance procedure documentation.

Procedures are typically developed using prescribed templates in tools such as Microsoft Word or Excel. Regardless of the software used to write them, they are generally stored as PDFs. These templates tend to change over time as organisational processes and policies change. For example, legacy documentation may not contain a "comments" field for technicians to provide feedback about the quality of the procedure, whereas new documentation may include this field. Organisations develop new templates over time and old procedures are rarely updated. Given the diversity in how PDF documents are structured, it is arduous and costly for an engineer to perform updates across the dataset and it is difficult

*Corresponding author. E-mail: caitlin.woods@uwa.edu.au.

to extract this information into a machine-readable format. To further complicate matters, maintenance procedure documents vary in content, structure and style, as engineers write procedure documentation in different ways. For example one engineer may assume that technicians have a certain level of knowledge about a task and may not choose to list *all* tools that are required for a task. A different engineer may not have made this assumption and may choose to include common tools such as "spanners". To query data from procedures that are stored in unstructured formats we must be able to map data to appropriate classes regardless of its representation in the document. In addition, we need to recognise different levels of detail within the procedure texts, and manage situations when data is absent from a procedure document.

Industrial organisations are transitioning towards Industry 4.0 (Lee et al., 2015). This means that they are developing cyber-physical maintenance and production systems. Data, including PDF-based maintenance procedures, currently collected and stored by organisations, needs to be transformed into meaningful information to advise these cyber-physical systems (Lee et al., 2015). Therefore, organisations will require "formal" (i.e. machine-readable) processes to store and access procedure information that is both "explicit" (i.e. understood the same way by individuals and software systems) and "shared" (i.e. used consistently between teams and software systems).

Ontologies, "a *formal*, *explicit* specification of a *shared* conceptualization of concepts within a domain" (Studer et al., 1998), can satisfy this requirement. The benefit of supplementing data with an ontology is threefold. First, ontologies are a controlled vocabulary used to define specific concepts that are consistent within and, perhaps, across organisations. Second, being "formal, and explicit", ontologies improve data interoperability and reduce data governance overheads. Finally, ontologies are "machine-interpretable". They capture the semantics of data and can infer new information using sophisticated logical reasoning techniques (Sirin et al., 2007; Glimm et al., 2014).

The challenge addressed in this paper is the application of ontologies to query data contained in procedure documentation, at an appropriate level of abstraction. We first discuss how procedure documents are currently used in industry (Section 2) and then examine relevant past works (Section 3). Conforming to suggestions by Ferrario and Grüninger (Ferrario and Grüninger, 2020), we provide ontological choices (Section 4) and implementation details (Section 5) for an *ontology for maintenance procedure documentation* (OMPD). This discussion includes the conformance of OMPD to the ISO 15926 Part 14 upper ontology (International Organization for Standardization, 2019). Our evaluation of OMPD (Section 6) includes executing SPARQL queries that answer each of the presented competency questions and mapping data from two diverse organisations to the procedure ontology. We finish the paper with a concept-level comparison of our ontology against three similar ontologies, a discussion of abstractions presented in this ontology, and considerations for the applied ontology community (Section 7). Finally, we and discuss suggestions for future work (Section 8).

2. Background

Maintenance procedures are instructions for technicians that describe how to perform a maintenance activity. Figures 1 and 2 show two procedure documents from the same continuous manufacturing (process) plant. The document shown in Fig. 1 describes a mechanical inspection procedure for a pump. The procedure contains metadata (i.e. reference documentation and documentation change history) as well as a series of tasks. Each task has a step number, job description, limits, required actions to complete the job and a blank box to fill in any corrective actions that the technician performs if these limits are not met. Tasks 2 and 3 in the procedure are represented using both and images and text.

WO number:	
Completed by:	
Date completed:	

2M Mech Insp Leak Detection Pumps [equipment ID & model ID omitted]

Reason for maintenance and relevant background information		
[reason for maintenance omitted]		
Tools, equipment and materials required		
Item/Material (Items/materials not already on task list	Un	Quantity
Reference documentation		
Document ref.	Description	
Specific known job hazards / Previous safety event lessons		
• Harm to persons by electrical, mechanical and chemical impacts • Guards of moving parts must not be removed when the pumps are running. [additional hazards omitted]		
Document history		
Date	Changes Made	By who
Approval		
Approval Date	Owner Role	Name
Approval Date	Reviewer Role	Name

If you feel the execution steps below do not meet the intent of this PM or is missing important steps/checks then please provide feedback to the relevant engineer.

Work Execution				
Task	Job Description	Limits	Required Action	Corrective Action Taken
1	Check that suction and discharge points are	Obstruction free	Check that suction & discharge valves are clear	
	clear of obstruction and supply water to pump suction.		Check that suction & discharge piping are clear Look for any piping deformation or restriction	
[schematic omitted]				
2	Check that the level in the leak detection tank [equipment id omitted [is enough to ensure the NPSH in the pump section.	Pump start up when setting level is reached	Fill the tank with storage water till the start-up setting level and check that the pump start up.	
[schematic omitted]				
[Tasks 3 to 9 omitted]				

In the event that you are unable to complete all the corrective work please escalate to your supervisor in order to agree a course of action.

Comments

Fig. 1. A two-month maintenance procedure from a Process Plant (identifying information omitted for proprietary reasons).

The procedure shown in Fig. 2 is more complex. This procedure has similar metadata fields at the start of the document, but has three different "work execution" tables that describe the task. The first table gives two tasks that should be performed *iteratively* for each of the equipment items listed in the second table. Finally, there is a "work completion" table to be completed once all of the equipment items have been inspected. Notice that in this procedure, multiple actions have been assigned the same task number, and there are multiple limits that link to those actions. Furthermore, the table titles are non-uniform with the second table containing the "equipment" and "comments fields".

There are generally three different job roles that involve work with maintenance procedure documentation. *Engineers* are responsible for creating and updating procedure documents. *Schedulers* determine when a procedure will be executed, based on their organisation's maintenance strategy, production goals and resource constraints. Finally, *Technicians* use procedure documents as an aid to guide them in their work. For each of these perspectives, we provide competency questions that test the ontology.

2.1. Engineer's perspective

Maintenance engineers create procedure documentation based on a maintenance strategy for a specific item of equipment. To identify a suitable maintenance strategy, engineers perform risk-management processes such as Reliability Centered Maintenance (RCM) (Moubray, 2001). RCM is a process to

1W Purge Check [area omitted]

Work order record			
Critical?	**Work order number:**	**Completed by:**	**Date:**
Yes			

Reason for maintenance and relevant background information		
[reason description omitted]		
Tools, equipment & materials required		
Item/Material (Items/materials not already on the task list)	**Un**	**Quantity**
General electrician tools (e.g. multimeter, screwdriver set, radio etc.)		**1**
Personal gas monitor		**1**
Critical information		
If unsure of anything, ask your Team Leader for clarification		
Specific know job hazards / Previous safety event lessons		
Hazard	Location	Control
Reference documentation		
Document ref.	Description	
Approval		
Approval Date	Owner Role	Name
Approval Date	Reviewer Role	Name
PRT history		
Date	Changes Made	By who

Work Procedures				
Work Execution				
Task	**Job Description**	**Limits**	**Required Action**	**Completed (tick) / Corrective Action Taken**
1.	Perform adequate risk assessment to understand work risks and implement controls. Confirm with [name of work group omitted] that work is safe to proceed.		Complete [name of risk assessment omitted] Inform [name of work group omitted]	☐
2.	Conduct the following steps for each instrument: - Check visual condition - Confirm purge is not blocked and flow reading bely 40% of full scale (where applicable) - Ensure both transmitter indications are within 5% of each other (where applicable). - Confirm field readings match [software system name omitted] readings	Severe damage, leaks, display not working, illegible labels. Blockage, Reading >40% of full scale >5% error >5% error	Rectify and defects. If cannot, raise subsequent notification.	☐

Work Execution				
Task	**Equipment**	Description	**Comments**	**Completed / Corrective action taken**
1.	[ID omitted]	[ID omitted] seal water accumulator level		☐
2.	[ID omitted]	Agitator Seal Water Make-up	Inspect regulator and ensure set point is the same as PV.	☐
[tasks 3-51 ommitted]				

Work Completion				
Task	**Job Description**	**Limits**	**Required action**	**Completed (tick) / Corrective Action Taken**
1.	Reinstate all equipment and inform [work group name omitted] of work completion	N/A	N/A	☐
2.	Provide feedback to [maintenance team] if you have identified any improvement opportunities	N/A	Record required update.	☐

Comments

Fig. 2. A one week maintenance procedure from a Process Plant (identifying information omitted for proprietary reasons).

identify which maintenance strategy is appropriate given the consequence of functional failure and the failure behaviour. The aim is to maintain function of the assets and manage risks of functional failure(s). Maintenance procedures are then developed for routine and repetitive maintenance activities associated with preventative and predictive maintenance strategies (Hodkiewicz et al., 2021b). These procedures describe how the work should be done.

Engineers who create maintenance procedures typically adopt pre-existing templates implemented by their team or organisation, usually in PDF format. If no suitable template exists, they may create their own. Upon completion, the procedure is uploaded into a Computerised Maintenance Management System (CMMS) ready to be used by maintenance schedulers. Given that the documents are stored as PDFs (or a similar format), cascading changes throughout procedures, such as updating regulatory or resourcing requirements, is a time-consuming exercise for engineers.

Competency questions asked by an engineer are as follows:

- There has been a change in the regulations and an existing permit needs to be modified. Which procedures use this permit and can I update the relevant procedures?
- I would like to know which procedures describe an end of life event for my equipment. Which of my procedures contain a “replacement” task (i.e. the replacement of tyres on a truck)?

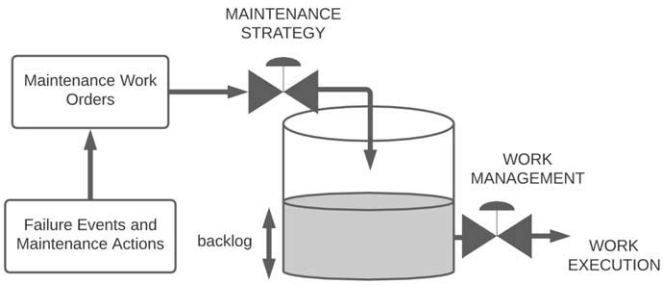


Fig. 3. An illustrative example of the maintenance work management process.

- Does my inspection procedure check all the failure modes outlined in the Failure Modes and Effects Analysis (FMEA) that was used in my RCM?

2.2. *Scheduler's perspective*

Maintenance schedulers decide when maintenance work is to be executed. Consider the analogy of maintenance work management as a tank system, shown in Fig. 3. Maintenance work orders are generated from both failure events and maintenance strategies. These work orders go into a backlog, depicted by the tank in Fig. 3. Schedulers determine which tasks from that backlog can be completed based on the availability of resources and the criticality of the work. To do this, schedulers need to know which resources are required to perform the task. Resources include qualified personnel, spare parts, tools and materials. This information is contained in maintenance procedures. However, when these procedures are in non-digital formats, it is difficult for schedulers to determine if the work being scheduled matches the available resources.

Competency questions asked by a scheduler are as follows:

- What resources do I require to execute the procedures used in maintenance work orders on next week's maintenance plan?

2.3. *The technician's perspective*

Technicians are the end users of procedure documentation. When technicians receive a work order, it typically has a maintenance procedure attached in the CMMS. The technician must open this procedure, print it and take it to their work location to use and annotate as they work. Once finished, they must scan the procedure and upload it back into the CMMS. This process is cumbersome for technicians. In a set of interviews that we performed with maintenance technicians in 2020 (Woods et al., 2021b), participants described situations where they could not find up-to-date procedures in the system and had to copy and re-copy information on to a paper-based procedure "three times" to complete their task. Participants also identified several benefits of having procedures presented in a digital format including the ability to view information in different presentation formats (Woods et al., 2021b). Digital representations of procedures provide much flexibility when presenting procedures to technicians. For example, we can explore adaptation of information that is presented to technicians based on their domain expertise. This is a core motivation for our team's exploration of the work presented in this paper.

Competency questions asked by a technicians are as follows:

- What tools, materials and permits do I require to execute a procedure?
- What steps need to be performed to execute my procedure?
- Given that I am up to task x, what task needs to be performed next?
- Does my assigned procedure have any safety hazards that I need to be aware of?
- What corrective action does my procedure suggest on observation of a failure mode in my inspection?

3. Past works

Since a procedure in maintenance can be viewed as a business process, it is relevant to describe this work and its relation to OMPD. In 2001, the graphical modelling language, UML, introduced a "business process" extension for business modelling (Sinogas et al., 2001). This allowed UML modellers to capture the relationships between *processes*, the *goals* of those processes and the *resources* that are used and consumed by those processes. In 2004, the Business Process Modelling Notation (BPMN) was developed (White, 2004). BPMN further captures the *events* and *gateways* that are likely to effect the sequence of a process. Both of these modelling languages are used widely today. However, being graphical languages, their primary objective is to give a human-readable representation of a process rather than supporting computational queries over a set of processes (as required by the competency questions given in Section 2).

In contrast, the Process Specification Language (PSL) (Gruninger and Menzel, 2003) was developed with a specific focus on supporting interoperability between software applications in the manufacturing domain. PSL is an upper ontology designed to represent the relationship between activities and their *occurrences*. Since our aim is to model procedure information that is currently stored in documentation, activity *occurrences* are out of OMPD's scope (discussed in Section 4.3). Instead, we have decided to use ISO 15926 Part 14 as an upper ontology. Our reasons for this are discussed in Section 4.1.

Maintenance procedure documentation shares many similarities with other types of procedure documents such as recipes used for cooking. Both maintenance procedure documentation and recipes contain a set of task descriptions and a list of resources (i.e. ingredients in recipes) that are required for the task. However, ontologies built for recipes tend to be too specific to meet the industrial needs of maintenance engineers, schedulers and technicians. For example (Hitzler and Krisnadhi, 2018) does not focus on the steps involved in the recipe. Rather this ontology describes the type of food produced and the nutritional information of the food contained in a recipe. Ribero et. al's (Ribeiro et al., 2006) ontology does raise interesting ideas about tasks in a recipe being either *ordered* and *optional*. However, the ontology contains classes such as *food* and the ontology uses reasoning to classify recipes based on characteristics such as *spiciness*. Recipe ontologies are a nice working abstraction of procedures in general. However, the reasoning examples presented in these ontologies focus on classification of recipes into categories that cannot be directly adapted to industrial maintenance procedures.

A closer domain to our industrial domain is the surgical domain. Similar to maintenance procedures, surgical processes involve tasks, actors and tools. An initiative called OntoSPM collaborative action is a combined effort from the surgical community to create ontological process models for their domain (Gibaud et al., 2018). The group has produced an ontology, OntoSPM, that is aligned to BFO and was last updated in 2019 (OntoSPM Collaborative Action, 2019). This ontology has some relevant concepts such as procedure *stages*, *phases* and *steps*. However, no assertions have been made about how steps, phases

and stages of a surgical procedure relate to one another. OntoSPM also contains concepts that are irrelevant to the maintenance domain such as *body part*, *performing surgery* and *duration of surgical process*. Finally, the ontology requires information that is not commonly contained in industrial procedure documentation. For example, the ontology describes atomic human actions including *language_acts* such as *ordering* and *manipulating actions by a human* such as *grabbing*, *giving* and *releasing* objects. This initiative demonstrates interest in ontological procedure representation from other domains. However, similar to recipes for cooking, current state of the art models cannot be directly applied to industrial maintenance procedures.

Few works propose ontologies for procedure documentation in an industrial setting. In 2008, NASA developed the Procedure Representation Language (PRL), an XML schema to be used in training and spaceflight operations (Kortenkamp et al., 2008). PRL was created to support adjustable autonomy for procedure execution where some parts can be completed by humans and other parts by computers. This representation language is *machine-interpretable* and generic enough to be *shared* across organisations. However, it has a strong focus on task *execution* and cannot be directly adapted to our competency questions. In 2010, Nemeth et al. presented a procedure ontology to be used for diagnosis of process systems (Németh et al., 2010). This ontology, however, relies heavily on data properties for storing values and does not conform with an upper ontology. This problem also present in a technical documentation ontology presented by Koukias and Kiritsis (2015) and an ontological analysis of manufacturing processes by Nagy et al. (2021). This makes it difficult to re-use these ontologies and to integrate the ontology with existing ontologies (Katsumi and Grüninger, 2016). It is worth mentioning that the ontology for manufacturing process models (Nagy et al., 2021) demonstrates successful development of an ontology using the data-centric perspective and use cases. However, the rigid relationships between concepts, such as "*Station workstationHasResource Resource*", means it has limited applicability to our work (as maintenance technicians rarely have a work station). In Section 7, we perform a further concept-level comparison between OMPD, and the ontologies described in Kortenkamp et al. (2008), Németh et al. (2010) and Koukias and Kiritsis (2015).

We have identified three gaps that are the research contribution of OMPD

- Goal 1: The ontology should be *generic* so that many industrial organisations can apply the procedure ontology to their data.
- Goal 2: The ontology should model information *currently stored in procedure documentation in industry* so that organisations can use the ontology with no new data requirements.
- Goal 3: The ontology should answer competency questions (given in Section 2) that *support maintenance technicians, engineers and schedulers* when creating or using procedure documentation in their work.

4. Ontological choices

The goals (Goal 1, Goal 2, and Goal 3) outlined in the previous section guide the overarching ontological choices discussed in this section. Further concept-level and axiom-level choices are discussed in Section 5. In this section, we highlight the trade-offs involved in meeting ontological best practices while designing a solution that our users will accept and use in practice.

4.1. Ontological choice 1: Which foundational ontology?

Upper ontologies including BFO (Arp et al., 2015), DOLCE (Masolo et al., 2003), PSL (Gruninger and Menzel, 2003) and ISO 15926 Part 14 (International Organization for Standardization, 2019) are core artefacts in top-down ontology development. Top-down ontology development starts with generic concepts that are common across many applications. This is different from bottom-up ontology development which has been criticized for being difficult to modify and integrate with other ontologies (Batres et al., 2007), a view supported by Souza et al. (2013) in their systematic review.

We have chosen to align this work to the ISO CD/TR 15926 *Part 14* upper ontology (International Organization for Standardization, 2019). CD stands for "community draft" and TR stands for "technical report" (we refer to the ontology as ISO 15926 Part 14 for brevity). Its development is driven by the POSC Caesar Association (PCA) and drafts have already been made available online (ISO/TC184/SC4/WG3, 2020).

ISO 15926 Part 14 draws on elements of BFO and the ISO15926 data model. While ISO15926 Part 2 is a well-established data model (Batres et al., 2007), its lack of support for semantic reasoning has attracted criticism (Jordan et al., 2014). ISO 15926 Part 14's development is driven by the need to use the expressive power of the OWL-2 Standard for reasoning that is not possible with the ISO 15926-2/4 work (ISO/TC184/SC4/WG3, 2020; Kiritsis, 2013).

The intention is that ISO15926 Part 14 is an *industrial* upper ontology, mapped to ISO15926 Part 2 and BFO. BFO is currently used in industrial community ontology efforts such as the Industrial Ontologies Foundry (Karray et al., 2021) so this mapping is important to ensure widespread use of OMPD in the industrial ontology community. ISO15926 Part 14 borrows many principles and relationships from BFO while adopting nomenclature from ISO 15926 Part 2. This is language that is familiar to industrial users and, in particular, users of the existing ISO 15926 standard. For example, ISO15926 uses the term `Activity`, rather than BFO's `Occurrent`. Regardless, BFO and ISO15926 Part 14 bear many similarities such as the use of the term `Disposition`, that is inspired by the BFO concept of the same name (International Organization for Standardization, 2019).

We have decided to align with this upper ontology to adhere with current community practices for industrial ontologies. ISO15926 Part 14 is getting increasing attention in industry for commercial applications. For example, ISO 15926 Part 14 is used for Aibel's Material Master Data project (Skjæveland et al., 2018). To make ISO 15926 Part 14 further accessible for industry professionals, ontology templates have been developed based on terms described by Klüwer et al. (2008). Examples of these can be found in Skjæveland et al. (2019) and Forssell et al. (2017). While ISO15926 Part 14 is a prominent effort, it has not been the only effort to convert the ISO 15926 data model into an OWL ontology (Batres et al., 2007; International Organization for Standardization, 2018a). Work was also done on this by the nuclear industry by Fiorentini et al. (2013) and more recently by Kwon et al. (2018).

A hierarchical diagram of the classes in ISO15926 Part 14 is provided in Fig. 4. The terms from ISO15926 Part 14 used in OMPD are `Activity`, `Object`, `Information Object`, `Physical Object`, `Role`, `Disposition`, `Quality` and `Person`. More about how these classes are used in OMPD is described in Section 5. Conformance to ISO 15926 Part 14 will future-proof OMPD by supporting ontology re-use (Goal 1) while ensuring that it is accessible to industrial users.

4.2. Ontological choice 2: Specificity vs generality

It is well known that ontology engineers must make trade-offs between specificity and generality (Hitzler and Krisnadhi, 2018). If OMPD was designed to be *general* it would model only the concepts that are

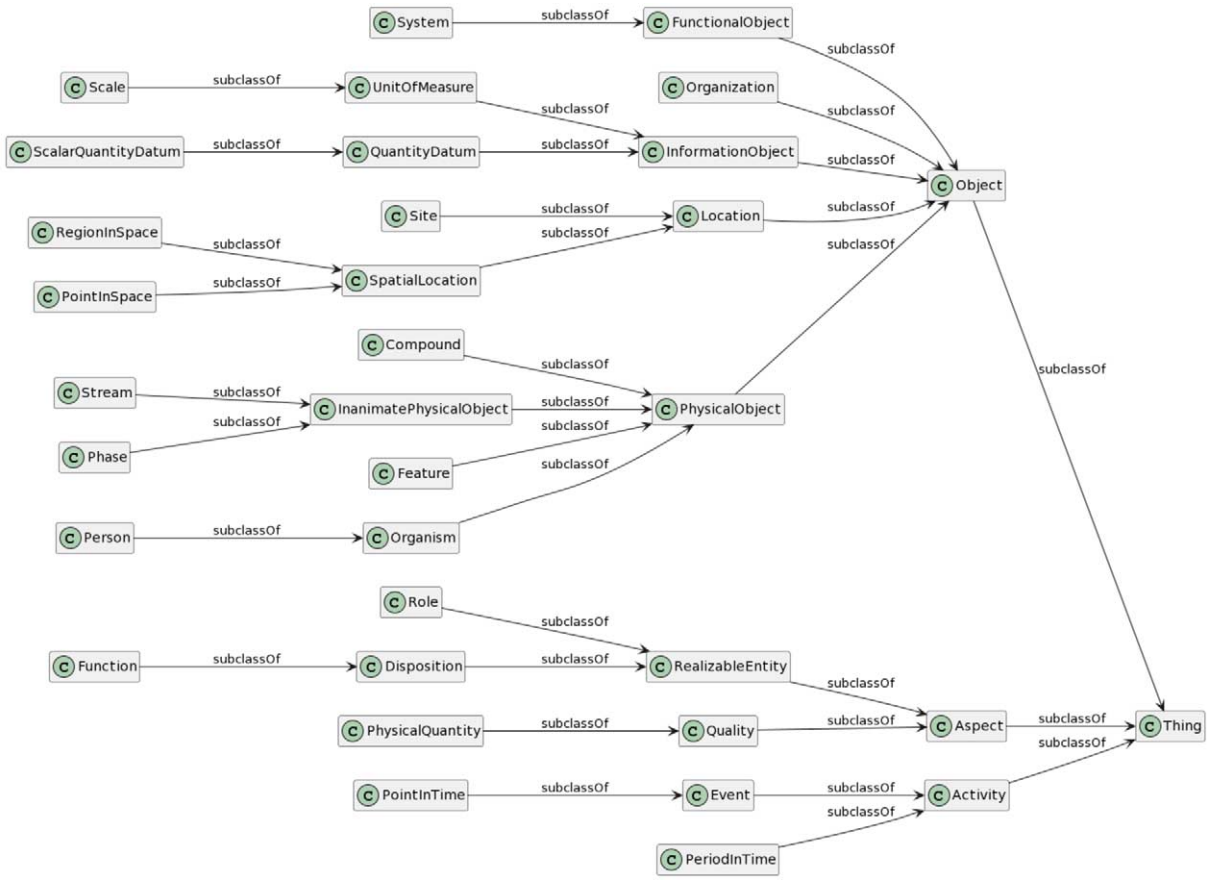


Fig. 4. Taxonomy of classes in ISO/CD TR 15946-14.

common across *all* procedure documentation (including cooking recipes, IKEA assembly instructions, etc). While ontologies of this nature are important, it does not help us to answer more specific competency questions that are important to engineers, schedulers and technicians. PSL (Gruninger and Menzel, 2003) is a good example of an ontology that has generalised to this extreme and has been very successful as a cross-domain tool. Instead, OMPD provides coverage of maintenance-procedure documentation. For example, concepts such as `Hazard` and `Permit` are important for maintenance procedures, but may not be necessary in everyday household procedures. The goals of OMPD are twofold. First, to be *generic enough* to support legacy maintenance procedure documentation from non-uniform data sources and across organisations (Goal 1). Second, to be *specific* enough to support engineers, schedulers and schedulers using real-world industrial data (Goal 3).

4.3. Ontological choice 3: Scope and modularisation

OMPD has two modules. These are the Static Procedure Ontology (SPO) and the Corrective Maintenance Task Ontology (CMTO). In designing the SPO, we recognised that maintenance procedure documentation that is currently used in practice is non-temporal (or "static"). For instance, if a document specifies that a tool is required for a procedure, the document does not know (or care) that the tool is available for use at a given time. While live resource modelling is a useful concept it is not within

the scope of OMPD. This choice was made so that OMPD can be used by organisations with no additional data requirements (Goal 2). While some organisations do store equipment availability and asset health information in a live feed or a digital twin (Tao et al., 2018), this is not yet a widespread practice. Therefore, future work can import SPO and add temporal concepts when industry data is available.

The second module is the CMTO. CMTO captures *limits* and *corrective actions*. These two concepts are sometimes (but not always) found in maintenance procedure documentation. *Limits* are assigned to observation tasks that inform maintenance technicians of the state or range of measurements that an asset should be within to pass an inspection. *Corrective actions* are conditional tasks that are to be performed if an asset is outside its *limits*. We have chosen to separate these concepts into a second module for two reasons. First, if an organisation does not capture corrective actions in their procedure documentation, they can just use SPO (Goal 1). Second, as we will discuss in Section 5.2 this module can integrate with other processes core to maintenance management such as RCM. The separation of this second module makes this integration easier for end-users of OMPD.

5. Implementation

This section includes descriptions of our concepts and axiom-level ontological choices made in the design of OMPD. An OWL instantiation of OMPD can be found in the following GitHub repository: https://github.com/uwasystemhealth/Paper_Archive_Procedure_Ontology. Note that the ontologies in this repository use the namespace, www.example.org, as a placeholder. Uploading this ontology to a public repository under a production-ready namespace is a focus of future work.

5.1. Concepts in SPO

This section contains implementation decisions underpinning the concepts and axioms in SPO, the core module of OMPD.

5.1.1. Documentation and processes

In OMPD, a `Maintenance Procedure Document` is modelled as an `Information Object` in ISO 15926 Part 14. An `Information Object` is remodelled from the ISO 15926 Part 2 concept `Class of Information Object`. In ISO 15926 Part 14, an `Information Object` is described as an `Object` such as a newspaper, or a paper document (POSC Caesar Association, 2022). In industry, maintenance procedures are often PDF documents but can also be software artefacts (where they may exist as a collection of database records). To further complicate matters, the same procedure can be copied into different physical locations (i.e. word document, PDF, database). For this reason, BFO models `Information Content Entities` (entities that are about something) as a `Generically Dependant Continuant` and the authors argue that there should be a separate entity, the `Information Bearing Entity` being the physical object that "bears" the information (Smith et al., 2013). In OMPD, however, we have made our classification based on the function of the maintenance procedure. In this domain, a `Maintenance Procedure Document` is a representation of a procedure that can be read, conformed to, and versioned. We do not intend to model the form of the original artifact (i.e. the PDF document), because we hope that, in the future, companies will cease to use paper-driven processes, in favour of digital procedure management processes (with procedures aligned to OMPD). If such a future becomes a reality, we will not have heterogeneous information bearers to

model. In this case, the class Information Object is sufficient for our purposes and we do not separate the information "bearer" from the information itself. This use of the Information Object class is also consistent with our use of ISO15926 Part 14 in previous works. For example, in our Failure Modes and Effects Analysis ontology (Hodkiewicz et al., 2021a). Further, to maintain generality in OMPD (Goal 1), we have not modelled the document's metadata (i.e. author). Instead users of OMPD can create an application-level ontology to capture this information.

Maintenance Procedure Documents are about one or more Maintenance Procedure Processes. ISO 15926 Part 14 offers two sub-properties of isAbout. These are, represents and quantifiesQuality but neither seem to capture the relationship between a Maintenance Procedure Document and a Maintenance Procedure Process. This process is an activity that consists of a set of steps (or Maintenance Tasks). A Maintenance Procedure Process should not have other Maintenance Procedure Processes as parts. In industry, a Maintenance Procedure Process is a fundamental unit in an organisation and these have rules attached to them. For example, maintenance technicians perform a single Maintenance Procedure Process at a time. Additionally, a procedure is recorded as a single event as a maintenance work order. From an organisational perspective, procedures are Merelogically indivisible. Maintenance Tasks are thus required to further break down the procedure into parts that a technician can read and understand.

We organise Maintenance Procedure Processes and Maintenance Tasks under a super-class called Maintenance Process. The concept of a Maintenance Process covers a wider range of maintenance activities. Maintenance is defined as "the actions intended to retain an item in, or restore it to, a state in which it can perform a required function" (IEC, 2016). We can use Maintenance Process to capture these actions (or collection of actions). We have not chosen to model the complexities behind "state" and "required functions" in OMPD, as this information is not currently stored in procedure documentation. We perform deeper analysis of these concepts in our prior work (Woods et al., 2021a). Regardless, to understand the difference between a Maintenance Process and a Maintenance Procedure Process or a Maintenance Task, we draw from the analysis in Jarrar and Ceusters (2017). While a Maintenance Procedure Process is *telic* (i.e. tending towards a goal that is described in a procedure), a Maintenance Process does not need to be. For example, a Maintenance Planning Process is a Maintenance Process that involves planning work that is to be executed on a particular item. A Maintenance Planning Process is an ongoing process, existing throughout the life-cycle of the equipment, and not tending towards some goal or end-state.

Maintenance Tasks are described with a Maintenance Task Description. This is typically a natural language text description (i.e. "unscrew bolt"). In the procedure shown in Fig. 1, these natural language descriptions are coupled with multimedia descriptions (in this case they are schematic diagrams). These descriptions can take many other forms including videos, computer-readable scripts (for automated equipment), 3D models and animations (for augmented reality). How organisations choose to represent the steps in their procedures can evolve as technology advances and organisational processes change. Therefore, similar to Maintenance Procedure Documents, we have not chosen to explicitly model the representation format of a Maintenance Task Description. Users of this ontology can use data properties, or an application-level ontology to model this information. In the evaluation in Section 6.2, we model a real-world procedure that uses both a text description and an image description to describe the same task. Using the Maintenance Task Description concept, we

can model both of these descriptions and use `hasText` and `hasImageUrl` data properties to capture their original format.

Another consideration for this ontology is how to manage procedure histories and versioning (i.e. when technical writers update procedures over time). In the current model, if a `Maintenance Task` described in a procedure document changes, no record is kept of the previous procedure. While this is reflective of current practice in industry (where a document is given a new version number and the previous document in overwritten), problems emerge if an organisation chooses to extend this model and model executions of maintenance tasks. In OMPD, we achieve versioning through a `document identifier` and a `version` annotation property. When a procedure is updated, a new `Maintenance Procedure Document` should be created with the same document identifier and an updated version number. Since this document now describes a different procedure to its previous version (whether that be due to a change of resources or activities), a new `Maintenance Procedure Process` must be created, with updated links to `Maintenance Tasks` and `Resources`. Since OMPD maintains a static perspective, we do not model the time period in which each procedure is the "active" procedure. However, this could be easily added in temporal extensions of OMPD.

The axioms for the classes introduced in this section are shown in Fig. 5 and Table 1. Classes in Table 1 have been presented in the form `PREFIX: Class Name` where the prefix is the ontology that the class comes from (`ISO` for ISO 15926 Part 14 and `OMPD` for the OMPD ontology). Natural language descriptions for the entities are given in their Aristotelian form (Arp et al., 2015) and formal axioms are given in Description Logic.

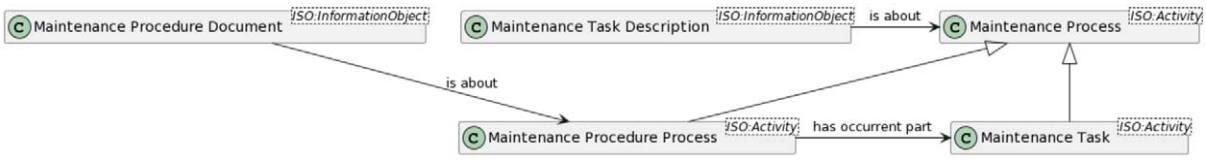


Fig. 5. Visual representation of documentation and processes in OMPD.

Table 1

Axioms for `Maintenance Procedure Document`, `Maintenance Process` `Maintenance Procedure Process` and `Maintenance Task Description`

Concept	Natural Language Description	Description Logic Axiomatisation in OMPD
Maintenance Procedure Document	An `ISO: Information object` that is about some `OMPD: Maintenance procedure process`	$MaintenanceProcedureDocument \equiv InformationObject \wedge (\exists isAbout.MaintenanceProcedureProcess)$
Maintenance Process	An `ISO: Activity`, intended to repair or restore an item to a state in which it can perform its required function	$MaintenanceProcess \sqsubseteq Activity$
Maintenance Procedure Process	An `OMPD: Maintenance process` that is representedBy some `OMPD: Maintenance procedure document` and hasActivityPart some `OMPD: Maintenance task`	$MaintenanceProcedureProcess \equiv MaintenanceProcess \wedge (\exists representedIn.MaintenanceProcedureDocument) \wedge (\exists hasActivityPart.MaintenanceTask)$
Maintenance Task Description	An `ISO: Information object` that is about some `OMPD: Maintenance Task`	$MaintenanceTaskDescription \equiv InformationObject \wedge (\exists isAbout.MaintenanceTask)$

Notice that this table does not contain an axiomatisation for `Maintenance Task`. In Sections 5.1.2, 5.1.3, 5.1.4 and 5.1.5 we will explore the ontological choices made when modelling `Maintenance Task` entities. Axioms for `Maintenance Task` is given in Fig. 9 and Table 2.

5.1.2. Tasks in context

A `Maintenance Procedure Process` has a set of steps, or `Maintenance Task` entities. A key ontological question is whether (or not) a task should be considered as an entity independent of the procedure that it is performed in. For example, if a pump overhaul procedure and a pump impeller replacement procedure on a particular pump both contain the task "replace impeller", then is this task the "same" in both procedures?

The benefit of decoupling a `Maintenance Task` from its context in a procedure is twofold. First, if a task in a procedure changes (e.g. it has new resource requirements), data-owners can easily determine which procedures contain the same task and change them accordingly. Second, this representation allows data-owners to check if a maintenance technician has completed similar tasks before (perhaps in a different procedure). This use case can help in developing adaptive work instructions for technicians, based on their experience in a particular task. Adaptive work instructions is one of the key motivations of our research group's enquiry into this space.

We considered two different ways to achieve this decoupling in OMPD. The first is to create a new concept called `Task in Context` that has a relationship to a `Maintenance Procedure Document` and the other instances of `Task in Context` described in the document. In this representation, `Task In Context` is different to a `Maintenance Task` individual (that is related to some `Maintenance Task Description`). This approach treats tasks as identical, so a task may appear in several procedures. We call this the "identity" approach. An alternative approach is to introduce an object property called `locallyEquivalentTo`. Data-owners can specify whether a `Maintenance Task` is locally equivalent to another task from a different procedure. This approach supposes that non-identical tasks appear in procedures. We call this the "local equivalence" approach.

Both approaches have pros and cons. The identity approach is pictured in Fig. 6. In this approach, sub-tasks must retain context so `Maintenance Tasks` and their corresponding `Task In Context` entities are interleaved, which creates schematic complexity. When a procedure involves subtasks (i.e. a task hierarchy with a depth greater than one), the local equivalence approach, pictured in Fig. 7, seems simpler and more intuitive. Our rational for choosing the local equivalence approach is that, when considering a single procedure in isolation, the schema is simpler. Procedures are treated as stand alone activities in practice and an engineer is likely to consider a single maintenance procedure in isolation. Note that the two approaches are similar in the information that they convey. A modeller can translate the local equivalence approach to the identity representation without losing information from the procedure.

In OMPD, the object property `locallyEquivalentTo` is used to realise the local equivalence approach to task modelling. This object property is not to imply that the objects are, necessarily, identical. Instead, the object property is used when tasks have the same maintenance description, use the same resources and realise the same hazards. Data-owners can use this feature of the ontology to improve the consistency of their maintenance task data. Suppose that an engineer has developed a `Maintenance Task` but has under-specified the tools required for that task. If another engineer writes a locally equivalent `Maintenance Task` with the appropriate tools, the following SWRL rule can infer that both `Maintenance Task` entities should have the same tools.

$$locallyEquivalentTo(?x, ?x2) \wedge Tool(?t) \wedge participantIn(?t, ?x) \rightarrow participantIn(?t, ?x2)$$

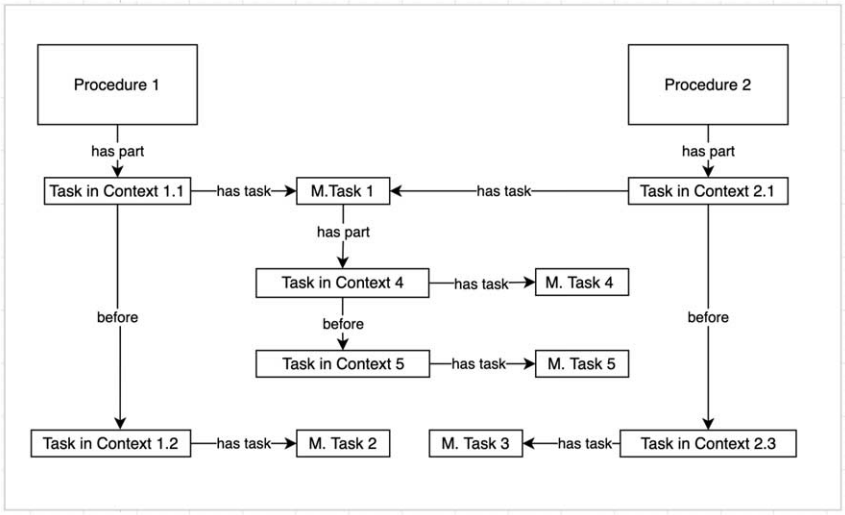


Fig. 6. Conceptual diagram of "Identity" approach to task modelling.

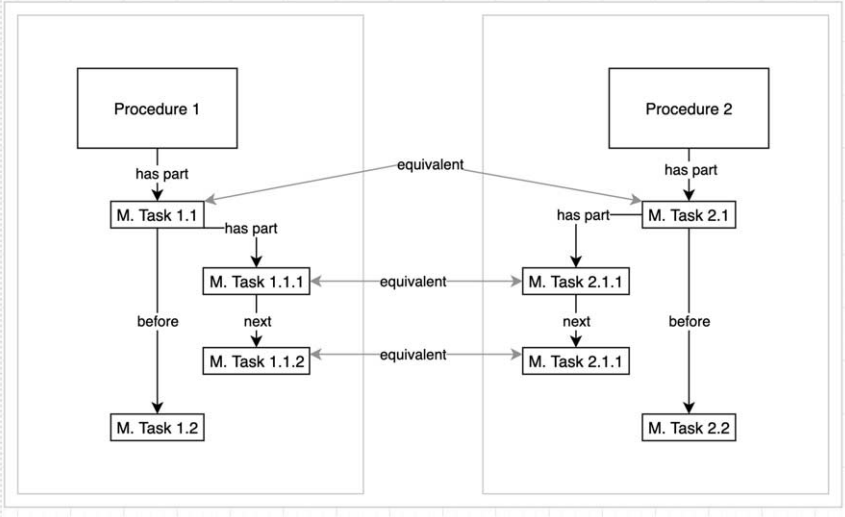


Fig. 7. Conceptual diagram of "Local equivalence" approach to task modelling.

The same technique can be applied to hazards. For example, if a technician finds a new hazard when completing a `Maintenance Task`. A SWRL rule can be applied to infer that all procedures containing locally equivalent `Maintenance Task` entities will also have that hazard. Note that we have not made this rule an ontological commitment in OMPD. We feel that some organisations may find this feature undesirable as incorrect data could be propagated through the ontology. However, we encourage users of OMPD to use this feature in their application-level implementations.

5.1.3. Considering procedure executions

One of the core goals of OMPD is to be usable by engineers, given the way that they work with procedures and the information that is currently stored in procedure documentation (Goal 2, Section 3). With

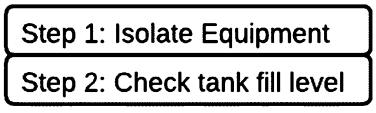
this in mind, OMPD contains a minimal set of classes and the tasks represented in the procedure are modelled as instances of `Maintenance Task` (a subclass of `Maintenance Process`). Tasks, as they are described in the procedure documentation, are modelled at the individual level in OWL. This way, users of the ontology do not have to update the class schema when a new procedure is added or a procedure is updated. From an operational perspective, we want engineers using the ontology to be able to enter their version of activities (instances of activities), not the mechanism to describe activities (classes of activities). In this way, our ontology is as close to the real world application as possible, without omitting use cases in the future. Our decision to model the `Maintenance Task` as it is described in current procedure documentation raises several fundamental ontological questions, which will not be addressed in depth here. This includes treatment of 'hypothetical' entities, objects, and situations as are commonly referenced in maintenance procedures.

5.1.4. Task sequencing

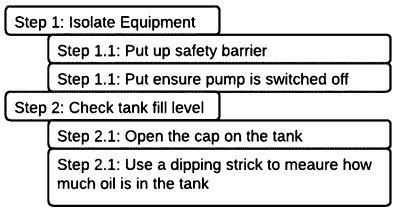
In OMPD, data-owners can specify the sequence of tasks in a `Maintenance Procedure Process`. OMPD models only the task sequence that is represented in a maintenance procedure document. Due to the "static" philosophy of OMPD (discussed in Section 4.3), we are not concerned with the sequence that these tasks occur in practice (i.e. as with activity occurrences in PSL). The industrial maintenance procedures that we have examined all represent *linear* task sequences thus we have not included parallel tasks in OMPD. Furthermore, maintenance procedure processes are designed to be performed by one technician. If tasks are to be performed in parallel, two separate procedures are created for this. To model task sequences, we used the Sequence design pattern from the *Ontology Design Patterns* repository (Gangemi, 2010). The implementation of this pattern is very similar to our generic task hierarchy (discussed in the following section) as we use the transitive property `indirectlyBefore` and its intransitive sub-property `directlyBefore`. We have added these properties as sub-properties of ISO15926 Part 14's intransitive `before` property. Re-using this design pattern means that we can answer questions such as "what tasks are remaining" and "what task is next". These competency questions are very useful for organisations who wish to power a user interface with this ontology. Using this functionality, technicians (or machines in the case of automated equipment) can "step through" tasks to see what activities need to be performed.

5.1.5. The generic task hierarchy

In industry, procedures are represented at different levels of detail both within and between organisations. While one procedure may contain a list of steps (as shown in Figure 8a), another procedure may have more detailed steps, organised as a hierarchy (Figure 8b). An example of when tasks will need to be more detailed is for organisations who use automated equipment to do particular tasks. While a human



(a) A task hierarchy with one level



(b) A task hierarchy with two levels

Fig. 8. Examples of different task hierarchies that can be represented in OMPD.

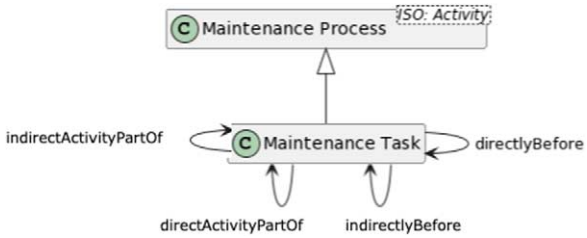


Fig. 9. Visual representation of maintenance task in OMPD.

reader may simply need to read "fill up the tank" and will know what to do, a robot arm will likely need to know the exact co-ordinates of the tank that it is filling. The procedure will need to specify the robot's motion as it moves its arm from position *a* to position *b* and back again. To ensure that both types of procedures can be represented, OMPD enables data-owners to maintain a *generic task hierarchy*.

The generic task hierarchy ensures that data-owners can retrieve a full task hierarchy without knowing how deep the hierarchy is. This functionality is useful because engineers will not need to know how many levels the tasks need to have when writing the procedure. If the organisation requires it, more levels of detail can be added later. This is essential for organisations who simply want to digitize their current procedures today, but intend to create adaptive procedures or manage procedures for automated equipment in the future.

We implement a generic hierarchy of activities using part-of relations between `Maintenance Tasks`. ISO15926 Part 14 contains an intransitive property called `activityPartOf`. To realise our generic hierarchy, we require a transitive sub-property of `activityPartOf` called `OMPD:indirectActivityPartOf`.

The second object property that we use is `OMPD:directActivityPartOf` that is a subclass of `OMPD:indirectActivityPartOf` and is intransitive. This object property is important so that ontology interrogators can re-construct the asset hierarchy and know where a task sits in the task hierarchy. This allows the task hierarchy to be reconstructed and shown to technicians (perhaps as a digital user interface). The modelling for `Maintenance Task` in OMPD is shown in Fig. 9. Table 2 contains the formal axiomatisation.

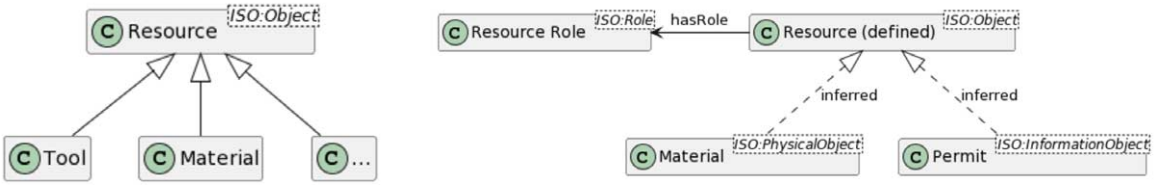
5.1.6. The role of resources

`Resources` are required for procedures to be executed. These include tools, materials, people and permits. We examined two options for representing resources in OMPD. The first option is depicted in Fig. 10a. In this representation, `Resource` is an asserted class that is a subclass of `ISO:Object`. Specific resources in maintenance procedures (i.e. `Tool` and `Permit`) are then asserted as subclass of `Resource`. This means that any individual of type `Tool` is also of type `Resource`. This representation has two issues. Firstly it goes against the ontology best practice, the *principle of asserted single inheritance* (Arp et al., 2015). This is because a `Permit` is both an `Information Object` and a `Resource` whereas a `Tool` is a `Physical Object` and a `Resource`. Secondly, this representation may introduce problems when inserting individuals into the ontology. If an organisation has a `Tool` such as a wrench. If that wrench is never used in a procedure, it is not a resource in OMPD. This would mean that a new system that stores components that are not resources is required.

To resolve these issues, OMPD implements resources as shown in Fig. 10b. This representation is in line with how the Industrial Ontologies Foundry represents resources. In this model, `Resource` is a defined class. It is defined as the bearer of a `Resource Role`. Now, if an individual is the bearer

Table 2
Definition for the maintenance task concept in OMPD

Natural Language Description	Description Logic Axiomatisation in OMPD
An `OMPD: Maintenance process` that is: • a direct activity part of zero or one `OMPD: maintenance procedure process` • an activity part of zero or more `OMPD: maintenance procedure process` • a direct activity part of zero or one `OMPD: maintenance task` • an activity part of zero or more `OMPD: maintenance task` • occurs directly before zero or one `OMPD: maintenance task` • occurs before zero or more `OMPD: maintenance task` • occurs directly after zero or one `OMPD: maintenance task` • occurs after zero or more `OMPD: maintenance task` • is locally equivalent to only `OMPD: maintenance task`	$MaintenanceTask \sqsubseteq MaintenanceProcess$ $\wedge(\leqslant 1 directActivityPartOf.MaintenanceProcedureProcess)$ $\wedge(\leqslant 1 directActivityPartOf.MaintenanceTask)$ $\wedge(\leqslant 1 directlyBefore.MaintenanceTask)$ $\wedge(\leqslant 1 directlyAfter.MaintenanceTask)$ $\wedge(\forall locallyEquivalentTo.MaintenanceTask)$ **`indirectActivityPartOf` is a transitive subproperty of `activityPartOf`, as with `indirectlyBefore` and `indirectlyAfter`.



(a) Resources as an asserted class

(b) Resources as a defined class (implemented in OMPD)

Fig. 10. Two representations for resources considered in the design of OMPD.

of some `Resource Role`, then the ontology will automatically classify the object as a `Resource` type. This has the added benefit that classes such as `Material` and `Permit` can sit in more suitable positions in the ISO 15926 Part 14 class hierarchy (i.e. `Material` is a `Physical Object` and `Permit` is an `Information Object`).

We have decided not to further constrain the definitions of specific resource classes in OMPD as these constraints would not contribute to the use-case for this ontology. The number of necessary constraints for these classes may also be different between organisations. For example, in one organisation a `Tool` may be an object with an OEM (Original Equipment Manufacturer) ID. In another (perhaps larger) organisation, this may not be the case. Such constraints should exist in an application-level implementation of OMPD. Of course, in the future we would like provide necessary and sufficient conditions for these classes, in a manner that is applicable across organisations. However, this will require further collaboration with both industry and the industrial ontology community. This is an avenue for future work. For now, we have provided a primitive class for `Tools`, `Permits`, and other resource types that we encountered in the industrial procedures examined in this work. Data-owners can use these classes as a mechanism to organise their data, and to enable the competency questions provided in this paper. Axiomatisations for the `Resources` that are modelled in OMPD are shown in Table 3.

Table 3
Resources in OMPD

Concept	Natural Language Description	Description Logic Axiomatisation
Resource	An ISO:Object that has some OMPD: Resource Role	*Resource* ≡ *Object* ∧ (∃*hasRole*.*ResourceRole*)
Resource Role	An ISO:Role that is the role of an ISO:Object that is required to execute some ISO:Activity	*ResourceRole* ≡ *Role*∧ ∀*roleOf* (*Object* ∧ ∃*requirementOf* .*Activity*)
Component	A ISO:Physical Object that may have some OMPD: Resource role	*Component* ⊑ *PhysicalObject* **If hasRole some Resource Role then individuals of type Component can also be of type Resource
Permit	A ISO:Information Object that may have some OMPD: Resource role	*Permit* ⊑ *InformationObject* **If hasRole some Resource Role then individuals of type Permit can also be of type Resource
Tool	A ISO:Physical Object that may have some OMPD: Resource role	*Tool* ⊑ *PhysicalObject* **If hasRole some Resource Role then individuals of type Tool can also be of type Resource
Material	A ISO:Physical Object that may have some OMPD: Resource role	*Material* ⊑ *PhysicalObject* **If hasRole some Resource Role then individuals of type Material can also be of type Resource
Qualified Person	A ISO:Person that may have some OMPD: Resource role and has a Qualification	*QualifiedPerson* ≡ *Person*∧ ∃*hasQuality*.*Qualification*∧ ∃*requirementOf* .*Activity* **If hasRole some Resource Role then individuals of type QualifiedPerson can also be of type Resource
Qualification	A ISO:Quality that is borne by zero or more OMPD: Qualitified Person	*Qualification* ⊑ *Quality* **relationship to qualification is inferred from Qualified Person axiomatisation

5.1.7. Entities at the procedure level and the task level

Many maintenance procedures in industry describe the Resources that are required for execution of the procedure, the Maintainable Item that work is performed on, and the Hazards that could emerge in the procedure's execution. These entities can be represented at the procedure level or the task level, depending on both the procedure (i.e. an inspection or a replacement) and the organisation.

Resources are defined in Section 5.1.6, but we are yet to define Hazards and Maintainable Items. A Hazard is defined in the engineering standard AS IEC 61882:2017 as a "source of potential harm" (Standards Australia, 2017). This "source" refers to characteristics of a context in which work is performed (i.e. radiation). ISO15926 Part 14 contains a class called Realizable Entity, modelled after the BFO term of the same name. A Realizeable Entity in BFO is a thing that inheres in an item (i.e. radiation) or group of items (i.e. electrical hazards), and is realised in some process. We model a Hazard as a Realizeable Entity that is realized in some Hazard Realization Process. We do not say that Hazards are realised in Maintenance Processes because it is possible to perform a Maintenance Process *without* realizing the potential hazards that are described in the procedure. It is also incorrect to say that a Maintenance Process causes a Hazard, as the hazard can be realised through an external event separate to the maintenance process.

Table 4
Definitions for `Hazard` and `Maintainable Item` in OMPD

Concept	Natural Language Description	Description Logic Axiomatisation
Hazard	An `ISO:Realizeable Entity` that is the source of potential harm that could be realized in some `OMPD: Hazard Realization Process`	$Hazard \equiv RealizeableEntity \wedge (\exists realizedIn.HazardRealizationProcess)$
Hazard Realization Process	An `ISO:Activity` that has the potential to cause harm	$HazardRealizationProcess \sqsubseteq Activity$
Hazard In Maintenance Process	An `OMPD:Hazard` that is realized in some `OMPD:Hazard Realisation Process` that occurs relative to some `Maintenance Process`	$HazardInMaintenanceProcess \equiv Hazard \wedge \wedge \exists realizedIn(HazardRealizationProcess \wedge \exists occursRelativeTo.MaintenanceProcess)$
Maintainable Item	An `ISO:Physical object` that has a role `OMPD:Maintainable item role`	$MaintainableItem \sqsubseteq PhysicalObject \wedge (\exists hasRole.MaintainableItemRole)$

Finally, we do not want to constrain `Hazards` only to `Maintenance Processes` as hazards can cause potential harm in a wide range of work processes. With these considerations in mind, we create a new class called `Hazard In Maintenance Process`. This is a `Hazard` that has some `Hazard Realization Process` that `occurs Relative To` a `Maintenance Process`. This new class allows data-owners to query for all hazards that are considered across our maintenance processes.

To define `Maintainable Item`, we draw from the existing state of the art reference ontology for maintenance work management, ROMAIN (Karray et al., 2019). ROMAIN models a `Maintainable Item` as the bearer of a `Maintainable Item Role`. Instead of BFO's `bearer of` relationship, we use ISO 15926 Part 14's `has role` relationship. In OMPD, `Maintainable Items` participate in `Maintenance Processes`. Definitions for `Hazard` and `Maintainable Item` are given in Table 4.

As per competency questions 1, 4 and 8, data-owners need to query for all `Resources`, `Hazards` and `Maintainable Items` that are involved in a procedure, regardless of what level these entities are represented. To implement this in OMPD, we have used the following SWRL rules:

$$hasDirectActivityPart(?y, ?x) \wedge participantIn(?z, ?x) \rightarrow participantIn(?z, ?y)$$

$$hasDirectActivityPart(?y, ?x) \wedge HazardRealizationProcess(?p) \wedge HazardInMaintenanceProcess(?z) \wedge realizedIn(?z, ?p) \wedge occursRelativeTo(?z, ?x) \rightarrow occursRelativeTo(?z, ?y)$$

Note that these SWRL rules can also be represented as OWL property chain if an organisation's specific implementation requirements allow. Axiomatisations for `Maintainable Item` and `Hazard` are shown in Table 4. Figure 11 shows a visual representation of the SPO module of OMPD.

5.2. Concepts in CMTO

The second module of OMPD is the CMTO. There are two fields in the procedure shown in Fig. 1 that have not been captured in the SPO module. These fields are "Limits" and "Corrective Action Taken". Limits are often found in maintenance inspection documentation where technicians must check that equipment is healthy, or its attributes are within a healthy range. For example, in the procedure in Fig. 1,

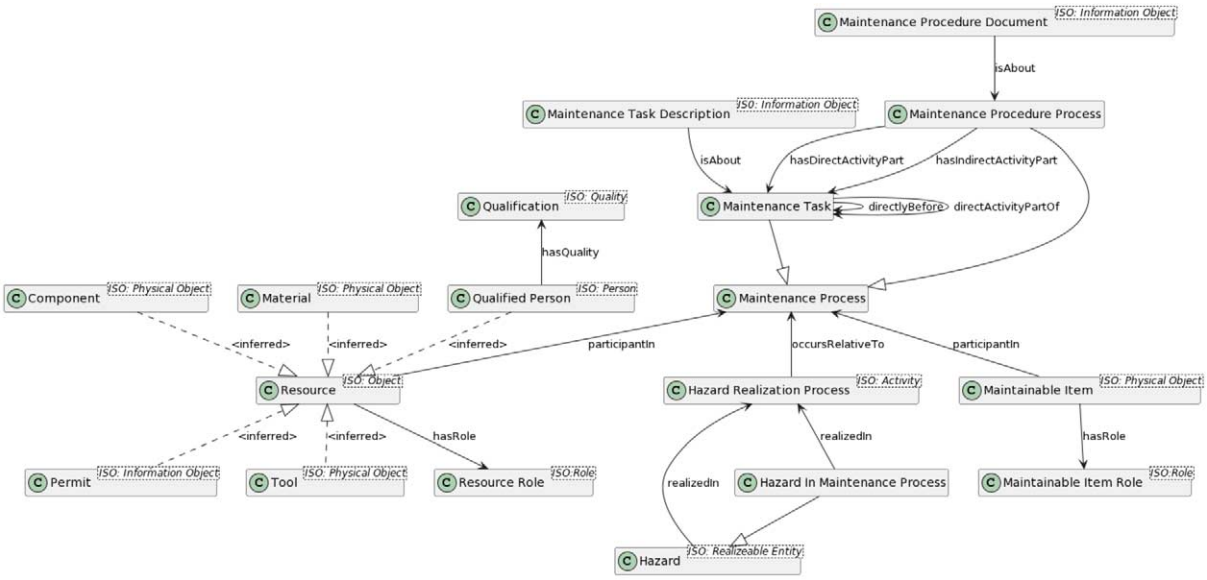


Fig. 11. The Static Procedure Ontology (SPO) Module.

the technician must "check that suction and discharge points are clear of obstruction" and the limit for this maintenance task is "obstruction free" (i.e. not plugged or blocked). We have made the observation that these limits maintenance procedures correspond to *functional failures*. A functional failure is defined as the "loss of the ability of an item to perform a required function" (EFNMS, 2017) and the performance is assessed by qualitative or quantitative observations. According to previous work completed on an Ontology for Failure Modes and Effects Analysis, a `Functional Failure` is the result of some `Failure Event` and is represented by some `Failure Mode Observation`, specified in the RCM process (Hodkiewicz et al., 2021a).

`Functional Failures` are identified in some `Maintenance Task` that is specified in the procedure document (i.e. check for x). If this `Functional Failure` is identified, then the procedure generally suggests that a `Corrective Maintenance Task` should be performed to address the issue. In the procedure in Fig. 1 this `Corrective Maintenance Task` is simply an empty box. The purpose of this empty box is to allow the technician to decide what needs to be done to address the problem. For example, if a task, "inspect bolts" realises a functional failure, "bolt loose", then the technician could write "tightened bolt" in the corrective action field on the PDF. If the fix to the problem is more complex, then the technician could write "raised work order" and raise the issue in their CMMS so that a fix to the problem can be scheduled. To capture these types of tasks in OMPD, we have created a new class that is a subclass of `Maintenance Task`. This new class is called `Corrective Maintenance Task` and is defined as a `Maintenance Task` that `addresses` some `Functional Failure`. This task can have a `Task Description` just like any other `Maintenance Task` in OMPD. However, in the case of the procedure in Fig. 1, no `Task Description` has been prescribed because the field is blank in the procedure document.

To model the idea that a `Failure Event` results in a `Functional Failure`, we create a new sub-property of `before` called `results in`. ISO 15926 has another sub-property of `before` called `causes` but we do not want to use this. In maintenance, there is much complexity to root-cause analysis for failures. For example, an upstream failure (i.e. failure of a pump) may affect other equipment (i.e. the

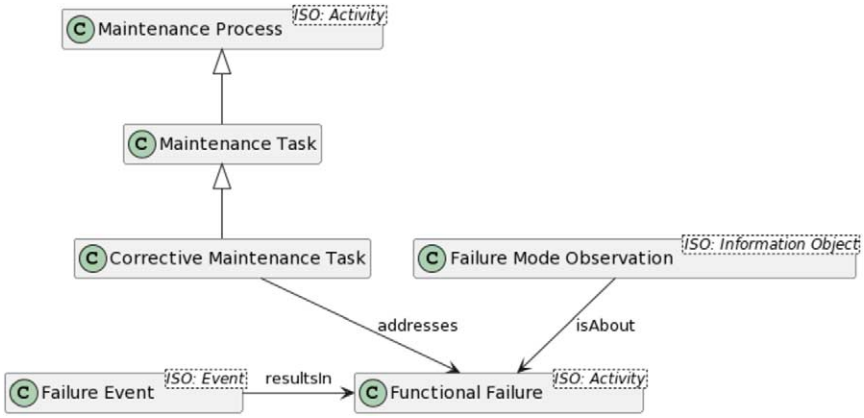


Fig. 12. Diagram of the Corrective Maintenance Task Ontology (CMTO).

level of a tank). Therefore, we believe that modelling a `failure event` as the cause of a functional failure is confusing for engineers. This is consistent with analysis conducted in prior work (Hodkiewicz et al., 2020).

In addition, to model the idea that a `Corrective Maintenance Task` addresses a `Functional Failure`, we create another new sub-property. The new property, `addresses`, is a sub-property of ISO 15926 Part 14's `ends`. In this case, there is an intentional relationship between the `corrective maintenance task` and the `functional failure` that is intended to cause a functional failure to cease. There is another relationship in ISO15925 Part 14 for such intentional relationships called `has interest in`. However, in the current latest community draft of the ontology (ISO/TC184/SC4/WG3, 2020), this relationship has no definition, and its intended use is unclear. Analysis of the suitability of the `has interest in` object property is a subject of future work.

The fact that "limits" in industrial maintenance procedures correspond to functional failures is an interesting outcome of this research because two disparate concepts (i.e. procedures and FMEA performed in RCM) in industry can now be integrated. It is a demonstration of how current data-management practices have failed industrial organisations. By thinking of concepts ontologically, and by modelling the data at the same level of abstraction as given in the original engineering artifact, we are able to align two data sources that appear quite distinct in industry and a perform reasoning across them. Further enquiry in this space will be the subject of future work.

A diagram of CMTO has been provided in Fig. 12 and axiomatisations for each of the concepts are provided in Table 5.

6. Evaluation

Evaluation of an ontology includes both verification and validation activities. Ontology verification is a check of the ontology's consistency (i.e. does the reasoner produce any errors) and validation is a demonstration of the ontology's ability to provide answers to competency questions. Ontology verification was performed using the HermiT reasoner (Glimm et al., 2014). HermiT is an OWL-2 DL reasoner that is built into the Protégé ontology development environment. Readers of this paper can replicate this verification by opening the ontology provided in Section 5 in Protégé and selecting the HermiT reasoner.

Table 5
Terms and axiomatisations in the corrective maintenance task ontology

Concept	Natural Language Description	Description Logic Axiomatisation
Functional Failure	An `ISO: Activity` that prevents an item from performing its required function and is represented in a `OMPD: Failure mode observation`. It has a maintainable item as a participant.	$FunctionalFailure \sqsubseteq Activity \wedge (\exists hasParticipant.MaintainableItem)$ **relationship to functional failure observation (represented in) is inferred from the `Failure Mode Observation` axiomatisation.**
Failure Mode Observation	An `ISO: Information object` that is about a `OMPD: Functional failure`	$FailureModeObservation \equiv InformationObject \wedge (\exists isAbout.FunctionalFailure)$
Failure Event	An `ISO: Event` that resultsIn a `OMPD: Functional failure`	$FailureEvent \sqsubseteq Event \wedge (\exists resultsIn.FunctionalFailure)$
Corrective Maintenance Task	A `OMPD: Maintenance task` that addresses a `OMPD: Functional failure`	$CorrectiveMaintenanceTask \equiv MaintenanceTask \wedge (\exists addresses.FunctionalFailure)$ **contextual relationships (i.e. directly before and directly after are modelled in `MaintenanceTask`)

The validation activities performed for this ontology are twofold. First, we demonstrate the ontology's ability to execute our provided competency questions using SPARQL (Pérez et al., 2009). Second, we map two real-world industrial procedure datasets to our ontology to ensure that relevant concepts can be represented.

6.1. Competency question execution

The competency questions provided in Section 2 have been executed using some test data. We demonstrate the reasoning capability of the ontology according to each competency question in a series of SPARQL queries. These queries, and a description of the reasoning that occurs are given in Tables 6, 7 and 8. Note that implementations of OMPD that use the `version` and `documentIdentifier` object properties to manage procedure histories should include reference to these in the queries to ensure that only the current/latest version of a procedure document is referenced. We have not included these checks in the queries in this paper for simplicity and demonstration purposes. The prefixes used for the queries in this section are as follows:

- prefix spo: `http://www.example.org/static-procedure-ontology`
- prefix iso: `http://rds.posccaesar.org/ontology/lis14/ont/core/1.0/`
- prefix cmto: `http://www.example.org/corrective-maintenance-task-ontology`

6.2. Data-driven validation

6.2.1. Company 1: Continuous manufacturing (process) plant

To determine the suitability of OMPD for real-world maintenance procedures, we first map it to a procedure from a continuous manufacturing (process) plant. The procedure used in this example is shown in Fig. 13. The numbers on Fig. 13 show how information from the document maps to the ontology (shown in Figs 11–14). We use the notation (n) to indicate the part of the procedure that is being discussed (where n is 1 to 10).

Table 6

SPARQL queries for technician competency questions

Technician competency questions

Query	**Explanation**
1. What tools, materials and permits do I require to execute a procedure	

`SELECT ?resource`
`WHERE {`
`  VALUES ?procedure_process { spo:procedure_process_001 }`
`  VALUES ?type { spo:Tool spo:Material spo:Permit }`
`  ?resource iso:participantIn ?procedure_process; a ?type`
`}`
	The query selects all individuals that are a participant in `procedure_process_001` (or an activity part of this process) and belong to the class `Tool`, `Material` or `Permit`. This query uses the SWRL rule described in Section 5.1.7 to ensure that tools, materials and permits at all levels of the task hierarchy are captured.

2. What steps need to be performed to execute my procedure?	

`SELECT ?maintenance_task ?parent`
`WHERE {`
`  VALUES ?procedure_process`
`  { spo:procedure_process_001 }`
`  ?maintenance_task iso:activityPartOf ?procedure_process .`
`  ?maintenance_task spo:directActivityPartOf ?parent .`
`}`
	This query selects all tasks that are an activity part of `procedure_process_001`. This query uses the transitive property, `indirectActivityPartOf` to ensure that tasks at all levels of the task hierarchy are selected. The `?parent` of each task is also selected to retain hierarchical information. Note that this query returns tasks in no particular order. The retrieval of ordered tasks is shown in Competency Question 3.

3. Given that I am up to task *x* in a procedure, what task needs to be performed next?	

`SELECT ?next_maintenance_task`
`WHERE {`
`  VALUES ?current_maintenance_task`
`  { spo:maint_task_isolate_conveyor }`
`  ?next_maintenance_task spo:directlyAfter`
`  ?current_maintenance_task .`
`}`
	This query gets the task that occurs directly after `maint_task_isolate_conveyor`. Our ontology has been populated using the `directlyBefore` relationship, therefore reasoning has been used to retrieve the property's inverse (`directlyAfter`). Note that this query could be rewritten so that reasoning is not used. However, we have chosen to use `directlyAfter` so that the vocabulary in the query matches the competency question.

Table 6
(Continued)

Technician competency questions	
Query	**Explanation**
4. Does my assigned procedure have any safety hazards that I need to be aware of?	

`SELECT ?hazard`
`WHERE {`
`  VALUES ?procedure_process { spo:procedure_process_001 }`
`  ?hazard_realization_process iso:occursRelativeTo ?procedure_process .`
`  ?hazard_realization_process iso:realizes ?hazard .`
`  ?hazard a spo:Hazard`
`}`
	This query selects all hazards that are realized in a `hazard_realisation_process` that occurs relative to `procedure_process _001` or in any of its sub-processes. This query uses the SWRL rule defined in Section 5.1.7 to ensure that hazards realized at any level of the task hierarchy are captured.

5. What corrective action does my procedure suggest on observation of a failure mode in my inspection?	

`SELECT ?corrective_maintenance_task`
`WHERE {`
`  ?corrective_maintenance_task cmto:addresses cmto:functional_failure_001 .`
`  ?corrective_maintenance_task iso:activityPartOf spo:procedure_process_001;`
`  a cmto:CorrectiveMaintenanceTask .`
`}`
	This query selects all individuals of type `Corrective Maintenance Task` that address a `functional_failure_001` and are an activity part of `procedure_process_001`. Since `Corrective Maintenance Task` is a defined class in OMPD, reasoning is required to select all individuals of this type.

Table 7
SPARQL queries for engineer competency questions

Engineer competency questions	
Query	**Explanation**
6. There has been a change in the regulations and an existing permit needs to be modified. Which procedures use this permit and can I update the relevant procedures?	
`DELETE { spo:permit_001 ompd:requirementOf ?maintenance_process. spo:permit_001 iso:hasRole spo:resource_role . } INSERT { spo:permit_002 ompd:requirementOf ?maintenance_process. spo:permit_002 iso:hasRole spo:resource_role . } WHERE { spo:permit_001 ompd:requirementOf ?maintenance_process. ?maintenance_process a spo:MaintenanceProcess }`	This query selects all maintenance processes where `permit_001` is used in the process and replaces the permit with `permit_002`. The query uses class subsumption so that permits are replaced regardless of being represented at the procedure or task level. This query also assigns a `ResourceRole` to `permit_002`.
7. I would like to know which procedures describe an end of life event for my equipment. Which of my procedures contain a "replacement" task?	
`SELECT DISTINCT ?procedure_process ?task ?text_value WHERE { ?task_description spo:hasTextValue ?text_value. ?task_description iso:isAbout ?task. ?task iso:activityPartOf ?procedure_process. ?procedure_process a spo:MaintenanceProcedureProcess FILTER (contains(str(?text_value),'replace')) }`	This query finds all maintenance task descriptions that contain the word "replace" and selects the procedure processes that these tasks are part of. This query shows how ontologies can work alongside natural language processing techniques to answer complex queries that are currently not possible in industry.

Table 7
(Continued)

Engineer competency questions	
Query	**Explanation**
8. Does my inspection procedure check all the failure modes outlined in the Failure Modes and Effects Analysis (FMEA) that was used in my RCM?	

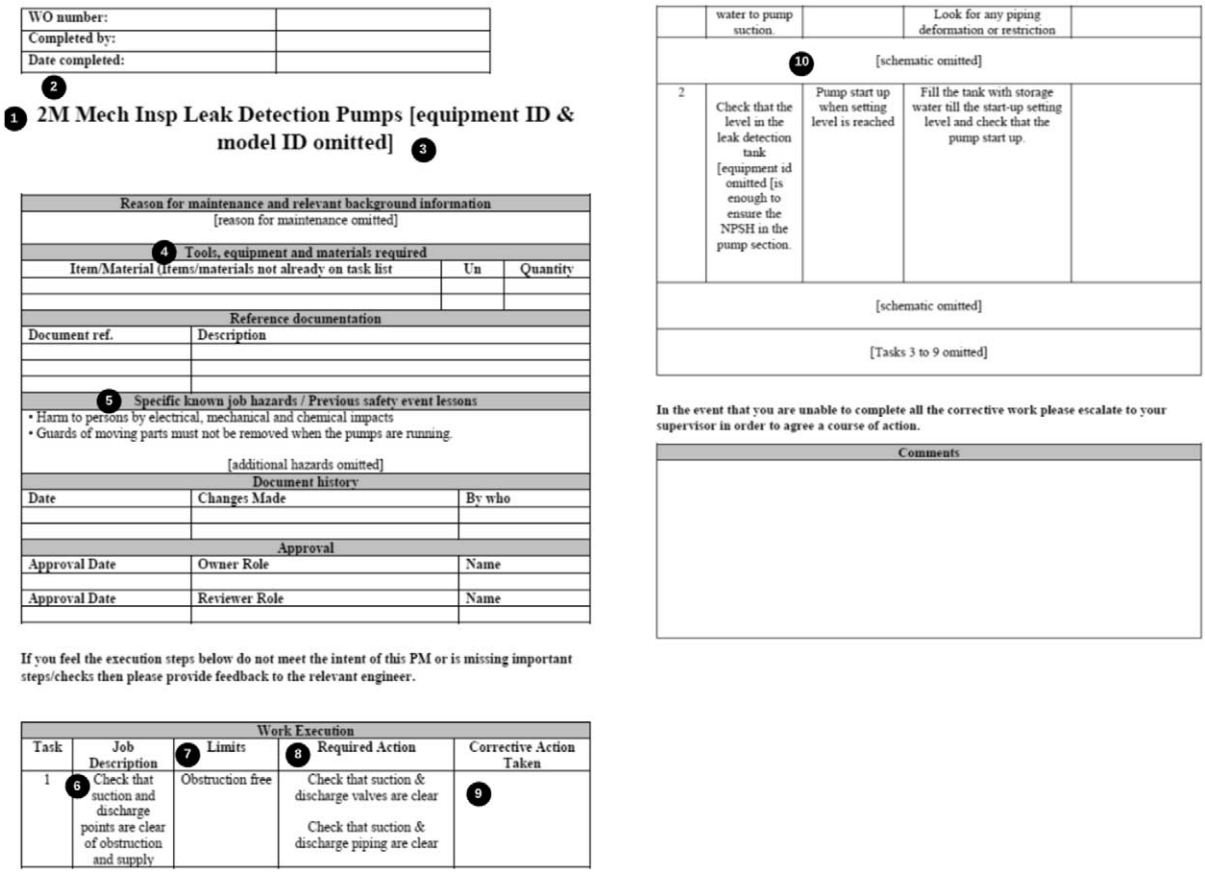
`SELECT ?functional_failure ?maintainable_item`
`WHERE {`
`  VALUES ?failure_modes_in_fmea { cmto:NOI } .`
`  VALUES ?maintainable_item { cmto:maintainable_item_001 } .`
`  VALUES ?procedure_process { cmto:procedure_process_001 }`
`  ?functional_failure cmto:addressedBy ?corrective_maint_task; a cmto:FunctionalFailure .`
`  ?corrective_maint_task iso:activityPartOf ?procedure_process .`
`  ?failure_mode iso:isAbout ?functional_failure`
`  FILTER NOT EXISTS {`
`  ?functional_failure iso:representedIn ?failure_modes_in_fmea`
`  }`
`}`
	This query takes three inputs, these are (1) a list of `Failure Mode Observations` from a FMEA for a given maintainable item, (2) a `Maintainable Item`, and (3) a `Procedure Process` (i.e. an inspection procedure). The query selects all functional failures that are addressed by corrective maintenance tasks in the procedure process. It will then filter out all functional failures that do not have a corresponding entry in the FMEA (given as input 1).

Table 8

SPARQL queries for scheduler competency questions

Scheduler competency questions	
Query	**Explanation**
9. What resources to I require to execute this week's procedures?	

`SELECT ?resource {`
`  VALUES ?procedure_process { spo:procedure_process_001 spo:procedure_process_002 }`
`  ?resource iso:participantIn ?procedure_process; a spo:Resource`
`}`
	This query is similar to Competency Question 1. However, an additional reasoning capability is demonstrated. That is, it selects individuals of type Resource which is a defined class in our ontology.
	Note that the concept of "this week" is not captured in this query. This is due to the static nature of OMPD (explained in Section 4.3). Instead, we assume that the query writer knows what procedures need to be executed "this week" and use this as an input to the query.



WO number:	
Completed by:	
Date completed:	

1 2M Mech Insp Leak Detection Pumps [equipment ID & model ID omitted] 3

Reason for maintenance and relevant background information
[reason for maintenance omitted]

4 Tools, equipment and materials required		
Item/Material (Items/materials not already on task list	Un	Quantity

Reference documentation	
Document ref.	Description

5 Specific known job hazards / Previous safety event lessons
• Harm to persons by electrical, mechanical and chemical impacts • Guards of moving parts must not be removed when the pumps are running. [additional hazards omitted]

Document history		
Date	Changes Made	By who

Approval		
Approval Date	Owner Role	Name
Approval Date	Reviewer Role	Name

If you feel the execution steps below do not meet the intent of this PM or is missing important steps/checks then please provide feedback to the relevant engineer.

Work Execution				
Task	Job Description	7 Limits	8 Required Action	Corrective Action Taken
1	6 Check that suction and discharge points are clear of obstruction and supply	Obstruction free	Check that suction & discharge valves are clear Check that suction & discharge piping are clear	9
	water to pump suction		Look for any piping deformation or restriction	
10 [schematic omitted]				
2	Check that the level in the leak detection tank [equipment id omitted [is enough to ensure the NPSH in the pump section.	Pump start up when setting level is reached	Fill the tank with storage water till the start-up setting level and check that the pump start up.	
[schematic omitted]				
[Tasks 3 to 9 omitted]				

In the event that you are unable to complete all the corrective work please escalate to your supervisor in order to agree a course of action.

Comments

Fig. 13. Numbered procedure used for mapping to OMPD.

Mapping Description: In Fig. 14, we show that the maintenance procedure in Fig. 13 corresponds to one individual of type `Maintenance Procedure Document` (1). This individual (`procedure_document_2M_ mech_inspection`) is about a `Maintenance Procedure Process` of the same name. This name has been extracted from the example procedure's title. The `Maintainable Item` to be inspected in the procedure is also described in the title of the document (3). For confidentiality reasons, the equipment ID and the model ID of this maintainable item has been omitted. A placeholder individual of type `Maintainable Item` (`maintainable_item_1`) has been created to capture this information. Already, benefits of OMPD can be realised. Engineers, schedulers and technicians will no longer need to trawl through PDF documents to find those that have a specific equipment identifier in their title. Instead, a query can retrieve a maintainable item find all instances of `Maintenance Procedure Process` that this maintainable item participates in.

The procedure document also contains two lines describing hazards associated with the procedure's execution (5). To map to OMPD, we have transformed the first row into three individuals of type `Hazard` (i.e. `hazard_harm_to_persons_by_electrical_impacts`) and included the second row as a fourth individual. The mapping shown in Fig. 14 displays how these hazards are realised in the `procedure_process_2M_Mech_ inspection` individual. This implementation captures the procedure-level hazard representation implemented by this process plant.

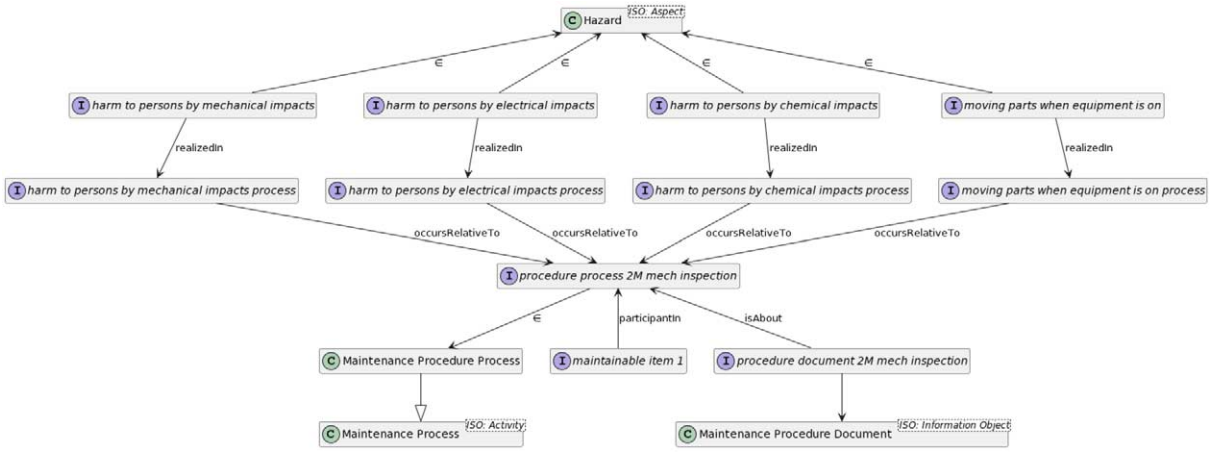


Fig. 14. Hazards and maintainable item mapped from company 1's procedure.

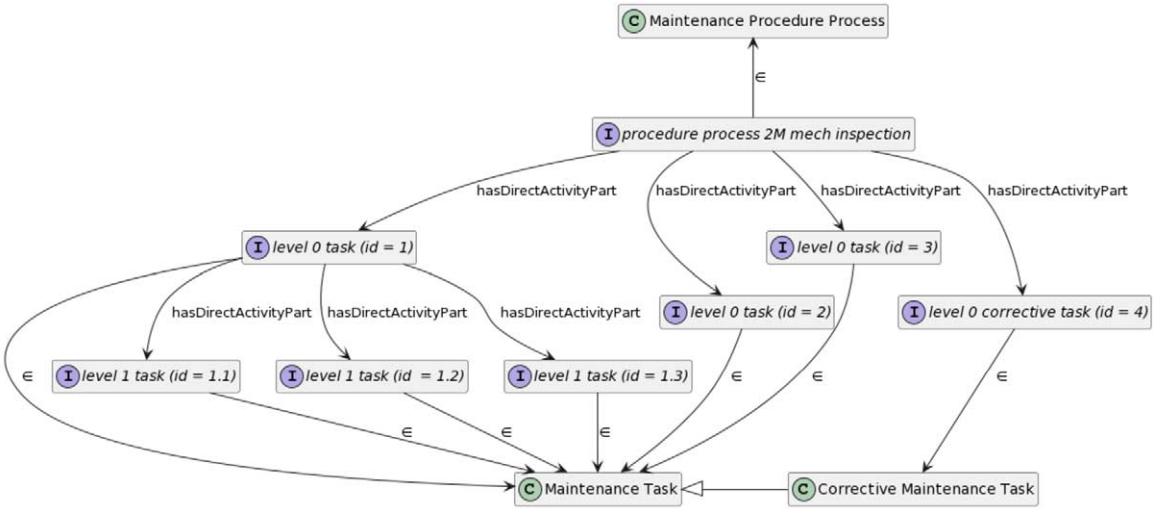


Fig. 15. Task hierarchy and corrective maintenance tasks mapped from company 1's procedure.

Another table, named "Work Execution" exists on the first page of the document in Fig. 13. This table contains `Maintenance Tasks`. Figure 15 shows how this table maps to OMPD's generic task hierarchy. The generated hierarchy has two levels. Level 0 of the hierarchy contains high-level job descriptions (6). Level 1 of the task hierarchy contains a finer breakdown of each job description. This information is given in the "required action" column of the work execution table (8). For each task, the documentation contains unstructured text and (for steps 1 and 2) images that provide further information about the task. These values have been assigned to a `Task Description` individual as shown in Fig. 16. The content is captured using the `hasText` and `hasImageUrl` data properties in OMPD.

Further tasks may (optionally) be required if a "limit" (i.e. "obstruction free") (7) is not met. In the example procedure, there is a column called "corrective action taken" (9) but there are no task descriptions for these actions. Rather they leave the corrective action up to the technician that sees the issues. For example, the technician may see an obstruction in the pump when performing Task 1 and may write "raised

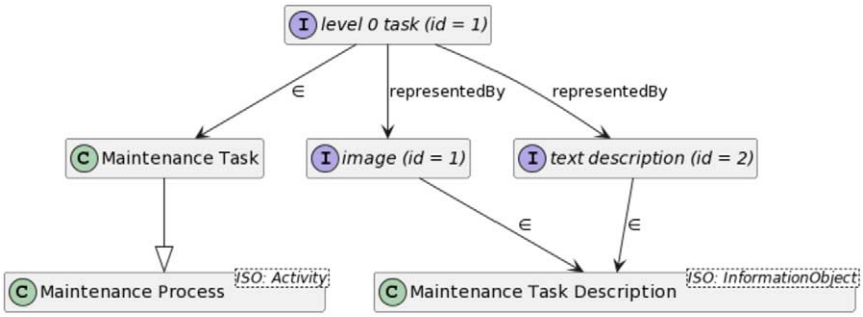


Fig. 16. Task descriptions mapped from Company 1's procedure.

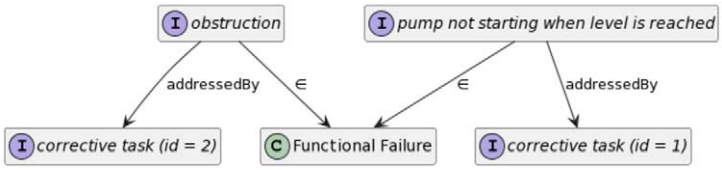


Fig. 17. Functional failures mapped from Company 1's procedure.

a subsequent notification" in that field. In OMPD, this field (8) corresponds to a `Maintenance Task` that does not have a corresponding `Maintenance Task Description` in the document. Due to the "static" philosophy of OMPD, we do not need to model whether or not this task is actually completed in practice. If necessary, OMPD can be imported into application-level ontologies that captures this information. Rather, we treat these "empty" tasks as Level 0 tasks that occur `directlyAfter` the task described in (6).

The final column to be mapped from the "Work Execution" table is "Limits" (8). We have taken the antithesis of the text in this column to create two `Functional Failure` individuals in OMPD. For example, we changed "Obstruction free" to "Obstruction". We relate these two `Functional Failures` to the their corresponding corrective maintenance task via a `addressed by` relationship. These mappings are shown in Fig. 17.

Mapping Limitations: This exercise requires manual extraction of the data from the PDF document into the ontology. To extract this information automatically, natural language processing will need to be used to interpret the information. Automatic extraction is under consideration in the maintenance and technical language processing communities (Wu et al., 2022) but is out of scope for the present paper. The next company that we examine automatically extracts information from PDF procedures with sophisticated proprietary software. The extracted procedures are stored in a relational database schema.

6.2.2. Company 2: Procedure curation company

Further evaluation of OMPD is performed using data from a procedure curation company. This company extracts information from PDF procedures, like those presented in the previous sections, using a hybrid (automated and manual) approach. The extracted information is stored in their own relational database schema. They have kindly provided us with access to a development database that contains thousands of procedures from various companies in their relational format. Since this database is for development, not all of these procedures are production-ready. Some procedures are duplicates and some contain incomplete, null or "test" values. Company 2 differs from Company 1 because they already

Table 9
Summary statistics for Company 2's procedure data

Statistic Name	Statistic Value
Number of procedures	4405
Task hierarchy depth	2
Mean Number of level 0 tasks per procedure	39
Mean Number of level 1 tasks per level 0 Task	3
Max number of level 1 tasks	646

have a digital representation for procedures. Regardless, the reasoning capability of OMPD and the ability to answer competency questions without custom scripts and complex SQL queries is valuable for Company 2.

In this section, we describe how we map this data to OMPD using Owlready2 (Lamy, 2017). We cannot share the complete relational database schema or the data for confidentiality reasons. Instead, we give an overview of the data and a description of the successes and limitations discovered in mapping this data to OMPD. Table 9 gives a summary of the data contained in Company 2's database. We have counted the number of "level 0" tasks and "level 1" tasks to show the size and complexity of procedures in the database. How the records in the database are mapped to "level 0" or "level 1" is explained in this section.

The database contains 5350 "strategy task" records. These records represent activities that need to be performed, according to a maintenance strategy. Of these activities, 4405 have associated "work instructions" (i.e. a procedure). For each "work instruction" record, a `maintenance procedure document` and corresponding `maintenance procedure process` individual is created using an Owlready script. Converting a single "work order" record into these two concepts means that data owners add data properties to the `maintenance procedure document` capture metadata from the "work orders" table (i.e. createdAt, updatedAt). This gives a conceptual separation between the documentation (i.e. the actual database record in this case), and the process that the record describes, related to the `maintenance procedure process`.

Company 2's database has two tables that represent tasks in a procedure. These tables are:"job operations", and "step". A job operation is a repeatable group of steps. In OMPD, these tables correspond to a task hierarchy of two levels (where job operations are "level 0" tasks and steps are "level 1" tasks). On inspection of the text representation of steps in the "step" table, we notice that it is possible to break these steps into further parts by separating the text on full stops. However, we have kept the hierarchy at two levels for simplicity. For each row in the "job operations" table and for each row in the "steps" table, a `Maintenance Task` was created is Owlready2. In Company 2's database, job operations are reused in multiple procedures. However, in OMPD, we have to create a new task for each task/job operation and indicate that tasks are the same using our `locallyEquivalentTo` property. Although this creates extra data and may seem redundant, we describe the reasoning for this in Section 5.1.2.

Similar to Company 1, resources are represented at the procedure level in Company 2's database.The resource types in Company 2's database are tool, spare part (`Component` in OMPD), permit, skill (`Qualified Agent` in OMPD) and personal protective equipment (PPE). PPE is the only resource that does not have its own class in OMPD. Further work is required to determine if PPE is conceptually different to a `Tool` in OMPD. For the purpose of this mapping, records in the PPE table mapped to the `Tool` class.

We will explain the mapping of resources from Company 2's database using a hypothetical scenario. Assume that we have a procedure that requires two spanners and a welding machine. In the database,

this information exists as two records. One for record "spanner" with a quantity of 2, and one for record "welding machine" with a quantity of 1. In the database, if any procedure requires two spanners, they will contain a foreign key relationship to this "spanner" record with a quantity of 2. To map this example to OMPD, three unique individuals of type `Tool` are created. In OMPD, these tools are a `participant in` the procedure. Note that a `Tool` is only a `Resource` in OMPD if it is the `hasRole some Resource Role`. For this example, we create two individuals of type `Resource Role`. These are `spanner role` and `welding machine role`. The two spanners have the `spanner role` and the welding machine has a `welding machine role`. During this mapping, we noticed an interesting benefit of this pattern. We can now query to say, "how many of my procedures use a spanner" (i.e. an individual that has the `spanner role`). In this example, we have mapped the required tools for each procedure, while retaining quantity information.

While `Resources` in Company 2's database are represented at the procedure level, `Hazards` associated with a procedure are represented at the task level (the second level of the task hierarchy). As discussed in Section 5.1.7, OMPD can support this. In Owlready2, we iterate through all `Hazards` from the database and create a unique individual of type `Hazard` and `Hazard Realization Process`. We then create a `occursRelativeTo` relationship between the hazard realization process and one or more `MaintenanceTask` individuals. As discussed in Section 5.1.7, the ontology can be queried to check which procedures have which hazards.

On top of `Hazards`, there are some cases in Company 2's database where "additional information" or "advice" is appended to a step. This includes specific information such as torque settings and the weight of an object. In our mapping, we simply include this information as a `Maintenance Task Description` alongside the task's text description. However, this company may wish to create an application-level ontology module that can capture this information more thoroughly.

In this exercise, we demonstrated the applicability of OMPD to a different data source than that described in Section 6.2.1. While we did see some practical limitations such as the ontology's inability to fully capture the "advice" table in Company 2's database, most of the concepts in Company 2's database could be represented. Further work alongside Company 2 will involve an implementation of OMPD on their development servers and an exploration of the practicalities of OMPD when in-use.

7. Discussion

7.1. Comparison with related ontologies

This section contains a concept-level comparison of OMPD with three related ontologies that were introduced in Section 3. The purpose of this is to analyse where OMPD sits in relation to these previous efforts and highlight the material impacts of different ontological choices.

Ontology 1. Procedure Representation Language (Kortenkamp et al., 2008). The Procedure Representation Language (PRL) was developed in 2008 by NASA and is defined as an XML schema. PRL was designed to shift NASA spacecraft operations gradually towards automation. Automation is also a motivation for the generic task hierarchy in OMPD, thus PRL is a suitable ontology for comparison. As reasoning capability was not a primary goal of the authors of PRL, we will perform a concept comparison between the two ontologies.

In PRL, the top-level entity is a `Procedure` that has a human-readable title, can contain metadata such as "author, comments, revisions, etc" and "has as its body one or more steps" (Kortenkamp

et al., 2008). In PRL, there is no separation between the `Procedure Document` (an information content entity with an author, revisions, etc) and a `Procedure Process` (an activity containing a series of steps). In OMPD, the classes `Maintenance Procedure Document` (a subclass of `iso:Information Content Entity`) and `Maintenance Procedure Process` (a subclass of `iso:Activity`) capture this separation.

PRL is intended to be useable by both humans and machines. `Steps` in PRL (called `Maintenance Tasks` in OMPD) are not organised in a hierarchical structure. Therefore, human-readable text associated with each step can only be represented at one level of detail. Since the steps will need to be at a level of detail understandable by machines, this human-readable text is likely far too detailed for an experienced technician. The generic task hierarchy in OMPD combats this issue. Using this feature of the ontology, data-owners are able to choose the level of granularity required for a specific application.

In PRL, a `Block` is a set of `Instructions` that is required to accomplish a step. These blocks can be "ordered" or "unordered". This feature provides flexibility in how blocks are executed and is not currently supported in OMPD. In OMPD, a linear sequential structure is enforced on `Maintenance Tasks` (discussed in Section 5.1.4). We recognise that, in the future, as industries move towards automated machinery, this could be a useful feature of OMPD and should be the subject of future enquiry. However, this is not a feature of the existent industrial maintenance procedures demonstrated in this paper. Therefore, asynchronous task execution is currently an unnecessary competency for OMPD.

PRL is a powerful representation language. NASA has put much thought into the information that needs to be captured for machines and humans to work alongside each other in a semi-automated way. We design OMPD to meet a different industrial need (described in Sections 2 and 3) and we aim to represent the static information that is *currently* stored in maintenance procedure documentation in industry. However, in the future, as further temporal modules are added to OMPD, much inspiration can be drawn from PRL.

Ontology 2. A procedure ontology for advanced diagnosis of process systems (Németh et al. (2010)). Németh et. al's ontology (Ontology 2) was designed for process plants, the same domain as the procedures in Figs 1 and 2. Ontology 2, however, captures operating, safety and control procedures in these process plants rather than maintenance procedures. Ontology 2 models the phenomena where functional failures cause an operating procedure to cease execution. When an equipment failure causes a procedure to "stop", Ontology 2 helps data-owners to understand the cause of the failure. To do this, Ontology 2 captures a link between operating procedures and FMEA.

This link to FMEA is significant because we identify a similar link in our design of OMPD (discussed in Section 5.2). However, unlike operating procedures, maintenance procedures are intended to identify equipment failures or fix equipment when a failure has already occurred. This is fundamentally different to Ontology 2 because these equipment failures do not mean that the procedure execution will cease. For example, when performing an inspection procedure for pumps, a technician could find a leak in the pump and take the "corrective action" of informing their supervisor. The technician will then continue the inspection procedure to check for other potential failures.

Ontology 2 relies on relationships such as `hasSteps`, `hasAim` and `hasComponentOrState`. This is a common consequence of not conforming to an upper ontology and greatly limits the ontology's re-usability. These relationships also limit the semantic information stored in the ontology and makes it difficult for new users to query the ontology. In addition, the Ontology 2 relies heavily on data properties such as `status` and `action type`. If the various action types and status options are represented as classes or individuals in the ontology instead, it greatly improved the reasoning capability of the ontology.

Ontology 2 demonstrates the capability of ontologies to integrate disparate industrial data sources. This integration can lead to new insights that are not currently possible in industry. However, a difference in perspective means that the applicability of Ontology 2 for our use case is limited. Furthermore, it is difficult to re-use parts of the ontology as it is not aligned to an upper ontology.

Ontology 3. Rule-based Mechanism to Optimise Asset Management Using Technical Documentation Ontology (Koukias and Kiritsis, 2015). Koukias and Kiristsis's ontology (Ontology 3) is designed to capture industrial technical documentation. The use case for Ontology 3 is to ensure that both humans and computers have the same understanding of the contents of technical documentation. This use case is similar to that of PRL (Ontology 1) but Ontology 3 was also designed to support semantic reasoning. Ontology 3 was chosen for analysis because it contains many concepts that are present in OMPD.

Unlike Ontology 1 and Ontology 2, this Ontology 3 models `Resources` and `Accessories` that are used in the procedure (similar to OMPD). However, it is unclear what separates an `Accessory` from a `Resource` and whether this distinction is necessary for its use case. These classes are also asserted classes in Ontology 3. As discussed in 6.1.5, this is likely to introduce difficulty when individuals are added to the ontology.

Similar to Ontology 2, Ontology 3 does not conform to an upper ontology thus relies on many custom object properties. These include `uses_accessory` and `uses_resource` and this makes it hard to extend or reuse the ontology. Instead, OMPD uses more generic relationships. For example, `participantIn` relationship from ISO 15926 Part 14 links resources to a procedure process.

This problem is further realised in the axiom, `Procedure performs_activity Activity`. This implies that the procedure itself performs an activity. In reality some agent will perform that activity. This agent may be a person, computer program or piece of automated equipment). In OMPD, the relationship between a `Maintenance Process` and a `Maintenance Activity` is `hasActivityPart` from ISO 15926 Part 14. This relationship and its mapping to our upper ontology has enabled much of the reasoning that we describe in Section 6.1.

Ontology 3 is similar to OMPD because it models some of the same concepts. Capturing resources and their relationship to technical documentation is especially useful for industrial users. However, narrow custom relationships that are not aligned to an upper ontology limits its applicability to OMPD's use case.

On Task Preconditions and Postconditions. All three examined ontologies have the concept of a `Precondition` for some (or all) tasks. Ontology 1 and Ontology 2 use preconditions to link tasks together. Ontology 3 uses `has_next_element` to do this but has a class called `Conditional Branch` that relies on a precondition. In our experience, this information is not stored in existing maintenance procedures used in industry, therefore it has not been included in OMPD. Furthermore, executing a procedure according to preconditions assumes that the procedure reader has information about the live-on goings of the maintenance environment. However, if further ontology modules are added to OMPD to capture additional data on top of what is currently stored in maintenance procedures in industry, the three ontologies can be used to inform these decisions.

7.2. Contributions for the applied ontology community

The presented ontology contains novelties that stem from our commitment to create a model that considers the (sometimes conflicting) goals of both industry and academia. We balance rigorous ontological modelling with an understanding of what our users will accept and use in practice. In this section,

we summarise the implications that this modelling perspective had on the outcomes of this work. Our user-focused perspective creates implications for the ontology's scope, schematic design and degree of axiomatisation. Each of these implications contain ontological questions for future consideration in the applied ontology community.

When determining the scope of OMPD, we consider our design goal of modelling *information currently stored in maintenance procedure documentation in industry*. Maintenance procedure documents are engineering artifacts where the level of abstraction has been given a-priori. The information held in these documents has been optimised over time, and tested in practice as they are used in these organisations. The information is accessible to engineers, and usable by schedulers and technicians. When analysing the data contained in these procedures, we uncovered interesting omissions; knowledge that we might expect these documents to contain from an ontological perspective, and knowledge we may consider to best practice in a process specification, is missing. For example, no information about task pre- and post-conditions is included in these documents. This is likely due to their inherent uncertainty. For instance, if a pump is serviced, a post-condition that we may like to specify is that the pump is fault-free. In practice this might not be the case (where the pump might be more likely to fail if it has not been put back together properly after a service). From an ontological perspective, it would be nice to build a full causal-physical model of an engineer's mental model surrounding procedures. However, our analysis has shown us that this information is not accessible to engineers and requiring this information for an OMPD implementation would be an enormous barrier to the industrial acceptance of our model. This design implication (based on our perspective throughout this paper) is a demonstration of a potential disconnect between ontology engineering processes and real-world use of data.

In Section 5.1.2, we discuss *task identity* and *task local equivalence*. We examine whether a `Maintenance Task` should be considered separately from its context within a procedure. We describe what we call the "identity approach" and the "local equivalence approach". Both approaches are informationally similar, but have significant impacts on how information is mapped to the ontology, and how the ontology is used in practice. In OMPD we use the "local equivalence" approach despite the data-redundancy trade-offs. We show that the "local equivalence" approach reduced the schematic-complexity of OMPD and matches the mental model of the engineer (where procedures are considered in isolation). For readers interested in processes and identity, this discussion raises interesting ontological issues that can be the subject of future work.

Finally, consideration of the end-users impacted the degree of axiomatisation in OMPD. In many cases, we rely on the assumptions of the foundational ontology and only place constraints where we deem they are necessary. This had an impact on the assertions that we could make about various concepts. For example, in Table 3 there are *Tools*, *Permits* and *Materials*. Each of these are described as `Objects` that have some `ResourceRole`. No other axioms are included to further describe what separates a `Tool` from a `Material`. Ontologically, we would like to ensure that these concepts are distinctive from one another to ensure that classes are not misused. However, the number of necessary constraints for an item to be considered a `Tool` or a `Material` in one organisation is likely different to another organisation. In one organisation a tool might need an OEM (Original Equipment Manufacturer). This restriction may be different in another organisation. We cannot make a decision about what separates a Tool from a Material at a domain-ontology level, because we risk excluding large portions of our user base. Despite being a domain-level ontology, some of the concepts read more like foundational-level concepts. This work raises interesting issues about the level of axiomatisation expected at certain levels of ontological modelling and how these expectations align with the real-world use of the ontology.

8. Conclusion and future work

In this paper, we have introduced a design for OMPD and demonstrated its applicability to real world industry use cases. OMPD conforms to the ISO 15926 Part 14 upper ontology to ensure that it is *generic* and reusable across many industrial organisations. Having a "static" philosophy to guide its design, we ensure that OMPD models information that is *currently stored in procedure documentation in industry*. Finally we have demonstrated competency questions to answer typical queries asked by three core roles within a maintenance team, *technicians, engineers and schedulers*.

The broader focus of our research is to find the best ways to design and build user interfaces to be used by technicians who execute maintenance procedures in their work. One of the key requirements that has emerged in our analysis is that ontologies should be able to "adapt" to a user's needs (i.e. domain expertise). However, from our exploration and first-hand experience, we have discovered that industrial procedure data is not currently in a state where these types of developments are possible. This ontology is a step towards representing industrial maintenance procedures in a rigorous, standardised manner. In future work, we intend to integrate this ontology with an adaptive user interface for maintenance procedures. We can use the ontology's generic task hierarchy to display tasks at different levels of granularity to the users with different levels of domain expertise.

Despite our intended use of the ontology, OMPD has been designed such that it can be used by any organisation that currently manage maintenance procedures, regardless of their technological goals. Further avenues for future work include an examination of the temporal aspects of procedures and creating a OWL-Lite version of the ontology to further support industrial use.

Machine-readable procedures will improve the utility, findability and maintainability of procedures that are already being used in industry every day. This digital reform of procedures is a positive and necessary step towards preparing organisations for Industry 4.0 ready maintenance systems.

Acknowledgements

This research is supported by an Australian Government Research Training Program (RTP) Scholarship. The authors would like to acknowledge a mining company for access to industrial maintenance procedures to assist in the work reported in this paper. The authors would also like to acknowledge the directors at OnPlan Technologies Pty Ltd for access to industrial data used to inform this paper. Finally, this work would not have been possible without funding from the BHP Fellowship for Engineering for Remote Operations – supporting community projects in areas in which BHP operates.

References

Arp, R., Smith, B. & Spear, A.D. (2015). *Building Ontologies with Basic Formal Ontology*. MIT Press.

Batres, R., West, M., Leal, D., Price, D., Masaki, K., Shimada, Y., Fuchino, T. & Naka, Y. (2007). An upper ontology based on ISO 15926. *Computers & Chemical Engineering*, *31*(5–6), 519–534. doi:10.1016/j.compchemeng.2006.07.004.

EFNMS (2017). *Maintenance – Maintenance Terminology. Standard En 13306, the European Federation of National Maintenance Societies*.

Ferrario, R. & Grüninger, M. (2020). Proposed guidelines for publishing ontology papers. *Applied Ontology*, *15*(1), 1–5. doi:10.3233/AO-200227.

Fiorentini, X., Paviot, T., Fortineau, V., Goblet, J.-L. & Lamouri, S. (2013). Modeling nuclear power plants engineering data using ISO 15926. In *Proceedings of 2013 International Conference on Industrial Engineering and Systems Management (IESM)* (pp. 1–6). IEEE.

Forssell, J.H., Lupp, D.P., Skjæveland, M.G. & Thorstensen, E. (2017). Reasonable macros for ontology construction and maintenance. In *CEUR Workshop Proceedings* (Vol. 1879). Technical University of Aachen.

Gangemi, A. (2010). Submissions: Sequence. http://ontologydesignpatterns.org/wiki/Submissions:Sequence.

Gibaud, B., Forestier, G., Feldmann, C., Ferrigno, G., Gonç, P., Haidegger, T., Julliard, C., Katić, D., Kenngott, H., Maier-Hein, L., et al. (2018). Toward a standard ontology of surgical process models. *International journal of computer assisted radiology and surgery*, *13*(9), 1397–1408. doi:10.1007/s11548-018-1824-5.

Glimm, B., Horrocks, I., Motik, B., Stoilos, G. & Wang, Z. (2014). HermiT: An OWL 2 reasoner. *Journal of Automated Reasoning*, *53*(3), 245–269. doi:10.1007/s10817-014-9305-1.

Gruninger, M. & Menzel, C. (2003). In *The Process Specification Language (PSL) Theory and Applications*. AI Magazine (Vol. 24, pp. 63–63).

Hitzler, P. & Krisnadhi, A. (2018). A tutorial on modular ontology modeling with ontology design patterns: The cooking recipes ontology. ArXiv preprint. arXiv:1808.08433.

Hodkiewicz, M., Klüwer, J.W., Woods, C., Smoker, T. & French, T. (2020). Digitalization and reasoning over engineering textual data stored in spreadsheet tables. *IFAC-PapersOnLine*, *53*(3), 239–244. doi:10.1016/j.ifacol.2020.11.039.

Hodkiewicz, M., Klüwer, J.W., Woods, C., Smoker, T. & Low, E. (2021a). An ontology for reasoning over engineering textual data stored in FMEA spreadsheet tables. *Computers in Industry*, *131*, 103496. doi:10.1016/j.compind.2021.103496.

Hodkiewicz, M., Lukens, S., Brundage, M.P. & Sexton, T. (2021b). Rethinking maintenance terminology for an industry 4.0 future. *International Journal of Prognostics and Health Management*, *12*(1). doi:10.36001/ijphm.2021.v12i1.2932.

IEC (2016). *Dependability Management – Maintenance and Maintenance Support. Standard AS IEC 60300.3.14*. Geneva, Switzerland: International Electrotechnical Commission.

International Organization for Standardization (2018a). ISO/TS 15926-12 Industrial automation systems and integration – Integration of life-cycle data for process plants including oil and gas production facilities – Part 12: Life-cycle integration ontology represented in Web Ontology Language (OWL). Technical report ISO/TS 15926-12:2018.

International Organization for Standardization (2019). ISO 15926-14: Industrial automation systems and integration – Integration of life-cycle data for process plants including oil and gas production facilities – Part 14: Data model adopted for OWL 2 Direct Semantics. *Community Draft.*

ISO/TC184/SC4/WG3 (2020). Industrial automation systems and integration – Integration of life-cycle data for process plants including oil and gas production facilities – Part 14: Industrial top-level ontology. Working Draft Proposal, https://readi-jip.org/wp-content/uploads/2020/10/ISO_15926-14_2020-09-READI-Deliverable.pdf.

Jarrar, M. & Ceusters, W. (2017). Classifying processes and basic formal ontology. In *Proceedings of the International Conference on Biomedical Ontology (ICBO 2017)*.

Jordan, A., Selway, M., Grossmann, G., Mayer, W. & Stumptner, M. (2014). Ontology-based process modelling for design. In *Design Computing and Cognition (DCC'14)* (pp. 1–20). Springer.

Kanse, L., Parkes, K., Hodkiewicz, M., Hu, X. & Griffin, M. (2018). Are you sure you want me to follow this? A study of procedure management, user perceptions and compliance behaviour. *Safety Science*, *101*, 19–32. doi:10.1016/j.ssci.2017.08.003.

Karray, M., Otte, N., Rai, R., Ameri, F., Kulvatunyou, B., Smith, B., Kiritsis, D., Will, C., Arista, R., et al. (2021). The industrial ontologies foundry (IOF) perspectives.

Karray, M.H., Ameri, F., Hodkiewicz, M. & Louge, T. (2019). ROMAIN: Towards a BFO compliant reference ontology for industrial maintenance. *Applied Ontology*, *14*(2), 155–177. doi:10.3233/AO-190208.

Katsumi, M. & Grüninger, M. (2016). What is ontology reuse? In *Formal Ontologies in Information Systems FOIS* (pp. 9–22).

Kiritsis, D. (2013). Semantic technologies for engineering asset life cycle management. *International Journal of Production Research*, *51*(23–24), 7345–7371. doi:10.1080/00207543.2012.761364.

Klüwer, J.W., Skjæveland, M.G. & Valen-Sendstad, M. (2008). ISO 15926 templates and the Semantic Web. In *Position Paper for W3C Workshop on Semantic Web in Energy Industries; Part I: Oil and Gas*.

Kortenkamp, D., Bonasso, R.P., Schreckenghost, D., Dalal, K.M., Verma, V. & Wang, L. (2008). A procedure representation language for human spaceflight operations. In *Proceedings of the 9th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS-08)*.

Koukias, A. & Kiritsis, D. (2015). Rule-based mechanism to optimize asset management using a technical documentation ontology. *IFAC-PapersOnLine*, *48*(3), 1001–1006. doi:10.1016/j.ifacol.2015.06.214.

Kwon, S., Kim, B., An, K., Ryu, D., Mun, D. & Han, S. (2018). Standardized exchange of plant equipment and materials data based on ISO 15926 methodology in nuclear power plants. *Annals of Nuclear Energy*, *118*, 185–198. doi:10.1016/j.anucene.2018.04.001.

Lamy, J.B. (2017). Owlready: Ontology-oriented programming in Python with automatic classification and high level constructs for biomedical ontologies. *Artificial Intelligence in Medicine*, *80*, 11–28. doi:10.1016/j.artmed.2017.07.002.

Lee, J., Bagheri, B. & Kao, H.-A. (2015). A cyber-physical systems architecture for Industry 4.0-based manufacturing systems. *Manufacturing letters*, *3*, 18–23. doi:10.1016/j.mfglet.2014.12.001.

Masolo, C., Borgo, S., Gangemi, A., Guarino, N., Oltramari, A. & Schneider, L. (2003). *The WonderWeb Library of Foundational Ontologies and the DOLCE Ontology*. WonderWeb Deliverable D18, Final Report vr. 1.0.

Moubray, J. (2001). *Reliability-Centered Maintenance*. Industrial Press Inc.

Nagy, L., Ruppert, T. & Abonyi, J. (2021). Ontology-Based Analysis of Manufacturing Processes: Lessons Learned from the Case Study of Wire Harness Production. *Complexity*, *2021*.

Németh, E., Hangos, K.M. & Lakner, R. (2010). A procedure ontology for advanced diagnosis of process systems. *Journal of Intelligent & Fuzzy Systems*, *21*(1, 2), 19–31. doi:10.3233/IFS-2010-0432.

OntoSPM Collaborative Action (2019). Ontologies for Representing Surgical Procedure Models. https://ontospm.univ-rennes1.fr/doku.php?id=ontology.

Pérez, J., Arenas, M. & Gutierrez, C. (2009). Semantics and complexity of SPARQL. *ACM Transactions on Database Systems (TODS)*, *34*(3), 1–45. doi:10.1145/1567274.1567278.

POSC Caesar Association (2022). *PCA Reference Data and Services.* https://rds.posccaesar.org/ontology/lis14/ont/core/.

Ribeiro, R., Batista, F., Pardal, J.P., Mamede, N.J. & Pinto, H.S. (2006). Cooking an ontology. In *International Conference on Artificial Intelligence: Methodology, Systems, and Applications* (pp. 213–221). Springer.

Sinogas, P., Vasconcelos, A., Caetano, A., Neves, J., Mendes, R. & Tribolet, J.M. (2001). Business processes extensions to UML profile for business modeling. In *ICEIS* (Vol. 2, pp. 673–678).

Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A. & Katz, Y. (2007). Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics*, *5*(2), 51–53. doi:10.1016/j.websem.2007.03.004.

Skjæveland, M.G., Forssell, H., Klüwer, J.W., Lupp, D., Thorstensen, E. & Waaler, A. (2019). Pattern-based ontology design and instantiation with Reasonable Ontology Templates. *A Higher-Level View of Ontological Modeling*, *69*.

Skjæveland, M.G., Gjerver, A., Hansen, C.M., Klüwer, J.W., Strand, M.R., Waaler, A. & Øverli, P. (2018). Semantic material master data management at aibel. In *International Semantic Web Conference (P&D/Industry/BlueSky)*.

Smith, B., Malyuta, T., Rudnicki, R., Mandrick, W., Salmen, D., Morosoff, P., Duff, D.K., Schoening, J. & Parent, K. (2013). IAO-Intel: An ontology of information artifacts in the intelligence domain.

Souza, É.F., Falbo, R.A. & Vijaykumar, N. (2013). Ontologies in software testing: A systematic literature review. In *VI Seminar on Ontology Research in Brazil* (p. 71).

Standards Australia (2017). *AS IEC 61882:2017 Hazard and Operability Studies (HAZOP Studies) – Application Guide. Standard AS IEC 61882:2017*. Standards Australias, Sydney, Australia.

Studer, R., Benjamins, V.R. & Fensel, D. (1998). Knowledge engineering: Principles and methods. *Data & Knowledge Engineering*, *25*(1–2), 161–197. doi:10.1016/S0169-023X(97)00056-6.

Tao, F., Zhang, H., Liu, A. & Nee, A.Y. (2018). Digital twin in industry: State-of-the-art. *IEEE Transactions on Industrial Informatics*, *15*(4), 2405–2415. doi:10.1109/TII.2018.2873186.

White, S.A. (2004). *Introduction to BPMN*. IBM Cooperation (Vol. 2).

Woods, C., Griffin, M.A., French, T. & Hodkiewicz, M. (2021b). Using job characteristics to inform interface design for industrial maintenance procedures. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (pp. 1–10).

Woods, C., Selway, M., Hodkiewicz, M., Ameri, F., Stumptner, M. & Sobel, W. (2021a). On the notion of maintenance state for industrial assets. In *CEUR Workshop Proceedings (Vol. 2969)*. Rheinisch-Westfaelische Technische Hochschule Aachen* Lehrstuhl Informatik V.

Wu, H., French, T., Liu, W. & Hodkiewicz, M. (2022). Automatic semantic knowledge extraction from electronic forms. In *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*. arXiv:1748006X221098272.