

On the Notion of Maintenance State for Industrial Assets

Caitlin Woods¹, Matt Selway², Melinda Hodkiewicz¹, Farhad Ameri³,
Markus Stumptner² and William Sobel⁴

¹The University of Western Australia, Crawley Western Australia, 6009, Australia

²University of South Australia, Industrial AI, 1 University Blvd., Mawson Lakes, South Australia, 5095, Australia

³Texas State University, San Marcos, TX 78666, U.S.A

⁴MTCConnect Institute, McLean, VA, 22102, U.S.A

Abstract

Maintenance is vital to ensure our manufacturing assets operate safely and efficiently. Maintenance work is triggered by a number of factors. One of these factors is a change in health *state* of an asset and the impact of this change on the ability of the asset to perform its functions. In addition, the act of performing maintenance changes the health *state* of the asset, restoring or retaining it to a *state* in which it can perform its required function. The ability to perform maintenance can also depend on the *state*, up state or down state, of the asset. The notion of *state* and the precursors and consequence of a change in *state* of an asset are central to developing formal maintenance ontologies and their related reasoning mechanisms to support maintenance management process automation.

This paper explores this notion in a Basic Formal Ontology (BFO) context using an illustrative case study. In modelling a simple example, we have encountered many issues that are under active discussion within the industrial ontology community. We propose a formalization for *maintenance state* and discuss practical and ontological issues. Our main focus is on the role of *maintenance state* in the interplay between function *realization* and process *participation*, conditioned on a *state*. The paper is a call-to-arms to the ontology community to engage with defining a notion of *state* to support the use of ontologies and reasoning to assist in automation of industrial manufacturing processes.

Keywords

state, stasis, maintenance, asset, Basic-Formal Ontology, function, industrial ontology

1. Motivation

The notion of an asset's state is central to maintenance language. Maintenance is defined as "combination of all technical, administrative and managerial actions during the life cycle of an

FOMI 2021: 11th International Workshop on Formal Ontologies meet Industry, held at JOWO 2021: Episode VII The Bolzano Summer of Knowledge, September 11–18, 2021, Bolzano, Italy

✉ caitlin.woods@uwa.edu.au (C. Woods); matt.selway@unisa.edu.au (M. Selway);

melinda.hodkiewicz@uwa.edu.au (M. Hodkiewicz); ameri@txstate.edu (F. Ameri); mst@cs.unisa.edu.au

(M. Stumptner); will@wvsobel.llc (W. Sobel)

🌐 <https://people.unisa.edu.au/Matt.Selway> (M. Selway);

<https://research-repository.uwa.edu.au/en/persons/melinda-hodkiewicz> (M. Hodkiewicz);


<https://people.unisa.edu.au/Markus.Stumptner> (M. Stumptner)

🆔 0000-0001-7829-7674 (C. Woods); 0000-0001-6220-6352 (M. Selway); 0000-0002-7336-3932 (M. Hodkiewicz);

0000-0001-6470-8073 (F. Ameri); 0000-0002-7125-3289 (M. Stumptner); 0000-0002-1358-9139 (W. Sobel)



© 2021 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

item intended to retain it in, or restore it to, a *state* in which it can perform the required function" [1]. Maintenance actions and their timing are determined by the maintenance strategies for an asset and the operating plan for the facility. Manufacturing plants need to plan and execute hundreds sometimes thousands of maintenance actions each month. Typical maintenance actions include inspections, calibrations, repairs, replacements, and modifications. The up or down state of an asset impacts what maintenance task can be executed.

Maintenance actions may be triggered by the change in health *state* of an asset and the impact of this change on the ability of the asset to perform its primary and secondary functions. The ability to perform maintenance also depends on the operating *state*, up state or down state, of the asset. The act of performing maintenance is often, but not always, intended to change the health *state* of the asset, restoring or retaining it to a *state* in which it can perform its required function. An exception to this is an inspection task which does not change the state of the asset.

Historically information about the health of an asset has seldom been available in near real time to maintainers and operators. However, this situation is changing due to rapid developments in industrial internet and analytics associated with Industry 4.0 and prognostics health management initiatives. Both these initiatives hope to improve asset availability and reduce costs through maintaining assets proactively before failure, reducing unnecessary maintenance work on healthy assets, and optimising when asset maintenance actions occur to balance cost, risk and performance. The availability of digital asset health information and the ability to infer a health *state* from sensor data opens the way to automated reasoning about maintenance actions to be taken conditioned on certain health and operating states. For example, a failed state notification could be used to trigger a repair maintenance action. The challenge of linking the outputs of health assessment with maintenance task generation, leveraging automated reasoning, is of interest to the industrial ontology community. The authors are part of the Maintenance Working Group of the Industrial Ontology Foundry (IOF) and engaged in the development of open, shared ontologies aligned with the Basic Formal Ontology (BFO) top-level ontology. It is within this framework that we explore the notion of state in a maintenance context and describe our ontological challenges.

2. Background

We start with identifying terms of interest and their natural language definitions. We draw the definitions from International Standards such as those developed by the International Standards Organisation (ISO), International Electrotechnical Commission (IEC), European Committee for Standardization (CEN) and the Society of Automotive Engineers (SAE). The shared vocabulary provided by formal standards is an important feature in how engineers communicate and these standards are widely used, and in many cases are mandatory, in engineering work. The terms in Table 1 and their definitions inform the development of classes in our proposed ontology but there is not a direct mapping.

2.1. On function, loss of function, and maintenance action

Each asset has a primary function, this describes the main reason(s) for owning or using the asset. It may also have a set of secondary functions due to the need to fulfill regulatory requirements

Table 1

Natural language definitions for asset, function, failure and maintenance related terms

Term	Natural language definition from International Standards	Standard
Physical asset	Item that has potential or actual value to an organization	[1]
Item	Part, component, device, subsystem, functional unit, equipment or system that can be individually described and considered	[1, 2]
Required function	Function, combination of functions, or a total combination of functions of an item which are considered necessary to fulfil a given requirement	[1]
Failure	Loss of the ability of an item to perform a required function	[1]
Degraded state	State of reduced ability to perform as required, but with acceptable reduced performance	[1]
Up state	State of an item being able to perform a required function, assuming that the external resources, if required, are provided	[1]
Down state	State of an item being unable to perform a required function due to preventive maintenance or a fault	[1]
Primary function	Function(s) which constitute the main reason(s) why a physical asset or system is acquired by its owner or user	[3]
Secondary function	Functions which a physical asset or system has to fulfil apart from its primary function(s), such as those needed to fulfil regulatory requirements and those which concern issues such as protection, control, containment, comfort, appearance, energy efficiency and structural integrity	[3]
Maintenance	Combination of all technical, administrative and managerial actions during the life cycle of an item intended to retain it in, or restore it to, a state in which it can perform the required function"	[1]

and requirements concerning issues of protection, control, containment, comfort, appearance, energy efficiency, and structural integrity [3]. Each maintenance action performed on an asset depends on the function to be maintained and the consequence of a functional failure. Each significant asset will have an asset management (AM) plan that sets out what maintenance tasks should be performed and when it should be performed. Additional maintenance work is identified when maintainers perform inspections and find anomalies or operators notice issues such as assets not performing functions as required and when failures occur. When considering these situations, we ask what is meant by the *loss of a function*? We know from the definitions in Table 1 that a function is considered necessary to fulfil a given requirement. How is a requirement deemed to be met, and at what stage is it lost? How is this transition captured in an ontology? In our case, we propose to move forward without having to resolve these bigger ontological issues by using the notion of a state—such issues can be modelled indefinitely whereas we aim to develop a fit-for-purpose domain/application ontology to achieve practical goals and reasoning tasks. We want to say that being in one state enables a function and being in another state disables the same function. The act of disabling a function means that a process that realises that function cannot be performed. To follow this line of thinking requires us to define a notion of state and change of state. This has been discussed in the ontology community, particularly by [4] who suggested that a thing will stay in a state until an external event happens and that an external event can result in a change to a stable state.

2.2. The Need for the Notion of State

Cambridge Dictionary defines *state* as ‘a condition or way of being that exists at a particular time’. In the domains of systems engineering and software development, *state* is used to describe and understand the behavior of complex systems. In those domains, a *state* simply represents a stage in the behavior pattern of an object. A transition is a progression from one state to another and will be triggered by an internal or external event [5].

For ontological definitions of state, we first consider DOLCE which defines a class *state* as a perdurant (approximately equivalent to BFO ‘process’), particularly a subclass of *stative* alongside *process*. State is understood as actual occurrences of situations, which are temporally located and unchanging [6]. In contrast to non-stative perdurants, DOLCE statives are cumulative (i.e., the sum of two operating state occurrences is still an occurrence of operating state); while in contrast to DOLCE processes, states are homeomeric which captures their unchanging nature [7].

This unchanging nature is also reflected in the Common Core Ontologies (CCO)[8]—a mid-level extension of BFO 2020 in which the class *state* or similar does not appear. Instead CCO uses the notion of *stasis* to represent a process in which an [BFO] independent continuant endures across some temporal region and bears some [BFO] (generically or specifically) dependent continuant that does not change in intensity. CCO give as examples: a process during which a light switch remains in the off position and, the stasis of being scheduled for maintenance. Other authors consider states as more like continuants than occurrents, in that they allow certain events to occur [9]. Process Specification Language (PSL) uses state to represent the ‘state of the world’ before and after an activity [5], this allows PSL to use input and output states to specify a set of conditions to constrain what activities occur [10]. Finally the ISO 15531 Standard uses the notion of *resources status* defined by a *resources status type* to provide feedback on the state of each manufacturing resource (e.g. machine) [11] for a particular moment in time.

In general, discussions on ‘state’ and ‘stasis’, particularly when discussing systems, share several common attributes, including:

- states occur/hold for some time; and,
- states describe some unchanging aspect of interest.

As such we consider a very general notion of state/stasis as ‘an occurrence that is of a particular entity (possibly an aggregation or assembly) and during which some aspect of interest (specified by the state kind) remains unchanged (completely or within some tolerance).’ This captures a variety of phenomena of interest, which we can narrow down with more specific notions necessary for our domain.

Coverage in the literature of the notion of the state of an asset and what functions can and cannot be realised in a state is sparse in engineering-related domain and application ontologies. We know that a motor in a powered off state cannot produce torque (its function), however we have not found ontology patterns that represent this idea. Engineering ontologies that mention states include the PROTEUS e-maintenance platform 2006 which identified four asset states: Normal state, Degraded state, Failure state, Programmed stop state [12]. Also the ROMAIN ontology defined two states, a State of Failure and a State of Degradation, both are sub-classes of process class [13]. Transition into a State of Degradation is represented by a process described

by a Triggering Event. The triggering event class is used to describe the transition between states. In neither case is there any detailed description of how the state class is related to other classes or any use of the state class in reasoning. However it is reassuring to see the developers acknowledging the need for the class of state and thinking about different types of states.

The notion of an *Object in Fault State* has been used in reasoning in a recent Failure Modes and Effects (FMEA) ontology [14]. This work describes possible types of malfunctions and the propagation of the state of an *Object in Fault state* at the component level to produce an *object in fault state* at the asset system level. A *malfunction* is defined as “the disposition of an item to fail to perform a required function”, and *object in fault state* as “the state of an item resulting from the realization of a malfunction”. A *malfunction* is a disposition and an object in fault state is a functional object that has some realized malfunction. This FMEA ontology is focused on modelling the negative aspects of malfunctions whereas we are trying to describe the functioning aspects and specifically what functions can be realised depending on a specific state of the asset. Nevertheless, we intend that our work here to define state for functions we may want to realise, such as the ability to operate and perform maintenance, should be coherent with the work by [14] on malfunctions which have only negative consequences.

3. Proposed Implementation

In this section, we propose an implementation of the notion of state in maintenance that aims to address specific goals and reasoning tasks encapsulated by the competency questions listed in the next section. The ontology is demonstrated on an illustrative use case of a washing machine. The competency questions for this ontology are shown in Section 4.

3.1. Use Case Description

We describe an illustrative use case using a washing machine. Assume, for the purpose of this discussion, that our washing machine has four functions. These are: (1) to wash clothes (primary function), (2) to turn on (secondary function), (3) to turn off (secondary function), (4) to operate at 5-Star energy efficiency (secondary function). The state of the washing machine determines which of these functions can be realized. Assume that our washing machine has five possible states.

1. Operating State (the equipment is operating normally, i.e., broadly related to the ‘up state’ in the terminology of Table 1)
2. Degraded State (e.g., the washing machine’s filter is degraded i.e., the ‘degraded state’ in the terminology of Table 1)
3. Failed State (the equipment cannot perform its primary function, i.e., the ‘down state’ in the terminology of Table 1)
4. On State (the equipment has been switched *on* by an external agent)
5. Off State (the equipment has been switched *off* by an external agent)

The first three states offer a familiar view of equipment state in reliability and maintenance work management. When making decisions about the timing of maintenance actions, reliability engineers will consider which of these three states the asset is currently in. States (4) and (5), rather, refer to the state of the washing machine after it has been acted upon by an external

agent. This information is used in two ways by engineers. The first is for troubleshooting, knowing which health state (1-3) and which On/Off State (4-5) the asset was in at a particular time is necessary for fault identification. The second is for maintenance task planning: while most maintenance work can only be executed when an asset is in the Off State, there are certain tasks such as calibration that need the asset to be in the On State. We use this distinction as a basis for our conceptualisation of *Maintenance State*, described in the following section.

3.2. Conceptualisation

We introduce the notion of *Maintenance State* as a ‘stasis that holds during a temporal interval when the realizable functions and capabilities of the participating maintainable item, or the grade of realization of those functions and capabilities, remain unchanged and where the stasis may prevent specific functions and/or capabilities from being realized.’ This definition is based on the notions of *function* and *capability* from the BFO and IOF Core ontologies, which map approximately to the maintenance notion of primary and secondary functions, respectively, and which are both types of disposition. These functions and capabilities are associated with types of process that realize the disposition; together, the functions/capabilities and their correlated processes represent both aspects of “engineering function”, i.e., the (intended) ability to do something, and the actual performance of the function/capability. By making this ontological commitment, it helps identify these subtle distinctions which helps to further refine the conceptualisation.

This is applied to the use case described in Section 3.1, using a “two-tier filter” conceptualisation of which functions and capabilities may be realizable at any given time. The first tier filter is primarily based around the maintenance concepts of *operating*, *degraded*, and *failed states*, which impacts which functions and capabilities are realizable: all are realizable in the operating state; (primary) functions are still realizable in the degraded state, while one or more capabilities are no longer realizable; and the (primary) function is no longer realizable in the failed state (i.e., there is a malfunction). The second tier filter is characterised by external factors impacting the functions and capabilities that can be realized, without implying anything about a malfunction. An example of this conceptualisation is shown in Figure 1.

The grey box labeled “dispositions 1” contains all of the washing machine’s dispositions (i.e., all of its primary and secondary functions described in Section 3.1). When there is no state information, we can assume that all of these dispositions are realizable. However, we can filter this set of dispositions depending on whether the washing machine is in an *operating*, *degraded* or *failed* state. We call this our “tier 1 filter”. So, in Figure 1, the washing machine is in an *Operating State*. Since an operating state does not affect which dispositions are realizable, the box “disposition 2” still contains all four of the washing machine’s dispositions. Now, we can “pass” our dispositions through a second filter. This filter is the set of states that are influenced by an external agent (i.e., On State and Off State). In Figure 1, the washing machine is in an *Off State*. When it is off, it cannot turn off, wash clothes or operate at 5-star efficiency. Therefore, after passing the set of possible dispositions through this filter, we are left with only one realizable function: to turn on. The fact the washing machine cannot wash clothes does not mean that the equipment is malfunctioning when it is turned off, rather it is in need of intervention from an external agent in order to realise this function. In summary, if a washing machine is in an

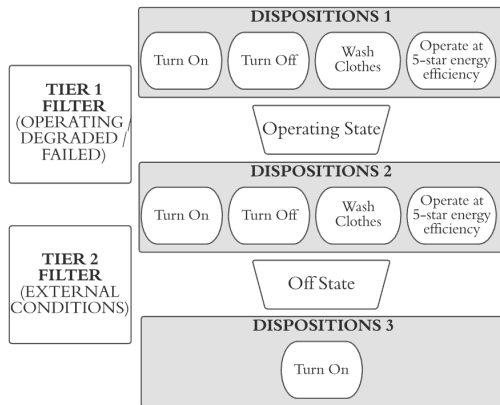


Figure 1: Conceptualisation of a washing machine in an operating state and an off state

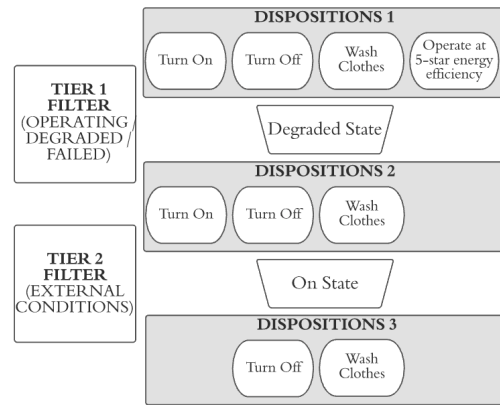


Figure 2: Conceptualisation of a washing machine in a degraded state and an off state

operating state, but is switched off then it can only turn on (it cannot perform its other three dispositions).

Now consider the example shown in Figure 2. In this case, our washing machine is in a *Degraded State*. This is a state in which it fails to fulfil a secondary requirement. For the purpose of this example, this means that the washing machine has a degraded filter and fails to meet an efficiency requirement. Therefore, Figure 2 shows that after the “tier 1 filter”, the box labeled “dispositions 2” shows that only three functions remain for the washing machine. Finally, since the washing machine is in an *On State*, it cannot be turned on because it is already on. Therefore, after “passing” our dispositions through the “second-tier filter”, the washing machine has two realizable dispositions. In summary, if the washing machine is in a degraded state, but is switched on, then it can turn off and wash clothes; however, it cannot turn on or operate at 5-star energy efficiency.

The introduction of *Maintenance State* as a distinguished kind of *State* separates the axioms specific to the maintenance domain from the axiomatisation of the mid- and upper-ontology. This helps future proof the ontology as we expect that not all axioms that we define in future application ontologies for maintenance will hold for the general notion of *State*. It also helps to modularise the ontology and support mappings of the maintenance application ontologies into other upper ontologies in the future.

3.3. Implementation

A BFO-aligned OWL implementation of the conceptualisation described in Section 3.2 can be found at https://github.com/uwasystemhealth/FOMI_maintenance_state. Figure 3 shows the four functions of a washing machine and their organisation under *Function* and *Capability* where the primary function is considered a *Function* and the secondary functions are considered *Capabilities*. The organisation of functions and capabilities in this implementation is discussed further in Section 5.1. The figure also shows that there exist individuals (as specifically dependent continuants) for each of these functions that *inhere in* an instance of *Washing Machine*. The washing machine itself being a *Maintainable Item* and, as such, will generally be in one or more

Maintenance States.

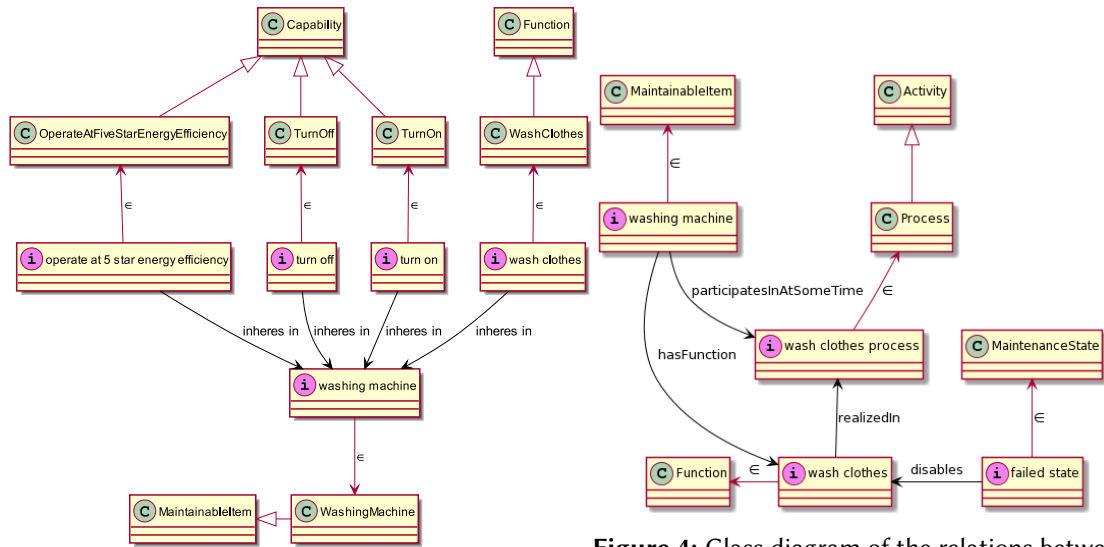


Figure 3: Class Diagram of *Functions* and *Capabilities* in a *Maintenance State* application ontology. ‘C’ represents classes, and ‘i’ represents individuals.

Figure 4: Class diagram of the relations between *function*, *process*, *maintainable item* and *maintenance state*

The “two-tier filter” conceptualisation for *Maintenance State* described in Section 3.2 is modeled in OWL using the Value Partition design pattern [15]. In UML, used for Figure 5, this is represented by the generalization set construct. The figure shows how each of the state types are arranged under the *Tier1ValuePartition* and *Tier2ValuePartition* classes, which each constitute an instance of the Value Partition design pattern.

We introduce new object properties in the ontology implementation. To capture the idea that functions and/or capabilities may be made unrealizable in a particular state, we introduce the relation *disables* (domain: *Maintenance State*, range: *Function* or *Capability*): i.e., if a *Maintenance State* disables a *Function*/*Capability*, it ‘prevents the Function/Capability from being realized’. In terms of temporalized BFO, it is a sub-property of ‘has participant at all times’ indicating that

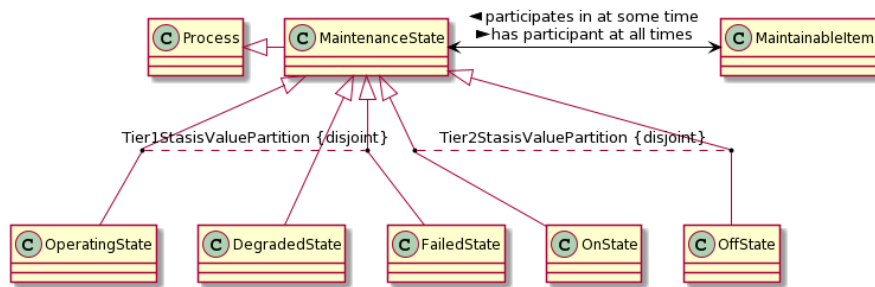


Figure 5: Class Diagram of *Maintenance State* organised into two disjoint covering partitions.

the Function/Capability is disabled for the entire time that the state is in effect. The inverse relation, *disabledBy* is a sub-property of ‘participates in at some time’, as function/capability may only be disabled during some parts of its existence, We use this object property to model which instances of *Maintenance State* disabled which *functions* or *capabilities* of our washing machine. This idea is captured in Figure 4.

We also have the object property *hasState* (domain: *Maintainable Item*, range: *Maintenance State*), a sub-property of *BFO:participates in at some time* and its inverse *stateOf*, which is a sub-property of *BFO:has participant at all times*. We use this object property to capture the current state(s) of a *Maintainable Item* (i.e. the washing machine) and the idea that each state (individual) is of a specific *Maintainable Item*.

We capture which *States* disable which *Functions* using a series of SWRL rules. Two examples are as follows:

$$(1) \text{WashClothes}(?f) \wedge \text{MaintainableItem}(?i) \wedge \text{bearerOf}(?i, ?f) \wedge \text{hasState}(?i, ?s) \wedge (\text{FailedState}(?s) \vee \text{OffState}(?s)) \rightarrow \text{disabledBy}(?f, ?s)$$

$$(2) \text{OperateAtFiveStarEfficiency}(?f) \wedge \text{MaintainableItem}(?i) \wedge \text{bearerOf}(?i, ?f) \wedge \text{hasState}(?i, ?s) \wedge (\text{FailedState}(?s) \vee \text{DegradedState}(?s) \vee \text{OffState}(?s)) \rightarrow \text{disabledBy}(?f, ?s)$$

These rules say that for a given function or capability (?f) and state (?s), if a maintainable item (?i) has that disposition and state and the state is of a relevant type, then the function or capability is *disabledBy* the state.

Now that there are *dispositions* that can be disabled by various *states*, we need to model the effect of this *disabled* property. We need to say that if a *disposition* is *disabled* then a *maintainable item* cannot participate in the process that realizes that disposition. For example, if the *wash clothes* function is disabled, the *washing machine* cannot participate in (as the “agent/instrument/effector”) a *washing clothes process* that would realize the *wash clothes* function, while the disablement is in effect. We represent this by inferring a relation *unableToParticipateIn* between the *Maintainable Item* and the realizing *Process*, for example¹:

$$(3) \text{MaintainableItem}(?i) \wedge \text{bearerOf}(?i, ?f) \wedge \text{hasState}(?i, ?s) \wedge \text{participatesInAtSomeTime}(?i, ?p) \wedge \text{realizes}(?p, ?f) \wedge \text{disabledBy}(?f, ?s) \rightarrow \text{unableToParticipateIn}(?i, ?p)$$

This relation is purely a support for reasoning and analysis and is extra-ontological. This approach is along the lines of [16] and their addition of the *Unexpected Malfunction* class. The alternative is to use consistency checking to determine if a maintainable item is participating in a process that realizes a disabled function. However, as mentioned by [16], this could lead to integration problems with other ontologies and prevents further reasoning. Moreover, while a consistent ontology (particularly under the realist view inherited from BFO) would mean that the conflicting process individual would never be present, it may be present either as the result of an expectation (e.g., an event triggers the process but it cannot start) or for analysis/investigative purposes. Additional object properties used for reasoning support are situated in the ontology implementation under the *reasoningSupport* object property. In future work, we intend to move these reasoning support object properties to a separate OWL file. This way, users of

¹For brevity we exclude the temporal aspects of the state and the process in which *Maintainable Item* participates

the ontology will not have extra-ontological relations in their ontology and can choose to use consistency-checking instead. The implementation described is sufficient to answer the competency questions demonstrated in the following section. We discuss open questions and limitations of this implementation in Section 5.

4. Evaluation

Validation was performed for this work by testing the ontology against competency questions that are of relevance to a maintenance engineer. In the following, we assume the execution of an OWL reasoner and that the queries are performed over the asserted and inferred information. For brevity we do not query for within a specific time-frame.

4.1. Competency Question 1

What are the conditions that need to be met so that an asset can perform its primary function? i.e., what state does the asset need to be in? We illustrate this by considering if there is a *Maintenance State* in which the asset cannot perform its required function. This requires the querying of specification information for which we introduced the relation *typeDisabledBy*. To use this we posit a *Washing Machine* individual and query for which states it must be in that would not disable the function (as indicated by *typeDisabledBy*).

```
PREFIX bfo:<http://purl.obolibrary.org/obo/>
PREFIX stasis:<http://www.semanticweb.org/stasis-ontology-filter#>
SELECT ?maintainable_item ?state ?primary_function
WHERE {
  VALUES ?maintainable_item {stasis:washing_machine_001} .
  ?maintainable_item bfo:BFO_0000196 ?primary_function . # BFO_0000196 = "bearer of"
  ?primary_function a stasis:PrimaryFunction .
  { ?state rdfs:subClassOf stasis:Tier1StasisValuePartition } UNION { ?state rdfs:
    subClassOf stasis:Tier2StasisValuePartition } .
  FILTER NOT EXISTS { ?primary_function stasis:typeDisabledBy ?state }}
```

4.2. Competency Question 2

Can an asset perform its function if it is in an Operating State but is switched off (i.e., in its Off State)? Or more generally, is the *Primary Function* of a *Maintainable Item* disabled in its current *Maintenance State*? If it is not, then it can perform its function, otherwise it cannot. This can be captured by the query:

```
PREFIX bfo:<http://purl.obolibrary.org/obo/>
PREFIX core:<http://www.industrialontologies.org/core/>
PREFIX stasis:<http://www.semanticweb.org/stasis-ontology-filter#>
SELECT ?maintainable_item ?primary_function
WHERE {
  VALUES ?maintainable_item {stasis:washing_machine_001 stasis:washing_machine_002
    stasis:washing_machine_003} .
  ?maintainable_item bfo:BFO_0000196 ?primary_function . # BFO_0000196 = "bearer of"
  ?primary_function a stasis:PrimaryFunction .
  FILTER NOT EXISTS { ?primary_function stasis:disabledBy ?state } }
```

4.3. Competency Question 3

If the asset can perform its primary function, but it is not operating to full efficiency, what conditions have not been met? I.e., what *Maintenance State(s)* does that asset need to be in to be able to realize the capability. For this we ensure that the asset is participating in, and able to be participating in, the process of its primary function and identifying the relevant states.

```
SELECT ?item ?functioning_process ?capability ?state
WHERE {
VALUES ?item {stasis:washing_machine_001 stasis:washing_machine_002 stasis:
washing_machine_003}
?functioning_process a stasis:FunctioningProcess .
?item bfo:BFO_0000056 ?functioning_process # 'participates in at some time'
FILTER NOT EXISTS { ?item stasis:unableToParticipateIn ?functioning_process } .
?item bfo:BFO_0000196 ?capability . # bearer of
?capability a stasis:OperateAtFiveStarEfficiency .
OPTIONAL { ?capability bfo:BFO_0000054 ?process } FILTER ( !BOUND(?process) ) .
{ ?state rdfs:subClassOf stasis:Tier1StasisValuePartition } UNION { ?state rdfs:
subClassOf stasis:Tier2StasisValuePartition } MINUS { {?capability stasis:
typeDisabledBy ?state} UNION {?item stasis:hasState/a ?state} } }
```

5. Discussion

5.1. Functions vs Capabilities

In simple English, we define a function as something that the equipment is designed to do (that is the purpose of its existence), whereas a capability is something that it can do. In the ontology presented here, the classes *function* (from BFO) and *capability* (from IOF Core) are both subclasses of *disposition*. A *function* is defined as a “disposition that exists in virtue of the bearer’s physical make-up and this physical make-up is something the bearer possesses because it came into being...through intentional design (in the case of artefacts), in order to realize processes of a certain sort” [17] [18]. The notion of *capability* is not defined in BFO [17] [18] but is included in IOF Core as a disposition in whose realization some Agent has an interest and for all times during which the capability inheres in its bearer [19].

The notion of ‘capability’ occurs frequently in the manufacturing domain. In ISO 15531 [20], ‘capability’ is defined as a ‘quality of being able to perform an activity,’ which is intended to be used to specify the functional aspects of a manufacturing resource [11]. Such a definition is more general than that of *function* and *capability* from BFO and IOF Core. While the manufacturing usage of ‘capability’ subsumes both *function* and *capability*, it does not distinguish between the two nor identify intent or purpose. However, such distinctions are important in the maintenance domain.

In maintenance, there is the notion of *primary function* and *secondary function* as described in Table 1. This distinction is essential in the maintenance domain and aligns with the notions of *function* and *capability* from BFO/IOF Core, respectively. In general, the maintenance of equipment is intended to ensure that it is able to fulfil its primary function upon request, while secondary functions may fulfil other requirements (e.g., regulatory, efficiency, protection, etc.) and may not intrinsically impact the ability to fulfil the primary function. For example, while the failure to fulfil a safety requirement (in a particular context) may prevent an equipment

from being turned on, it does not change the nature of the equipment nor its ability to fulfil its primary function (if it were to be turned on).

In the case of a washing machine, to *wash clothes* is the machine's primary function. To *turn on*, *turn off* and *operate at five star energy efficiency* are secondary functions—for simplicity of this presentation we do not consider protection, safety, regulatory, etc., secondary functions. According to the vocabulary used in industry, it is intuitive for both primary and secondary functions to be a subclass of function.

In the implementation described in Figure 3, however, we chose to make *wash clothes* a function and *turn on*, *turn off* and *operate at 5 star efficiency* all capabilities. We made the commitment align with the BFO elucidation of *function*. In our implementation, *wash clothes* is the only *function* because to *wash clothes* is the reason for the washing machine's existence. In contrast, secondary functions (i.e. *operate at five star efficiency*) are not the primary reason for the washing machine's existence, thus secondary functions have been asserted as *capabilities*. Moreover, we believe this distinction will be beneficial in the analysis of equipment and system decompositions where the interplay between functions and capabilities at different levels will be highlighted. Even so, this terminology may be confusing for industrial users. We invite further discussion on this topic from the industrial ontology community.

5.2. Conditional Participation

In BFO compliant ontologies, *continuants* have *functions* that are realised in a *process*. The *continuant* is then a participant in that *process*. The implementation described in Section 3.3 expands this pattern. In the implementation, there is a fourth entity, the *maintenance state*, that has an impact on the *maintainable item's* ability to perform a function. Therefore, we require some mechanism to model the situation where: *a maintainable item has a function that is disabled, therefore the maintainable item cannot participate in any process that realises that function*. The pattern used is demonstrated in Figure 4 and in competency questions 2 and 3.

In the maintenance domain, the function of an asset does not change throughout its lifetime. Within the BFO framework, it could be argued that when equipment is in a failed state, the equipment has changed so significantly that the function is non-existent. To align to the maintenance domain, we want continuity of the function individual. Therefore, we use the *disables* object property to model if functions or capabilities are realisable.

5.3. State or Stasis

The IOF has been using the term *stasis* in line with the Common Core Ontologies (CCO). CCO is a collection of ontologies in OWL that extend BFO. The IOF's argument for this is that the word 'state' is overloaded with other entities (such as location). The term 'stasis', as defined in CCO, is something to *maintain*, a condition in which things are unchanging. In engineering the term 'state' is often associated with a specific context such as 'operating state'. In an operating state, qualities such as the load on an asset or its speed can be changing. Our interest here is in how to represent functions that can be realised, or not, conditioned on the state of the asset e.g. if it is operating or not. If we use the term 'stasis' to describe 'state' in this context we have to explain what stasis means and why we are using it to our engineering colleagues. One view is

that ontologies should be invisible to end users, in our case, engineers. This is enabled by the use of patterns and pipelines that allowing ontology experts to build and manage a library of templates and domain experts to provide content in the form of structurally simple template instances [16]. An alternate view is that even if the ‘internal’ names in the ontology are not seen by the end users there is an intervening layer that explicitly maps the ontology concepts to the names (labels) seen by the end users. Choosing the internal name against the convention of the field makes it more difficult for those who maintain the layer to keep track of the implications.

One way forward might be to use ‘stasis’ at the reference ontology level, but subclass it with our more appropriate domain class, e.g. ‘Maintainable Item State’. Apart from clarifying the terminology, it means the assertions we are making about ‘stasis’ are not universal but specific to our domain, since not all ‘stasis’ necessarily control whether functions can be realised or not.

6. Conclusion and Future Work

This paper explores the notion of state in a BFO context to model the transition between an asset being able to perform a function and not perform it. We propose the notion of *maintenance state* as a means of modelling the relationship between function *realization* and process *participation* conditioned on a *state*. We discuss the difference between a *function* and a *capability* of an engineered asset when there are primary and secondary functions, choosing to model the first as a function and the latter as capabilities. The current formulation and reasoning treats the ontology largely atemporally, as a snapshot. However, a more complete solution will require consideration of temporal aspects including temporal data properties and relations between states and processes. Although some consideration has been made towards temporalisation, e.g., *unableToParticipateIn* links to the temporal nature of a state, immediate future work will incorporate a more complete treatment to allow reasoning across time.

In this work we explored including the event that initiates the state change in our ontology. Lengthy discussions were had on the notion of triggering events and the role that a process or process boundary might play (e.g. a failure event). We decided this should be developed in a separate modular ontology. We propose the community put more focus on developing small modular ontologies aligned with a top level ontology and associated reference and domain ontologies, to address specific modelling needs. This approach would support a more rapid deployment of ontologies that are fit-for-purpose for use by the engineering community.

Acknowledgements

The authors wish to acknowledge members of the IOF community particularly in the Maintenance Working Group. Melinda Hodkiewicz would also like to acknowledge funding from the BHP Fellowship for Engineering for Remote Operations.

References

- [1] CEN-EN, Maintenance - Maintenance terminology, Standard EN 13306, European Committee for Standardization (CEN), 2017.

- [2] IEC, Dependability Management – Maintenance and Maintenance Support, Standard AS IEC 60300.3.14, International Electrotechnical Commission, Geneva, Switzerland, 2016.
- [3] SAE International, A Guide to the Reliability-Centered Maintenance (RCM) Standard, Standard SAE JA1012, SAE International, 2018.
- [4] Y. Wand, Ontology as a foundation for meta-modelling and method engineering, *Information and Software Technology* 38 (1996) 281–287.
- [5] C. Bock, M. Gruninger, Psl: A semantic domain for flow models, *Software & Systems Modeling* 4 (2005) 209–231.
- [6] N. Guarino, G. Guizzardi, Events and their context, in: *Joint Ontology Workshops (JOWO)*, Medical University of Graz, 2019.
- [7] C. Masolo, S. Borgo, A. Gangemi, N. Guarino, A. Oltramari, WonderWeb Deliverable D18: Ontology Library (final), Technical Report D18, Laboratory For Applied Ontology - ISTC-CNR, 2003. URL: <http://wonderweb.man.ac.uk/deliverables/D18.shtml>.
- [8] CUBRIC, An overview of the common Core Ontologies 1.3, 2020. URL: <https://github.com/CommonCoreOntology/CommonCoreOntologies>.
- [9] A. Galton, States, processes and events, and the ontology of causal relations (2012).
- [10] C. Bock, C. Bock, M. Gruninger, *Inputs and outputs in the process specification language*, Citeseer, 2004.
- [11] A.-F. Cutting-Decelle, J.-J. Michel, ISO 15531 MANDATE: a standardised data model for manufacturing management, *International journal of computer applications in technology* 18 (2003) 43–61.
- [12] A. Matsokis, H. M. Karray, B. Chebel-Morello, D. Kiritsis, An ontology-based model for providing semantic maintenance, *IFAC Proceedings Volumes* 43 (2010) 12–17.
- [13] M. H. Karray, F. Ameri, M. Hodkiewicz, T. Louge, ROMAIN: Towards a BFO compliant reference ontology for industrial maintenance, *Applied Ontology* 14 (2019) 155–177.
- [14] M. Hodkiewicz, J. W. Klüwer, C. Woods, T. Smoker, T. French, An ontology for reasoning over engineering textual data stored in fmea spreadsheet tables, *Computers in Industry* 131 (2021).
- [15] M. E. Aranguren, *Role and Application of Ontology Design Patterns in Bio-Ontologies*, Ph.D. thesis, Citeseer, 2009.
- [16] D. P. Lupp, M. Hodkiewicz, M. G. Skjæveland, Template libraries for industrial asset maintenance: A methodology for scalable and maintainable ontologies, in: *CEUR Workshop Proceedings*, volume 2757, Technical University of Aachen, 2020, pp. 49–64.
- [17] B. Smith, M. Almeida, J. Bona, M. Brochhausen, W. Ceusters, M. Courtot, R. Dipert, A. Goldfain, P. Grenon, J. Hastings, et al., *Basic Formal Ontology 2.0: Specification and user’s guide*, National Center for Ontological Research: Buffalo, NY, USA (2015).
- [18] A. Ruttenberg, BFO-2020, 2021. URL: <https://github.com/BFO-ontology/BFO-2020>.
- [19] C. Will, Industrial Ontology foundry (IOF) Core, 2021. URL: <https://github.com/NCOR-US/IOF-BFO/tree/IOF-Core-2020>.
- [20] ISO, Industrial automation systems and integration – Industrial manufacturing management data, Standard ISO15531, International Organization for Standardization, 2004.