ROMAIN: Towards a BFO compliant reference ontology for industrial maintenance

Article *in* Applied ontology · April 2019

DOI: 10.3233/AO-190208

CITATIONS
4

READS
479

4 authors:

Hedi Karray
Ecole Nationale d'Ingénieur de Tarbes
43 PUBLICATIONS **187** CITATIONS
SEE PROFILE

Farhad Ameri
Texas State University
61 PUBLICATIONS **833** CITATIONS
SEE PROFILE

Melinda Hodkiewicz
University of Western Australia
86 PUBLICATIONS **1,058** CITATIONS
SEE PROFILE

Thierry Louge
CNRS
15 PUBLICATIONS **174** CITATIONS
SEE PROFILE

ROMAIN: Towards a BFO compliant Reference Ontology for Industrial Maintenance

Mohamed Hedi Karray [a,*], Farhad Ameri [b], Melinda Hodkiewicz [c] and Thierry Louge [d]
[a] *LGP-INP-ENIT, Université de Toulouse, Tarbes, France*
E-mail: mkarray@enit.fr
[b] *Department of Engineering Technology, Texas State University, TX, U.S.A*
E-mail: ameri@txstate.edu
[c] *Faculty of Engineering and Mathematical Sciences, University of Western Australia, Australia*
E-mail: melinda.hodkiewicz@uwa.edu.au
[d] *CALMIP, Université de Toulouse-CNRS-INPT-INSA-UPS, UMS 3667, France*
E-mail: tlouge@inp-toulouse.fr

Abstract.
In this paper, we present a domain-specific, open access, reference ontology (ROMAIN) for the maintenance management domain. We use a hybrid approach, based on a top-down alignment to an open source top-level ontology, the Basic Formal Ontology (BFO), and a bottom up focus on classes that are grounded in maintenance practice. We constrain the scope of the ontology to the classes that are unique to the maintenance management practice, such as maintenance strategy, degradation, and work order management, rather than modeling the entire domain of maintenance. This approach reduces the scope of the development task and enables reasoning to be tested at a manageable scale. ROMAIN provides a unifying framework that can be used in conjunction with other BFO compliant sub-domain ontologies, such as planning and scheduling ontologies. The proposed ontology is validated using real-life data in the context of a use case related to evaluating the effectiveness of maintenance strategy.

Keywords: Basic Formal Ontology, industrial maintenance, maintenance management

1. Introduction

The competitiveness and responsiveness of manufacturing firms depend, in part, on how effectively they maintain their assets and ensure the availability of their resources. 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, 2011). Almost all assets of any value need maintenance and, as a result, there has evolved organizational structures and processes to manage maintenance with theory and practice to support its delivery. Maintenance management practices are documented in Standards (IEC, 2016), textbooks (Kelly, 2006, 1997; Palmer, 1999) and by professional societies (SMRP, 2009; GFMAM, 2016).

Data are key to the effective and safe maintenance of assets over their life cycle. This data is generated by, and drawn from, various sources such as maintenance management systems, design documentation,

*Corresponding author. E-mail: mkarray@enit.fr.

original equipment manufacturer manuals, process control systems, third party service providers, risk management assessments and failure investigations (Morello et al., 2010), to name just a few. However, due to the heterogeneity of data sources and diversity of data types, unlocking the real value of data and discovering the useful patterns of knowledge embedded in the maintenance data has always presented a major challenge (Bendell, 1988; Karray et al., 2011; Knapp and Hasibether, 2011). Ontologies can effectively address this challenge by semantic annotation, integration, consistency checking and organization of data (Karray et al., 2010).

Nowadays most asset-owning organizations will have a Computerized Maintenance Management System (CMMS) whose data schemes conform to a common understanding of work flows in maintenance practice. As a result, the way maintenance work is managed across different sectors such as defense, manufacturing, and service, is remarkably consistent. The details of the work vary with the asset type but the modeling entities and workflow processes are common. This makes maintenance an ideal candidate for an ontology that can have wide-spread application as the classes used to describe the maintenance management process and information are commonly used by the maintenance community.

Over the last 25 years, groups in industry, academia, and on standards committees have developed ontologies for engineering and asset management applications. These works include maintenance concepts but do not focus on maintenance management practice and processes. As a result, there has been limited uptake by maintenance practitioners (Hodkiewicz, 2018). There is a need for a reference ontology in the maintenance domain that can serve as the overarching semantic model connecting various application ontologies related to maintenance. The work proposed in this paper is positioned to fulfill this need.

In this paper, we present a reference ontology for industrial maintenance management. The reference ontology developed in this work uses a top-level ontology. The development of ROMAIN started with identifying the general terms most commonly used in maintenance practice and continued with the representation of more complex entities in the domain. Although the ontology, in its current state, covers most of the key notions specific to the maintenance domain, it can be extended in order to take account of the technological and scientific advances. The rest of the paper is organized as follows. We start with the motivation involving a description of the maintenance management domain. An overview of the current state of the art in ontology practice is provided in section 2, next followed by a critical review of previous work related to maintenance in section 3. Section 4 is devoted to discuss best practices of ontology development. The development approach for ROMAIN is then described and the structure of the ontology is presented in sections 5 and 6. We evaluate ROMAIN against its design principles and validate it with a use case related to measuring the effectiveness of maintenance strategy in section 7. Finally, we discuss challenges and future directions.

2. Background of Maintenance Management

This section provides the definitions and the necessary background information related to some of the important entities in maintenance management. As all assets are likely to fail at some time, there exists a business function to maintain the asset so that the production process or service provided by the asset can continue. This function is called *maintenance management*. The maintenance management process follows a process such as that shown in Figure 1. Key steps in this process are developing maintenance strategy and life plans, administration steps involving planning, procurement and scheduling, task execution, administration associated with cost and performance assessment and strategy optimization. The steps in this process that are specific to maintenance are shown as shaded. Non-shaded steps contain processes, such as planning, scheduling and cost management, that are commonly used in other functions in an organization.

Maintenance management is not a core subject in many engineering curricula and the maintenance world is largely impenetrable to those who have not worked in it as it has its own tribes, language and customs (Reiman and Oedewald, 2004). With this in mind, core maintenance and work management concepts are explained in the following section. Italics are used to identify concepts that will be represented

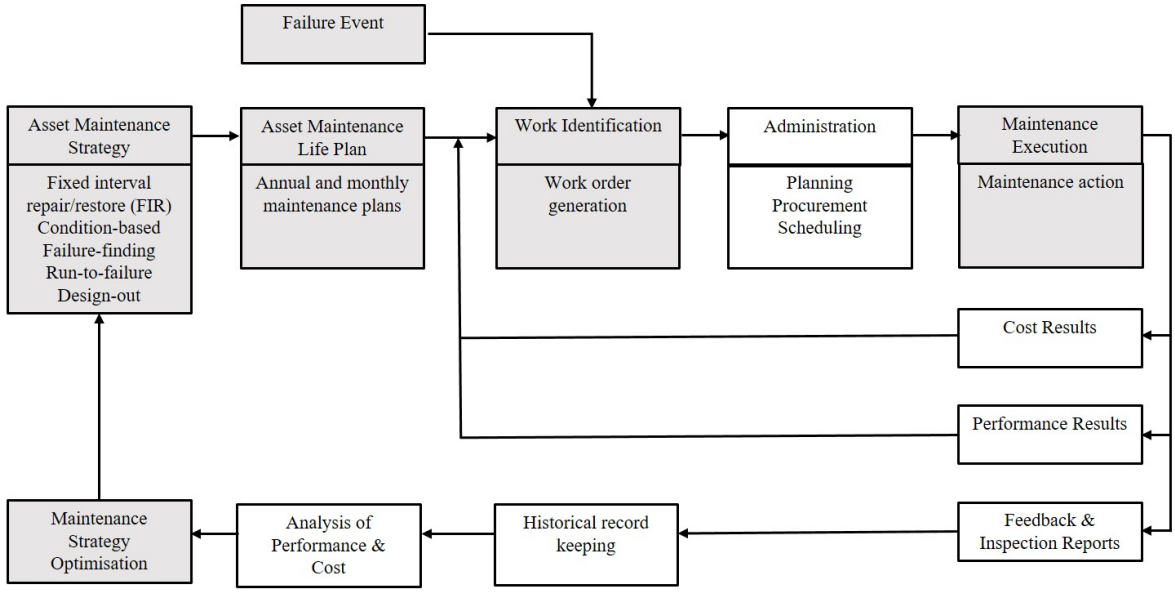

Fig. 1. Maintenance practitioner view of the management process, adapted from Coetzee (2004)

as classes in the domain ontology. These classes are drawn from maintenance practitioner texts and references (Stuber, 2016; Coetzee, 2004; Kelly, 2006; Palmer, 1999).

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 and requirements concerning issues of protection, control, containment, comfort, appearance, energy efficiency, and structural integrity (SAE, 2011). The *maintenance action* performed on an asset depends on the *function* to be maintained and the consequence of a functional failure. Since the 1970s, the most common approach to developing an asset's *maintenance plan* is a process called Reliability-centered Maintenance (RCM). In RCM, the emphasis is on identifying each functional failure and selecting an appropriate *maintenance strategy* to manage each failure mode. The output of the RCM process is the identification of *maintenance strategies* (IEC, 2011) and associated maintenance tasks for each *maintainable item*. A collection of strategies and the associated tasks over the life cycle is called a *maintenance plan*. The concepts described above for *function, maintenance strategy, maintenance plan, and maintainable item* are specific to maintenance and need to be represented in a maintenance domain-specific ontology.

There are six types maintenance strategies in RCM. These are a) condition-based, b) fixed interval restoration (also called scheduled restoration), c) fixed interval replacement (also called scheduled discard), d) failure-finding, e) run-to-failure, and f) design-out (also called modification) (Moubray, 2007). The strategy selected depends on failure behaviour and the consequence of failure. Following strategy selection, a specific *maintenance standard work procedure* and interval are developed and this information is stored in the CMMS. Some of the strategies, except run-to-failure and modification, are preventative or predictive in nature. Individual preventative tasks are triggered at the appropriate time by the CMMS and each instance is represented by a *maintenance work order* record (MWO). Corrective work is triggered by a *failure event* on a *component* resulting from a *non-conformity*. The observed failure is documented using a *maintenance work notification* as part of the work identification stage. Once work is identified, either through failure events or actions triggered by maintenance strategies, then the work is planned and scheduled. Planning involves the procurement of parts and the organization of personnel, tools, other resources and the availability of the asset. Once complete work can be scheduled. Finally, the value-adding step is when the work is executed by maintenance personnel. After execution work there should be analysis of the quality and effectiveness of the maintenance work and opportunities for improvement identified. We

suggest that the processes of planning, scheduling, and execution are generic to many industrial processes. As a result they do not need to be developed specifically for this maintenance ontology. Instead existing ontologies that are aligned to the same top-level ontology can be used to represent these processes.

3. Ontologies in Maintenance

Maintenance management can be viewed through two different lenses. The first is the view of maintenance practitioners of the work management process described in the previous section. In this view, shown in Figure 1, maintenance is a process in which work on many assets is identified, planned, scheduled, executed and analyzed. This view is seldom apparent to operators and owners and is almost entirely absent in maintenance ontologies built to date.

The alternate view considers the requirements of a specific asset, how it should be maintained over its life, its reliability and the risk that its failure presents to the organization. This view, shown in Figure 2, takes an asset life cycle perspective that is primarily of interest to operators and owners of assets (ISO, 2014), rather than maintenance management practitioners. Ontologies that mention maintenance usually have this perspective (Ebrahimipour and Yacout, 2016; Karray et al., 2012; Matsokis and Kiritsis, 2010; Otte et al., 2018). Examples include the Product Life Cycle Management Ontology proposed by Matsokis and Kiritsis (2010) and a more evolved version of this ontology focusing on the operating phase of the life cycle and integrating concepts related to the maintenance field such as alarms, events, and activities (Matsokis et al., 2010). Likewise, Karray et al. (2012) proposed IMAMO (Industrial Maintenance Management Ontology) as a modular ontology that decomposes the maintenance field to according stakeholders' views. Another notable ontology is CDM-Core, a publicly available manufacturing ontology, developed according to condition monitoring data model from ISO13372 (ISO, 2012) to semantically annotate sensor data. The CDM-Core work focuses on an asset's performance rather than on maintenance work management but has useful descriptions of quality measures and a detailed verification process (Mazzola et al., 2016). It is also to note that most of these ontologies do not use any axioms which prevents them from being used effectively for reasoning and consistency checking services.

Ruiz et al. (2004) developed a semi-formal ontology that represents products, roles, activities, processes, workflow, and actions related to software maintenance. Software failure modes and maintenance processes are different from those of physical assets. Therefore, this work is not considered further here. Other ontologies in the maintenance domain have been built for specific use cases such as for failure modes and effects analysis (Ebrahimipour et al., 2010) and maintenance on a pneumatic values (Ebrahimipour and Yacout, 2015).

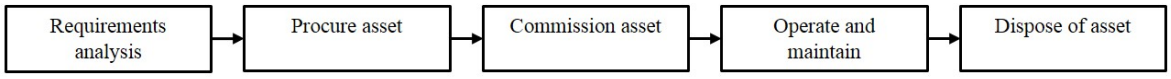

Fig. 2. An asset-centric view of maintenance, commonly represented as a phase in one part of an asset's life cycle

None of the ontologies mentioned above use a top-level ontology. We observe that when individual groups develop domain ontologies that are focused on their specific local needs, the resulting ontologies are incompatible, isolated, and non-sharable. These problems are widely recognized in the ontology sphere, where it is commonly assumed that they can be (or even are already being) addressed by the use of sophisticated ontology mappings or similar technology (Arnold and Rahm, 2014). In our experience, ontologies developed independently and on the basis of distinct sets of ontology development principles (or even on the basis of no principles at all) remain in almost every case unmapped. Moreover, even where mappings are created, they are rarely able to be updated in the needed ways because the ontologies on both sides of the mapping-relation are themselves being changed independently and on asynchronous cycles.

The result is incremental decay of almost all existing mappings. We believe that the use of a top-level ontology as well as a set of shared principles provide a common overarching framework that ensures consistent development of ontologies across communities of researchers and ontology developers such as the case of OBO Foundry in the biomedical domain (Smith et al., 2007).

There were some efforts related to development of the top-level ontology for process plant design engineering in the 1990s. This was driven by the oil and gas and chemical industries. The resulting ontology is documented in the ISO Standard 15926 (ISO, 2003) and described by Batres et al. (2007). The ISO 15296 Standard resulted in a number of applications relevant to the process plant design phase (Batres et al., 2007) but has not resulted in the development of a domain-specific ontology for maintenance management activities.

As part of manufacturing ontology suite, a Basic Formal Ontology (BFO; Arp et al., 2015) compliant ontology of maintenance was proposed in the scope of CHAMP project (Otte et al., 2018). Because the main focus of CHAMP project is on manufacturing processes and entities, the coverage of maintenance in this ontology is limited and the maintenance terminology used drawn from less technical and reliable sources such as Wikipedia. It is not apparent that there has been a review of the maintenance classes against maintenance standards and practice.

4. Ontology Development Methodology

Ontologies have been widely used for knowledge representation as they provide a common vocabulary for modeling a specific domain by capturing knowledge in a structured and formal way. Gruber (1995) defined ontology as 'an explicit and formal specification of a shared conceptualization'. The quality and viability of an ontology depends on the methodology followed for ontology development. Also, the requirements for an ontology is affected by how the ontology is positioned in the stack of ontologies ranging from foundational ontologies to application ontologies. In this section, an overview of different types of ontologies is provided to set the stage for positioning the proposed maintenance ontology as a reference ontology. Best practices for ontology development are presented next. One of the novelties of the proposed ontology is conforming with a top-level ontology. Therefore, the benefits of using top-level ontologies are discussed toward the end of this section.

4.1. Types of Ontologies

Several classifications are presented in the literature to differentiate ontologies based on their level of formality and generality. Top-level ontologies are generic ontologies that can be viewed as ontologies describing the top-level classes that can be used for building other ontologies. They enable interoperability between domain ontologies that incorporate the same top-level ontology (Noy, 2004), thus facilitating data integration and knowledge reuse. Examples of foundational ontologies include the Suggested Upper Merged Ontology (SUMO; Pease, 2006), the Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE; Masolo et al., 2002), the Basic Formal Ontology (BFO; Arp et al., 2015), the Unified Foundational Ontology (UFO; Guizzardi and Wagner, 2010), Yet Another More Advanced Top-level Ontology (YAMATO; Mizoguchi, 2010) and ISO15926 (ISO, 2003). A comparison between some of the previously mentioned top-level ontologies can be found in the work by Mascardi et al. (2007). In contrast, domain ontologies are only applicable to specific domains with a specific point of view. They describe the vocabulary related to a particular area of knowledge (such as biology, environmental sciences, maintenance, etc.) and use classes introduced in the top-level ontology. A *reference ontology* brings together the existing domain ontologies by providing canonical and precise definition of the common entities used in the domain. Reference ontologies focus on rich and axiomatic representation of the intended meanings of common terms in a domain (Andersson et al., 2006). For this reason, reference ontologies often go through several rigorous quality checks according to principled methodologies. Reference ontologies are typically non-proprietary and non-implementation-specific and can be adopted as a sort of standard in a community to

create application ontologies (Falquet et al., 2011). Application ontologies are specializations of reference ontologies and they are created to accomplish some specified local tasks or applications.

4.2. Best practices in ontology building

According to Simperl et al. (2010), most ontology development works are based on an unprincipled approach to ontology engineering which results in non-interoperable and inconsistent ontologies . For the purpose of construction of any ontology, many works (Arp et al., 2015; Arpirez et al., 1998; Gruber, 1995), elaborate clearly principles to respect in order to design useful ontologies. Many of these principles turn on the fact that, since ontologies are built to be shared between actors, it is beneficial if the ontologies used in a given domain share a common upper layer of well-defined terms (Smith, 2006). This upper layer should be domain neutral, since its aim is to represent the most general categories of entities and the most general relations within and between them (Arp et al., 2015). The categories taken together must provide the means to define classes whose instances can be found in any realm of reality. According to Smith and Ceusters (2010), a reference ontology should cover the terminological content of the settled portions of given scientific discipline. In practice the ontology building process is more efficient if key concepts are defined according to a top-level ontology first rather than developing the ontology from the bottom up and then trying to align it retrospectively to a top-level ontology. It is also important that each ontology should have an is-a hierarchy having the structure of a directed rooted tree. Terms in the ontology should be defined in a consistent manner. Finally, the ontology must be available to be used by all potential users with few constraints.

4.3. Value of alignment to a Top level Ontology

When multiple maintenance ontologies are developed in relative isolation from each other by different communities, disconnected information silos will be created. To ensure that different application ontologies employ comparable and mutually-understandable definitions and knowledge constructs, they need to share a common top-level formal ontology. The use of a top-level ontologies facilitates the alignment between several domain ontologies and promote greater data interoperability through conformity to a common semantic model. Top-level ontologies also play the same role as libraries in software programming tasks. These libraries of defined classes and relationships can be reused thereby reducing development time. Using a top-level ontology also enables more effective quality assurance and governance when developing ontologies. The usage of top-level ontologies for integrating information and sharing knowledge among heterogeneous sources has been motivated in various related works (Baumgartner and Retschitzegger, 2006). Top-level ontologies have been used in various domains including situation awareness, pervasive systems (Stevenson et al., 2009), biomedical information systems, government and US military system (Semy et al., 2004), emergency management (Elmhadhbi et al., 2019) and manufacturing (Otte et al., 2018).

4.4. Basic Formal Ontology (BFO)

BFO is a top-level ontology, which has been introduced to support information retrieval, analysis and integration in diverse domains. BFO is at the core of the Open Biological and Biomedical Ontology (OBO) foundry[1], which is an initiative aiming to develop interoperable ontologies based upon shared principles and architecture. BFO establishes a common ground for ontology development and is the top-level ontology used by OBO foundry ontologies.

The Industrial Ontologies Foundry (IOF)[2] is an initiative similar to OBO, dedicated to the development of ontologies for industry. As part of IOF work, the choice of a relevant top-level ontology is under study.

[1] http://www.obofoundry.org/

[2] http://www.industrialontologies.org

Both DOLCE and BFO were used when developing proof-of-concept for IOF ontologies, as they are two well-known top-level ontologies that are widely used in different domains (Guarino, 2017).

For ROMAIN, we consider the use of BFO as it has already been used successfully in OBO Foundry for providing a common ground for several interoperable ontologies and it is in the process of becoming an ISO Standard (ISO 21838-2). Additionally, BFO is sufficiently small and covers classes to model functions and roles, which are necessary for engineering knowledge representation. Also, it is quality controlled through aggressive review and reuse by a sizable community of users. ROMAIN imports portions of BFO compliant mid-level ontologies such as Common Core Ontologies (CCO; CUBRC, 2014) and Information Artifact Ontology (IAO; Ceusters, 2012).

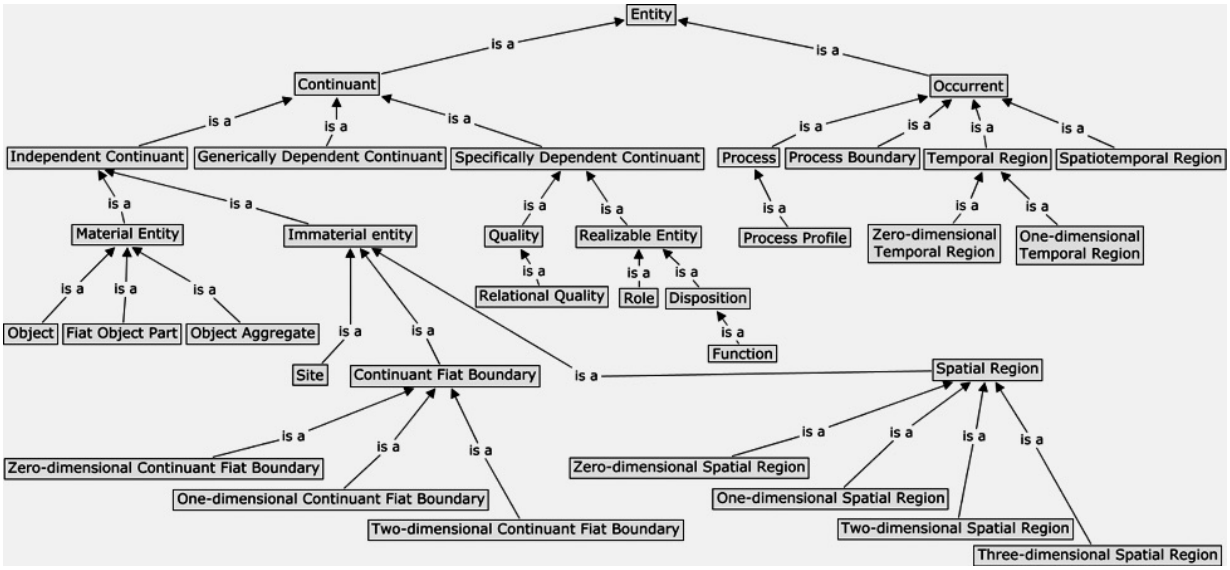

Fig. 3. The structure of BFO 2.0

As shown in Figure 3, BFO 2.0 consists of two main categories of entities: continuants and occurrents. Continuants are entities which continue to exist through time while maintaining their identity and have no temporal parts but are associated with a history. A history names an entity belonging to the realm of occurrents (which can have temporal parts). Occurrents are entities that occur, happen, unfold or evolve with time. BFO:occurrent can be either an entity that unfolds itself in time, or it is the instantaneous boundary of such an entity. Table 1 summarizes key categories encountered in BFO as defined by Arp et al. (2015).

5. Reference Ontology of Industrial Maintenance (ROMAIN)

This section provides more details about the proposed maintenance ontology in terms of objectives, requirements, and competency questions.

5.1. ROMAIN objectives

The scope of the current version of the ontology is limited to those classes and relationships needed to represent various entities related to the maintenance management process as described in Section 2.

5.2. ROMAIN Requirements

According to what is discussed in the previous sections, the key requirements of the ROMAIN ontology are listed as follows:

Table 1
BFO's classes definitions

Class	Definition
BFO:entity	Anything that exists or has existed or will exist.
BFO:continuant	An entity that continues or persists through time while maintaining its identity, and has no temporal parts. It is a dependent or independent object.
BFO:occurrent	An entity that occurs, happens, unfolds or develops in time: events or processes or happenings.
BFO:independent continuant	A continuant entity that is the bearer of qualities. It can maintain its identity and existence through gain and loss of parts, dispositions and/or roles, and through changes in their qualities.
BFO:generically dependent continuant	An entity that is dependent on one or more other independent continuants that can serve as its bearer. It is similar to complex continuant patterns of the sort created by authors or through the process of evolution.
BFO:specifically dependent continuant	An entity that depends on one or more specific independent continuants for its existence. It exhibits existential dependence and has two subcategories: quality and realizable entity.
BFO:process	An occurrent entity that exists in time by occurring or happening, has temporal parts, and always depends on at least one material entity. It can be partitioned into temporal parts in different ways and at different levels of granularity.
BFO:quality	A specifically dependent continuant that depends or inheres in an entity at all and is fully exhibited or manifested or realized in that entity.
BFO:disposition	A realizable entity whose bearer is some material entity.
BFO:function	A function is a disposition that exists in virtue of the bearerâĂŹs physical make-up and this physical make-up is something the bearer possesses because it came into being either through evolution (in the case of natural biological entities) or through intentional design (in the case of artifacts), in order to realize processes of a certain sort.
BFO:role	A realizable entity which exists because the bearer is in some special physical, social, or institutional set of circumstances in which the bearer does not have to be, and is not such that, if it ceases to exist, then the physical make-up of the bearer is thereby changed.
BFO:temporal region	An occurrent entity that is part of time as defined relative to some reference frame.

- The ontology has to be aligned with a top-level ontology and reuses classes from mid-level ontologies.
- The ontology has to capture the core notions related to the domain of maintenance management.
- The ontology has to use the established terminology of the maintenance management domain.
- The ontology has to follow the design principles for ontology development.

5.3. Competency questions

Grüninger and Fox (1995) proposed competency questions (CQs) as a technique for defining the ontology specifications. These CQs consist of a set of questions, stated in natural language, that the ontology must be able to answer. Below are some of the examples of the CQs that ROMAIN, and its derivative application ontologies, should be able to answer:

- How many breakdown work orders were executed?
- How many replace orders were executed before the expected replacement interval <x> of operating hours?
- What is the total cost of failures on the maintainable item <A> which should have no failures under its scheduled restoration maintenance strategy?
- Did any scheduled restorations occur before the target interval of <x> operating hours?
- What is the cost of the work done initiated by an inspection?

6. ROMAIN structure

This section describes the core classes of ROMAIN. Figure 4 illustrates the structure of the ontology using some of its main classes. Descriptions of the various classes and their relationships are given in the sections below. We note that the prefix 'ROM' is used to identify ROMAIN classes.

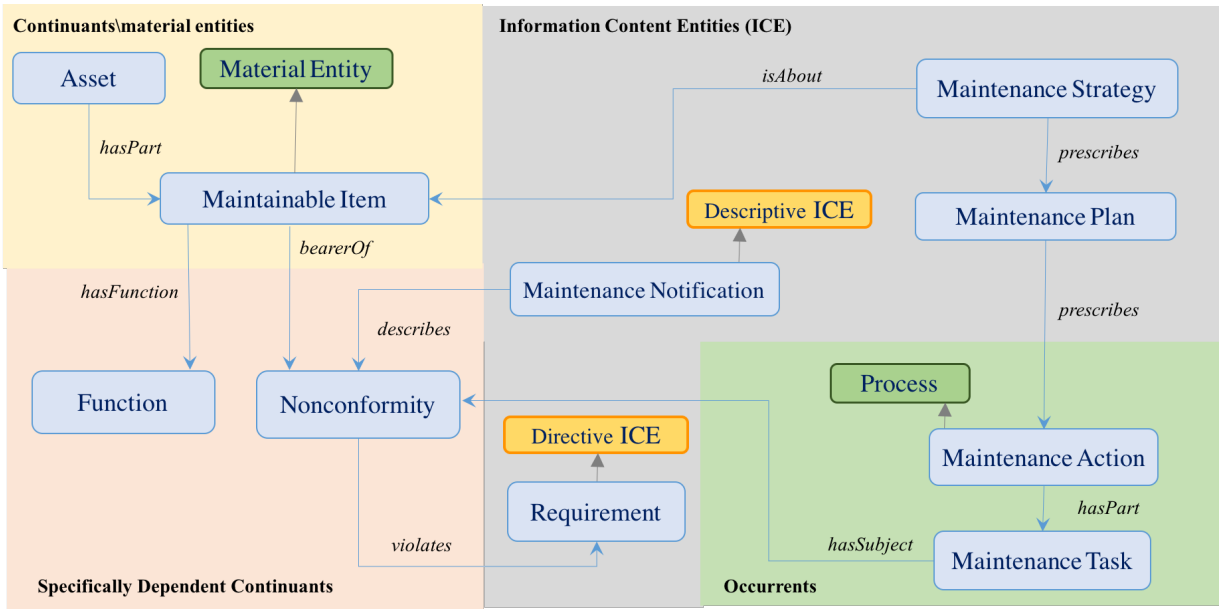

Fig. 4. The class diagram showing the relationships between some of the continuants, occurrents, and information entities

6.1. Relations in ROMAIN

Before introducing ROMAIN classes, we introduce the used relations in order to well understand the definitions of these classes and their proprieties. There are two groups of relations in the proposed ontology. The first group is composed of the relations specific to ROMAIN and the second group is composed of the generic relations imported from other external ontologies. The Relations Ontology (RO; Smith et al., 2007) in particular is used extensively. RO is a collection of relations intended primarily for standardization across ontologies in OBO Foundry. RO is developed in the context of this Foundry initiative and, therefore, it is in compliance with BFO. Table 2 summarizes all relations used in ROMAIN.

6.2. Continuants

Continuants are the entities that continue or persist through time, such as objects and their functions and qualities. There are three types of continuants in BFO, namely, generically dependent, independent, and specifically dependent continuants that are discussed separately in the context of maintenance management in this section . The definitions of the ROMAIN's generically dependent continuants, independent continuants, and specifically dependent continuants are provided in Table 3, Table 4, and Table 5 respectively.

6.2.1. Independent continuants

Independent continuants do not depend on other entities for their existence. Examples of independent continuants in ROMAIN include the machines, equipment, devices, or components that become the subjects of maintenance strategies and actions. They are considered to be instances of *Artifact* class in ROMAIN. An *Artifact*, a class imported from CCO, is an object that is designed by some agent to realize

Table 2
Terms used for ROMAIN's relations

Relation	Definition
is a	entity A is a entity B means that A is subclass of B.
RO: bearer of	a relation between an independent continuant (the bearer) and a specifically dependent continuant (the dependent), in which the dependent specifically depends on the bearer for its existence.
RO: has role	a relation between an independent continuant (the bearer) and a role, in which the role specifically depends on the bearer for its existence.
RO: has function	a relation between an independent continuant (the bearer) and a function, in which the function specifically depends on the bearer for its existence.
RO: has part	a core relation that holds between a whole and its part.
RO: is about	A primitive (i.e. undefined) relationship between an information content entity and some entity.
RO: prescribes	For all classes T1 and T2, if T1 prescribes T2, then there is some instance of T1, t1, that serves as a rule or guide to some instance of T2, t2 (if T2 is a type of BFO:occurrent) or that serves as a model for some instance of T2, t2 (if T2 is a type of BFO:continuant)..
ROM: has subject	a relation between an occurrent and a dependent continuant in which the dependent continuant is the reason to the appearance of the occurent.
RO: affects	If p is a process and c is a continuant, then p affects c if and only if p influences c in some manner, most often by producing a change in c.
RO: describes	For all classes T1 and T2, if T1 describes T2 then there is some instance of T1, t1 that presents the characteristics by which some instance of T2, t2 can be recognized or visualized.
RO: has output	a relation between a processual entity and a Continuant such that the presence of the Continuant at the end of the processual entity is a necessary condition for the completion of the processual entity.
RO: participates in	a relation between a continuant and a process, in which the continuant is somehow involved in the process.
RO: proceeds	a relation between two occurents. Occurent x precedes occurent y if and only if the time point at which x ends is before or equivalent to the time point at which y starts.
RO: occurs on	An instance level relation which holds between some processual entity and some temporal region whenever the duration of the processual entity is contained by the temporal region.
RO: is input of	Inverse of has input, a relation between a processual entity and a Continuant such that the presence of the Continuant at the beginning of the processual entity is a necessary condition for the start of the processual entity.
ROM: violates	A relation between a continuant and a directive ICE in which the continuant does not fulfill the directions expressed in the directive ICE.

certain functions. An *Artifact* is an asserted class that can assume different roles in maintenance scenario and become an *asset*, a *component*, or a *maintainable item* as shown in Figure 5. These notions are further defined in the following sections.

Asset An *Asset*, as an *independent continuant*, is an *Artifact* that has potential or actual value to an organization (ISO, 2014). *Asset* is treated as a defined class in ROMAIN. It is formally defined as an *Artifact* that bears an *Asset Role* for an organization.

Component A *Component* is an *Artifact* that bears a *Component Role*. A *Component Role* is a *role* that is borne by an *Artifact* when it become a part of another artifact, to provide a particular function.

Maintainable Item A *Maintainable Item* is an *artifact* when it bears the *Maintainable Item Role* in the context of a maintenance strategy. An *Asset* can be a *Maintainable Item* or it can be composed of one or more maintainable items. Each *Maintainable Item* has one or more functional failures. A single *Component* can also become an instance of *Maintainable Item*. The selection of what is or what is not a *Main-*

tainable Item is a decision of the maintenance group and a result of *maintenance strategy development process*. It depends on the resources and capabilities available at a specific site. For example, they may choose to rebuild pumps, in which case the maintainable items will be the bearings and other items that make up the pump. Or they may choose to send the pump to an external rebuild shop in which case the pump is the maintainable unit.

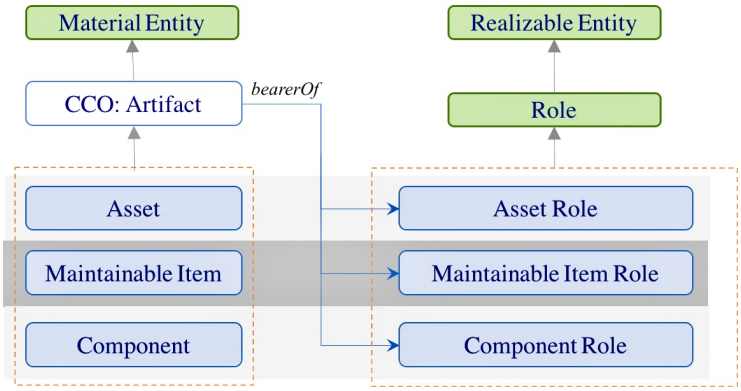

Fig. 5. Different roles an artifact can bear in the domain of maintenance

6.2.2. *Specifically Dependant Continuants*

A specifically dependent continuant (SDC) is an entity that depends on other independent continuants for its existence. Quality, function, and role are sub-types of this SDC class. Functions, in particular, are directly used when representing a functional failure in an artifact since a failure is the termination of the ability of an artifact to perform a required function. A failure is often a consequence of some change or degradation in one or more qualities of the artifact or its parts. A failure can be attributed to one or more nonconformities. A nonconformity is a deviation from a requirement. It is one of the core classes in ROMAIN and is hence described in more detail below.

Nonconformity In general, nonconformity is non-fulfillment of a requirement or expectation. In ROMAIN, *Nonconformity* is formally defined as a specifically dependant continuant that inheres in a material entity and deviates from a design specification. It can be a loss of quality, function, or capability in an item that results from a degradation process. Each nonconformity can also trigger other degradation processes that may cause more nonconformities in the same bearer or other external bearers. For example, reduced traction in a tire (degraded function = functional nonconformity) can be the result of worn tire treads. In this example, “shallow tread" is a physical change or a degraded quality that is the basis for the functional nonconformity in the tire and “tread depth" is a quality borne by the tire that changes during the course of “wear process". Both “shallow tread" and “reduced traction" can be regarded as instances of nonconformity class in connection with specific design requirements. Also, it should be noted that nonconformity is a defined class since it is not an inherent characteristic of a quality or function.

6.2.3. *Generically Dependant Continuants*

In ROMAIN, the *Information Content Entity (ICE)* class, a subclass of *generically dependent continuant*, is used for different purposes as shown in Table 3. Descriptive, designative, and directive ICE are the three sub-types of ICE in ROMAIN that are imported from CCO. Descriptive ICEs are used to describe various entities such as maintenance strategies, actions, work orders, and notifications. Directive ICEs are used to describe plans and procedures, such as maintenance plan, through a set of prepositions. *Maintenance Work Order Record* is an example of a descriptive ICE that describes a desired *Maintenance Action*. It is also used to record the actual action taken and captures the costs and resources required and actually used in the execution of the maintenance action. Figure 6 shows the main fields of a *Maintenance Work Order Record*. While there are many data fields associated with a single maintenance work order record, the ones listed in this figure are the most common fields. For example, most work order records contain

data fields that correspond to start date and system status code that are considered to be sub-types of *date identifier* and *code identifier*, respectively. *Maintenance Notification* is a descriptive ICE that describes an observed *Nonconformity*.

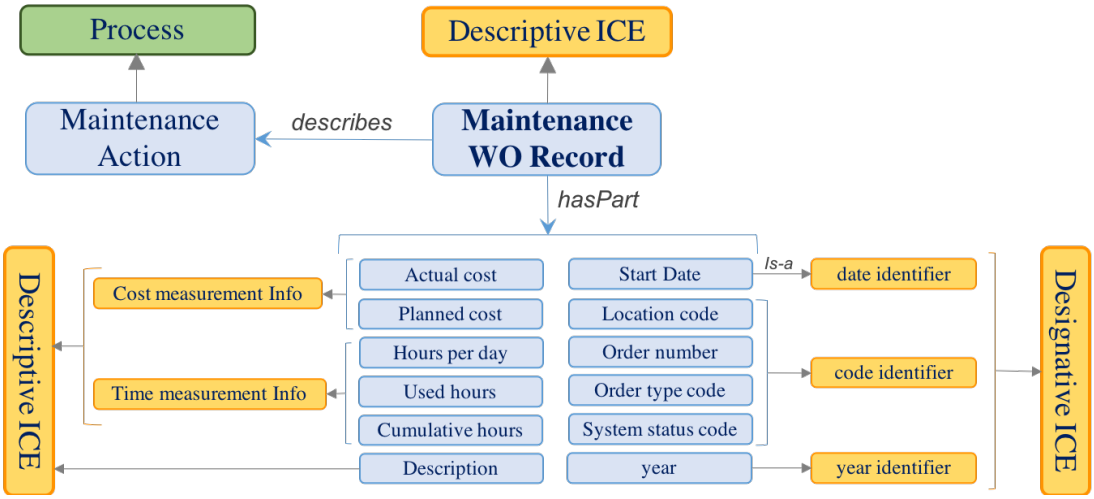

Fig. 6. Data fields associated with a maintenance work order record

Table 3
ROMAIN Generically dependent continuants

Common term (ROMAIN class label)	**Definition**
Descriptive Information Content Entity (Descriptive Information Content Entity)	An IAO:information content entity that consists of a set of propositions that describe some Entity.
Designative Information Content Entity (Designative Information Content Entity)	An IAO:Information Content Entity that consists of a set of symbols that denote some Entity.
Directive Information Content Entity (Directive Information Content Entity)	An IAO:Information Content Entity that consists of a set of propositions or images (as in the case of a blueprint) that prescribe some Entity.
Maintenance work order record (Maintenance Work Order Specification)	An IAO:information content entity describing a desired ROM:Maintenance Action
Maintenance notification (Maintenance Notification)	An IAO:information content entity describing a ROM:Non-conformity.
Maintenance plan (Maintenance Plan Specification)	An IAO:plan specification containing Maintenance Strategies and Maintenance Standard Work Procedures for a ROM:Maintainable Item over its lifecycle.
Maintenance strategy specification (Maintenance Strategy Specification)	An IOA:information content entity describing the ROM:Maintenance Strategy Type to manage a specific functional failure.
Maintenance strategy type (Maintenance Strategy Type)	An IAO:information content entity resulting from ROM:maintenance strategy development process for a ROM:Maintainable Item.
Maintenance schedule (Maintenance Schedule List)	An IOA:information content entity describing a list of ROM:Maintenance Work Order Specification to be executed in a defined BFO:one-dimensional temporal region (i.e. temporal interval).
Maintenance standard work procedure (Standard Work Procedure Specification)	An IAO:plan specification that describes the steps required for a specific ROM:Maintenance Action.
Requirement (Requirement Specification)	An IAO:information content entity that prescribes a need or expectation to be met

Table 4
ROMAIN Independent continuant

Common terms (ROMAIN class descriptor)	**Definition**
Asset (Asset)	A CCO:artifact that bears an ROM:Asset Role.
Component (Component)	A CCO:artifact that bears a ROM:Component Role.
Maintainable item (Maintainable Item)	A CCO:artifact when it bears the ROM:Maintainable Item Role

Table 5
ROMAIN Specifically dependant continuant

Common term (ROMAIN class label)	**Formal Definition**
Component role (Component Role)	A BFO:role that is borne by an CCO:artifact(s) when it becomes a part of another BFO:material entity, to provide a particular BFO:function.
Maintainable item role (Maintainable Item Role)	A BFO:role borne by a group of one or more CCO:artifact(s) as an output of a ROM:Maintenance Strategy Process.
Asset role (Asset Role)	A BFO:role that is borne by an CCO:artifact to deliver potential or actual value to an CCO:organization
Function (BFO:function)	A BFO:function is a disposition that exists through intentional design in order to realize processes of a certain sort.
Non-conformity (Nonconformity)	A BFO:specifically dependent continuant that inheres in a material entity and deviates from a design specification.

6.3. Occurrent

Processes and events are the main types of occurrents in ROMAIN as shown in Table 6 and Figure 7.

Table 6
ROMAIN Occurrent

Common term (ROMAIN class label	**Formal Definition**
Degradation process (Process Of Degradation)	A BFO:process that results in the loss of a desired quality or function.
CCO:Stasis (State)	A BFO:Process in which some BFO:independent continuant endures and one or more of the dependent entities it bears does not change in kind or intensity
Functional failure (State Of Failure)	A ROM:state during which a CCO:artifact is unable to perform its BFO:function.
Degraded state (State Of Degradation)	A ROM:state during which a CCO:artifact bears an undesirable BFO:quality or BFO:function.
Maintenance action (Maintenance Action)	A BFO:process to perform work on a CCO:artifact according to a ROM:Maintenance Work Order Specification.
Triggering event (Triggering Event)	A BFO:process resulting in an action.
Failure (Failure Event)	A BFO:process that precedes the ROM:State of Failure.

Process of Degradation Artifacts can undergo a change in their specifically dependant continuants. The *Process Of Degradation* is a process in which a desired quality, function, or capability is lost over time or an undesirable quality, function, or capability is gained over time. The output of the *Process Of Degradation* is a *Non-conformity* that may or may not be observable. When a *Non-conformity* inheres in an artifact, then the artifact enters the *State Of Degradation*. When an item is in *State Of Degradation*, it doesn't necessarily mean that the item is failed. However, degradation might lead to failure.

Failure Event In general, an *event* is treated as a *process* in BFO. A *Failure Event* is a process that precedes a *State Of Failure*. When a *Component* is in *State Of Failure* it is unable to perform its intended function. The relation between an event and a state is inspired here by the the point of view presented by Galton (2012), in which a state can be initiated or terminated by an event. It should be noted that failure event is not a temporal region to specify the moment that failure happens. Rather, it is a process that occurs at some point in time or during a time interval. A *failure event* occupies a temporal region,

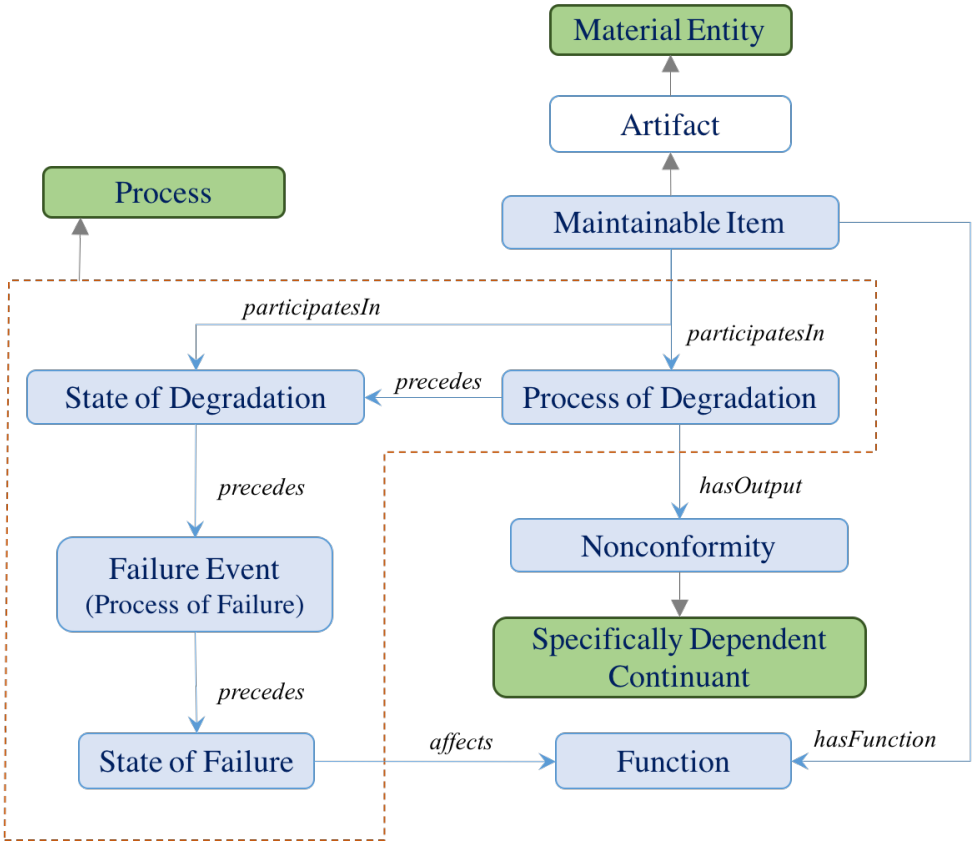

Fig. 7. The occurrents related to degradation and failure processes

either zero-dimensional or one dimensional. If the *failure event* is instantaneous, then it occupies a zero-dimensional temporal region and if it takes place during an interval, then it occupies a one-dimensional temporal region.

Triggering Event Triggering Event is a process. For example, it might be that a preset maintenance time interval has been reached, resulting in initiation of a maintenance process as shown in Figure 8. This figure also demonstrates how a fixed-interval repair strategy, as a directive ICE, can prescribe a triggering event.

6.4. Axioms

Axioms represents that facts that are always true in the domain. In ROMAIN, different types of class axioms, including equivalent, subclass, and disjoint class axioms, have been used to impart more formality to the ontology. Table 7 shows some examples of class axioms implemented in ROMAIN. The axioms are represented using Manchester Syntax from OWL 2.0 as it is a more compact and readable representation.

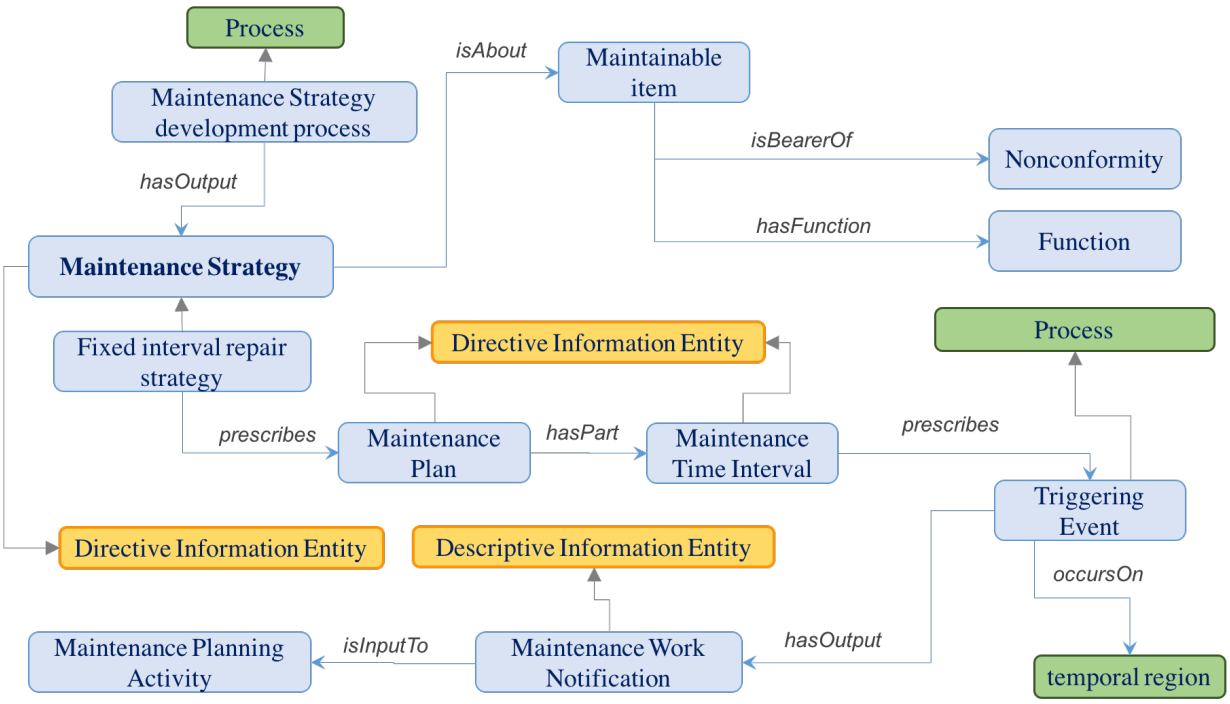

Fig. 8. Relationships between a Maintainable item, maintenance strategy, and maintenance action

Table 7
Some examples of class axioms in ROMAIN

Axiom	Description
`Class: MaintainableItem EquivalentTo: cco:Artifact AND ( hasRole SOME MaintainableItemRole)`	Equivalent class axiom for Maintainable Item.
`Class: Asset EquivalentTo: cco:Artifact AND ( hasRole SOME Asset Role)`	Equivalent class axiom for Asset.
`Class: StateOfFailure SubClassOf: (precededBy SOME ProcessOfFailure) AND (hasParticipant SOME Artifact )`	Every State of Failure is preceded by a Process of Failure and has at least one Artifact participant
`Class: ProcessOfDegradation SubclassOf: (precedes SOME StateOfDegradation) AND (hasOutput SOME Nonconformity)`	Every Process of Degradation precedes a State of Degradation and has a Nonconformity as the output

7. Validation, Verification and Discussion

Ontology verification is the ontology evaluation which compares the ontology against the ontology specification document (ontology requirements and competency questions cf. section 5) thus ensuring that the ontology is built correctly (Suarez-Figueroa and Gómez-Pérez, 2008). In other words, it allows us to answer the question “are we producing the ontology right?”. Ontology validation is the ontology evaluation that compares the meaning of the ontology definitions against the intended model of the world aiming to conceptualize. In other words, it permits to answer the question “Are we producing the right ontology?". Hence, in order verify and validate a reference ontology as ROMAIN, we estimate that we need to consider a real industrial context and real data set. In the following section, we examine the competency questions listed in Section 5.3 according to the maintenance strategy effectiveness use case by using real-life data.

7.1. Use case: maintenance strategy effectiveness

Our motivating question is to determine how effective the selected maintenance strategy is in practice. Having an effective maintenance strategy will reduce unanticipated events leading to improved uptime, reduced costs and more effective use of maintenance resources. Common manifestations of ineffective strategy are assets failing before their scheduled restoration or discard times and assets being replaced or restored while they are still functional. In addition, the execution of the strategy can be ineffective. For example if assets with on-condition tasks are failing unexpectedly then the on-condition task is obviously not effective at detecting the deterioration. Once ineffective tasks are identified, engineers can investigate and understand why the strategy is ineffective. Currently the process to investigate maintenance strategy effectiveness is triggered by observations by maintenance personnel after an unexpected failure event. The investigation, usually done by an engineer or maintenance planner, requires access to data in a number of locations. For example, specific work order records, preventative maintenance strategy and work history in the CMMS (all stored in different tables), failure investigations in Excel or Word files, RCM analysis (also usually in Excel file). It can take days or even weeks to pull together the right data and perform the analysis. A major issue is that most of the data is stored as unstructured, free text. As a result some pre-processing of the unstructured free text using natural language processing or rule-based methods may be required.

The data for this maintenance strategy evaluation use case comes from the CMMS used by an operator of Heavy Mobile Equipment (HME), one of the most expensive maintainable items is the engine. We have records of all tasks executed on the engines of four identical HME assets in their first 18,000 operating hours. We show an extract of these records in Figure 9. We aligned the data schema with ROMAIN classes and then we manually instantiated this part of data via the Protégé editor in order to use it in our experiments.

In this table, columns 1, 5, 6, 7 and 8 are as-downloaded from the CMMS. Data in columns 2, 3 and 4 have been added by the engineer to assist with analysis. While the data represented may seem straightforward to analyze, the engine is only one of many maintainable units on this asset, and operators can own, tens to thousands of assets that might benefit from maintenance strategy effectiveness analysis. Therefore, automation of this process is an inevitable necessity.

The competency question ‘how many breakdown orders were executed?’ is straightforward and would be usually done by an engineer generating a pre-configured report for EQ1 engine from the CMMS, add in columns 2-4, and then use PivotTables in Excel to filter the data. This exercise then needs to be repeated for each sub-system of interest each time the analysis needs to be done. Some engineers will use Excel macros to help. In contrast, the SPARQL query for this first competency question is presented in figure 10, with its associated result (the SPARQL query language can be used to express queries across diverse data sources). Breakdown orders are triggered by failure events, which are zero-dimensional temporal regions. Therefore, counting the number of breakdown orders comes down to counting the number of failure events, expressed in the request in figure 10.

Asset Functional location	Utilized hours	Cumulative utilized hours	Work order trigger	Work order Description	Work order code	Actual costs	Planned costs
EQ1_engine				Starts			
EQ1_engine	210	210	Breakdown	REPAIR ENGINE SHUTTING DOWN	PM03	1,014	-
EQ1_engine	1085	1295	Breakdown	REPAIR LOW ENG OIL LEVEL ERROR	PM03	-	507
EQ1_engine	3080	4375	FromInspection	Replace Eng Oil Level Switch	PM02	768	605
EQ1_engine	1540	5915	Minor	REPAIR LACK OF POWER	PM03	357	357
EQ1_engine	1768	7683	Fixed Interval Restoration	OEM Model 1 LOADER 8000HR ENGINE MIDLIFE	PM01	44,235	49,184
EQ1_engine	0	7683	FromInspection	Replace LH Eng Fuel Rail Clamp	PM02	159	390
EQ1_engine	578	8260	Breakdown	Engine derating.	PM03	312	-
EQ1_engine	3850	12110	Breakdown	REPAIR ENG OIL LEAK	PM03	795	1,102
EQ1_engine	508	12618	FromInspection	REPAIR BROKEN LH ENG COVER MOUNT BRACKET	PM02	323	323
EQ1_engine	35	12653	FromInspection	REPLACE FAULTY ENGINE SPEED TIMINGSENSOR	PM02	1,279	584
EQ1_engine	175	12828	FromInspection	INSPECT/REPAIR ENGINE STARTER MOTOR	PM02	785	785
EQ1_engine	88	12915	FromInspection	WELD UP CRACKED ENGINE PANELS	PM02	521	-
EQ1_engine	1138	14053	FromInspection	INVESTIGATE ENG OIL LEAK L/HFRONT	PM02	2	388
EQ1_engine	735	14788	Fixed Interval Replacement	OEM Model 1 LOADER ENGINE C/O	PM01	225,849	37,277
EQ1_engine	0	14788	FromInspection	REPAIR OIL LEAKS AT FRONT OF ENGINE	PM02	645	4,629
EQ1_engine	368	15155	Fixed Interval Restoration	OEM Model 1 LOADER 8000HR ENGINE MIDLIFE	PM01	-	53,618
EQ1_engine	0	15155	FromInspection	REPLACE CRACKED ENGINE COOLING FAN	PM02	6,547	5,972
EQ1_engine	2625	17780	FromInspection	CMT_engine _Fuel dilution	PM02	6,426	3,729
EQ1_engine	1295	19075	FromInspection	REPLACE ENGINE SPEED SENSOR	PM02	312	137
EQ1_engine	2083	21158	FromInspection	RESEAL LEAKING A/C DRIVE ON ENGINE	PM02	1,134	1,134
EQ1_engine	1838	22995		Ends			

Fig. 9. All work order records associated with work on the engine of heavy mobile equipment EQ1 over its operating life to 23,000 hours

SPARQL query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX romain: <http://www.semanticweb.org/tlouge/ontologies/2018/11/ROMAIN#>
SELECT (count(?failure) as ?failureCount)
        WHERE {?failure rdf:type romain:failureEvent .}
```

failureCount
"4"^^<http://www.w3.org/2001/XMLSchema#integer>

Fig. 10. SPARQL query for first competency question

The competency question ‘did any scheduled repairs or restoration occur before the target interval of <x> operating hours?’ The RCM process of this maintainable item has identified a fixed interval restoration task at 8,000 operating hours and a fixed interval replacement task at 16,000 operating hours. Assuming a window of + or – 500 hours as being acceptable we can see that the first fixed interval restoration

task at 7,683 hours is acceptable but the second one at 15,155 operating hours was not executed. We know it was not executed because there are no actual costs associated with the work. We also note that there is a fixed interval replacement task that occurred at 14,788 hours. There are two alarming issues here. The first is that the replacement work occurred at 14,788 operating hours, well short of the expected 16,000 hours. The second issue is that there are duplicate fixed interval tasks in the CMMS system and one (the fixed interval restoration task at 16,000) hours should be removed as there is a fixed interval replacement also at 16,000 hours. Obviously, the planner has ignored the fixed interval restoration task generated by the CMMS at 15,155 hours but the task is still being generated in the CMMS system and should be removed. Since the records at 14,788 and 15,155 will be identified in the analysis, the duplicate can be removed. The SPARQL request for this second competency question is presented in Figure 11. The results show the identifier of the work order, the identifier of the event triggering the work order, and the value of operating hours at which the work order occurred. The first step of the request consists of querying the events triggering scheduled work orders (first line of the query). Then, work orders related to those events are identified (second and third line of the query). Finally, only the work orders with less than 16,000 cumulative hours are kept in the results. The results present the identifier of the work orders and the events in the ontology, together with the cumulative hours value for each work order.

SPARQL query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX romain: <http://www.semanticweb.org/tlouge/ontologies/2018/11/ROMAIN#>
SELECT distinct ?worder ?event ?val
        WHERE {?event rdf:type romain:event_tirggered_by_Fixed_Interval_Preventative_Maintenance_Task_execution .
        ?worder rdf:type romain:Work_order .
        ?worder romain:isAbout ?event .
        ?worder romain:ROM:_cumulative_hours ?val. FILTER(?val< 1600).
}
```

worder	event	val
WO16	Event16	"15155"^^<http://www.w3.org/2001/XMLSchema#integer>
WO5	Event5	"7683"^^<http://www.w3.org/2001/XMLSchema#integer>
WO14	Event14	"14788"^^<http://www.w3.org/2001/XMLSchema#integer>

Fig. 11. SPARQL query for second competency question

The ability to write a single SPARQL query which can be run for a list of assets at different functional locations over the existing methods is significantly more efficient and transparent to current engineering practice. As mentioned earlier, the current approach is to have an Excel spreadsheet and pivot table for each query. Version control is difficult to manage and the number of Excel sheets quickly grows to be unmanageable. Not to mention that the process of cutting and pasting, organizing and making pivot tables is laborious and prone to human error.

The SPARQL queries presented in this section have been run on a version of ROMAIN containing the work orders presented in figure 9. The results presented in figure 11 (WO16, WO5, WO14) refer to the 17th, 6th, and 15th lines of figure 9.

7.2. Evaluation of ROMAIN according to best practices and principles

In evaluating the ROMAIN ontology we consider the requirements listed in 5.2. The ontology is aligned with a top-level ontology, the BFO. Therefore, it can be aligned and extended with classes from BFO compliant mid-level ontologies such CCO and IAO as shown in Figure 12. This figure also shows how BFO-aligned ontologies for activities in the maintenance management process that use processes such as planning, scheduling and costing, can be tailored to maintenance by reference to ROMAIN concepts.

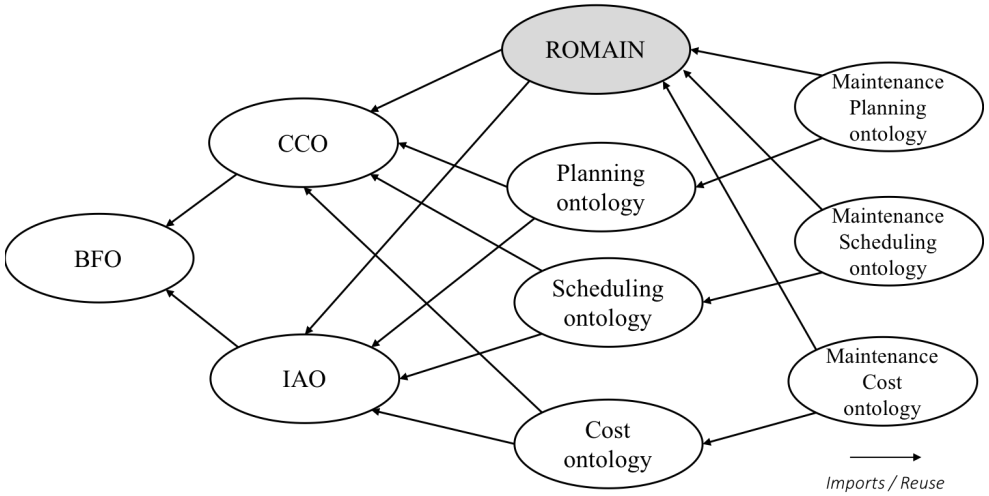

Fig. 12. A mapping of the relationships between BFO-compliant ontologies to enable development of an ontologies for maintenance management activities. [The arrow sense shows the sense of import, e.g. CCO imports BFO, and ROMAIN imports CCO].

The ROMAIN ontology captures core notions and terminology related to maintenance management as demonstrated by the alignment to authoritative maintenance literature (Coetzee, 2004; Kelly, 2006; IEC, 2011, 2016) as well as maintenance practitioner input. Finally the ontology follows the design principles discussed above.

The main challenge in the development of a BFO-conformant ontology for maintenance has been the absence of tried, tested and accepted BFO-aligned classes and relations for the engineering domain. In 2017, an international group of volunteers started the Industrial Ontology Foundry (IOF) with the objective of developing open and principles-based reference ontologies for the manufacturing domain. While work on this is underway, the group has not yet reached the point of agreeing on some of the classes that are used in the ROMAIN ontology described here. As a result, the ROMAIN authors have made judgments based on views in the IOF community that have not yet stabilized.

One challenge has been the use of the class of *maintenance action* and whether this is an *Information Entity* as in a description of the work to be done, or a process in the doing of the work. In ROMAIN we place *Maintenance Action* in the *occurrent* section of the ontology but describe the work to be done in a *Maintenance Work Order Specification* as a *directive information entity*.

Another challenge has been related to defining a condition or state of a component. In maintenance we have the notion of a component degrading. 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). Hence maintenance actions are dependent on the state of the component and the effect of this state its ability to perform a required function. In ROMAIN we have defined two states, a *State of Failure* and a *State of Degradation*. These are sub-classes of process class. Transition into a *State of Degradation* is represented by a process described by a *Triggering Event*. The triggering event class proved to be important in resolving issues around describing this transition. The transition between a *State of Failure* and a *State of Degradation* remains challenging as it depends on defining a function and the level for that required function. For example, one person may decide that the tire tread should be 0.4 mm deep but another that 0.8 mm deep is necessary. A tire tread depth of 0.6 mm will be regarded as in a *State of Degradation* by the first but a *State of Failure* by the second. We expect dealing with transitions to be an area of fertile discussion in the future. For ROMAIN, the main challenge has been the immature status of the application of the BFO as a top-level ontology in engineering. While we are committed to the importance of exploiting foundation ontologies in our work, there are challenges in BFO concerning some aspects such as the definition of Generically Dependent Continuant, mereology and constitution. All of these challenges have been bought to the attention of the BFO community.

Defining the notion of *nonconformity* also presented a challenge when developing the ontology. Nonconformity is the violation of both articulated and unarticulated requirements. However, representation of unarticulated requirements is not possible as they are not specified in design specifications and technical documents. Therefore, we had to limit the formalization of the notion of nonconformity to explicit requirements only. Additionally, nonconformity is a multifaceted entity as it can be an undesired quality or function. Therefore, we had to directly classify it under Specifically Dependant Continuant such that it covers different types of nonconformities. Perhaps application ontologies may need to create more specific classes of nonconformity depending on their needs. Finally, in many cases, nonconformity is realized when an item lacks a function but stating 'A lacks B' is difficult due to the constraints imposed by a realist approach. There are workarounds for this situation but it needs to be stated outside of OWL since it involves instance-to-class relations (Ceusters et al., 2007).

The subject of *causality* has always been a challenging subject for ontologists and they have adopted different approaches for handling this subject philosophically and ontologically (Galton, 2012). In the domain of maintenance, proper formulation and elucidation of the notion of *causality* is essential for effective root-cause analysis. Usually conjunction of multiple states of degradation may cause the maintainable item to enter the state of failure. Defining causality links between various states, processes, and events need further analysis. Also, there are different types of causal relations such as affecting, enabling, and facilitating between various entities that need to be delineated formally when creating a reference ontology for maintenance.

Like others before us, developing a coherent set of axioms has proved challenging. These have to be selected carefully as there are many hidden dependencies in maintenance management. These will be more fully considered in future work.

One of the unique aspects of this project was including a maintenance expert in the development team. As a results, the semantics of the ontology was aligned early on with the established standards and models in the field. One of the pitfalls of most ontology development projects is involving domain experts towards the last stages of design process only for test and validation purposes. We used GitHub as the platform for collaborative ontology development. The issue tracking mechanism of GitHub was particularly useful for suggesting revisions and tracking changes. The ontology is available on github[3].

8. Conclusion

This work presents the first building blocks of ROMAIN, a BFO-compliant reference ontology for maintenance management. We intend to expand this work further while making it open for others to develop a set of ROMAIN compliant modules covering other processes and applications in maintenance. Extension of this initial work will allow the community to move towards an expanded reference ontology for maintenance management. It is intended to help stimulate developments in maintenance-related domain ontologies considering using BFO as a top-level ontology (as explained in fig 12). One of the strengths of the adopted methodology for ontology development in this work was relying on real-life maintenance data for testing and evaluating the ontology. We believe conforming to the vocabulary that is widely used by maintenance professionals and practitioners is a major catalyst for widespread acceptance and uptake. As the ontology evolves, some of the less generic classes of the ontology might be moved to the application-specific extensions of the ontology, thus leaving the core classes application-agnostic.

Acknowledgment

Authors would like to thank members of the IOF (Industrial Ontologies Foundry) community for their many useful comments.

[3]https://github.com/HediKarray/ReferenceOntologyOfMaintenance

Melinda Hodkiewicz thanks the BHP Fellowship for Engineering for Remote Operations and the Australian Research Council through the Industrial Transformation Research Program (Grant No.IC180100030) for their support.

References

Ameri, F., Urbanovsky, C. & McArthur, C. (2012). A systematic approach to developing ontologies for manufacturing service modeling. *CEUR Workshop Proceedings*, *886*, 1–14.

Andersson, B., Bergholtz, M., Edirisuriya, A., Ilayperuma, T., Johannesson, P., Gordijn, J., Grégoire, B., Schmitt, M., Dubois, E., Abels, S., et al. (2006). Towards a reference ontology for business models. In *International Conference on Conceptual Modeling* (pp. 482–496). Springer.

Arnold, P. & Rahm, E. (2014). Enriching ontology mappings with semantic relations. *Data & Knowledge Engineering*, *93*, 1–18.

Arp, R., Smith, B. & Spear, A. (2015). *Building ontologies with Basic Formal Ontology*. Cambridge USA: MIT Press.

Arpirez, J., Gómez-Pérez, A., Lozano, A. & Pinto, H.S. (1998). 2Agent: An ontology-based WWW broker to select ontologies. In *Workshop on Applications of Ontologies and PSMs, Brighton, England*.

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.

Batres, R., Fujihara, S., Shimada, Y. & Fuchino, T. (2014). The use of ontologies for enhancing the use of accident information. *Process Safety and Environmental Protection*, *92*(2), 119–130.

Baumgartner, N. & Retschitzegger, W. (2006). A survey of upper ontologies for situation awareness. In *Proc. of the 4th IASTED International Conference on Knowledge Sharing and Collaborative Engineering, St. Thomas, US VI* (pp. 1–9).

Bendell, T. (1988). An overview of collection, analysis, and application of reliability data in the process industries. *IEEE Transactions on Reliability*, *37*(2), 132–137.

Borgo, S., Guarino, N. & Masolo, C. (1996). Stratified ontologies: the case of physical objects. In *Proceedings of ECAI-96 Workshop on Ontological Engineering* (pp. 5–15).

Ceusters, W. (2012). An information artifact ontology perspective on data collections and associated representational artifacts. In *MIE* (pp. 68–72).

Ceusters, W., Elkin, P. & Smith, B. (2007). Negative Findings in Electronic Health Records and Biomedical Ontologies: A Realist Approach. *International Journal of Medical Informatics*, *76*(3), 326–333.

Coetzee, J.L. (2004). *Maintenance*. Victoria Canada: Trafford Publishing.

CUBRC (2014). Common Core Ontologies for Data Integration. https://www.cubrc.org/index.php/data-science-and-information-fusion/ontology.

Doerr, M. (2003). The CIDOC conceptual reference module: an ontological approach to semantic interoperability of metadata. *AI magazine*, *24*(3), 75.

Ebrahimipour, V., Rezaie, K. & Shokravi, S. (2010). An ontology approach to support FMEA studies. *Expert Systems with Applications*, *37*(1), 671–677.

Ebrahimipour, V. & Yacout, S. (2015). Ontology-based schema to support maintenance knowledge representation with a case study of a pneumatic valve. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *45*(4), 702–712.

Ebrahimipour, V. & Yacout, S. (2016). *Ontology modeling in physical asset integrity management*. New York: Springer.

Elmhadhbi, L., Karray, M.H. & Archimede, B. (2019). Toward the use of upper level ontologies for semantically interoperable systems: An emergency management use case. In *Enterprise Interoperability VIII* (Vol. 9). Springer. Chapter 11.

Falquet, G., Métral, C., Teller, J. & Tweed, C. (2011). *Ontologies in urban development projects*. Springer Science & Business Media.

Galton, A. (2012). States, processes and events, and the ontology of causal relations.

GFMAM (2016). *Maintenance Framework*. London, England.

Gruber, T.R. (1995). Toward principles for the design of ontologies used for knowledge sharing? *International journal of human-computer studies*, *43*(5-6), 907–928.

Grüninger, M. & Fox, M.S. (1995). The role of competency questions in enterprise engineering. In *BenchmarkingâĂŤTheory and practice* (pp. 22–31). Springer.

Guarino, N. (2017). BFO and DOLCE: So Far, So Close... *COSMOS+ TAXIS*.

Guizzardi, G. & Wagner, G. (2010). Using the unified foundational ontology (UFO) as a foundation for general conceptual modeling languages. In *Theory and Applications of Ontology: Computer Applications* (pp. 175–196). Springer.

Hodkiewicz, M. (2018). Working towards a Siri for Maintenance. In *Mastering SAP conference, Gold Coast, Australia*.

IEC (2011). *AS IEC 60300.3.11 Dependability management Application guide - Reliability-centred Maintenance*. Geneva Switzerland.

IEC (2016). *AS IEC 60300.3.14 Dependability management Application guide - Maintenance and maintenance support*. Geneva Switzerland.

ISO (2003). *ISO 15926 Industrial automation systems and integration – Integration of life-cycle data for process plants including oil and gas production facilities – Part 2: Data model*. Geneva Switzerland.

ISO (2012). *ISO 13372 Condition monitoring and diagnostics of machines - Vocabulary*. Geneva, Switzerland.

ISO (2014). *ISO 55001 Asset Management – Management systems – Requirements*. Geneva Switzerland.

Karray, M.H., Chebel-Morello, B. & Zerhouni, N. (2012). A formal ontology for industrial maintenance. *Applied Ontology*, *7*(3), 269–310.

Karray, M.H., Morello, B.C. & Zerhouni, N. (2010). Towards a maintenance semantic architecture. In *Engineering Asset Lifecycle Management* (pp. 98–111). Springer.
Karray, M.-H., Chebel-Morello, B., Lang, C. & Zerhouni, N. (2011). A component based system for s-maintenance. In *Industrial Informatics (INDIN), 2011 9th IEEE International Conference on* (pp. 519–526). IEEE.
Kelly, A. (1997). *Maintenance organization and systems*. Butterworth-Heinemann.
Kelly, A. (2006). *Strategic maintenance planning* (Vol. 1). Oxford: Elsevier.
Kiritsis, D. (2013). Semantic technologies for engineering asset life cycle management. *International Journal of Production Research*, *51*(23-24), 7345–7371.
Knapp, M. & Hasibether, F. (2011). Material master data quality. In *Concurrent Enterprising (ICE), 2011 17th International Conference on* (pp. 1–8). IEEE.
Kuraoka, K. & Batres, R. (2003). An Ontological Approach to Represent HAZOP Information. *Process Systems Engineering Laboratory, Tokyo Institute of Technology*.
Mascardi, V., Cordì, V. & Rosso, P. (2007). A Comparison of Upper Ontologies. In *Woa* (Vol. 2007, pp. 55–64).
Masolo, C., Borgo, S., Gangemi, A., Guarino, N., Oltramari, A. & Schneider, L. (2002). The wonderweb library of foundational ontologies.
Matsokis, A. & Kiritsis, D. (2010). An ontology-based approach for Product Lifecycle Management. *Computers in industry*, *61*(8), 787–797.
Matsokis, A., Karray, H.M., Chebel-Morello, B. & Kiritsis, D. (2010). An ontology-based model for providing semantic maintenance. *IFAC Proceedings Volumes*, *43*(3), 12–17.
Mazzola, L., Kapahnke, P., Vujic, M. & Klusch, M. (2016). CDM-Core: A Manufacturing Domain Ontology in OWL2 for Production and Maintenance. In *KEOD* (pp. 136–143).
Mizoguchi, R. (2010). YAMATO: yet another more advanced top-level ontology. In *Proceedings of the Sixth Australasian Ontology Workshop* (pp. 1–16).
Molina, R., Unsworth, K., Hodkiewicz, M. & Adriasola, E. (2013). Are managerial pressure, technological control and intrinsic motivation effective in improving data quality? *Reliability Engineering & System Safety*, *119*, 26–34.
Morello, B., Karray, M. & Zerhouni, N. (2010). New Persepctives of Maintenance Systems: Towards S-Maintenance. *Proceedings from the IMS2020 Summer School on Sustainable Manufacturing*, *26*, 223–236.
Moubray, J. (2007). *Reliability-centered Maintenance RCM II* (2nd. ed.). Oxford: Butterworth-Heinemann.
Noy, N.F. (2004). Semantic integration: a survey of ontology-based approaches. *ACM Sigmod Record*, *33*(4), 65–70.
Otte, J.N., Rudnicki, R. & Smith, B. (2018). *Final Report For Coordinated Holistic Alignment of Manufacturing Processes (CHAMP)*. New York: University of Buffalo.
Palmer, D. (1999). *Maintenance planning and scheduling handbook*. McGraw-Hill Professional Publishing.
Pease, A. (2006). Formal representation of concepts: The Suggested Upper Merged Ontology and its use in linguistics. *Ontolinguistics. How Ontological Status Shapes the Linguistic Coding of Concepts*.
Rajpathak, D.G. (2013). An ontology based text mining system for knowledge discovery from the diagnosis data in the automotive domain. *Computers in Industry*, *64*(5), 565–580.
Reiman, T. & Oedewald, P. (2004). Measuring maintenance culture and maintenance core task with CULTURE-questionnaire—-a case study in the power industry. *Safety Science*, *42*(9), 859–889.
Ruiz, F., Vizcaíno, A., Piattini, M. & García, F. (2004). An ontology for the management of software maintenance projects. *International Journal of Software Engineering and Knowledge Engineering*, *14*(03), 323–349.
SAE (2011). *SAE JA1012 A guide to the Reliability-centered maintenance (RCM) Standard*. London.
Schulz, S., Balkanyi, L., Cornet, R. & Bodenreider, O. (2013). From concept representations to ontologies: a paradigm shift in health informatics? *Healthcare informatics research*, *19*(4), 235–242.
Semy, S.K., Pulvermacher, M.K. & Obrst, L.J. (2004). Toward the Use of an Upper Ontology for U . S . Government and U . S . Military Domains : An Evaluation Military Domains : An Evaluation.
Simperl, E.P.B., Mochol, M. & Bürger, T. (2010). Achieving Maturity: The State of Practice in Ontology Engineering in 2009. *IJCSA*, *7*(1), 45–65.
Smith, B. (2006). Against idiosyncrasy in ontology development. *Frontiers in Artificial Intelligence and Applications*, *150*, 15.
Smith, B. & Ceusters, W. (2010). Ontological realism: A methodology for coordinated evolution of scientific ontologies. *Applied ontology*, *5*(3-4), 139–188.
Smith, B., Ashburner, M., Rosse, C., Bard, J., Bug, W., Ceusters, W., Goldberg, L.J., Eilbeck, K., Ireland, A., Mungall, C.J., et al. (2007). The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. *Nature biotechnology*, *25*(11), 1251.
SMRP (2009). *SMRP Best Practice - Maintenance |& Reliability Body of Knowledge* (5th. ed.). Atlanta, GA: Society of Maintenance and Reliability Professionals.
Stevenson, G., Knox, S., Dobson, S. & Nixon, P. (2009). Ontonym: a collection of upper ontologies for developing pervasive systems. In *Proceedings of the 1st Workshop on Context, Information and Ontologies* (p.9). ACM.
Stuber, A. (2016). *The Maintenance Framework*. Global Forum on Maintenance and Asset Management.
Suarez-Figueroa, M.C. & Gómez-Pérez, A. (2008). First attempt towards a standard glossary of ontology engineering terminology.