

Research Article

Cite this article: Diao X, Zhao Y, Vaddi PK, Pietrykowski M, Khafizov M and Smidts C (2024). Multiple aspects maintenance ontology-based intelligent maintenance optimization framework for safety-critical systems. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, **38**, e3, 1–19
<https://doi.org/10.1017/S0890060423000215>

Received: 04 October 2022
Revised: 10 October 2023
Accepted: 03 November 2023


Keywords:

maintenance optimization; ontology development; online monitoring; decision-making; safety-critical systems

Corresponding author:

Xiaoxu Diao;
Email: diao.38@osu.edu

Multiple aspects maintenance ontology-based intelligent maintenance optimization framework for safety-critical systems

Xiaoxu Diao , Yunfei Zhao, Pavan K. Vaddi, Michael Pietrykowski, Marat Khafizov and Carol Smidts

The Department of Mechanical and Aerospace Engineering, The Ohio State University, Columbus, OH, USA

Abstract

Maintenance optimization is a process for improving the efficiency of maintenance strategies and activities, considering various aspects of the target system and components, such as the probabilities of system failures and the cost of repair and replacement of a failed component. The improvement of maintenance optimization algorithms generally requires information from various data sources. For example, it may require the system risk information derived from risk analysis tools or the residual lifetime of a component from fault prognosis tools. The requirements of data acquisition (DAQ) and aggregation pose new challenges for maintenance management systems (MMSs) that implement and use these maintenance optimization algorithms. This paper proposes a multiple aspects maintenance ontology-based framework to facilitate DAQ from MMSs, online monitoring systems, fault detection and discrimination tools, risk assessment tools, decision-making tools, and component identification tools, and accelerate the implementation and verification of contemporary maintenance optimization models and algorithms. The proposed framework consists of a multi-aspect maintenance ontology with critical information for maintenance optimization and application interfaces for collecting information from various data sources, such as fault prognosis tools, online monitoring tools, risk assessment tools, and decision-making algorithms. In addition, this paper proposes a heuristic method for integrating concepts and properties from other existing ontologies into the proposed framework when the existing ontology is not fully compatible with the ontology under construction. Finally, the paper verifies the proposed ontology framework using a feedwater system designed for nuclear power plants with valves and filters as the components under maintenance.

Introduction

Safety-critical systems are systems whose failure will cause catastrophic consequences, such as massive property damage or fatalities. Safety-critical systems are commonly used in modern industrial systems, such as flight control systems in airplanes and cooling systems in nuclear power plants (NPPs). Component or structure degradations and failures during the operation of such systems may significantly impact the functions implemented by such components and subsequently increase the risk of system failures and catastrophic consequences. System maintenance activities aim to keep system components reliable and functional, which is essential for the long-term, safe, and economic operation of industrial systems. However, maintenance activities usually are faced with challenges, such as being time-consuming and costly. Therefore, optimization is required to reduce maintenance activities' expenses and maintain system outages to a minimum. The optimization process should ensure that the most appropriate type of maintenance is performed on system, structure, and components (SSCs) and determine at which periodicity maintenance should be performed on account of regulatory requirements and maintenance targets related to safety, reliability, availability, and cost (IAEA, 2018).

As a modern maintenance optimization methodology, condition-based maintenance (CBM) collects and assesses real-time information and recommends maintenance decisions based on the current operational condition of the target system (Alaswad and Xiang, 2017). Many CBM models have been proposed (Sharma et al., 2011), such as the hidden Markov process model (Zhao and Smidts, 2022), the Wiener process model (Guo et al., 2013), and the Gamma process model (Yuan et al., 2021). However, the mathematical models cannot be applied without real-time data collected from system operations, failures, modifications, and costs (Sharma et al., 2011). As an example, these models require data from condition monitoring to identify the state of system functions (e.g., operating or failed) and components (e.g., on or off) (de Jonge and Scarf, 2020). Therefore, several maintenance optimization algorithms are developed (Kobbacy, 2004) to implement the CBM models designed for various specific industrial systems. These optimization algorithms and their implementations provide interfaces for collecting monitoring data from

© The Author(s), 2024. Published by Cambridge University Press. This is an Open Access article, distributed under the terms of the Creative Commons Attribution licence (<http://creativecommons.org/licenses/by/4.0>), which permits unrestricted re-use, distribution and reproduction, provided the original article is properly cited.

various sources as model inputs, running algorithms for evaluating the effectiveness and cost of maintenance activities and making decisions, and interacting with existing maintenance management systems (MMSs) for maintenance work order deployment and maintenance model evaluation (Lee *et al.*, 2020).

Nevertheless, developing a maintenance optimization system requires us to overcome several challenges. These challenges are becoming more severe as the variety and complexity of CBM models increase (Apeland and Aven, 2000; Alaswad and Xiang, 2017). Existing optimization systems only focus on one or two aspects of maintenance, such as security and economic loss (Gu *et al.*, 2009) or making decisions based on fault prognosis (Wang *et al.*, 2006).

This paper implements an ontology-based framework to bridge the gap between the maintenance optimization models and their implementations. This framework facilitates model applications for maintenance optimization by unifying the concepts and algorithms used by models and industrial applications, providing interfaces to the modules required by the maintenance optimization process, and automatically routing required data to these modules. The following list summarizes the contributions of this paper.

- **An MuAMO with unified concepts and properties required by maintenance optimization considering multiple aspects**, such as SSCs, monitoring data, system risk, and decision-making. The proposed ontology serves as a knowledge repository with generic and specific information for running various maintenance optimization models.
- **An ontology-based framework for implementation and evaluation of maintenance optimization models**. The optimization system developers do not need to repeatedly adapt the optimization models for different target systems. Using the unified concepts and measures defined by MuAMO, various optimization models can be quickly integrated into the proposed framework and applied to various specific systems. In addition, the proposed framework can automatically infer the models with the best fitness for the target component or system. The satisfaction of preconditions defined for different optimization models can be verified automatically by the ontology build-in solvers.
- **A series of application interfaces for integrating with other information resources**. Generally, a maintenance optimization algorithm requires various system information, such as system structures, configurations, failure modes, and real-time monitoring data from sensors and data acquisition systems. As an information aggregation platform, the proposed framework includes interfaces designed and implemented for collecting information from third-party tools that provide the information required by the maintenance optimization algorithms.
- **A heuristic method to extend the existing ontology and integrate it with other domain ontologies**. To efficiently enrich the concepts and properties of MuAMO, we propose a heuristic method for ontology integration. Specifically, the proposed method can minimize the hierarchical changes to the classes in existing ontologies.

In the following sections, the “Related work” section introduces related work which has used ontologies for maintenance optimization. The “Overview” section overviews the structure of the proposed framework. The development methodology and process used to develop MuAMO are detailed in the “Ontology development methodology” section. The “Ontology framework” section details the algorithms used by the framework to perform maintenance

optimization in real time. The “Implementation” section discusses the implementation of the framework. The “Case study” section uses two components in a feedwater system for an NPP as the case study. The “Discussion” section describes the use of the proposed framework. The “Conclusion” section concludes the paper.

Related work

“An ontology is an explicit specification of a conceptualization” (Gruber, 1993). As an effective way for standardizing and sharing information, ontologies have become increasingly valuable tools in computer science for their utility in enabling a thorough and well-defined discourse and for building logical models of systems. Ontology has been applied to various system engineering processes, such as collaborative assembly design (Kim *et al.*, 2009), engineering knowledge modeling (Harrison and Chan, 2009) and maintenance (Ajit *et al.*, 2008), and project management (Wu *et al.*, 2021). However, these ontologies are developed for the development stages of industrial systems and cannot support the tasks performed in the maintenance stage.

For system maintenance and optimization, some ontologies were developed for specific industrial fields. For example, Elhdad *et al.* (2013) proposed an ontology for process monitoring and maintenance in petroleum plants. The proposed ontology can enhance maintenance decisions based on the knowledge gathered through the process of monitoring. This monitoring process is based on signals which are triggered during the plant safety shutdown process. The proposed ontology is extended to ensure that decision-makers have sufficient information to make the right decision at the right time. Ebrahimipour and Yacout (2015) developed an ontology-based schema to support maintenance knowledge representation for physical components. The schema can overcome the problems of heterogeneity and inconsistency in maintenance records. A bond graph model is employed to model the structure of equipment functions involved in fault propagation at the part-component level. In addition, this method provides a generic technical understanding, which enriches semantic extraction and knowledge discovery in a typical maintenance report. Anquetil *et al.* (2006) developed an ontology for software maintenance optimization. They defined two common maintenance scenarios and considered the industrial issues associated with them. Campos (2007) proposed an ontology for asset management for mechanical systems. The ontology records the data structure and information needed for the maintenance management of the industrial assets with their objects, attributes, and relationships. Dimitrova *et al.* (2020) developed PADTUN, a novel intelligent decision support system that assists with pathology diagnosis and assessment of tunnels with respect to their disorders and diagnosis influencing factors. Nevertheless, these ontologies are designed for specific industrial fields or for a specific type of industrial systems.

For the purposes of generalization, the industrial maintenance management ontology (Karray *et al.*, 2012) was developed for industrial maintenance, ensuring semantic interoperability and generating new knowledge that supports decision-making in the maintenance process. ROMAIN (Karray *et al.*, 2019) is a domain-specific ontology for the maintenance management domain. It constrains the classes that are unique to the maintenance management practice, such as maintenance strategy, degradation, and work order management. Emmanouilidis *et al.* (2020) conducted a study of maintenance ontologies from the viewpoint of

Table 1. Literature comparison of the ontologies related to maintenance optimization (IMAMO: industrial maintenance management ontology; MuAMO: multiple aspects maintenance ontology; ROMAIN: a domain-specific ontology for the maintenance management)

Methodology	Maintenance activities	Asset management	Failures	Monitoring	Risk	Cost	Decision-making
IMAMO (Karray et al., 2012)	Yes	Yes	Yes	Yes	No	No	No
ROMAIN (Karray et al., 2019)	Yes	Yes	Yes	Yes	No	Yes	No
(Emmanouilidis et al., 2020)	Yes	Yes	Yes	Yes	No	No	Yes
(Canito et al., 2021)	Yes	Yes	Yes	Yes	No	No	Yes
MuAMO	Yes	Yes	Yes	Yes	Yes	Yes	Yes

MuAMO is the methodology/ontology proposed in this paper.

reliability-oriented context information management and proposes a baseline context information management ontology aligned with the needs of maintenance services for connected production machines. This ontology is applied on an industrial case study relevant to maintenance services for a distributed fleet of connected industrial printers. Canito et al. (2021) proposed an ontology to bridge the gap between data sampling from different types of sensors and equipment for predictive maintenance. It also provides interface for machine learning algorithms to train data mining models. Table 1 compares the existing ontologies or methodologies recently designed for system maintenance in terms of features important to this paper's maintenance optimization, such as the maintenance activities, assets, failures, risk, and cost. From the table, we can observe that none of these ontologies are able to cover all the aspects considered in the proposed ontology MuAMO.

Framework and methodology

Overview

The objective of the maintenance ontology framework is to facilitate the development of a maintenance optimization system that requires information from various data sources. Figure 1 describes the methodology used for achieving the proposed target. This section first introduces the ontology development methodology used for establishing the maintenance ontology framework. The method proposed for creating classes and properties is based on existing standards and reports. Also, this section includes a method to integrate existing ontologies into the proposed framework. The outcome of the introduced ontology development method is the maintenance ontology framework with its associated views. This section details the classes and properties defined for each view (see the "Ontology framework" section) and then introduces the use of the proposed framework in the "Implementation" section, in which

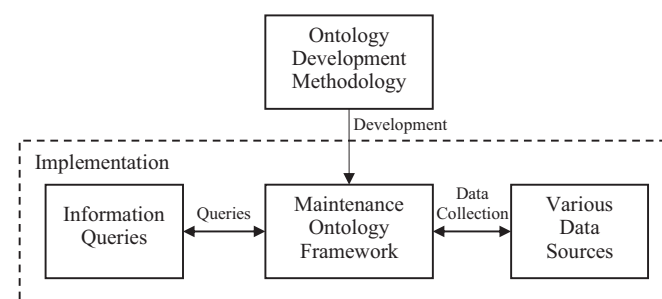


Figure 1. Roadmap of the paper.

the related algorithms and interfaces are discussed, and the queries generated for eliciting required information are explained.

Ontology development methodology

The ontology development methodology defines the steps and the pathway to design, implement, and verify a domain-specific ontology. This paper refers to and adjusts some mature ontology engineering methods that are already applied in other industrial domains (López et al., 1999; El-Diraby, 2013; Chantrapornchai and Choksuchat, 2016; Kethavarapu and Saraswathi, 2016; Qu et al., 2016). In Steps II and III, a heuristic methodology for defining and integrating ontologies is detailed since they are part of contributions of this paper. The development of the proposed maintenance ontology follows an ontology engineering approach which includes the following steps (Noy and McGuinness, 2017).

Step I. Determine the domain and scope of the ontology

Generally, an ontology is developed to solve a series of problems for a specific type of target system or a specific domain, and the competency questions are used to define the requirements for the ontology, that is, the competency questions are the necessary questions the ontology needs to answer (Noy and McGuinness, 2017). Therefore, this paper defines the domain of the proposed ontology as supporting maintenance optimization and related activities.

We also defined the competency questions that the proposed ontology needs to answer. These competency questions are used as necessary criteria for verifying the completeness of the proposed ontology in the "Case study" section. Since the objective of the proposed ontology is to facilitate the development of a maintenance optimization system, the following competency questions are defined, derived from our understanding of industry needs and that should be answered by maintenance optimization systems:

- To decrease the risk of the target system, what is the most efficient maintenance activity (MEMA)?
- Considering maintenance costs and system risks, what is the most critical part of the target system/component?
- What is the maximum achievable improvement of system safety in terms of all feasible maintenance activities given a specific budget?

Step II. Consider reusing existing ontologies

Many domain ontologies have been developed to solve knowledge representation and sharing issues. Reusing existing ontologies can save time and maximize compatibility with other existing ontologies. However, reusing terms in one domain may cause inconsistencies or

Table 2. The advantages and disadvantages of reusing existing ontologies

Advantages	Disadvantages
<ul style="list-style-type: none"> Unify terms and concepts accepted by a wide range of experts and professionals. 	<ul style="list-style-type: none"> Must accept the predefined terms leading to loss of flexibility.
<ul style="list-style-type: none"> Reuse the concepts and constraints defined by other ontologies. 	<ul style="list-style-type: none"> May cause inconsistencies and contradictions.
<ul style="list-style-type: none"> Easy to be reused by other ontologies. The proposed ontology can be smoothly integrated with the other upcoming ontologies. 	<ul style="list-style-type: none"> May introduce a large number of unused concepts and terms.

even contradictions since the terms commonly used in different domains may have different meanings and explanations.

Table 2 summarizes the advantages and disadvantages of reusing existing ontologies. This paper balances these advantages and disadvantages when investigating and integrating existing ontologies into MuAMO.

By considering the problems that may be encountered as discussed above, existing ontologies are carefully evaluated to ensure their compatibility with the field of maintenance. Table 3 summarizes the domain ontologies considered for reuse in the proposed maintenance ontology framework. It is worth noting that upper-level ontologies are also investigated for consideration in the proposed ontology since these ontologies provide meta-concepts and properties useful for bridging concepts defined in different domain ontologies.

In recent years, researchers have provided many ontologies for conceptualizing and sharing knowledge between research domains, such as the basic formal ontology (BFO) (Smith, 1998), the suggested upper merged ontology (Niles and Pease, 2001), and the descriptive ontology for linguistic and cognitive engineering (Gangemi et al., 2003). Since most of the reused ontologies are compatible with BFO, the proposed framework will also be built based on BFO (Smith, 1998).

Since some of the ontologies do not use BFO (Smith, 1998) as their meta-ontology (i.e., some of the ontologies are not compatible with BFO), the existing ontology integration methods are not applicable; hence, reusing and integrating the concepts and relations in such ontologies requires extra efforts to prevent inconsistencies and conflicts. This paper proposes an expert-aided incremental method to integrate portions of the existing domain ontologies into the proposed ontology (MuAMO). To minimize the changes in the relations between the concepts in MuAMO and the concepts in the integrated ontology, we propose a heuristic method for ontology integration. The following definitions are required by the method.

Assume that the ontology of the proposed framework $\mathcal{O} = \{\mathcal{C}, \mathcal{R}\}$ contains a set of concepts $\mathcal{C} = \{c_1, \dots, c_m\}$ (i.e., classes) and properties $\mathcal{R} = \{r_1, \dots, r_n\}$ (i.e., links), where c represents a class and r represents a property (i.e., a link). If we restrict \mathcal{R} as the set of all “is_a” links between two classes, the notation $r(c_i, c_j)$ or r_{ij} denotes that a “is_a” link exists between class c_i and class c_j , that is, class c_i is the parent class of class c_j .

Definition 1. Function $\mathcal{L}(c_i, c_j): \mathcal{C} \times \mathcal{C} \rightarrow \mathcal{R}$ returns the set of links R which start from c_i and end at c_j . Based on this definition, the notation $\mathcal{L}(c_i, *)$ represents the set of links starting from c_i and $\mathcal{L}(*, c_j)$ represents the set of links ending at c_j .

Definition 2. Function $\mathcal{D}(c_i): \mathcal{C} \rightarrow \mathcal{C}$ returns the descendants of class c_i . The descendants include all the subclasses of class c_i , all the subclasses of these subclasses, and so on.

Definition 3. Function $\mathcal{D}_{\mathcal{S}}(c_i): \mathcal{C} \rightarrow \mathcal{C}$ returns the direct descendants (children) of class c_i . The direct descendants include only the subclasses of class c_i .

Definition 4. Function $\mathcal{U}(c_i): \mathcal{C} \rightarrow \mathcal{C}$ returns the ancestors of class c_i . The ancestors include all the parent classes of class c_i , all the parent classes of these parent classes, and so on.

Definition 5. Function $\mathcal{U}_{\mathcal{S}}(c_i): \mathcal{C} \rightarrow \mathcal{C}$ returns the direct ancestors (parents) of class c_i .

Definition 6. Top Nodes $\mathcal{T}(C)$ return the set of top classes in the class set C so that $\forall c \in C, c \in \mathcal{T}(C)$ if $\mathcal{U}_{\mathcal{S}}(c) = \emptyset$.

Definition 7. Block $B = \{C, R\}, C \subseteq \mathcal{C}, R \subseteq \mathcal{R}, \mathcal{L}(c_i, c_j) \neq \emptyset$, where $c_i \in C, c_j \in C, c_i \neq c_j$. The notation means that all the classes in a block are linked, that is, there is at least one link starting from or ending at every concept. Generally, an ontology \mathcal{O} can be divided into various block set $\mathcal{B} = \{B_1, \dots, B_l\}$. For each B_i in $\mathcal{B}, \mathcal{T}(B_i) = \{c_T\}$ has only one element which is the top class in block B_i .

Definition 8. Minmax block set \mathcal{B}_M is a block set in which there is no link between the concepts belonging to different blocks. $\mathcal{B}_M = \{B_1, \dots, B_l\}, \mathcal{L}(c_i, c_j) = \emptyset, c_i \in B_x, c_j \in B_y, B_x \in \mathcal{B}_M, B_y \in \mathcal{B}_M, x \neq y$.

Assume that the new ontology with new concepts and links $\mathcal{O}' = \{\mathcal{C}', \mathcal{R}'\}$ will be integrated into the ontology $\mathcal{O} = \{\mathcal{C}, \mathcal{R}\}$. The Minmax block set of the new ontology \mathcal{O}' is $\mathcal{B}'_M = \{B'_1, \dots, B'_l\}$. Then the integration process can be classified into the following cases.

Case 1, for each B'_i in \mathcal{B}'_M , no common class between B'_i and \mathcal{C} exists, that is, $C' \cap C = \emptyset, C' \in B'_i$.

In this case, the top class $\mathcal{T}(B'_i)$ is selected first to be integrated into the ontology \mathcal{O} . Expert knowledge is required in this step to identify the concept c_0 that is closely related to the top class $\mathcal{T}(B'_i)$ and add the top class as the subclass of c_0 by adding the relation $r(c_0, \mathcal{T}(B'_i))$. Then the other classes and relations in B'_i can be added into the ontology \mathcal{O} . For example, Figure 4 displays the block of classes B'_i that will be integrated into the existing ontology. The block $B'_i = \{C', R'\}$ contains six classes $C' = \{c'_1, \dots, c'_6\}$ and their relations $R' = \{r'_{1,2}, r'_{1,3}, r'_{2,4}, r'_{2,5}, r'_{3,6}\}$. The top class $\mathcal{T}(B'_i) = \{c'_1\}$.

In this case, the top class c'_1 will be integrated into the ontology \mathcal{O} first by using expert knowledge. As shown in Figure 5, the class c_3 is selected as the parent class of c'_1 , and then the relation $r(c_3, c'_1)$ is added into the ontology \mathcal{O} . Finally, the set of classes and relations of block B'_i are added into the ontology \mathcal{O} .

It is worth noting that the class c'_1 will never be the top class of the ontology \mathcal{O} since the ontology \mathcal{O} defined by the propose framework uses BFO (Smith, 1998) as the meta-ontology which defines “Thing” as the top class.

Case 2, a common class c'_1 exists between B'_i and $\mathcal{C}, C' \cap \mathcal{C} = \{c'_1\}, C' \in B'_i$.

This case can be further classified as $\{c'_1\} = \mathcal{T}(B'_i)$ and $\{c'_1\} \neq \mathcal{T}(B'_i)$.

Table 3. Domain ontologies considered for reuse

Domains	Descriptions	Existing ontologies
System maintenance	The ontologies represent maintenance strategies, activities, and related factors.	<ul style="list-style-type: none"> Industrial maintenance management ontology (IMAMO) (Karray et al., 2012): an ontology for general industrial equipment maintenance. (Elhdad et al., 2013) An ontology-based framework for process monitoring and maintenance in petroleum plants. (Ebrahimipour and Yacout, 2015) An integrated ontology for representing maintenance knowledge for industrial system components. A domain-specific, open access, reference ontology (ROMAIN) (Karray et al., 2019): a basic formal ontology (BFO)-compatible ontology (see below in table) for the general industrial maintenance domain. Manufacturing semantic ontology (MASON) (Alkahtani et al., 2019): a decision support system based on ontology and data mining to improve the design. MIMOSA Cris (common relational information schema) (Campos, 2007): an ontology for industrial asset management.
Fault and failure	The ontologies represent knowledge about faults and failures.	<ul style="list-style-type: none"> A fault ontology (Diao et al., 2022) is used for the analysis of faults at the design stage.
Safety and risk	The ontologies represent knowledge about risk assessment, initial events, consequences, probabilities, and severities, and so forth.	<ul style="list-style-type: none"> An ontology-based framework for risk assessment in road scenes (Mohammad et al., 2015). HUFO (Oltramari et al., 2015): a human factors ontology for cybersecurity risk assessment.
Decision-making	The ontologies represent knowledge about the criteria, the goals, and the algorithms for decision-making.	<ul style="list-style-type: none"> A decision-making ontology for information system engineering (Kornysheva and Deneckère, 2010).
Upper-level ontologies	The ontologies define high-level concepts that guarantee the compatibility of the new ontologies with existing ontologies.	<ul style="list-style-type: none"> BFO (Arp et al., 2016): an upper-level (formal, domain-neutral) ontology to support the creation of lower-level domain ontologies. BFO is at the core of the open biological and biomedical ontology. The open biological and biomedical ontology foundry, which is an initiative aiming to develop interoperable ontologies based upon shared principles and architecture.

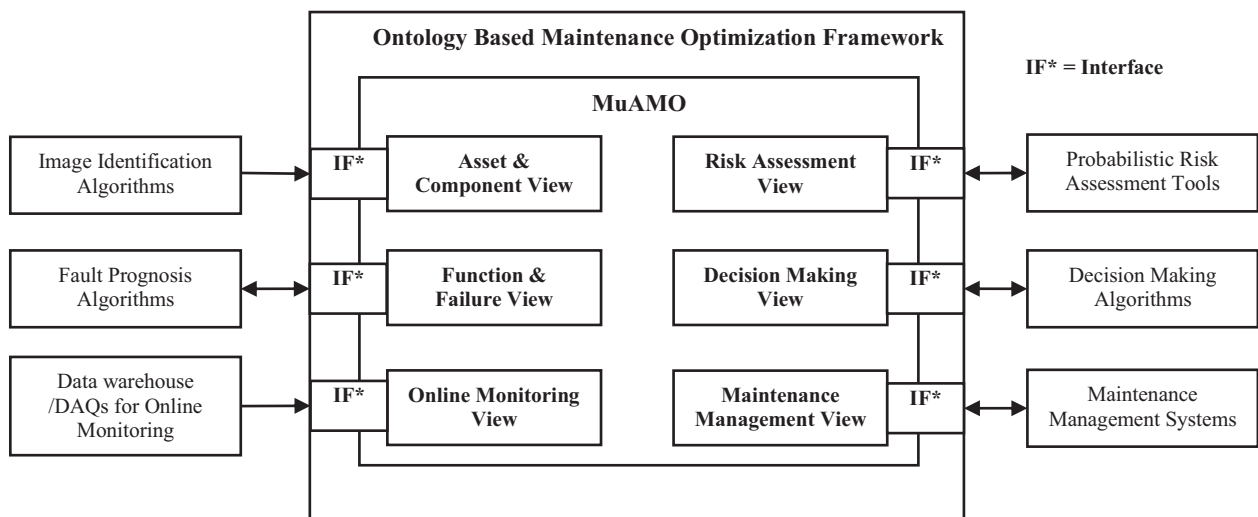


Figure 2. Architecture of the ontology-based maintenance optimization framework.

- Case 2.1,** $\{c'_1\} = \mathcal{T}(B'_i)$. In this case, the other classes $\{c'_2, \dots, c'_6\}$ and relations in B'_i can be added into the ontology \mathcal{O} following the procedure in Case 1. As shown in Figure 6, the class c'_1 is the common class that exists both in C' and \mathcal{E} . Then the block B'_i is integrated into \mathcal{E} without the changes of relations between the classes in C' and \mathcal{E} .
- Case 2.2,** $\{c'_1\} \neq \mathcal{T}(B'_i)$. In this case, if we define the block with class c'_1 as B'_{i_0} and the other derived blocks as $B'_{i_1}, \dots, B'_{i_n}$, then we can add B'_{i_0} into ontology \mathcal{O} using the method in Case 2.1 and

add the other blocks $B'_{i_1}, \dots, B'_{i_n}$ using the method in Case 1. As shown in Figure 7, c'_2 is the common class with \mathcal{E} , but c'_2 is not the top class of B'_i . In this case, the link $r'_{1,2}$ is pruned, and the descendants of c'_2 are added into \mathcal{E} . Then the new block with c'_1 and its descendants are integrated by using the method in Case 1. In this example, c'_1 is added as the child class of c_1 .

Case 3, more than one common classes exist between B'_i and \mathcal{E} , $C' \cap \mathcal{E} = \{c'_1, \dots, c'_t\}, C' \in B'_i$.

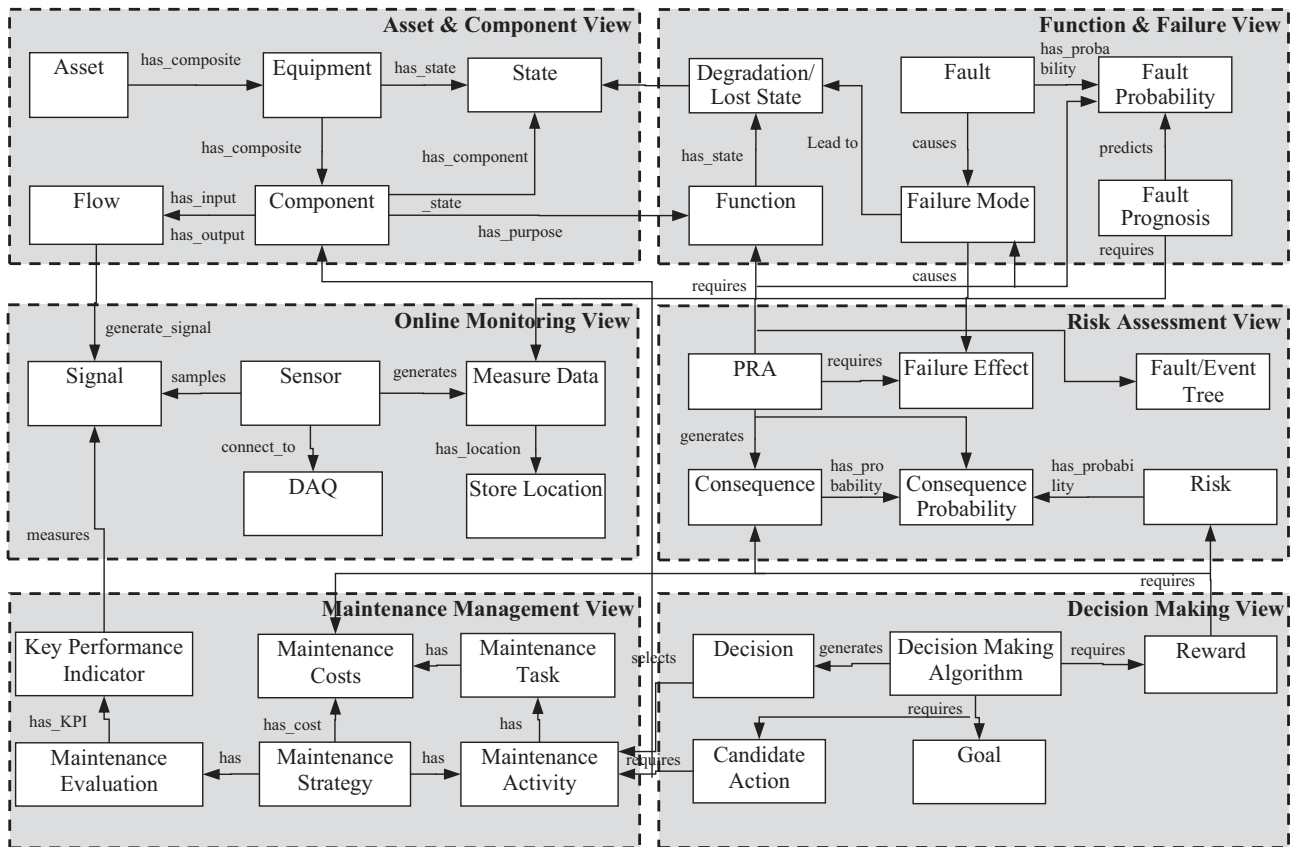


Figure 3. Dependencies between the views of the proposed framework.

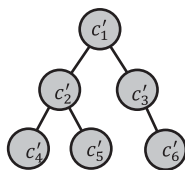


Figure 4. Example block that will be integrated into the existing ontology.

- **Case 3.1.** $\nexists r'(c'_i, c'_j), c'_i \in C, c'_j \in C, r(c'_i, c'_j) \in \mathcal{R}$. In this case, no common link exists in both B'_i and \mathcal{O} . We can break the links $\mathcal{L}(*, c'_i), c'_i \in C$ and generate new blocks. For each new block, we can use the methods in Case 2. Figure 8 shows an example of this case.
- **Case 3.2.** $\exists r'(c'_i, c'_j), c'_i \in C, c'_j \in C, r(c'_i, c'_j) \in \mathcal{R}$. In this case, there is at least one common link existing in both B'_i and \mathcal{O} . Since c'_i, c'_j and r'_{ij} are already in ontology \mathcal{O} , we can break the link $\mathcal{L}(*, c'_i)$ and add the derived blocks using the methods in Case 2. Figure 9 shows an example of this case.

Step III. Enumerate important terms in the ontology

Since the proposed ontology is used to model and manage knowledge for maintenance optimization, the important terms describing concepts and relations related to this target need to be included in the proposed ontology. This step collects and determines the important terms widely used in the target domain. These terms are collected from several industry standards and reports that domain

experts and researchers commonly accept. In this paper, the importance of these terms is evaluated by the experts. In the future, natural language processing tools can be reused in this step.

The following content describes the steps of the methodology used for collecting information from these documents.

Step III.1. Locate descriptions of key concepts, definitions, restrictions, or dependencies in the target document.

Step III.2. Select the terms denoting the key concepts or relations in the target domain.

Step III.3. Iterate on the selected terms w_0 :

- **Step III.3.1.** For each noun, compare it with the classes \mathcal{C} already in the ontology \mathcal{O} . If the selected term w_0 has the same meaning as a class c_i already defined by the ontology \mathcal{O} , link the selected term as an alias of the class *aliases* (w_0, c_i). Otherwise, from the ontology \mathcal{O} , find a class c_i whose meaning is closest to the new term and define the new term as a subclass of it $\mathcal{L}(c_i, w_0) = \{r(c_i, w_0)\}$.
- **Step III.3.2.** For each verb, compare it with the links \mathcal{R} already in the ontology \mathcal{O} . If the selected term has the same meaning as a link r_{ij} already defined by the ontology \mathcal{O} , define the selected term as an alias of the link *aliases* (w_0, r_{ij}). Otherwise, if the subject(s) and the object(s) of the verb, that is, the concepts c_i and c_j related to the link, are already in the ontology \mathcal{O} , then add the new link to the ontology $\mathcal{O}, \mathcal{L}(c_i, c_j) = \{r_{ij}\}$. If one or more of the related concepts are not defined in the ontology \mathcal{O} , repeat Step III.3.1 for adding the missing concepts.
- **Step III.3.3.** If a rule or constraint is defined by a link of some concepts, an axiom represented by the Semantic Web Rule

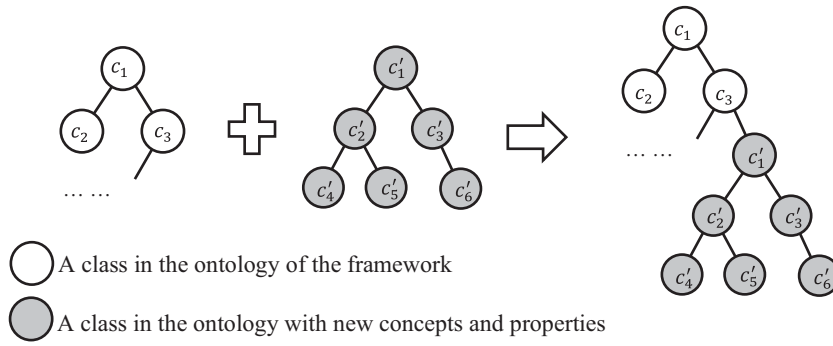


Figure 5. Example of Case 1.

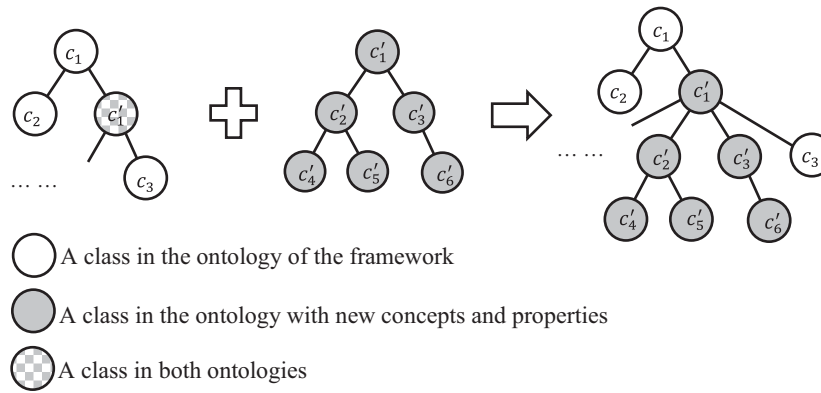


Figure 6. Example of Case 2.1.

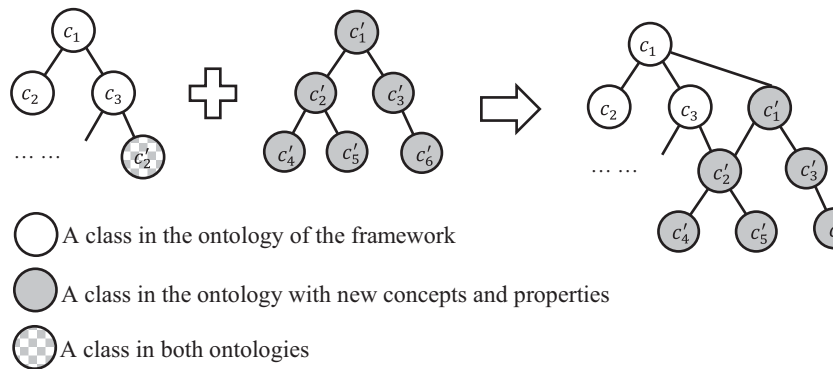


Figure 7. Example of Case 2.2.

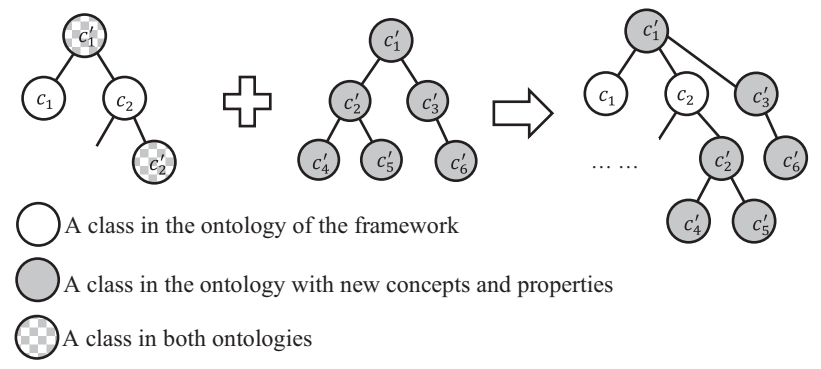


Figure 8. Example of Case 3.1.

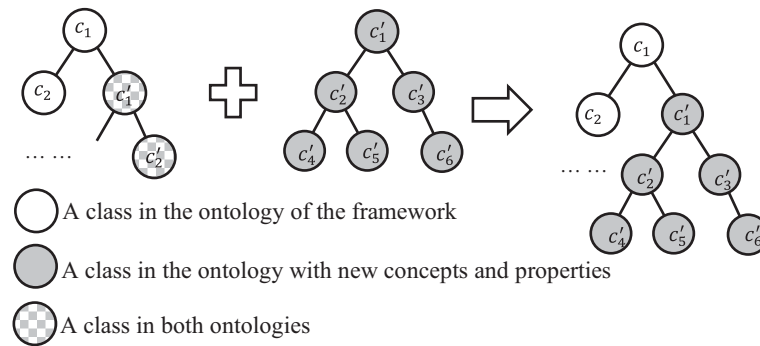


Figure 9. Example of Case 3.2.

Table 4. Documents referenced by the proposed ontology (NPP: nuclear power plant; PRA: probabilistic risk assessment)

Doc no.	Doc title	Description	Elicited information
ASME-PRA-S-2000	Standard for PRA for NPP applications	This standard sets forth requirements for PRAs used to support risk-informed decisions for commercial NPPs and prescribes a method for adapting these requirements for specific applications.	The main concepts related to risk assessment are collected from this document, including basic events, event tree analysis, and fault tree analysis. These terms mainly derive from initiating events, accident sequence, success criteria, system, and data analysis.
DOE G 433.1	Nuclear Facility Maintenance Management Program Guide	This guide introduces essential maintenance management program elements.	Concepts extracted from this document include maintenance activities, organization and administration, facilities and tools, procedures, and procurement of parts, materials, and services.
IAEA NP-T-3.8	Maintenance optimization program for NPPs	This publication describes the latest NPP maintenance optimization programs, including critical requirements, and provides strategies for their successful implementation.	The proposed framework reuses the terms regarding maintenance optimization processes, prerequisites of maintenance optimization, scoping and classification of structures, systems, and components, approaches to maintenance optimization, work management, and online maintenance.
IAEA TECDOC 1383	Guidance for optimizing NPP maintenance programs	This publication was developed to collect and analyze proven maintenance optimization methods and techniques (engineering and organizational).	This publication supplements condition-based maintenance activities, monitoring technologies, and maintenance performance indicators.
NUREG CR-6928	Industry-Average Performance for Components and Initiating Events at U.S. Commercial Nuclear Power Plants	This report addresses four types of risk model events: component unreliability, component or train unavailability, special event probabilities, and initial event frequencies.	The proposed ontology references the probability and distribution that the system component cannot perform its safety function because of maintenance outages. Some initial events are referenced as well.

Language (SWRL) (Horrocks et al., 2004) will be defined since some complex relations cannot be represented by adding a new link.

For example, the following sentence is extracted from a maintenance optimization document and is selected since it contains key concepts and relations in the maintenance domain.

“Maintenance techniques can also check the status of components that are not malfunctioning and seek to increase the interval of inspection.”

The key terms “Maintenance techniques,” “components,” “status of components,” “malfunctioning,” “inspection,” “interval of inspection,” “check,” “seek,” and “increase” are included in the sentence. According to the methods introduced above, the terms “Maintenance technique” and “component” are already defined in the proposed framework. The term “status of component” is defined as an alias of “state of component,” and the term “malfunctioning” is defined as an alias of “failure.” The term “inspection” is defined as a child class of “Maintenance activity,” and the term “interval of inspection” is already defined by the class “Maintenance time

interval” based on expert’s knowledge. The term “check” builds the relationship between the concepts “maintenance technique” and “status of components” initially. But later this relationship is revised to be the link between the concepts “maintenance team” and “maintenable item” in Step III. The terms “seek” and “increase” are pruned since these relations are already represented by other links in the proposed ontology.

Step III.4. Refine the established ontology

- **Step III.4.1.** Remove contradictions or inconsistencies by using the ontology built-in reasoners (e.g., HermiT; Glimm et al., 2014). (Inconsistencies may exist between different documents; some terms in the nuclear power field may be inconsistent with the ones in other fields.)
- **Step III.4.2.** If there are too many direct subclasses of a class, then add more intermediate classes to optimize the ontology hierarchy.
- **Step III.4.3.** Eliminate redundant classes, properties, or axioms and revise inappropriate classes, properties, or axioms based on expert’s knowledge, if applicable.

Algorithm 1: Optimal Maintenance Activity Selection

```

CM=Get_Maintainable_Items()
FOR cm in CM
AC=Get_applicable_Maintenance_activity(cm)
DM=Get_applicable_Optimization_Algorithms(AC, cm)
IF DM is not NULL
  IF If_require_failure_information(DM) THEN
    FF=Get_failure_information(DM, cm)
  ELSE
    FF=NULL
  ENDIF
  IF If_require_risk_information(DM) THEN
    RI=Get_risk_information(DM, cm)
  ELSE
    RI=NULL
  ENDIF
  OP=Get_Optimal_Activity(DM, AC, FF, RI)
  IF OP is not NULL
    Update_Maintenane_strategy(OP)
  ENDIF
ENDIFOR

```

Figure 10. Algorithm of maintenance optimization process using the ontology repository.

Algorithm 2: Get Functional Failure Information of the Target System**Inputs:**

DM, the decision making algorithm
cm, the target maintenance component

```

FN=Get_Functions_relatedto_component(cm)
ME=Get_required_measure(DM)
RN=Create_empty_list()
FOR fn in FN
  FM=Get_failure_modes(fn)
  FOR fm in FM
    MV=Get_required_measure_value(ME)
    IF MV is not NULL
      Append_List(RN, (fn, fm, MV))
    ENDIF
  ENDFOR
ENDFOR
Outputs:
RN, the list of functional failure with their probabilities.

```

Figure 11. Algorithm for obtaining failure information.

Step III.5. Validate the established ontology. Check if all the key concepts, rules, and constraints have been covered. This step is usually completed by domain experts.

The current method depends on experts' judgment to determine which terms are important. Usually, an important concept or relation will appear multiple times in a standard or document. Experts can select the sentences or paragraphs that contain the key concepts, definitions, restrictions, or dependencies described using nouns or verbs. If there is no sentence or paragraph that describes the important concepts, definitions, restrictions, or dependencies using nouns or verbs, the experts can rewrite that sentence or paragraph and then apply the proposed method.

Algorithm 3: Get System Risk Information of the Target System**Inputs:**

DM, the decision making algorithm
cm, the target maintenance component

```

ET=Get_Event_Trees()
FT=Get_Fault_Trees()
FOR et in ET
  IE=Get_init_events(et)
  FOR ie in IE
    Update_event_probability(ie)
  ENDFOR
ENDFOR
FOR ft in FT
  BE=Get_basic_events(ft)
  FOR be in BE
    Update_event_probability(be)
  ENDFOR
ENDFOR
RK=Get_system_risk(ET, FT)
Outputs:
RK, the system risk with possible consequences and their probabilities.

```

Figure 12. Algorithm for obtaining risk information.

Table 4 describes the referenced documents to which we applied the methodology described above for establishing the proposed ontology MuAMO. It is worth noting that some documents in the nuclear field are being referenced since the case study system used in this paper is a safety-critical system designed for NPPs. The methodology mentioned above can be applied to other industrial fields as well.

This section identifies the terms and relations that need to be integrated into the proposed ontology. The following subsections perform the integration in which the corresponding classes, properties and facets, are defined.

Step IV. Define classes, properties and facets

Several terms and concepts are collected by applying the information collection methodology (see the "Step II. Consider reusing existing ontologies" and "Step III. Enumerate important terms in the ontology" sections) to the industrial standards and guidelines, and their corresponding ontology classes are established. The "Ontology framework" section details the classes defined for the views in MuAMO based on the methodologies introduced above.

The properties and facets link the classes established in MuAMO and define constraints for using these classes. MuAMO reuses several properties defined by the existing ontologies, for example, fault ontology (Diao et al., 2022), ROMAIN (Karray et al., 2019), and information artifact ontology (Ceusters, 2012). Due to the page limitation, these properties are not presented in this paper.

Step V. Create instances

Creating instances for an ontology is applying the ontology to a practical problem. The "Case study" section details the instances of the concepts and properties introduced in this paper to demonstrate the proposed ontology's capability and complete the validation.

Table 5. Critical classes defined for the proposed framework (DAQ: data acquisition)

Class name	Parent class	Description
MuAMO:Component State	ROMAIN:State	The state concept is extended to component states.
MuAMO:Monitoring Technology	BFO:Realizable Entity	The technologies used by sensors for monitoring physical variables, e.g., thermocouples.
MuAMO:Signal	IAO:Material Information Bearer	The signals that are sampled by sensors.
MuAMO:Performance Level	IEO:Measurement Unit	The performance levels defined for evaluating the effectiveness of maintenance activities.
MuAMO:DAQ	AO:Sensor System	The DAQ equipment used for transferring analog signals transmitted from sensors to digital signals that can be recorded by computers.
MuAMO:Risk	BFO:Specifically Dependent Continuant	Risk is defined as the possible consequences of abnormal events with their probabilities.
MuAMO:Initiating Event	ROMAIN:Failure Event	Any event internal or external to the system that perturbs the steady-state operation of the system.
MuAMO:Accident Condition	BFO:Process	Conditions resulting from deleterious environmental effects or degraded equipment, components, or systems, occurring during events that are not expected in the course of system operation.
MuAMO:Accident Sequence	BFO:Process Aggregate	A combination of events beginning with an initiating event that challenges safety systems and resulting in an undesired consequence.
MuAMO:Event Tree	IAO:Directive Information Entity	A quantifiable, logical network that begins with an initiating event or condition and progresses through a series of branches that represent expected system or operator performance that either succeeds or fails and arrives at either a successful or failed end state.
MuAMO:Fault Tree	IAO:Directive Information Entity	A deductive logic diagram that depicts how a particular undesired event can occur as a logical combination of other undesired events.
MuAMO:Decision Making Algorithm	BFO:Algorithm	The decision-making algorithm.
MuAMO:Goal	IAO:Objective Specification	The goal required by the decision-making algorithm.
MuAMO:Reward	BFO:Specifically Dependent Continuant	The reward required by the decision-making algorithm.
MuAMO:Objective Function	AO:Artifact Function Specification	The function defining the objective of the decision-making algorithm.
MuAMO:Candidate Action	IAO>Action Specification	The possible actions that the decision-making algorithm can select.
MuAMO:Decision	IEO:Directive Information Content Entity	The outcomes of the decision-making algorithm.
MuAMO:Maintenance Equipment	MuAMO:Equipment	The equipment required for performing a maintenance activity.
MuAMO:Maintenance Team	BFO:Independent Continuant	The people required for performing a maintenance activity.

Ontology framework

Based on the aforementioned methodology, an ontology framework can be created to collect, represent, manage, and maintain the knowledge that will be used, synthesized, inferred, and verified during the maintenance optimization process. Figure 2 illustrates the architecture of the maintenance ontology framework with its related external tools or algorithms. In the figure, the proposed maintenance optimization framework consists of MuAMO, the maintenance optimization ontology, and several interfaces (labeled by IF). The concepts in the proposed ontology are categorized into six views, including the asset and component view (ACV), the function and failure view (FFV), the online monitoring view (OMV), the risk assessment view (RAV), the decision-making view (DMV), and the maintenance management view (MMV).

Knowledge and information will be shared between these views. One view may include several concepts that depend on the concepts included in other views. As an example of these dependencies, the maintenance costs included by various maintenance strategies and the system risk caused by system failures, recorded by the MMV and the RAV, respectively, are required by the reward function of algorithms included in the DMV to select the MEMA. In the proposed framework, the failures of system components and the maintenance activities are correlated through the concepts related to risk assessment. In detail, the failures of system components are quantified by their probabilities and further quantified by the risk caused by these failures. On the other hand, maintenance activities will impact the failure rates of the components under maintenance and further impact the risk after each maintenance activity.

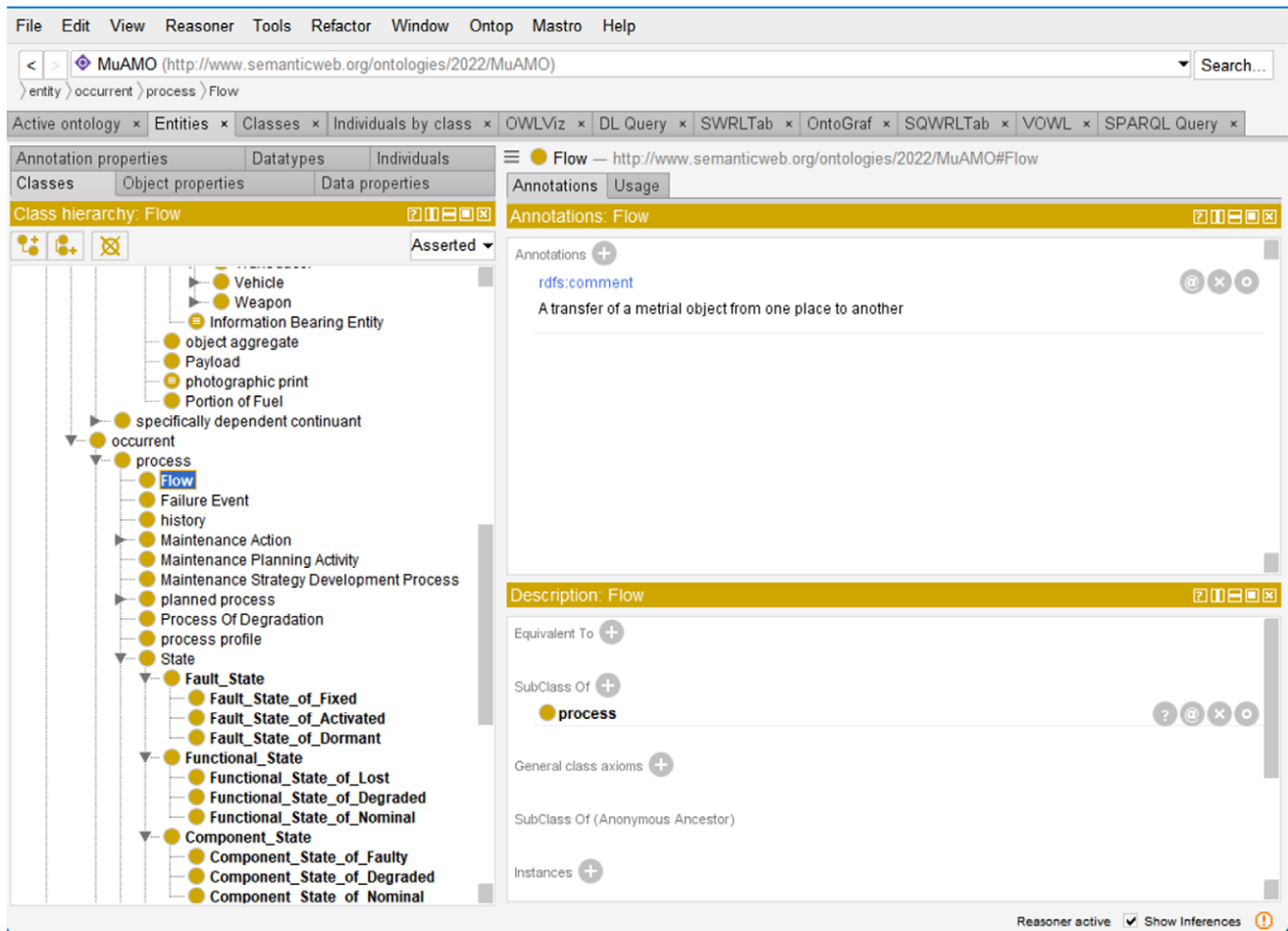


Figure 13. Screenshot of the ontology editor Protégé.

However, the proposed framework implemented the information flow through these concepts and recorded/updated the corresponding information in real time. The quantification involved in the information flow, such as the calculation of the failure rates of the components under maintenance, is performed by external tools. This mechanism provides flexibility in the integration of various tools related to maintenance optimization.

Each view in the framework possesses an application programming interface (API) that implements the communications between the proposed ontology and other external tools or algorithms. Figure 3 uses a portion of the critical concepts contained in each view to depict the dependencies between views. Besides the reused classes from other ontologies, Table 5 lists some of the critical classes defined in the proposed ontology MuAMO.

Implementation

Tools and interfaces

The proposed ontology is implemented using Protégé (Musen, 2015), an open-source ontology editor with built-in description logic reasoners. Figure 13 is a screenshot of the tool. Protégé supports the SWRL (Horrocks et al., 2004) for defining complex constraints. This research utilizes Protégé to reuse the existing ontologies, add and manage new concepts, and check for inconsistencies that may occur during the insertion of new concepts and properties.

The interfaces of the proposed framework are implemented using the Python programming language and the OWLReady2 library (Lamy, 2017). OWLReady2 provides packages for modifying and saving ontologies and performing reasoning. The library has been optimized for managing an ontology repository with a large number of classes and instances. OWLReady2 uses HermiT (Glimm et al., 2014) as a default reasoner. In addition, this library also includes an efficient API for connecting to external databases. Several technical solutions, such as pipelines, network sockets, or file systems, can be used with the APIs provided by the proposed framework to implement the interactions with external tools. For example, a reinforcement learning-based maintenance optimization algorithm (Zhao and Smidts, 2022) can be executed in a process running in parallel with the proposed framework. By using the APIs of the proposed framework, a well-formatted file can be generated, which includes all the parameters required by the optimization algorithm (e.g., the information on components and online monitoring data). The proposed framework can automatically prepare the required file and launch the algorithm. After the execution of the algorithm, the output result can be fed to the ontology framework through the APIs as well. For instance, the output file with the optimal maintenance activity can be parsed by the APIs and then recorded by the ontology repository. For the MMV, if the existing MMS is running remotely, i.e., running on another computer with network connections, an adapter can be implemented by using the APIs to feed information to the ontology

Table 6. Functions used by the algorithms

Function name	Description
<i>Get_Maintainable_Items</i>	Get the list with all the components that need to be maintained in the target system.
<i>Get_applicable_Maintenance_activity</i>	Get the maintenance activities that can be applied to the target system.
<i>Get_applicable_Optimization_Algorithms</i>	Get the maintenance optimization algorithms that can be applied to the target system.
<i>If_requirefailure_information</i>	Identify if the current algorithm requires system failure information.
<i>If_require_risk_information</i>	Identify if the current algorithm requires system risk information.
<i>Get_Optimal_Activity</i>	Run the interface with the decision-making algorithm to select the optimal maintenance activity.
<i>Update_Maintenane_strategy</i>	Run the interface with the maintenance management system to update the current maintenance strategy.
<i>Get_Functions_relatedto_component</i>	Get the functions related to the current component.
<i>Get_required_measure</i>	Get the measures required by the current algorithm.
<i>Get_failure_modes</i>	Get the possible failure modes of the current function.
<i>Get_measure_value</i>	Get the sampled values of the required measures.
<i>Get_Event_Trees</i>	Generate the structure of the event tree for the target system. The generation process can be implemented by iterating the subsequent events belonging to the event tree, starting with the initial events.
<i>Get_Fault_Trees</i>	Generate the structure of the fault tree for the target system. The generation process can be implemented by iterating the cause events belonging to the fault tree, starting with the top events.
<i>Get_init_events</i>	Get the initial events of an event tree.
<i>Update_event_probability</i>	Update the probability of an event.
<i>Get_basic_events</i>	Get the basic events of a fault tree.
<i>Get_system_risk</i>	Run the interface with external probabilistic risk assessment tools to obtain the system risk information.

framework. Meanwhile, the adapter can use the interfaces provided by the MMS to obtain the required information. In practice, any algorithms or tools providing automation interfaces can be integrated with the proposed framework.

It is worth noting that although the current version of the proposed framework provides the ability to interact with external databases or tools, some algorithms, such as the algorithms for fault prognosis, risk assessment, and decision-making, are performed manually for the case study system.

Information processing algorithms

This section introduced the information processing algorithms used by the proposed framework to collect information for various views defined by MuAMO and provide maintenance recommendations to system operators. Figure 10 displays the pseudocode of the algorithm. This algorithm will be executed periodically to change the maintenance strategies dynamically. The algorithm can be triggered using a fixed period (e.g., every second or every minute) depending on the type of the component being maintained or by specific events (e.g., new online monitoring data are received in interrupt-based systems). The functions used by the algorithms are introduced in Table 6. These functions are implemented by directly inquiring about the ontology repository or calling the interfaces of the views in MuAMO.

The maintenance optimization process algorithm also requires two additional functions: “*Get_failure_information*” for obtaining the failure information for the component being maintained and “*Get_risk_information*” for obtaining the system risk if the component being maintained is failed. The algorithms of these two functions are detailed in Figures 11 and 12, respectively.

The algorithm in Figure 11 uses *list* operations. For example, function “*Create_empty_list*” can create an empty list and function “*Append_list*” can append a set of variables into the current list.

Information queries

Based on the ontology repository and the algorithms introduced above, the real-time data related to the maintenance target system can be collected and recorded by the ontology framework. However, the collected data cannot directly be used by maintenance optimization models or be referenced by the analyst for decision-making (i.e., be used to answer the competency questions proposed in the “Step I. Determine the domain and scope of the ontology” section). Data query for select information of interest is required. As a de facto standard, the ontology provides SQWRL, an SWRL-based information query language (Horrocks et al., 2004) for eliciting data from ontology repositories. SQWRL provides basic arithmetic and Boolean operators which can be used to acquire data based on logic expressions. Generally, an ontology query engine, for example, HerMiT (Glimm et al., 2014), can parse the SQWRL expressions and can derive the associated results.

This section builds the queries for answering the competency questions. It uses the format of SWRL for expressing the established queries. Due to the complexity of the competency questions, some of the questions are answered progressively, that is, one competency question is divided into several subquestions. After answering all these subquestions, the competency question can be finally answered. The following text details the queries created for each competency question mentioned in the “Step I. Determine the domain and scope of the ontology” section.

QA) To decrease the risk of the target system, what is the most efficient applicable maintenance activity?

This question is usually raised by maintenance analysts to prioritize the maintenance activities. However, since this question requires synthetic information from the different views of the proposed

Table 7. Information queries provided for answering competency questions

Competency questions	Related information queries
QA) To decrease the risk of the target system, what is the most efficient applicable maintenance activity?	1) Which components/subsystems need to be maintained in the target system? $component_under_analysis(?x)$ 2) What are the available maintenance activities for the target component? $applicable_maintenance_activity(?x,?m)$ 3) What is each failure mode's cost (risk)? $has_risk(?x,?f,?r)$ 4) What is the most efficient maintenance activity? $most_efficient_maintenance_activity(?x, ?m, ?a)$
QB) Considering maintenance costs and system risks, what is the most critical part of the target system/component?	$most_critical_component_part(?x, ?p)$
QC) What is the best improvement to system safety in terms of all feasible maintenance activities for a given budget?	1) What are the feasible maintenance activities for a given budget? $feasible_maintenance_activity(?x, ?m)$ 2) What is the maximum improvement of system safety that can be obtained at this time? $most_efficient_maintenance_activity(?x, ?m, ?s)$

framework, the original question is divided into the following subquestions.

QA.1) Which components/subsystems need to be maintained in the target system?

The information required by this question can be acquired by identifying the components belonging to the feedwater system. The following query can be used.

$$Component(?x) Maintenable_Item(?x) has_component(SYSTEM, ?x) \rightarrow component_under_analysis(?x)$$

In the expression, the variable "SYSTEM" will be replaced by the name of the system considered for maintenance. The variable x will contain the components/subsystems that need to be maintained.

QA.2) What are the available maintenance activities for the target component?

This question can be answered by searching the maintenance activities whose target component is one of the target system's components. Also, the input statement contains the expressions related to equipment and people since some of the maintenance activities require specific equipment or people to conduct them. The following query can be used.

$$Maintenance_Activity(?m) has_target_component(?m, ?x) Component_under_analysis(?x) Maintenance_Equipment(?e) Require_equipment(?m, ?e) sqwrl : count(has_instance(?e)) > 1 Maintenance_Team(?t) require_staff(?m, ?t) Sqwrl : count(has_available_worker(?t)) > 1 \rightarrow applicable_maintenance_activity(?x, ?m)$$

It is worth noting that this query requires the answer of the previous question. We can see that the condition $Component_under_analysis(?x)$ is one of the premises of this expression.

QA.3) What is each failure mode's cost (risk)?

The cost associated with the occurrence of particular failure modes can be acquired from the connection relationship between the failure view and the risk view.

$$component_under_analysis(?x) has_failure_mode(?x, ?f) has_failure_rate(?f, ?q) cause_loss(?f, ?l) swrlb : multiply(?r, ?q, ?l) \rightarrow has_risk(?x, ?f, ?r)$$

In the expression, the risk is calculated by multiplying the failure rate and the financial loss (dollars/year). The variable f will be the failure mode, and the variable r will be the corresponding risk associated with such a failure.

QA.4) What is the MEMA?

Based on the answers from the previous subquestions, we can build the query for answering the competency question A. The MEMA is the one that decreases the risk of the target system in its current state the most. Assume that in the current state, the failure rates of the seal and the leaf spring are both one failure/year and that inspecting a component does not impact its failure rate, and replacing a component restores its failure rate to its initial value. Assume that the MEMA can be calculated using the following equation:

$$MEMA = argmax \left(\frac{risk\ before\ maintenance - risk\ after\ maintenance}{maintenance\ cost} \right)$$

Then we can create the query expression below.

$$component_under_analysis(?x) applicable_maintenance_activity(?x, ?m) has_failure_mode(?x, ?f) cause_loss(?f, ?l) has_current_failure_rate(?f, ?c) has_risk(?x, ?f, ?r) has_maintenance_cost(?m, ?v) swrlb : multiple(?cl, ?c, ?l) swrlb : subtract(?s, ?cl, ?r) swrlb : divide(?a, ?s, ?v) sqwrl : max(?a) \rightarrow most_efficient_maintenance_activity(?x, ?m, ?a)$$

The variable m will be the MEMA the framework can identify. And, the variable x will be the component that is the target component of the maintenance activity m .

QB) Considering maintenance costs and system risks, what is the most critical part of the target system/component?

The most critical part of a component can be defined as contributing the most risk to the system or leading to the most costly

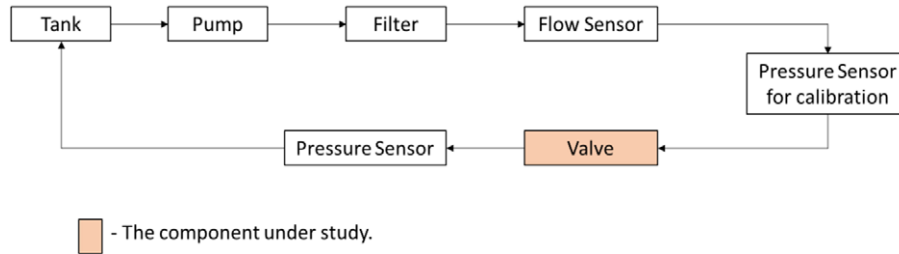


Figure 14. Hardware of the experimental system.

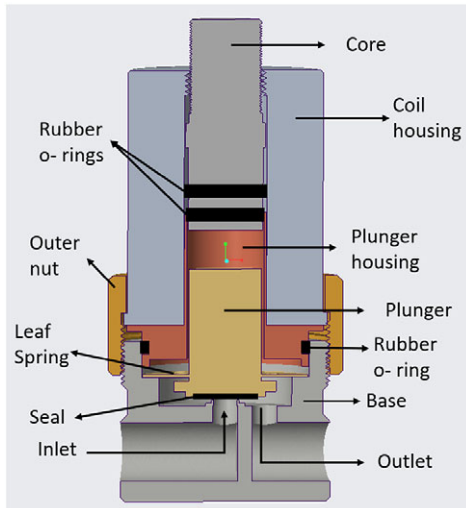


Figure 15. Solenoid operated valve cross-sectional view and parts.

maintenance. In practice, the proposed ontology evaluates the criticality of a component by the sum of the risks and maintenance costs related to the component. By reusing the queries created for competency questions QA.1 and QA.3, the statement necessary for finding the most critical part of a component can be built as below.

```

component_under_analysis(?x) has_composite(?x,?p)
  applicable_maintenance_activity(?x,?m)
  has_target_component(?m,?p) has_failure
  _mode(?x,?f) has_maintenance_cost(?m,?v)
  has_risk(?x,?f,?r) swrlb : add(?a,?r,?v) sqwrl : max(?a) →
  most_critical_component_part(?x,?p)
    
```

In the output of the query above, the variable *x* will be the most critical component of the target system and the variable *p* will be the most critical part of component *x*.

QC) What is the best improvement to system safety in terms of all feasible maintenance activities for a given budget?

Generally, the improvement to system safety can be defined as decreasing system risk. Based on that definition, this question can be divided into the following subquestions:

QC.1) What are the feasible maintenance activities for a given budget?

The feasible maintenance activities can be obtained by checking the candidate activities in the MMV using the following query.

```

component_under_analysis(?x) applicable_
  maintenance_activity(?x,?m)
  has_maintenance_cost(?m,?v)
  swrl : lessThanOrEqual(?v,BUDGET) →
  feasible_maintenance_activity(?x,?m)
    
```

After the execution of the query, the variable *m* will be the feasible maintenance activity. In the expression, the variable *BUDGET* will be replaced by the actual budget.

QC.2) What is the maximum improvement of system safety that can be obtained at this time?

By reusing the queries of competency questions QA.4 and QC.1, the best system safety improvements can be calculated using the following expression.

```

component_under_analysis(?x) applicable_
  maintenance_activity(?x,?m)
  has_failure_mode(?x,?f) has_risk(?x,?f,?r)
  has_current_failure_rate(?f,?c)
  cause_loss(?f,?l) swrlb : multiply(?cl,?c,?l) swrlb :
  subtract(?s,?cl,?r) sqwrl : max(?s)
  → most_efficient_maintenance_activity(?x,?m,?s)
    
```

The value of variable *s* will be the maximum improvement to system safety based on the information recorded by the ontology framework. Table 7 summarizes the information queries provided for answering competency questions.

Case study

In this paper, the proposed framework is verified using solenoid operated valves (SOVs) in a hardware-in-the-loop (HIL) experimental setup. SOVs are widely used in NPPs in pneumatic circuits that are utilized to control and operate important actuators, such as those used to move and position control rods. This section details the system structure and the components, the considered functions and the corresponding failure modes, the monitoring data and the available maintenance activities, and other required information for decision-making, generated from the proposed ontology and framework.

Figure 14 depicts the HIL experimental system. The experimental setup emulates the feedwater system on the secondary side of a pressurized water reactor. The target component, a solenoid-operated proportional flow control valve, acts as the feedwater flow

Table 8. Failure modes of the solenoid operated valve

No.	Comp	Failure modes	Initial failure rate (failures/year)	Description
1	Seal	Wearing out	1E-2	The seal can wear out due to aging or erosion caused by repeated exposure to the fluid.
2	Seal	Water absorption	1E-2	The seal is composed of rubber materials, which can absorb water and expand. Such bloated seals cover the inlet port at the center of the base of the valve even when the plunger moves upward for a certain range and thus interfere with the valve function by limiting the range of flowrate.
3	Seal	Detachment	1E-2	The seal is connected to the base of the plunger through adhesives, which may weaken over a while due to exposure to the fluid medium. A detached seal interferes with the flow through the valve.
4	Leaf spring	Decrease of stiffness	1E-3	The leaf spring undergoes compression loading every time the plunger moves to the top-most position, i.e., when the valve is fully opened. Such continuous loading can permanently deform the spring, i.e., can cause spring creep and reduce the spring stiffness, affecting the valve's performance. As a result, the valve opens completely at lower input power, and the flowrate through the valve cannot be controlled.
5	Leaf spring	Loss of spring elasticity	1E-3	In some cases, lime scaling can render the spring leaves inflexible, thereby resulting in a complete loss of functionality of the spring.

Table 9. Measurements for the solenoid operated valve

No.	Measurements	Description
1	The input current to the valve	The control signal of the target component.
2	The volume flowrate through the valve	The flow measurements are taken to infer the health of the spring.
3	The plunger leaf spring deformation measurements	The spring deformation is measured using images.
4	The weight of the plunger with the seal	The measure is to quantify the water absorption level and material loss in the seal.

control valve. The SOVs in the system are flow control valves and are not pressure control valves.

Components and structures

The major components of the solenoid valve used in our experimental setup are presented in this section. Figure 15 displays the structure and the components of the SOV under study. Table 8 presents the typical failure modes of solenoid valves considered in the case study. In the case study, the following assumptions are made to simplify the calculations related to risk assessment:

- 1) Only one system failure is considered, which will cause a financial loss equivalent to \$2E3.
- 2) Any failure mode of the valve will cause the system failure with a probability 100%. This information is modeled by the proposed ontological framework.

Table 9 summarizes the measurements sampled from the target system and the SOV. The data are recorded by the classes defined in the monitoring view.

Maintenance activities

Table 10 summarizes the maintenance activities defined by the maintenance process view. It shows the assumed total cost and the work duration for each maintenance activity. The decision-making algorithm will reference this information to select the optimal maintenance activity.

Results and achievements

By executing the information queries introduced in the “Information queries” section, we can acquire the outputs from the ontology repository to answer the competency questions raised in the “Step I. Determine the domain and scope of the ontology” section. Table 11 displays the ontology repository's outputs which have been verified by expert judgment. In the results, the components' names with the number 1 denote that they are instances of components, not the component classes.

The answer to QA.1 includes all the components that need to be maintained in the target system. The answer to QA.2 includes the available maintenance activities for the target component “Valve_1.” According to the answer to QA.4, the MEMA is “replace the seal,” which decreases the risk by 9.9 (dollar loss per year/dollars of maintenance cost). Also, the most critical part of the solenoid valve is the Leafspring, shown by the answer to QB.

Assume that the failure of any part of the valve will fail the valve. According to the feasible maintenance activities identified by QC.1, the best system safety improvement can only be achieved by the activity “replace the whole valve,” which reduces the risk by 1980.

Discussion

This paper proposes an ontology framework for supporting the development of a maintenance optimization system. The proposed framework can facilitate the tasks that need to be performed during the development of every maintenance optimization system. These tasks are discussed below.

Table 10. Maintenance activities for the solenoid operated valve

No.	Maintenance activities	Cost (worker salary + component cost)	Work duration
1	Inspect the whole valve	\$200	0.5 hr
2	Inspect coil test only	\$150	0.16 hr
3	Spring compression test only	\$120	0.25 hr
4	Weigh the plunger and seal only	\$100	0.08 hr
5	Replace the spring	\$150 + \$500	0.5 hr
6	Replace the seal	\$150 + \$50	0.75 hr
7	Replace the whole valve	\$100 + \$500	0.5 hr

Table 11. Results of the case study system for the competency questions

Competency questions	Outputs of the framework
QA.1) Which components/subsystems need to be maintained in the target system?	"Tank_1," "Pump_1," "Filter_1," "FlowSensor_1," "Valve_1," "PressureSensor_1"
QA.2) What are the available maintenance activities for the target component?	"Valve_1, inspect_the_whole_valve," "Valve_1, inspect_coil_test_only," "Valve_1, spring_compression_test_only," "Valve_1, weigh_the_plunger_and_seal_only," "Valve_1, replace_the_seal," "Valve_1, replace_the_spring," "Valve_1, replace_the_whole_valve"
QA.3) What is each failure mode's cost (risk)?	"Valve_1, seal_wearing_out, 20," "Valve_1, seal_water_absorption, 20," "Valve_1, seal_detachment, 20," "Valve_1, leafspring_decrease_of_stiffness, 2," "Valve_1, leafspring_loss_of_spring_elasticity, 2"
QA.4) What is the most efficient maintenance activity?	"Valve_1, replace_the_seal, 9.9"
QB) Considering maintenance costs and system risks, what is the most critical part of the target system/component?	"Valve_1, Leafspring"
QC.1) What are the feasible maintenance activities for a given budget?	"Valve_1, inspect_the_whole_valve," "Valve_1, inspect_coil_test_only," "Valve_1, spring_compression_test_only," "Valve_1, weigh_the_plunger_and_seal_only," "Valve_1, replace_the_spring"
QC.2) What is the maximum improvement of system safety?	"Valve_1, replace_the_whole_valve, 1980"

- Using the framework for optimization model selection. A maintenance optimization model usually contains one or more optimization algorithms that require various inputs. For example, a maintenance optimization model for a feedwater system may require the flow rates through the pipelines in the system to be sampled in real time by different sensors, the predicted residual lifetime of the regulating valves from a fault diagnosis tool, as well as the risk of system failures from a risk assessment tool. One can use the proposed framework to quickly identify whether the requirements of the optimization algorithm can be satisfied. The identification can be achieved by using ontology query languages, for example, SPARQL (Pérez *et al.*, 2006), or by defining the prerequisite rules using SWRL (Horrocks *et al.*, 2004). The optimization algorithms with unsatisfiable requirements can be filtered out.
- Using the proposed framework to implement a maintenance optimization model will be more straightforward than the traditional development process since the framework provides a standardized API that can supply the required information to the optimization model. As a result, the developers of the optimization model do not need to implement the functions related to data sampling and formatting, and the time for model implementation can be reduced.
- As a data provider for the maintenance optimization model, the proposed framework can provide online data for the optimization model. Also, it can be configured to provide test data to the

model for model evaluation. For example, the OMV can be configured to connect to a database that records a series of benchmark data. Meanwhile, if two optimization models are implemented based on the API provided by the framework, the framework can launch both algorithms using the same monitoring data, and their outcomes can be compared and evaluated.

Although the proposed framework itself cannot perform critical tasks required by system maintenance, such as system diagnosis and risk assessment, it provides various APIs that can be easily reused by external tools to provide the results of such tasks to the ontology repository. For example, by using the APIs provided by the component view, the system structure information represented by a system modeling diagram, such as piping and instruments diagrams (P&IDs), can be modeled and reused by other views. The interface developed for this view can be connected to an image identification tool to update the system components and structural information automatically (Gao *et al.*, 2020). The monitoring view can search and select data sampled from real systems. External diagnosis or health management tools can use these data to perform diagnosis and prognosis. The RAV can interact with external risk analysis tools to acquire risk evaluation results. The view can provide information about the event trees, fault trees, and the distributions and probabilities for various events.

In the current version of the proposed ontology framework, the classes of failure modes and applicable maintenance activities are

manually defined by domain experts. The classes of failure modes and the classes of maintenance activities are linked by the types of components, that is, specific failure modes are linked to specific types of components, and the same is true for the maintenance activities. In this case, the validity of failure modes and maintenance activities is ensured by expert knowledge. Domain experts can add failure modes for specific target systems to the ontology repository or identify the failure modes that can be reused. In the future, the conditions and constraints for validating failure modes and maintenance activities can be predefined using the ontology framework so that the ontology reasoner can automatically identify their applicability to various target systems.

In addition, it is worth noting that this paper describes a method for reusing existing ontologies and minimizing the modifications to the existing ontologies. However, conflicts between the classes in two different ontologies may exist. The existence of conflicts between two or more ontologies means that classes or properties given the same name in these ontologies may have different interpretations or that mistakes may exist in these ontologies. For example, the class “orange” can be a subclass of “fruit” in the fruit ontology but can also be a subclass of “color” in the color ontology. The proposed ontology integration method cannot prevent such semantic heterogeneity between existing ontologies. However, some ontology inference engines, for example, HerMiT (Glimm et al., 2014), can detect these types of conflicts automatically.

Conclusion

This paper establishes a maintenance optimization ontology, the MuAMO, by using the methodology introduced for ontology development. To integrate ontologies incompatible with the same meta-ontology, a heuristic method to reuse the concepts and properties defined by existing ontologies was developed. By using the ontology MuAMO as a data repository for aggregating information and a series of interfaces interacting with these data sources, an ontology framework has been created as a data repository for aggregating information. The proposed framework can provide various required information for different maintenance optimization algorithms to evaluate the maintenance activities and select the optimal one. In this paper, a solenoid valve deployed in a feedwater system as the target component has been used to verify the proposed methodology and validate the developed framework. As an example use of the proposed framework, the results from the case study system demonstrated the feasibility and effectiveness of the proposed framework. Based on the current version of the ontology framework, several tasks can be undertaken to improve the proposed framework.

In the future, we will enrich the interfaces of each view in MuAMO and connect the framework to more third-party tools. Specifically, the ACV will be connected to an image processing tool (Gao et al., 2020), which can identify system components in P&IDs and automatically model system structures. The FFV and the OMV will connect to a fault diagnosis algorithm (Li et al., 2022) for component degradation prediction based on online monitoring data. Meanwhile, the RAV will interact with a probabilistic risk assessment tool, for example, SAPHIRE (Smith et al., 2016), for system risk assessment, and the DMV will be connected to a decision-making algorithm (Zhao and Smidts, 2022). Also, the MMV will be enhanced to communicate with MMSs developed for specific target systems (e.g., the ones in NPPs).

Abbreviations

ACV	asset and component view
AO	artifact ontology
API	application programming interface
BFO	basic formal ontology
CBM	condition-based maintenance
DAQ	data acquisition
DMV	decision-making view
DOE	Department of Energy
FFV	function and failure view
HIL	hardware-in-the-loop
IAO	information artifact ontology
IEO	information entity ontology
IMAMO	industrial maintenance management ontology
MEMA	most efficient maintenance activity
MMS	maintenance management system
MMV	maintenance management view
MuAMO	multiple aspects maintenance ontology
NPP	nuclear power plant
OMV	online monitoring view
P&ID	pipng and instruments diagram
PRA	probabilistic risk assessment
RAV	risk assessment view
ROMAIN	A domain-specific ontology for the maintenance management
SOV	solenoid operated valve
SSC	system, structure, and component
SWRL	Semantic Web Rule Language

Acknowledgments. This research is being performed using funding received from the DOE Office of Nuclear Energy’s Nuclear Energy University Program, United States of America.

Competing interest. The authors declare none.

References

- Ajit S, Sleeman D, Fowler DW and Knott D (2008) Constraint capture and maintenance in engineering design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM* 22(4), 325–343. <https://doi.org/10.1017/S089006040800022X>.
- Alaswad S and Xiang Y (2017) A review on condition-based maintenance optimization models for stochastically deteriorating system. *Reliability Engineering and System Safety* 157, 54–63. <https://doi.org/10.1016/j.res.2016.08.009>.
- Alkahtani M, Choudhary A, De A and Harding JA (2019) A decision support system based on ontology and data mining to improve design using warranty data. *Computers and Industrial Engineering* 128, 1027–1039. <https://doi.org/10.1016/j.cie.2018.04.033>.
- Anquetil N, De Oliveira KM and Dias MGB (2006) Software maintenance ontology. In *Ontologies for Software Engineering and Software Technology*. Berlin–Heidelberg: Springer, pp. 153–173. https://doi.org/10.1007/3-540-34518-3_5.
- Apeland S and Aven T (2000) Risk based maintenance optimization: Foundational issues. *Reliability Engineering and System Safety* 67(3), 285–292. [https://doi.org/10.1016/S0951-8320\(99\)00068-X](https://doi.org/10.1016/S0951-8320(99)00068-X).
- Arp R, Smith B and Spear AD (2016) *Building Ontologies with Basic Formal Ontology*. Cambridge, MA: MIT Press. <https://doi.org/10.7551/mitpress/9780262527811.001.0001>.
- Campos J (2007) An ontology for asset management. *IFAC Proceedings Volumes (IFAC-PapersOnline)* 40(19), 36–41. <https://doi.org/10.3182/20071002-mx-4-3906.00007>.

- Canito A, Corchado J and Marreiros G (2021) Bridging the gap between domain ontologies for predictive maintenance with machine learning. In *Trends and Applications in Information Systems and Technologies*. Advances in Intelligent Systems and Computing, 1366. Cham: Springer, pp. 533–543. https://doi.org/10.1007/978-3-030-72651-5_51.
- Ceusters W (2012) An information artifact ontology perspective on data collections and associated representational artifacts. *Studies in Health Technology and Informatics* 180, 68–72. <https://doi.org/10.3233/978-1-61499-101-4-68>.
- Chantrapornchai C and Choksuchat C (2016) Ontology construction and application in practice case study of health tourism in Thailand. *SpringerPlus* 5(1), 2106. <https://doi.org/10.1186/s40064-016-3747-3>.
- de Jonge B and Scarf PA (2020) A review on maintenance optimization. *European Journal of Operational Research* 285(3), 805–824. <https://doi.org/10.1016/j.ejor.2019.09.047>.
- Diao X, Pietrykowski M, Huang F, Mutha C and Smidts C (2022) An ontology-based fault generation and fault propagation analysis approach for safety critical computer systems at the design stage. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM* 36, E1. <https://doi.org/10.1017/S0890060421000342>.
- Dimitrova V, Mehmood MO, Thakker D, Sage-Vallier B, Valdes J and Cohn AG (2020) An ontological approach for pathology assessment and diagnosis of tunnels. *Engineering Applications of Artificial Intelligence* 90, 103450. <https://doi.org/10.1016/j.engappai.2019.103450>.
- Doe G and Ramsey CB (2011) Nuclear facility maintenance management program guide for use with DOE O 433.1B. <https://www.directives.doe.gov/directives-documents/400-series/0433.EGuide-1-1a>.
- Ebrahimipour V and 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. <https://doi.org/10.1109/TSMC.2014.2383361>.
- El-Diraby TE (2013) Domain ontology for construction knowledge. *Journal of Construction Engineering and Management* 139(7), 768–784. [https://doi.org/10.1061/\(asce\)co.1943-7862.0000646](https://doi.org/10.1061/(asce)co.1943-7862.0000646).
- Elhaddad R, Chilamkurti N and Torabi T (2013) An ontology-based framework for process monitoring and maintenance in petroleum plant. *Journal of Loss Prevention in the Process Industries* 26(1), 104–116. <https://doi.org/10.1016/j.jlp.2012.10.001>.
- Emmanouilidis C, Gregori M and Al-Shdifat A (2020) Context ontology development for connected maintenance services. *IFAC-PapersOnLine* 53, 10923–10928. <https://doi.org/10.1016/j.ifacol.2020.12.2833>.
- Gangemi A, Guarino N, Masolo C and Oltramari A (2003) Sweetening WORDNET with DOLCE. *AI Magazine* 24(3), 13. <https://doi.org/10.1007/3-540-45810-7>.
- Gao W, Zhao Y and Smidts C (2020) Component detection in piping and instrumentation diagrams of nuclear power plants based on neural networks. *Progress in Nuclear Energy* 128. <https://doi.org/10.1016/j.pnucene.2020.103491>.
- Glimm B, Horrocks I, Motik B, Stoilos G and Wang Z (2014) Hermit: An OWL 2 reasoner. *Journal of Automated Reasoning* 53(3), 245–269.
- Gruber TR (1993) A translation approach to portable ontology specifications. *Knowledge Acquisition* 5(2), 199–220. <https://doi.org/10.1006/knac.1993.1008>.
- Gu YJ, Dong XF and Yang K (2009) Study on maintenance method intelligent decision support system used for power plant equipment. In *Asia-Pacific Power and Energy Engineering Conference, APPEEC, September*, pp. 96–100. <https://doi.org/10.1109/APPEEC.2009.4918824>.
- Guo C, Wang W, Guo B and Si X (2013) A maintenance optimization model for mission-oriented systems based on Wiener degradation. *Reliability Engineering and System Safety* 111, 183–194. <https://doi.org/10.1016/j.res.2012.10.015>.
- Harrison R and Chan CW (2009) A dynamic knowledge modeler. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM* 23(1), 53–69. <https://doi.org/10.1017/S0890060409000109>.
- Horrocks I, Patel-schneider PF, Boley H, Tabet S, Grosz B and Dean M (2004) SWRL: A Semantic Web Rule Language combining OWL and RuleML. In *W3C Member submission 21* (Issue May 2004, pp. 1–20). Available at <http://www.w3.org/Submission/SWRL/>.
- INTERNATIONAL ATOMIC ENERGY AGENCY (2018) *Maintenance Optimization Programme for Nuclear Power Plants.*, IAEA Nuclear Energy Series No. NP-T-3.8, IAEA, Vienna.
- Karray MH, Ameri F, Hodkiewicz M and Louge T (2019) ROMAIN: Towards a BFO compliant reference ontology for industrial maintenance. *Applied Ontology* 14(2), 155–177. <https://doi.org/10.3233/AO-190208>.
- Karray MH, Chebel-Morello B and Zerhouni N (2012) A formal ontology for industrial maintenance. *Applied Ontology* 7(3), 233–267. <https://doi.org/10.3233/AO-2012-0112>.
- Kethavarapu UPK and Saraswathi S (2016) Concept based dynamic ontology creation for job recommendation system. *Procedia Computer Science* 85, 915–921. <https://doi.org/10.1016/j.procs.2016.05.282>.
- Kim KY, Chin S, Kwon O and Ellis RD (2009) Ontology-based modeling and integration of morphological characteristics of assembly joints for network-based collaborative assembly design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM* 23(1), 71–88. <https://doi.org/10.1017/S0890060409000110>.
- Kobbacy KAH (2004) On the evolution of an intelligent maintenance optimization system. *Journal of the Operational Research Society* 55(2), 139–146. <https://doi.org/10.1057/palgrave.jors.2601696>.
- Kornysheva E and Deneckère R (2010) Decision-making ontology for information system engineering. In *Conceptual Modeling – ER 2010*. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 6412. Berlin–Heidelberg: Springer, pp. 104–117. https://doi.org/10.1007/978-3-642-16373-9_8.
- Lamy J (2017) Owlready: Ontology-oriented programming in python with automatic classification and high-level constructs for biomedical ontologies. *Artificial Intelligence in Medicine* 80, 11–28.
- Lee J, Ni J, Singh J, Jiang B, Azamfar M and Feng J (2020) Intelligent maintenance systems and predictive manufacturing. *Journal of Manufacturing Science and Engineering, Transactions of the ASME* 142(11), 110805. <https://doi.org/10.1115/1.4047856>.
- López MF, Gómez-Pérez A, Sierra JP and Sierra AP (1999) Building a chemical ontology using Methontology and the ontology design environment. *IEEE Intelligent Systems and Their Applications* 14(1), 37–46. <https://doi.org/10.1109/5254.747904>.
- Mohammad MA, Kaloskamps I, Hicks Y and Setchi R (2015) Ontology-based framework for risk assessment in road scenes using videos. *Procedia Computer Science* 60(1), 1532–1541. <https://doi.org/10.1016/j.procs.2015.08.300>.
- Musen MA (2015) The protégé project: A look back and a look forward. *AI Matters* 1(4), 4–12.
- Niles I and Pease A (2001) Towards a standard upper ontology. In *The 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001)*, pp. 2–9. <https://doi.org/10.1145/505168.505170>.
- Noy N and McGuinness DL (2017) Ontology development 101: A guide to creating your first ontology. *Sustainability (Switzerland)* 9(12), 1–25. <https://doi.org/10.3390/su9122317>.
- Oltramari A, Henshel D, Cains M and Hoffman B (2015) Towards a human factors ontology for cyber security. In *CEUR Workshop Proceedings*, Vol. 1523, pp. 26–33.
- Pérez J, Arenas M and Gutierrez C (2006) Semantics and complexity of SPARQL. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 4273 LNCS, 30–43. https://doi.org/10.1007/11926078_3/COVER.
- Qu C, Liu F, Yu H, Yuan R and Wang A (2016) User oriented semi-automatic method of constructing domain ontology. *Communications in Computer and Information Science* 575, 553–561. https://doi.org/10.1007/978-981-10-0356-1_58.
- Sharma A, Yadava GS and Deshmukh SG (2011) A literature review and future perspectives on maintenance optimization. *Journal of Quality in Maintenance Engineering* 17(1), 5–25. <https://doi.org/10.1108/13552511111116222>.
- Smith B (1998) Basic concepts of formal ontology. In *Formal Ontology in Information Systems*. Amsterdam: IOS Press, pp. 19–28.
- Smith C, Knudsen J, Vedros K, Michael M, Kvarfordt K and Wood T (2016) SAPHIRE 8 Basics An Introduction to Probabilistic Risk Assessment via the Systems Analysis Program for Hands-On Integrated Reliability Evaluations (SAPHIRE) Software P-201. <https://doi.org/10.2172/1467591>.

Wang L, Chu J, Mao W and Fu Y (2006) Advanced maintenance strategy for power plants – introducing intelligent maintenance system. *Proceedings of the World Congress on Intelligent Control and Automation (WCICA)* **2**, 7444–7448. <https://doi.org/10.1109/WCICA.2006.1713411>.

Wu C, Wu P, Wang J, Jiang R, Chen M and Wang X (2021) Ontological knowledge base for concrete bridge rehabilitation project management. *Automation in Construction* **121**, 103428. <https://doi.org/10.1016/j.aut-con.2020.103428>.

Xiaoxu Diao is a Research Associate in the Department of Mechanical and Aerospace Engineering at The Ohio State University, Columbus, OH, USA. His research interests are fault diagnosis and prognosis, software reliability and safety, and cybersecurity assessment for safety-critical systems. His research has been published in 11 journal papers, 3 book chapters, and 10 conference papers. He served as the technical session chair for the International Topical meeting on Probabilistic Safety Assessment, 2021 and served on the organizing committee for the Big Data for Nuclear Power Plants Workshop, 2018–2020.

Yunfei Zhao is a Research Associate in the Department of Mechanical and Aerospace Engineering at The Ohio State University, Columbus, OH, USA. His research interests include human reliability analysis, probabilistic risk assessment, cybersecurity risk assessment, and cyberattack detection and response. His research has been published in 22 journal papers, 1 book chapter, and 14 conference papers. He is the Secretary of the Program Committee and the Chair of the Communications Committee of ANS's NISD, served on the TPC for PSA2021, served as session chairs for RAMS 2020 and NPIC& HMIT 2019, and has been an active reviewer for more than 10 journals.

Pavan Kumar Vaddi is a Ph.D. candidate in the Reliability and Risk Laboratory in the Department of Mechanical and Aerospace Engineering at The Ohio State University, Columbus, OH, USA. He received his B.Tech. degree in Mechanical Engineering from IIT Madras, Chennai, India. His research interests are cyberattack detection and response, cybersecurity risk assessment, probabilistic risk assessment, and fault diagnosis and prognosis. His research has been published in two journal papers, one book chapter, and six conference papers.

Yuan XX, Higo E and Pandey MD (2021) Estimation of the value of an inspection and maintenance program: A Bayesian gamma process model. *Reliability Engineering and System Safety* **216**, 107912. <https://doi.org/10.1016/j.res.2021.107912>.

Zhao Y and Smidts C (2022) Reinforcement learning for adaptive maintenance policy optimization under imperfect knowledge of the system degradation model and partial observability of system states. *Reliability Engineering and System Safety* **224**, 108541. <https://doi.org/10.1016/j.res.2022.108541>.

Michael Pietrykowski is a Postdoctoral Researcher at the Los Alamos National Laboratory, Santa Fe, NM, USA. He received his Ph.D. degree from the Department of Mechanical and Aerospace Engineering at The Ohio State University, Columbus, OH, USA. He received his B.S. degree in Electrical and Computer Engineering from The Ohio State University, specializing in computers and solid-state devices, and is an NRC Graduate Fellowship and DOE NEUP Fellowship recipient. His research is focused on investigating digital instrumentation and controls systems in nuclear power plants using hardware-in-the-loop.

Marat Khafizov is an Associate Professor in the Department of Mechanical and Aerospace Engineering at The Ohio State University, Columbus, OH, USA, where he directs the Thermal properties of Materials for Extreme environments laboratory. His research interests are in investigating the impact of radiation damage on properties of materials, development of experimental methods, and sensors for measuring material's physical properties in extreme environments. Dr. Khafizov also serves as the Deputy Director for the Center for Thermal Energy Transport under Irradiation, an Energy Frontiers Research Center funded by the U.S. Department of Energy, Office of Basic Energy Sciences.

Carol Smidts is a Professor in the Department of Mechanical and Aerospace Engineering at Ohio State University, Columbus, OH, USA. She graduated with a B.S./M.S. and Ph.D. degree from the Université Libre de Bruxelles, Bruxelles, Belgium, in 1986 and 1991, respectively. She was a Professor at the University of Maryland at College Park in the Reliability Engineering Program from 1994 to 2008. Her research interests are in software (SW) reliability, SW safety, SW testing, probabilistic risk assessment, and human reliability. She is a senior member of the Institute of Electrical and Electronic Engineers and a member of the editorial board of Software Testing, Verification, and Reliability.