INL/EXT-19-55610

Light Water Reactor Sustainability Program

Data Integration Aggregated Model and Ontology for Nuclear Deployment (DIAMOND): Preliminary Model and Ontology

September 2019

U.S. Department of Energy

Office of Nuclear Energy

INL/EXT-19-55610

Data Integration Aggregated Model and Ontology for Nuclear Deployment (DIAMOND): Preliminary Model and Ontology

Ahmad Al Rashdan, Jeren Browning, and Christopher Ritter
Idaho National Laboratory

September 2019

Prepared for the
U.S. Department of Energy,
Office of Nuclear Energy

EXECUTIVE SUMMARY

The data of a nuclear power plant are stored in various isolated forms in different systems. These data have different structures and tools and are, therefore, used independently. The integration of data sources is often performed manually and on an as-needed basis to achieve a particular goal. A holistic approach to integration of the data would enable creation of a data warehouse for a single nuclear power plant or multiple plants. A data warehouse results in direct and indirect cost savings because it automates the manual search for data, enables sharing and comparison of data from various tools from a single or multiple plants, enables digital transformation of data, increases frequencies of needed data that are sparse in nature (such as failure signatures), reduces the need for training on various tools for plant staff, enables a holistic staff perception of plant activities, reveals cost-saving opportunities, and improves the visual perception of all parts necessary for operations and maintenance. One of the main challenges to deploying a data warehouse is associated with the integration of data sources into a consistent data model using a standard ontology that is developed by and for the nuclear power industry. This model is currently nonexistent, but is necessary to enable the nuclear data warehouse function.

A data model contains objects that make up a given domain along with their attributes and relationships. It provides domain standardization and maintainability, setting forth the objects, attributes, and relationships within that domain and acting as the guide by which the various applications within that domain will structure their data and interchange. An ontology is defined as "a set of concepts and categories in a subject area or domain that shows their properties and the relations between them." Ontologies consist of a taxonomy in a directed and rooted tree. A survey of the models and ontologies available to support a data integration aggregated model and ontology for nuclear deployment (DIAMOND) was performed, and different standards that address specific aspects of the needed model and ontology were leveraged and integrated in this effort. Specifically, the basic formal ontology and lifecycle modeling language were leveraged for the base ontology. ISO 14224 and 15926 were leveraged in the development of the overlapping nuclear- and fossil-data aspects. The Electric Power Research Institute common information model was also reviewed, and the power distribution and transmission experience gained was leveraged in the power generation scope of DIAMOND.

The initial aim targeted by this effort was to develop ontological definitions and a model structure. This report documents the resulting model and ontology to develop DIAMOND. The report describes the various elements of model development based on a growing-tree-development approach—i.e., DIAMOND's tree structure is established with the core-data objects and is populated with a preliminary level of detail. Fifteen sources of data in a nuclear power plant, representing the majority of operations data sources in nuclear power plants, were considered and have been incorporated to various extents into DIAMOND at this stage of development. Additional data sources, scope, and a higher level of fidelity will be incorporated into DIAMOND next through use cases in collaboration with the nuclear industry and relevant stakeholders.

ACKNOWLEDGEMENTS

The authors would like to thank the Light Water Reactor Sustainability program for funding this work and the Idaho National Laboratory's Laboratory Directed Research & Development for funding the base ontology research that was leveraged in this work. The authors would also like to thank Curtiss Wright for their support on this work, and Cooper Nuclear Station, Department of Energy Versatile Test Reactor, Duke Energy, Electric Power Research Institute, KnowledgeRelay, NextAxiom, Nuclear Information Technology Strategic Leadership, Palo Verde Generating Station, and Xcel Energy for their valuable inputs.

CONTENTS

FIGURES

ACRONYMS

API	Application Programming Interface
BFO	Basic Formal Ontology
BORO	Business Objects Reference Ontology
CIM	Common Information Model
COSMO	COmmon Semantic Model
CRM	Conceptual Reference Model
DIAMOND	Data Integration Aggregated Model and Ontology for Nuclear Deployment
DOE	Department Of Energy
DoDAF	Department of Defense Architecture Framework
DOLCE	Descriptive Ontology for Linguistic and Cognitive Engineering
EMS	Energy Management System
EPRI	Electric Power Research Institute
GFO	General Formal Ontology
IDEAS	International Defense Enterprise Architecture Specification
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
LML	Lifecycle Modeling Language
LWRS	Light Water Reactor Sustainability
MarineTLO	Marine Top-Level Ontology
NPP	Nuclear Power Plants
OWL	Web Ontology Language
PROTON	PROTo Ontology
SUMO	Suggested Upper Merged Ontology
SysML	Systems Modeling Language
UFO	Unified Foundational Ontology
UMBEL	Upper Mapping and Binding Exchange Layer
UML	Unified Modeling Language
W3C	World Wide Web Consortium
YAMATO	Yet Another More Advanced Top Ontology

DATA INTEGRATION AGGREGATED MODEL AND ONTOLOGY FOR NUCLEAR DEPLOYMENT (DIAMOND): PRELIMINARY MODEL AND ONTOLOGY

1. INTRODUCTION

Nuclear power plants (NPPs) in the United States (U.S.) are challenged with sustaining an economically profitable profile due to the plants' running cost and an increasingly competitive market landscape. As a result, the Department of Energy (DOE's) Light Water Reactor Sustainability (LWRS) program launched multiple efforts targeting efficiency opportunities that can result in significant cost saving without impacting the safety of the plants. Some of these opportunities are related to the use of data to offset manual human activities through a seamless flow of data between plant entities and among multiple plants. The data of an NPP are stored in various forms for different systems (Figure 1), such as the plant computer, the work management system, scheduling, systems engineering, operator logs, condition reporting, etc. These data have different structures and tools and are, therefore, used independently. If a data-integration application is needed, it is performed manually and within a limited scope. While leveraging data in silos can still achieve significant cost saving, the collaborative use of data from all sources in a plant or from multiple plants can result in significantly higher cost saving.

An increasing number of domains are realizing the advantages of a central data repository (i.e., a data warehouse) which facilitates the integration of applications and hosts data according to a standard (Al Rashdan et al. 2019). The nuclear power industry needs to establish a data warehouse for a single NPP or multiple plants. A data warehouse results in direct and indirect cost savings because it automates the manual search for data, enables sharing and comparison of data from various tools, enables digital transformation of data, increases frequencies of needed data that are sparse in nature (such as failure signatures), reduces the need for training on various tools for plant staff, enables a holistic staff perception of plant activities, reveals cost-saving opportunities, and improves the visual perception of all parts necessary for operations and maintenance. One of the main challenges to deploy a data warehouse is associated with the integration of data sources.

For many years, point-to-point integration between individual applications has been the standard in enabling data exchange. However, this methodology is exponentially more time consuming, leads to integrations which can only be understood by their owners, prohibits expansion and modification of integration, and leads to confusion in domain terminology and knowledge. The number of manually created connections depends on the number of systems involved, but quickly grows as more and more applications are added. Figure 2 shows an example of a mesh and a star topology. For any n number of applications, the number of required mesh connections is $\frac{1}{2} \times n (n - 1)$ (Bennett 2008), while only n is needed for a star topology. This means that, to integrate five applications in a mesh topology, 10 connections are needed, and for 10 applications, 45 connections are needed.

A recent pilot to integrate four data sources in an NPP using a star topology highlighted the need for a data model and ontology, which currently does not exist for the nuclear industry (Al Rashdan et al. 2019). Having a standard by which to communicate can allow for rapid means of collecting, storing, and using data. With a defined central model and ontology in place, each application can use this standard for the import and export of data, because the blueprint by which data is exchanged is defined once (i.e., all applications have standard reference on means to communicate with each other and interpret the data received). The model and ontology also act as a standard for the terminology and relationships of the data within that domain. This report describes the first data integration aggregated model and ontology for nuclear deployment (DIAMOND) to enable the development of a data warehouse for the nuclear industry. DIAMOND is a living model and ontology which will meet the common requirements of various types of applications. When an individual application, with unique features, integrates with DIAMOND, it may be determined that additional entities or properties are necessary to accommodate the unique features.

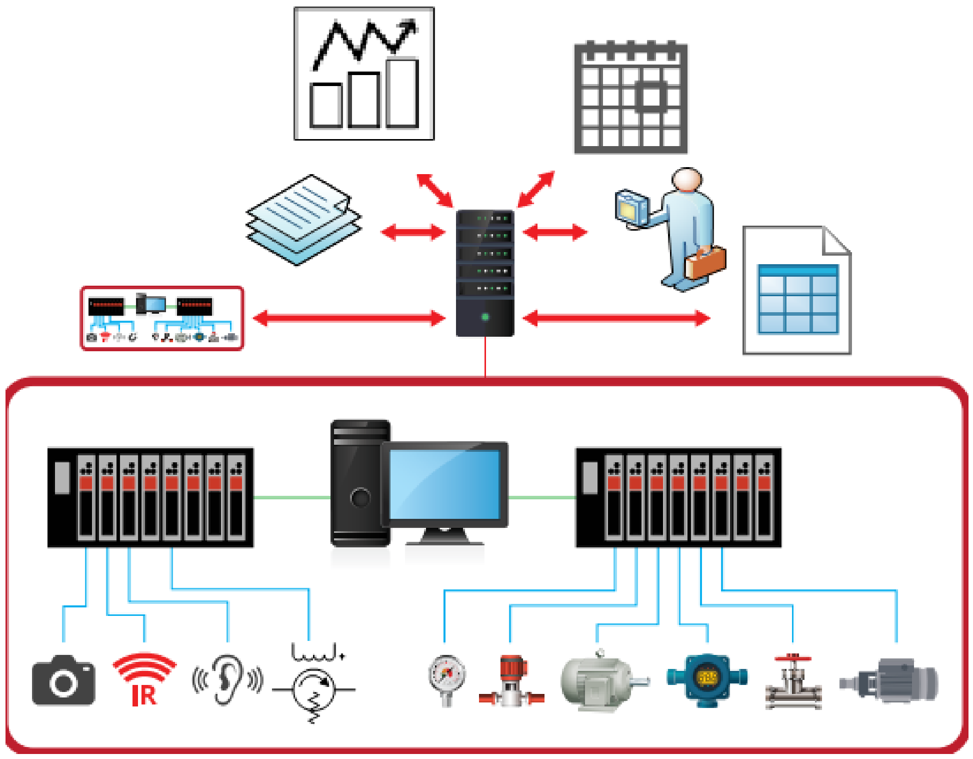
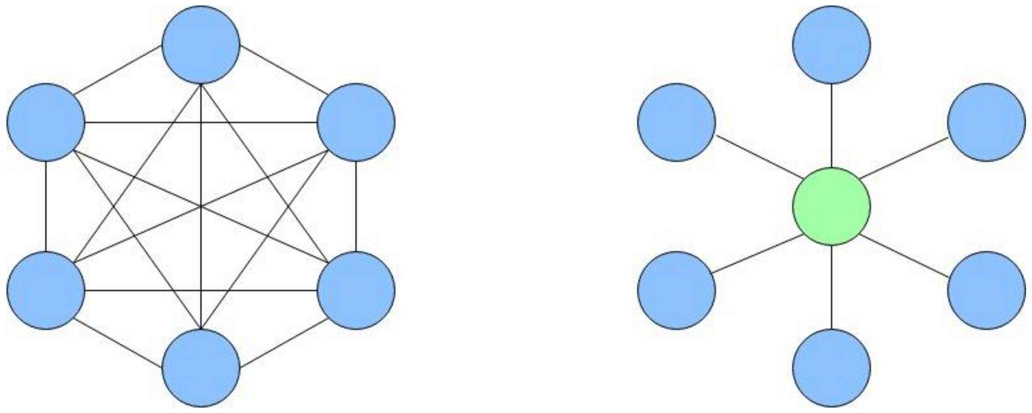


Figure 1. The concept of a data warehouse in an NPP (from Al Rashdan et al. 2019).



(a) Mesh or direct connection approach (b) Star or centralized data connection approach

Figure 2. Depictions of nodes (applications) and the connections between them using a mesh approach and a central (star) data repository approach.

Section 0 of this report describes the standards and literature reviewed or leveraged by DIAMOND. The report then describes DIAMOND data elements, structure, relationships, and attributes in Section 3. Section 4 describes key elements of the model from two perspectives: the model and data sources perspectives. The effort presented is a time snapshot of an on-going and active development of

DIAMOND. This report is not intended to describe the details of DIAMOND. Only key elements of DIAMOND are described herein. An exhaustive listing of all the elements of DIAMOND are in the actual data model and ontology (planned for open-access release in 2020). Additionally, over the next few years, additional data sources, scope, and higher levels of fidelity will be incorporated into DIAMOND through use cases in collaboration with the nuclear industry and relevant stakeholders. A summary of the future planned work is presented in Section 5.

2. STANDARDS AND LITERATURE REVIEW

DIAMOND is a data model, but more specifically, it is an ontology. This is because an ontology is a data model that stores the objects in a hierarchical order. It consists of a taxonomy in a tree structure. Prior to development of a new model and ontology, it was essential to review related existing efforts in order to better understand what work had already been accomplished in this area. This included considering whether any existing model or ontology might be suitable for extension (as an alternative to creating a new model and ontology).

2.1 Models

A data model contains objects that make up a given domain, along with their attributes and relationships. It provides domain standardization and maintainability, setting forth the objects, attributes, and relationships within that domain and acting as the guide by which the various applications within that domain will structure their data and interchange. The Electric Power Research Institute (EPRI) has developed, through the International Electrotechnical Commission (IEC), a common information model (CIM) EPRI (2015). The EPRI (2015) CIM is targeted towards energy markets and power transmission and distribution (i.e., not power generation) and provides common ground for organizations and applications that need to interact across these domains. While the CIM describes electrical components and their relationships, it also covers other aspects related to the exchange and management of these data, including asset tracking, work scheduling, and customer billing. The model provides for simple and straightforward application integration, extension, maintenance, file transfer, and version control. In addition to the EPRI (2015) CIM primer, three standards were reviewed.

- IEC 61970-301 is the primary model of the CIM. Some key packages here include the core, wires, and topology packages (McMorran 2007).
- IEC 61968-11 is an extension of IEC61970-301. This model is focused on exchange of data within distribution companies (McMorran 2007).
- IEC 62325-301 is focused on defining data exchanged between participants in electricity markets (IEC 2018).

International Organization for Standardization (ISO) 15926-13:2018 was also reviewed (ISO 2018). It specifies a model for “asset planning for process plants, including oil and gas production facilities.” This model was leveraged due to the overlap between what would be contained in a gas and oil plant and nuclear plant models. The standard includes a number of Microsoft Excel files that are openly available for use and grouped by category (connection materials, heat transfer, etc.). Each file contains a detailed dictionary of things within that category. Each entry contains a definition and lists the superclass, if applicable. For example, the angle-valve class is grouped within the superclass, valve. These lists were deemed useful for inspiring and validating parts of DIAMOND, particularly for classes of physical equipment. Because the model’s ontology is organized by its predefined categories rather than a representation of real objects, it was not a suitable candidate for extension.

ISO 14224:2106, “Petroleum, petrochemical and natural-gas industries: Collection and exchange of reliability and maintenance data equipment,” was also reviewed as a potential candidate for extension. ISO 14224 is similar in subject to ISO 15926, attempting to cover reliability and maintenance data for equipment in the petroleum, petrochemical, and natural-gas industries (ISO 2016). While the industrial complex this standard applies to is much wider than the generation of power, it includes as a subset the systems and components that are specific to power generation that are well correlated with nuclear power generation. It also aligns well with the mission of data collection, exchange, and analysis to support the needs of the industry for safe operation, reliability, and the maintainability of equipment. The scope of this standard is "…the collection of reliability and maintenance . . . data in a standard format for equipment . . . during the operational lifecycle of equipment." The standard defines the minimum required dataset to be collected and the data format required to facilitate the exchange of data. The organization is based on a taxonomy classification with the following nine segments: Industry, Business Category, Installation, Plant/Unit, System/Section, Equipment Unit, Subunit, Component/Maintainable Item, Part.

The ISO 14224 standard contains useful tables of information related to types of equipment, as well as a number of images depicting the boundary definitions of the various systems related to an equipment type. It presents a method that can be used for modeling data for components within the NPP, but it is limited both in describing relationships between the equipment and identifying specific data points for them. The model is also limited to the definition of specific classes of equipment. The standard appendices provide a rich ontology for the identification of failure maintenance and parameter data points. Included are failure mechanisms, failure causes and events, detection methods, maintenance activities, and failure modes grouped by equipment classification. There is then a mathematical analysis of failure and an appendix specifically for failure data analysis. Because the focus is on maintainability and reliability of equipment, this effort determined that, while the standard represents a good source of material for the breakdown and identification of equipment within the NPP, it does not serve as a valid standard as is for nuclear deployment. Although not used for nuclear model and ontology extension, the CIM, ISO 14224, and ISO 15926 were very useful as a reference for the development of DIAMOND.

2.2 Ontologies

An ontology is defined as "a set of concepts and categories in a subject area or domain that shows their properties and the relations between them" (Lexico 2019). Ontologies make use of the terminology of universals and particulars. A universal is an entity or "anything that exists, including objects, processes, and qualities" (Arp et al. 2015), while a particular is an instance of some universal. Thus, the classes that are defined in the ontology may be considered universals while the objects that instantiate them are particulars. Ontologies consist of a taxonomy in directed and rooted tree. This is a tree structure with a single root node which has some number of children nodes. Those children nodes may then have their own children and so on. These parent/child relationships make use of inheritance. Thus, the root node is the most general entity or thing represented, and each child node is some defined specialization of its parent. A child of some class will inherit all the properties of the parent class and can then have its own unique properties in addition to the properties inherited. The common phrasing used to denote this relationship is "is a," such as a mammal is an animal.

The basic structure of an ontology is common among all ontologies, but further characteristics can deviate based on the ideals being followed by the developer of the ontology. Thus, it becomes necessary in ontology development to define some set of rules and guides that the developer will follow. Important concepts must be considered, such as whether a child can have more than one parent, whether an attempt is being made to describe reality or concepts, etc. Ease of application integration, a defined set of domain-specific nomenclature, and a domain dictionary of sorts are all benefits of an ontology. Additionally, ontologies may be extended to include further levels of granularity or even other domains.

The concept of upper or top-level ontologies was also studied. Upper ontologies can be extended for use by virtually any domain, whether that domain is biomedicine or architecture. Each of the upper ontologies provides the uppermost classes or universals to be extended by a specific domain. An upper ontology is an ontology which consists of very general terms (such as "object", "property", "relation") that are common across all domains (Mascardi et al. 2007). There are several key benefits of incorporating an upper ontology into a domain ontology. These benefits include:

- A predefined and concise structure in which to place domain-specific objects
- A predefined structure that has been thoroughly vetted by some community
- The potential for integration with other ontologies that also make use of the same upper ontology.

The principles these upper ontologies follow and how they attempt to provide a framework for domains to use can vary greatly. A large number of upper ontologies were examined, looking for one or more that could be incorporated into DIAMOND to provide the benefits listed above. The evaluation of the available upper ontologies was largely based on their current use in scientific communities and industrial domains. A structure that was easy to understand and could accommodate the needs of DIAMOND (such as space to represent time-bound processes and actions) was also sought. The following upper ontologies and references were examined:

- Basic Formal Ontology (BFO, Smith 2019)
- Business Objects Reference Ontology (BORO, Cesare 2016)
- Conceptual Reference Model (CRM, Sinclair et al. 2006)
- COmmon Semantic MOdel (COSMO, Cassidy 2010)
- Cyc (OpenCyc and ResearchCyc, Mascardi et al. 2007)
- Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE, Masolo et al. 2003)
- Department of Defense Architecture Framework (DoDAF, Department of Defense 2010)
- General Formal Ontology (GFO, Herre et al. 2007)
- Gist (Semantic Arts 2019)
- International Defense Enterprise Architecture Specification for exchange Group (IDEAS, IDEAS Group 2011)
- Lifecycle Modeling Language (LML, Lifecyclemodeling 2017)
- Marine Top-Level Ontology (MarineTLO , Tzitzikas et al. 2013)
- PROTo Ontology (PROTON, Terziev et al. 2005)
- Suggested Upper Merged Ontology (SUMO, Niles and Pease 2001)
- Upper Mapping and Binding Exchange Layer (UMBEL 2016)
- Unified Foundational Ontology (UFO, Guizzardi and Wagner 2004)
- Yet Another More Advanced Top Ontology (YAMATO, Mizoguchi 2010).

Key differences between these include how they handle the representation of objects in reality (or a three-dimensional space) versus processes that occur over time, the level of granularity, rules around relationships, and so forth. For use in DIAMOND, some of these ontologies were disqualified due to designs for specific domains (MarineTLO, IDEAS, DOLCE, CRM, BORO, DoDAF, UMBEL), paid usage (Cyc), or a lack of needed information and use (GFO, COSMO, UFO). Of the remaining upper ontologies, it was determined that several had relatively less usage or seemed overly verbose for DIAMOND needs (YAMATO, SUMO, PROTON, Gist).

The BFO is an upper ontology in use by over 300 ontologies across a variety of domains, is actively maintained (Basic Formal Ontology Team 2019), contains a concise number of classes (less than 40) which meet the needs of the targeted ontology, and is supported and vetted by the scientific community through tools and services such as the Open Biological and Biomedical Ontologies Foundry. Due to its usage and the universal nature of the classes within BFO, it was considered a prime candidate for use as an upper ontology. LML is another upper ontology that also meets the goals stated above, albeit with a focus on business-process language and structure. It was determined that a combination of BFO and LML would best suit the needs of DIAMOND.

3. METHODOLOGY AND TOOL

Ontologies most often use Web Ontology Language (OWL). Web Ontology Language (OWL) is a World Wide Web Consortium (W3C) standard that is also utilized to allow for structuring entities as well as validation of the ontology structure. Due to these features and others, including the ability to define the relationships within the ontology, reuse of data properties, and the ability to easily express existential, universal, or cardinal relationships to properties, OWL is a preferable tool to alternatives like the Unified Modeling Language (UML).

OWL is a semantic web language, a "computational logic-based language such that knowledge expressed in OWL can be exploited by computer programs, e.g., to verify the consistency of that knowledge or to make implicit knowledge explicit" (W3C 2013). It provides a structure for representing ontologies, and specifically the objects and relationships inside of them. OWL not only provides a framework in which to structure the classes, relationships, and properties needed for an ontology, but also allows for verifying the contents of the ontology. This can be done through tools called reasoners, which examine the objects within the ontology and point out any possible errors where the ontology is inconsistent. Additionally, a reasoner can use the information asserted in the ontology to infer more details about the ontology's structure. The syntax used for OWL can take a variety of forms (such as functional syntax, OWL2 XML, Manchester) but the most commonly used form is RDF/XML syntax (W3C 2012a).

It is essential that an ontology developer choose a series of principles to follow during development. The following list highlights a number of principles which have helped previous ontologies achieve success (Arp et al. 2015):

- Analyze existing ontologies and determine what universals may be copied from them
- Stay as close as possible to the terms used by actual domain experts
- Don't use acronyms
- Use singular nouns
- Start with the most general terms in the domain (helps to define the scope of the ontology)
- Maintain an open-world assumption (i.e., the ontology can and will change as new things are discovered or granularity levels increased)
- Adhere to the rule of "objectivity", describing what exists, not what is known.

These principles were followed throughout this effort. They help to avoid pitfalls such as circularity, failure of nodes to trace back to the root node, and difficulty in multiple developers' working on the ontology simultaneously. The ontology is divided into the following elements that are discussed in the next sections:

- Classes
- Object properties, also referred to as relationships
- Data properties, also referred to as attributes.
- Annotation properties.

3.1 Classes

Classes are the objects with which the ontology is primarily constructed. Properties of a class that are inherited from parent classes are grouped into the "SubClass Of (Anonymous Ancestor)" group within the description of a class. Properties that are specific to that class (along with an automatically generated entry indicating the parent class) are placed in the "SubClass Of" group. Properties that are inherited to a class should not be edited; any new properties or modifications should be made to the SubClass Of group. For example, a document class exists under the LML artifact class because a document is a type of artifact. The document class will inherit all the properties from the artifact class, and then can be assigned any additional properties that are specific (or local) to it.

Development guidelines: In this effort, classes:

- Are labeled in lower case with their most common name. Alternative forms of the name may be added as "alternative term" annotations, and acronyms may also be added in uppercase using this same type of annotation.
- Should be provided a "definition" annotation to the extent possible. This statement should try to follow the Aristotelian format of "An x is a y that…" where x is the class being defined and y is the superclass of x. This format ensures that the inheritance relationship is followed between classes within the ontology and that each key differentiating factor of each class is specified in the latter portion of the definition (after "that").
- Names (or labels) should be unique (part of best practices in ontology design, Svátek and Šváb-Zamazal 2010). This makes the process of finding a class faster and easier and avoids any confusion, thus aiding in development of the ontology. Unique class names also ensure compatibility for any related software and application development by making class mappings straightforward and non-ambiguous. To fulfill this requirement, terms from a parent class may often be seen in children classes (e.g., role, document, system) in order to fully qualify what the entity is without having to infer any portion of the desired class name from parent or sibling entities. For example, operator role is a class under the role class.
- Ideally, should use singular nouns as much as possible. When a plural noun refers to a singular object, it can be acceptable, and therefore singular nouns should not be strictly enforced (Svátek and Šváb-Zamazal 2010). For example, when defining a type of equipment like gauge, the class label should be gauge and not gauges. Classes like system and its descendants may have the connotation of referring to multiple things, but as a class, refer to a singular identified system and are, therefore, an acceptable terminology to use in the ontology.

Particularly at higher levels in the ontology, it was important to mark sibling classes as disjoint. For example, the BFO classes temporal entity and non-temporal entity are marked as disjoint. This ensures that something that is a temporal entity (inherits from the temporal entity class) cannot also be a nontemporal entity, and vice versa. The ontology has made classes disjoint down to the LML level. A few classes further down the tree have been made disjoint as well, where it was deemed useful. These disjoint relationships help to ensure that DIAMOND follows a strict specialization hierarchy (where each class has only one parent) as much as possible. Multiple inheritance is not strictly disallowed within DIAMOND, but should be avoided to promote clear definitions of classes and an ontology structure that is easy to interpret.

3.2 Object Properties (Relationships)

The hierarchy view of classes only shows the inheritance relationship, so any other relationships must be defined by the user as object properties and then applied to classes. The relationships within the DIAMOND ontology are primarily based on LML relationships. Relationships created in addition to those taken from LML include "has role" and "includes" (along with its inverse "included by"). The majority of these relationships have an inverse. For example, decomposes is the inverse of decomposed by. The relationships that do not have an inverse are specified by (which relates a class to some quality), relates (which is a generic association relationship and is the inverse of itself), and has role (relates a class to some role). These relationships are placed on a class by adding a property and using the object restriction creator.

- caused by: Caused by is a relationship for the risk class (in LML) that denotes some asset, artifact, etc. that causes the risk.
  Example: Risk caused by Action
  Inverse: causes
- decomposed by: Identifies the children of some entity. This relationship subsumes every other wording for a parent/child relationship (e.g., part of, composed of, etc.)
  Example: Questionnaire decomposed by Survey Item
  Inverse: decomposes
- generates: Denotes that some entity has an output that is a result of some class or process.
  Example: Action generates Transition Template
  Inverse: generated by
- has_role: Denotes that the entity contains some role.
  Example: Smoke Detector has role Fire Detector Role
  Inverse: none
- includes: Denotes that the owner of the relationship includes the indicated entity but is not a parent to it.
  Example: Process includes Action
  Inverse: included by
- located at: A located at B if A is contained within a subset of the spatial region occupied by B at some time t.
  Example: Reactor located at Reactor Building
  Inverse: locates
- mitigated by: Denotes that a risk is mitigated (or that the probability or cost is reduced) by some entity.
  Example: Risk mitigated by precaution
  Inverse: mitigates
- performed by: A is performed by B if A is a process (or action) and B is a continuant (non-temporal) entity where B participates in or is involved in A.
  Example: Action performed by Role
  Inverse: performs
- referenced by: Denotes that the entity (usually an artifact or asset) is referred to by some other entity.
  Example: Purchase Order referenced by Incoming Invoice
  Inverse: references
- relates: A generic relationship to be used when other available relationships are insufficient. Can often be used for relationships between entities that have the same LML parent (e.g., artifact, asset).
  Example: Plant Documentation relates Document Type
  Inverse: none
- specified by: Denotes a relationship to some quality using the Characteristic naming convention from LML (specified by).
  Example: Part specified by Mass
  Inverse: none

Development guidelines: When adding object properties to a class:

- A restriction type must be specified. This indicates the desired cardinality for that property. The possible restriction types are Some, Only, Min, Max, and Exactly. The types Min, Max, and Exactly require a number to be provided. The Some type is an existential restriction and indicates that the ontology does not know for sure whether there will be a relationship here, but that one or more may exist. This restriction type is the default restriction type for the DIAMOND ontology because it allows for optional properties on a class. If a cardinality is known (for example, we know that this class requires this property in order to exist), the Min, Max, or Exactly restriction types may be used. The Only restriction type may be used for indicating enumerators by restricting the property to a predefined set of classes (through an object property) or data properties.
- Object property characteristics can be marked for each object property as applicable. These characteristics denote whether an object property is functional, transitive, symmetric, etc. Further details on the meaning of these characteristics can be found on Protégé's website. Protégé is the tool used to develop DIAMOND and is described in Subsection 3.5.

3.3 Data Properties

While classes can be thought of as the tables in a relational database, data properties are the fields that make up the content of those tables. Data properties must be a primitive type, such as a string, double, date timestamp, etc. These types mostly come from the XML Schema Definition Language (XSD) (W3C 2012b) and are named accordingly within the Protégé tool (e.g., xsd:integer). After a data property (such as "status" with type xsd:string) is defined, that property can be applied to any number of classes. As with classes, a developer of the ontology should first ensure that there are no existing data properties that meet their needs before creating new entities. The data properties within the ontology are grouped by their type. These categories include:

- boolean: Uses the xsd:boolean type.
- date: Uses the xsd:dateTimeStamp type.
- enumeration: Properties that are a list of static values (generally xsd:string type) from which one value would be chosen by an instance of a given class.
- number: The main datatype used here is xsd:double, which handles decimal points. If the number is known to only need two positions past the decimal (such as for a currency or some percentages), the xsd:decimal type may be used. The xsd:integer type may also be used when it is known that only whole numbers will be used.
- string: Uses the xsd:string type.
- uri: Uses the xsd:anyURI type to indicate paths to web or file resources.

Development guidelines: A best practice when using data properties is:

- Ensuring no data property currently exists that can meet the needs of the developer before adding new properties. One of the simplest ways to ensure this is to use Protégé's find command to verify whether a similar property exists. Searching for synonyms or related terms can reveal existing entities that may be reused.
- Group properties by head noun (in addition to organizing data properties by the types listed above). For example, if the data property "quantity" is needed for some entity and another entity requires a data property called "issued quantity", "issued quantity" may be created as a child of the quantity property (assuming they share the same type, in this case, number). This works well when the properties are all the same type and aids in reducing the time to find desired properties in the Data Properties view.

3.4 Annotation Properties

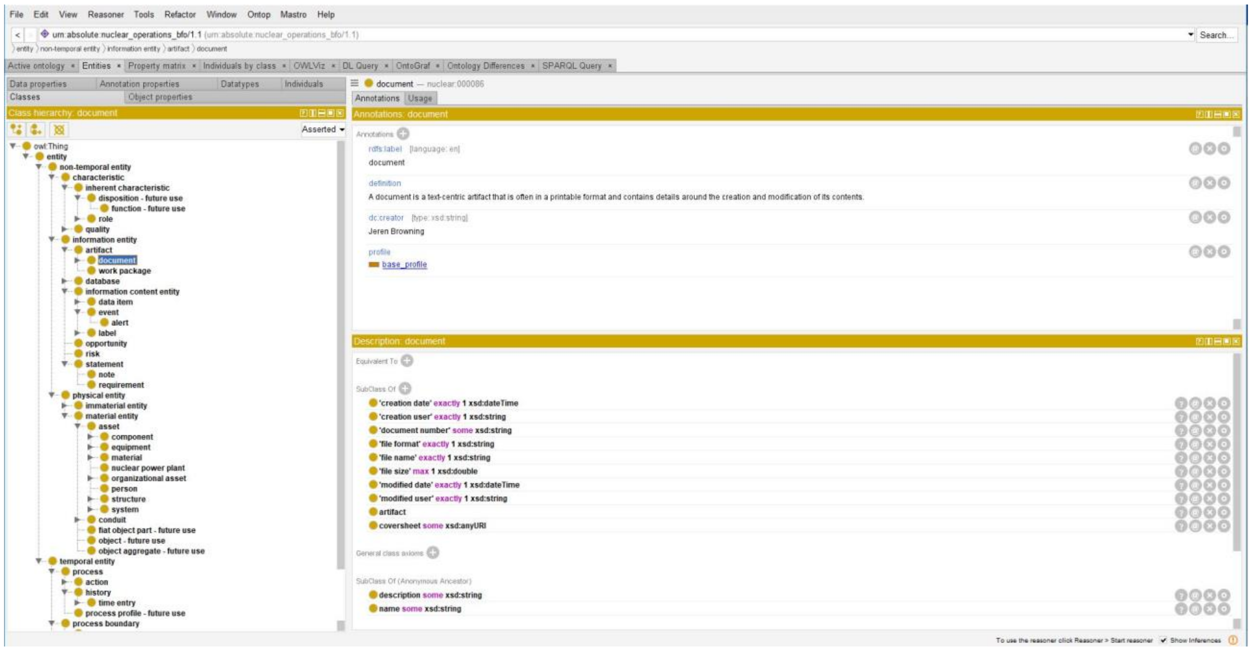
Annotation properties are primarily used for holding metadata related to some entity. OWL editors such as Protégé allow for the developer to define a format for automatic ID generation for new entities created within an ontology. In DIAMOND, this format is "nuclear:" followed by 6 numbers, which Protégé automatically increments as new entities are created. Therefore, the human-readable label of an entity is not the unique ID of the entity, but rather a rdfs:label annotation property on the entity. Other annotation properties which are commonly used in DIAMOND include definition, dc:creator (indicating who created the entity), profile (created specifically for DIAMOND and discussed in further detail in section 4.2.1), alternative term, and example of usage.

A list of available annotation properties can be seen in the Annotation Properties view within Protégé. Of particular use to the developer is the Usage tab of an annotation property, which allows the developer to see what entities make use of a given property.

3.5 Development Tool

The Protégé tool was used for the development of DIAMOND. Protégé is an established tool for working with OWL files and is provided as a free and open-source software (by Stanford University 2019). This tool allows for easy editing of OWL files. Protégé, and ontology editors in general, structure the ontology in hierarchical views (example shown in Figure 3).

Protégé organizes classes in a hierarchical tree by inheritance. This allows for easy viewing of the inheritance relationship. Visualizing other relationships that exist, through the object properties created and additions to classes, is more challenging with the base Protégé tool. However, a number of plugins are available for Protégé to extend its functionality. One of these is OntoGraf, which allows for easy visualization of both the inheritance relationship and relationships between classes (Falconer 2013). Examples of OntoGraf use in the context of DIAMOND are shown in Figure 4 and Figure 5.



Class hierarchy: document

```
owl:Thing
  entity
    non-temporal entity
      characteristic
        inherent characteristic
          disposition - future use
            function - future use
          role
        quality
      information entity
        artifact
          document
          work package
        database
        information content entity
          data item
          event
            alert
          label
        opportunity
        risk
        statement
          note
          requirement
      physical entity
        immaterial entity
        material entity
          asset
            component
            equipment
            material
            nuclear power plant
            organizational asset
            person
            structure
            system
          conduit
          fiat object part - future use
          object - future use
          object aggregate - future use
    temporal entity
      process
        action
        history
          time entry
        process profile - future use
      process boundary
```

document — nuclear:000086

Annotations: document

Annotations

rdfs:label [language: en]
document

definition
A document is a text-centric artifact that is often in a printable format and contains details around the creation and modification of its contents.

dc:creator [type: xsd:string]
Jeren Browning

profile
base_profile

Description: document

Equivalent To

SubClass Of

- 'creation date' exactly 1 xsd:dateTime
- 'creation user' exactly 1 xsd:string
- 'document number' some xsd:string
- 'file format' exactly 1 xsd:string
- 'file name' exactly 1 xsd:string
- 'file size' max 1 xsd:double
- 'modified date' exactly 1 xsd:dateTime
- 'modified user' exactly 1 xsd:string
- artifact
- coversheet some xsd:anyURI

General class axioms

SubClass Of (Anonymous Ancestor)

- description some xsd:string
- name some xsd:string

To use the reasoner click Reasoner > Start reasoner [✓ Show Inferences]

Figure 3. A view of DIAMOND within the Protégé editor.

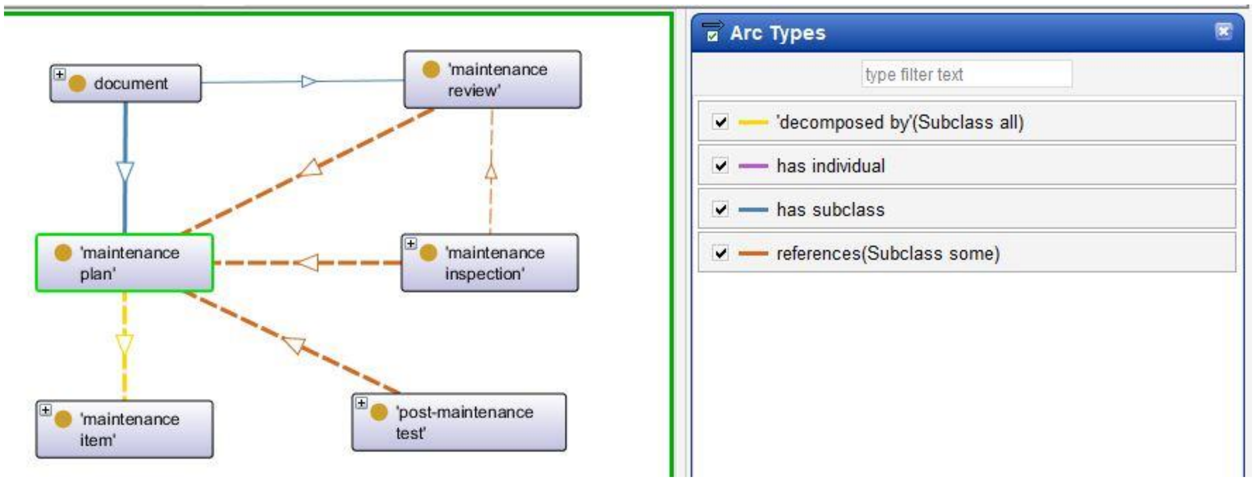


Figure 4. An example view of several of the key classes within the maintenance data source, shown in OntoGraf.

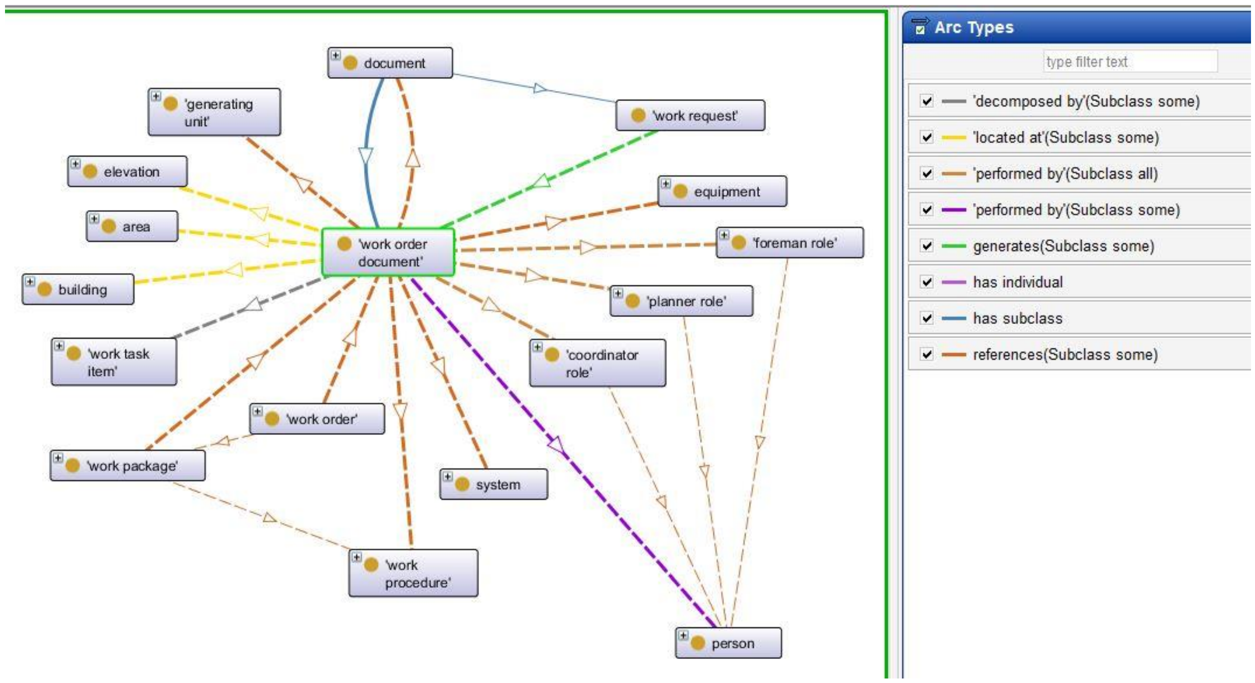


Figure 5. An example visualizing the several relationships of work order document with OntoGraf.

4. MODEL AND ONTOLOGY

The base of DIAMOND leverages a combination of BFO and LML (discussed in 2.2). BFO creates classes that can categorize any entity. This includes processes, material things, immaterial things, and some interesting approaches to concepts like qualities and roles. However, there were few detractors from BFO for this use case. The first was that several of the class names used by BFO are difficult to understand and require investigation into their definitions and an overview of examples in order to understand them properly. A key milestone for DIAMOND is to be made available to public for others to extend. This is to enable developers to view and modify the OWL file produced. DIAMOND's open-access requirement and extendable nature will allow for a continuous and straightforward expansion process. Since several of the high-level BFO classes have noninstinctive names, and to ensure the ontology is user-friendly, it was necessary to make the names of these BFO entities more intuitive. Alternative labels are provided in the ontology for these BFO classes. Although the name of the class in the hierarchy view of an editor such as Protégé has been changed, no other properties or relations associated with these BFO classes have been altered. This means that the original BFO IDs, along with annotations listing the original BFO class name, have been kept intact, so that DIAMOND's ontology is still fully compatible with other ontologies utilizing BFO.

Below is a list of the updates to BFO class labels with the BFO original label first, followed by DIAMOND's new label:

- Continuant: Nontemporal Entity
- Occurrent: Temporal Entity
- Generically Dependent Continuant: Information Entity
- Independent Continuant: Physical Entity
- Specifically Dependent Continuant: Characteristic
- Realizable Entity: Inherent Characteristic.

Some of the BFO classes were included in DIAMOND, but are not currently in use by the ontology. Their labels had " – future use" appended to indicate that the class is not in use, but may be used in the future.

The quality and role classes required particular attention. When making any modification to the ontology, it is important to consider whether a new entity (class or property) might fall under the BFO classes of quality or role. A clear understanding of these two classes and when they should be used will help to ensure consistency throughout the ontology. A definition was necessary to determine what should be represented as a class under BFO's class "quality" and what should be represented as an OWL datatype property. BFO defines quality as a specifically dependent continuant (characteristic) that does not require any sort of process to be realized (Arp et al. 2015). It should also exist at any time that the independent continuant (physical entity), to which it is associated, exists. This means that attributes of some physical entity that can be measured over time or scientifically evaluated should be included in the ontology as qualities. This includes entities such as mass, voltage, pressure, etc. To ensure this definition is adhered to, all qualities have the properties of a measurement, measurement date, and unit of measurement (pounds, kilograms, feet per second, etc.) Other class attributes should be created as data properties. For example, if the entity attempting to be represented is constant, assigned by a person or process, or is an enumeration (list of values), it is represented as a data property.

The other BFO class of interest is the "role" class. A role may be thought of as a characteristic which some entity may take on at some time. For example, in the DIAMOND ontology, a student would not be a subclass of person, but would rather be a role that can be associated with the person class. Considering things as they exist in reality, a student is not truly a type of person, but rather a set of properties a person may take on at a given time, possibly with some existing preconditions. A role may be applied to more than just the person class, and the DIAMOND ontology defines several classes beneath role to aid in the organization of types of roles. These classes currently include object role, organizational role, and personnel role.

Although BFO provides a solid foundation for a representation of objects as they exist in reality, it lacks many of the more business-oriented terms by which one might categorize the objects within a nuclear plant. It was apparent that perhaps an additional upper ontology placed directly beneath some of the BFO classes would provide the ideal foundation for DIAMOND. LML focuses on providing a structure and language that can be used to facilitate understanding between stakeholders throughout the lifecycle of a product (Lifecyclemodeling 2017), and was therefore incorporated. Not all of the classes and relationships from LML have been incorporated into the ontology, but several of the classes are applicable. The following LML classes were included in DIAMOND:

- Action – “Generates effects and may have pre-conditions before it can be executed. This Action can include transforming inputs into outputs.”
- Artifact – “Specifies a document or other source of information that is referenced by or generated in the knowledge base.”
- Asset – “Specifies an object, person, or organization that performs an action, such as a system, subsystem, component, or element.”
- Conduit – “Specifies the means for physically transporting Input/Output entities between Asset entities. It has limitations (attributes) of capability and latency.”
- Opportunity – “A circumstance with the potential for some benefit.”
- Requirement – “Identifies a capability, characteristic, or quality factor of a system that must exist for the system to have value and utility to the user.”
- Risk – “Specifies the combined probability and consequence in achieving objectives.”
- Statement – “Specifies text referenced by the knowledgebase and usually contained in an Artifact.”

With the combination of BFO and LML, DIAMOND has the advantages of a potential to integrate with users of either of these upper ontologies, as well as validation and buy-in from both communities.

4.1 Model View Perspective

The hierarchical structure of DIAMOND allows for easy navigation and the ability to intuitively find a desired class. For example, in order to find the valve class, one would first expand the nontemporal entity, as valve is a physical object and not a time-bound process (temporal entity). From the following three choices of characteristic, information entity, and physical entity, physical entity is clearly the next choice. Within physical entity, a valve is a material thing, so material entity is selected next. A valve is an asset, and after expanding that class, the most likely options of component or equipment are presented. The equipment class includes unique identifiers, and therefore contains the objects within a plant that might commonly be referred to in various processes and documents. The component class is for assets that are not given unique identifiers and commonly are part of some equipment. For example, bearing and belt are subclasses of component, but valve and pump are subclasses of equipment. With this definition, equipment can be expanded to find the valve class.

The root class of the ontology is BFO’s entity class. The entity definition is very generic, and it is meant to encompass everything which might be put into an ontology. Beneath the entity class are BFO’s temporal (BFO uses the label occurrent) and nontemporal (continuant) classes.

4.1.1 Temporal Entity

Temporal classes are entities that only exist along some timeline. Examples might include actions or the history of some entity. Underneath the temporal class, the process and process boundary classes are found, both from BFO. The definition of a process is very intuitive, something that occurs over time and may have some input or output and usually involves some actor who performs the process. The process boundary class is essentially the logical divider between two processes, something that might be thought of as a given point in time. There are two BFO sibling classes to process and process boundary, namely spatiotemporal region and temporal region. The spatiotemporal region class is an attempt to represent the region that covers both three-dimensional space as well as time for some process. A child class of this entity would hold details around the space and time in which a process was performed, not the process itself. A temporal region is similar, but only covers the time region of a process or process boundary. DIAMOND has not yet had a use case for these classes, so both of these classes along with the one-dimensional temporal region and zero-dimensional temporal region classes, which are children of the temporal region class, have been marked for future use.

Within the process boundary entity is a class defined as part of DIAMOND, planned transition. Planned transition is a template to be used as part of some process. This class can be best understood by gaining an appreciation for the relationship between the action and planned transition classes (Al Rashdan and Agarwal 2019). Beneath the process class is the LML action class, which is used as the primary class for the representation of a process within the ontology (along with its children classes). The relationship between the action class and planned transition class is a vital element of the representation of processes and workflows within the ontology. An action will hold the information related to the actual work or process being performed, such as who is performing the action, where it is located, what the original (or estimated) duration is, what instructions should be followed, etc. Between two or more actions it is necessary to have some sort of structure that represents the transition that occurs. This transition could depend on the results of the previous action. For example, consider a first action within a process such as "verify pump functionality." Now consider that the process might lead to two different actions, depending on the result of verify pump functionality. If the output of that first action states that the pump is functional, the next step may be something like "check pump flow rate." However, if the first action outputs that the pump is not functional, the next step may instead be something like "report pump." The planned transition class holds relationships to one or more incoming actions (as it is necessary to support various process constructs such as a merging of actions performed in parallel), as well as relationships to one or more outgoing actions. The planned transition holds the logic for determining which action or actions from a list of potential output actions should be chosen and can use the incoming data from the previous action to determine the result.

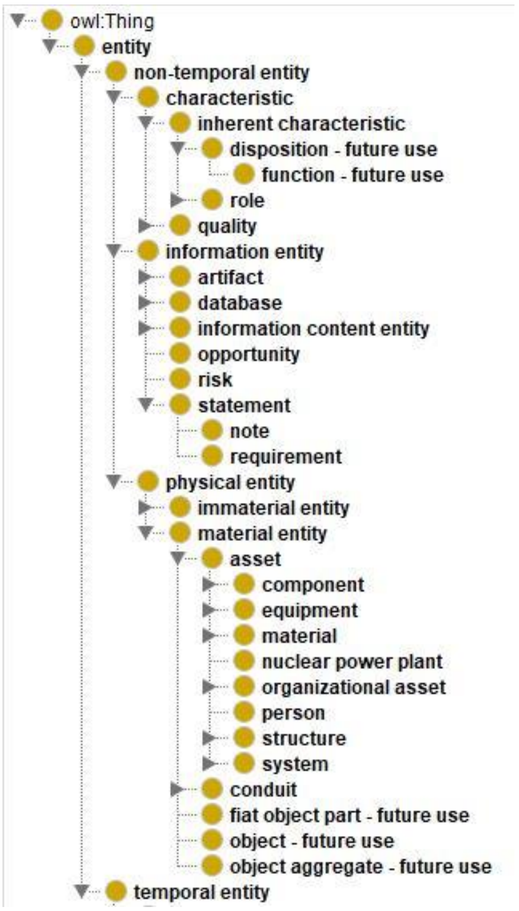
The planned transition class has one child, actual transition. This class holds additional information on top of what is already contained in planned transition and may be used for an individual instantiation of some process. Therefore, the actual transition would hold details unique to that performance of the process, such as what input or output properties may have been involved, what output actions were chosen, and when it occurred.

In a similar manner, action has the subclass planned action which has the subclass actual action. While the action class can hold high-level actions (generally in the form of verbs) for reuse across processes, the planned-action class may hold actions unique to a given process and the actual-action class contains additional detail unique to an individual instantiation of that action within a process. This includes attributes such as actual start date, actual finish date, actual duration, etc.

Sibling classes to the LML action class include the BFO classes of history and process profile. The process-profile class is not in use at this time. The history class has the subclass time entry, and this class and its descendants is used to store historical measurements and values. This allows for the representation of a running log that holds historical timestamped data.

4.1.2 Nontemporal Entity

The nontemporal entity is a parent entity for the majority of objects within DIAMOND. It includes physical items that are found within an NPP or describe those items in some way (such as qualities or roles). More specifically, a non-temporal entity is anything which exists in a form that excludes time. The nontemporal-entity class has three children classes from BFO (see Figure 6). Characteristic (BFO label is "specifically dependent continuant") is a class for entities that are inherent or realized in other entities. Children classes of a characteristic should generally be adjectives or similar terms that would be used to describe some other entity. The information-entity class (BFO "generically dependent continuant") is meant for entities that are generally digital or information-centered in nature. Documents, labels, and generic data structures (such as a note which would contain attributes for text, a timestamp, and the user who created the note) will be found under this class. The third child of nontemporal entity is physical entity (BFO "independent continuant"). This class is the parent for all of the physical objects one might find within a nuclear plant, as well as classes for immaterial entities primarily concerned with physical location.



- owl:Thing
  - entity
    - non-temporal entity
      - characteristic
        - inherent characteristic
          - disposition - future use
            - function - future use
          - role
        - quality
      - information entity
        - artifact
        - database
        - information content entity
        - opportunity
        - risk
        - statement
          - note
          - requirement
      - physical entity
        - immaterial entity
        - material entity
          - asset
            - component
            - equipment
            - material
            - nuclear power plant
            - organizational asset
            - person
            - structure
            - system
          - conduit
          - fiat object part - future use
          - object - future use
          - object aggregate - future use
    - temporal entity

Figure 6. A view of the non-temporal entities discussed within DIAMOND.

The children classes of the characteristic class are inherent characteristic (BFO "realizable entity") and the BFO class, quality. Examples of quality class (discussed in Section 4) children include mass, pressure, and voltage. These entities are related to other classes through object properties and play the role of attributes, much like data properties. However, qualities should be those scientifically measurable attributes of a class that may change over time. If one use case can be contrived where the attribute might change over time, it should be considered a quality. However, if no such use cases can be determined, the attribute should be represented as a data property. For example, if it is determined that for some piece of equipment in the plant, length is an attribute that changes and is measured, but that for another entity (e.g. fixed object), the length attribute will never change, length should still be represented as a quality.

The inherent characteristic class is meant to have children classes which are characteristics of other entities (either information or physical entities). They describe a characteristic of that other class that occurs under certain conditions or further elucidates some function or purpose of that class. The children classes of inherent characteristic are the BFO classes disposition and role. Disposition and its child class function are not used by DIAMOND. The class role has previously been described in Section 4. The children classes of role include object role, organizational role, and personnel role. The personnel role class should contain entities which are generally related to some person. This includes classes like author role, engineer role, and security officer role. While these types of roles will most commonly be related to a performance by some person, it is also possible for some of these roles to be filled by other entities such as a tool or other asset. The organizational role is meant to be a parent to roles which would be associated with a role filled by some logical organization (such as a department or other group). This includes classes like operations role and maintenance role. The final class, object role, is intended to hold all other roles. These are roles that are not associated with an organization and will also never be associated with a person. Examples include shielding-material role and neutron-moderator role.

In the information-entity class, several children classes are found that are not BFO classes. The artifact class is an LML class and is meant for text-centric entities which subsist and are referenced on their own. Therefore, the primary type of child class for artifact should relate to physical or digital documents. When considering attributes on BFO and LML classes, DIAMOND has left BFO classes alone (besides changing labels for some of the classes), and no properties have been added. The LML classes have generally been giving the string attributes of name and description, but more detailed attributes are generally left for subclasses of LML classes. An example of this can be seen with the artifact class, which has only two subclasses, document and work package. Both of these are classes that have been created for DIAMOND. The work-package class exists outside of documents because it is an entity that is more of a reference to other documents than it is a document itself. The document class contains many of the metadata fields related to an individual document, some of which may be thought of as the audit columns of a database table that track who created the document, who last edited the document, and the timestamps for each of these. A name and file size have also been provided. Children of the document class include many of the information and text-centric classes that are central to DIAMOND and the various data sources within it. Examples include classes such as clearance tag, work order, purchase order, and operator-round document.

The artifact class has five sibling classes, database, information-content entity, opportunity, risk, and statement. The database class has been developed as part of DIAMOND and is generally intended to hold metadata about some database. This metadata would then be useful in the exchange of data between applications within an NPP. The information-content entity class comes from the information artifact ontology (IAO, OBO Technical WG 2019). IAO uses BFO as an upper ontology; therefore, it was possible to reuse classes within that ontology in DIAMOND. Information-content entity is a very generic class and is primarily used within DIAMOND as a parent for information entities which do not belong to the sibling classes of information-content entity. The children classes of this entity are data item, event, and label. The event class is intuitive and has the child class alert, which is a type of event. The label class is also intuitive and has children entities which are referenced by other entities to describe some aspect or unique identifier of those entities. For example, identifier is a subclass of label, and serial number, a subclass of identifier. The equipment within an NPP hold a reference to the identifier class so that an association may be made between the unique identifier of some equipment and the data structure for that unique identifier. The data-item class also comes from IAO, and is intended to serve as a parent to entities that are referenced in other entities, primarily artifacts. The children of data item would, therefore, not be generally accessed individually, but through the viewing of some document. An example of this can be found in the operator-rounds data source. The ontology includes an operator-round document, which is the digital or physical document referred to in the planning and execution of some operator-round process.

The opportunity, risk, and statement classes all come from LML. An opportunity is a class intended to be used in relation to the risk class. The definition for this class was introduced as part of DIAMOND because it is not explicitly provided by LML. Opportunity is a concept related to risk, but not a class in LML. The definition and usage of the risk class is intuitive: entities that pose some form of risk to an organization or other entity. The statement class is intended to hold simple pieces of text that are fact-based entities. The realization and value of this class comes primarily from the child class of statement, requirement. Requirement is also an LML class. A requirement should identify some necessary condition with the NPP, such as “a pump shall weigh no more than 100 kilograms.” A sibling class of requirement is note, created by DIAMOND, which holds information about some piece of text and the user who created it (along with the timestamp).

The physical-entity class had two children classes, immaterial and material entity. The immaterial-entity class contains a number of children classes, all of which are from BFO, and may be thought of as a parent for entities that are primarily concerned with spatial regions or some location in space. The continuant fiat boundary (and its children) and site classes are not in use by DIAMOND at this time. The spatial region may be utilized through its four children, dividing spatial regions into the number of dimensions. A zero-dimensional spatial region can be thought of as some finite point in space, and is not currently utilized. A one-dimensional spatial region can be thought of as a line in space, measuring from one point to another. An example that is in use in the ontology is elevation, measuring a distance from a point at sea-level to some point above or below. A two-dimensional spatial region logically represents some surface. In the ontology this is realized as classes such as area or zone (referring to two-dimensional spaces within an NPP). A three-dimensional spatial region covers some three-dimensional space, and is not an appropriate parent for three-dimensional objects one might at first consider. Rather, a spatiotemporal region within BFO refers to a logical definition of space, rather than a definition of some existing object. For example, this class has a child, room. The room entity refers to the three-dimensional space within a room and not some structure, which should be a descendant of material entity.

The material-entity class from BFO contains several children which are not currently in use, fiat object part, object, and object aggregate. Although the object class is intuitive, it is perceived to conflict with one of its sibling classes, asset. Asset is a class from LML that is broad in scope, covering nearly anything physical that is of importance to a business. This includes people, physical items like structures and equipment, and even systems. The overlap in definition between asset and object and object aggregate, and the need to keep the structure and relationships of BFO intact, resulted in not using the BFO object and object aggregate classes, but use the LML asset class instead. The final child of material entity, the conduit class, comes from LML. Conduit is meant to be used for physical entities in the transportation of some input/output. Subclasses of conduit are entities such as cable or pipe.

The asset class contains several children, all of which have been created as part of the development of DIAMOND. Equipment contains attributes such as manufacturer information, condition, and an identifier. Examples include compressor, sensor, and valve. A component does not contain a unique identifier and decomposes the equipment class. Examples include piston, rotor, and seal. The material class contains classes that make up the composition of other material entities. This class has a number of subclasses that are both individually useful as well as useful for grouping other subclasses of material. Some of these classes are chemical element, fabric, gas, and metal. The nuclear power plant class beneath asset is intended to hold information specific to an NPP. Relationships to the various structures and equipment contained within the plant make up some of the attributes of this class. The organizational-asset class was designed to be a parent to children entities who hold information about some logical organization within the NPP. Example subclasses include department, domain, and licensee. Another child of asset is the person class, which holds various attributes unique to a person. It should be noted that the person class does not have any children, as entities like maintenance worker or security officer should be represented as roles. The structure class holds classes such as building and cooling tower and is meant to hold three-dimensional physical entities used for housing other assets and processes. The final child class of asset is system, intended to hold entities that describe one or more material entities that, when combined, are intended to accomplish some function or purpose.

4.2 Data Sources Perspective

The creation of the model commenced targeting fifteen data sources, identified in prior work as key categories of information which are pertinent to online monitoring for nuclear day-to-day operations (Al Rashdan and St Germain 2018). These data sources are not inclusive of all data sets in an NPP (because they are focused on online monitoring applications), but describe the majority of the data sources and were deemed as a good reference for developing DIAMOND. The data sources of Al Rashdan and St Germain (2018) include:

- Process Instruments and Control
- Maintenance
- Equipment Performance Testing
- Calibration
- Operator Rounds
- Radiation Protection
- Security
- Condition Reporting System Data
- Work Orders
- System Engineer's Notebooks
- Schedule
- Logistics
- Clearance Orders
- Vendor and Plant Documentation
- Industry Operating Experience.

The details of each source definition can be found in Al Rashdan and St Germain (2018). Before describing the data sources perspective, it is necessary to describe how profiles can be used to group data in DIAMOND.

4.2.1 Profile

It's often useful to split an ontology into smaller sections of grouped classes and properties that have some common characteristics. This is referred to as a profile. In the context of DIAMOND, a profile is used to group data based on their source, such as operator rounds, without having to send and parse through the entire ontology. This new subset can then be used for passing data related to that profile. Because the class and property details of the subset are consistent with the main ontology, the subset data is completely interchangeable.

Profiles were applied to DIAMOND using a tool called ROBOT (Jackson et al. 2019). ROBOT was created specifically for use with ontologies in the Open Biological and Biomedical Ontologies Foundry, but can work with any OWL ontology file[a]. A filter command can be used to create a new ontology that is a subset of DIAMOND. In order to specify the classes that should be included in a profile, a profile must first be defined within the ontology's annotation properties under the profile_property. This annotation property may then be applied to any desired class by choosing to add an annotation to that class[b].

A profile was created for each of the data sources in Section 4.2. The classes, contained within a profile, can easily be viewed by using Protégé's find command and entering the name of the profile, or by going to the Annotation Properties view and selecting the desired profile and the Usage tab within the profile's description. The following sections provide some insight into the classification of key model elements into each profile, or data source, along with unique challenges faced. As described in Section 1,

[a] ROBOT may be utilized by downloading the jar file (Jackson et al. 2019) and following the instructions listed there under Getting Started

[b] This is done in Protégé by selecting the profile_property annotation, and then from the entity Internationalized Resource Identifier menu selecting Annotation Properties and the desired property.

the level of information provided in the following section is to illustrate examples of the developed entities in DIAMOND and are not inclusive of all classes and properties of DIAMOND.

4.2.2 Process Instruments and Control

The process instruments and control data incorporated primarily the physical equipment related to the input (controls) and output (instruments) of processes within a nuclear plant. An initial step in determining what classes to include and how to add them in DIAMOND was determining the categorization of instruments and controls. The delineation between the classes component and equipment was critical in making this decision. Due to a need for unique identifiers and other attributes within the equipment class, the instruments class was placed beneath equipment, and has its own children such as chart logger and gauge. Looking at the example of the gauge class, much of the required information for the class was already inherited through the equipment class. This includes details around the manufacturer of the instrument, the status and safety classification of the equipment, as well as location, configuration, etc. After validating the information received through inheritance, the task remained to determine what further attributes were needed for this class. The data of primary concern here are related to the current value of a gauge, as well as minimum and maximum values within some acceptable threshold. Attributes like these were created as data properties (because they did not already exist) and were added to the gauge class.

Defining relationships in process instruments and control was also performed. For example, decomposed by relationships exist between a panel class (beneath equipment) and the instruments, controls, and display classes. Rather than attempting to place all of the information that might be found for a given panel and all of its parts into a single entity, it is vital that classes be created at a sufficiently granular level. Relationships can then be used to connect information together. In this example, the class instrument has multiple children, and the relationship from panel for any given panel may target one or more of these classes. Creating the relationship to the parent-class, instrument, ensures that the scope of the relationship is kept to the instrument and its subclasses, but prevents the need for a relationship to each potential type of instrument.

4.2.3 Maintenance

The maintenance data incorporated in DIAMOND contains documents, roles, and actions related to the maintenance process in an NPP. The data also cover the physical equipment involved, such as the tools used for some maintenance activity. It also includes the various documents and reviews related to these activities.

A maintenance plan is the document that holds the various details around the maintenance activities to be performed in an NPP. This class inherited the core document attributes from the document class (file name, file size, who created the document and when, etc.) and then required additional attributes be created and applied to the class to cover the remaining information necessary. A maintenance plan is primarily a list of maintenance items; therefore, a relationship was created to another class, maintenance item, which is a child of data item class. As discussed in Section 4.2.2, this relationship between a document and the repeated items within it allows for maximum data modularity and ease of access to data. The maintenance item class contains descriptive attributes about the work to be performed.

Maintenance-related activities should be performed by a person or organization with the maintenance role, a role class that allows for grouping properties and relationships unique to an entity involved in maintenance. Unique attributes to the class include properties such as a relation to the personal protective equipment (PPE) class, which acts as a parent to the various types of PPE which might be used by a maintenance worker. Relationships were also created between a maintenance-inspection action and maintenance plan and review, linking a maintenance action with the necessary documents.

When maintenance occurs on some asset, the data for that asset can be retrieved by a simple relationship that exists between the maintenance document classes (plan, item, and review) and the asset class. This allows the ontology to easily connect the process and text-focused classes with the asset on which work is actually being performed.

4.2.4 Equipment Performance Testing

Equipment performance testing includes the documented results of tests as well as any pertinent information related to the act of testing. Testing includes the act of ensuring some functionality of an asset, along with the measurements taken, process involved, conditions observed, future actions deemed necessary, etc. Equipment performance testing was incorporated into the ontology with a relatively small number of classes. With the established pattern of linking together assets, documents, and actions (seen above in maintenance), it was a straightforward process to add the necessary classes for the testing data. For example, the performance test document class is a descendant of document and contains attributes such as a threshold, minimum and maximum states, a measured value, and a reference to the asset on which testing occurred. A subclass of action, testing, allows for holding properties unique to that type of action. These classes are connected through a relationship and allow for reference of the performance test document and its contents as the action of performing the testing occurs.

The performance test document class is a subclass of test document and a child of document. Additional attributes to the document class that would be specific to test document classes were added. These attributes include the measurement taken and the unit of measurement used in a given test. This test document class can then be used to create subclasses that inherit these properties, in addition to the properties inherited from the document class, for various types of tests. Other test classes exist that are able to take advantage of the document and test document classes as well. These include operability test document and post-maintenance test document. The proper use of inheritance and use of other relationships is clearly key to a cohesive and understandable structure.

4.2.5 Calibration

The calibration data contain classes related to the process of calibrating equipment and related measurements. Measurements should generally be taken before, during, and after the process. The performance of calibration could be by some person, but it could also be performed by a tool or self-calibrating device. This relationship between calibration and various performers utilizes the BFO role class so that the role can be applied to whatever entity is performing the process.

The act of calibration should produce some calibration report. Beyond a reference relationship to some asset, the calibration report contains a number of attributes, including the action taken, the allowable range of measurements for the asset on which calibration is occurring, the various measurements taken, and information about who performed the calibration. The action taken as part of the calibration process, future actions required, and dates and measurements around the calibration process were all necessary to represent. The calibration process may create a type of profile for a given asset, listing historical measurements and performance for that asset. This profile document class was placed under document and makes use of the time entry class for storing historical measurements.

Sensors such as a thermocouple, infrared camera, and others can play an important part in the calibration process. Therefore, the sensor class was also included in this calibration data. Questions such as whether a camera is a sensor or some other type of equipment were brought up frequently throughout the development process. Having clear definitions for classes was essential so that classes such as these could be properly grouped under the correct parent class. For example, after consideration, the definitions analyzed indicated that the sensor class was the best fit as parent for the various types of camera classes within DIAMOND.

4.2.6 Operator Rounds

The operator-rounds data contain the various entities related to the performance and data associated with operator rounds within a nuclear plant. Similar to a maintenance plan, an operator round relies on a core document with step by step instructions to be performed on one or more assets. The performer of an operator round will follow these instructions, which generally include actions such as visiting certain assets within the plant and ensuring proper functionality and that no indications of potential future errors exist. To accomplish this, various information-entity classes were created. An operator-round document that inherits from the document class contains information like when the next round should occur as well as a decomposition relationship to the operator-round step item class. This class inherits from data item and contains the instructions and related information for each individual step on the operator-round document. Attributes like any recorded values, attachments (such as pictures), and the user who performed the step and the time of performance are all included in the operator-round step item class. The items share a common format, and would therefore be stored as a certain type of data structure that is referenced by the parent operator-round document class.

Another class, called operator-round entry, exists as a sibling to operator-round step item, and allows for the storage of information related to the actual performance of some step item. Attributes such as the sequence number of the item, instructions, reading type, etc., are found in this class. A relationship also exists between operator-round step item and the action class, as an operator round includes both the information entities that hold the majority of the information related to the operator-round process as well as the defined actions and transitions that would make up the actual performance of the operator round.

Operator-round approval item is another data-item class that decomposes operator-round document, thereby allowing users to associate an individual approval with the whole round, individual steps, or not require an approval at all. This class includes a Boolean flag, indicating whether approval is required, a description and status for the approval, and, of course, attributes for who performed the approval and when.

4.2.7 Radiation Protection

Radiation protection includes the various assets related to recording workers' dose within an NPP, along with the documentation associated with these risks. Documents such as a radiological work permit and records of estimated and actual dose were added. A contamination survey, which contains attributes around the radiation exposure of some area or asset within the plant, is also included in radiation protection. These documents are particularly concerned with the locations within an NPP and necessitated a clear structure for the representation of these areas, indicated by entities such as elevation, areas, and zones under the BFO spatial-region class and its children classes (grouped by number of dimensions and explained in Section 4.1).

In addition to the many properties of the document classes, radiation protection also involved adding physical-equipment classes under the PPE that might be used within the NPP, as well as a representation for instruments like a dosimeter. Sensor was the set as the parent class for the dosimeter class, and children classes of dosimeter include certain versions such as electronic personal dosimeter, film badge dosimeter, etc.

Roles also came into play for this data source. The shielding-material role, related to the ability of some material to reduce radiation exposure, was created after consideration that a role would be the best approach to use for any subclass of material or other asset which may serve this purpose within an NPP. For example, in one instance the material concrete may perform this role and need to make use of the properties within this role.

4.2.8 Security

The security data include the representation of physical aspects of security such as doors, access zones, and security badges. Locks, metal detectors, and other physical-entity classes were also created. The inheritance of properties from the equipment or component classes allowed for quick and easy additions, as many of the attributes these classes needed were already provided through inheritance. For classes such as security badge, relationships were critical in connecting the class to the other properties it would need to reference in order to be effective. This includes relationships like an association to the access-zone class in order to determine from a security-badge object what zones it may access. Additionally, a relationship exists to the person class, so that all of the details around the person assigned the security badge can stay with the person object, but be referenced by the security-badge class for use. The person class holds attributes that are unique to a person, and preferably not associated with a role that the person might assume or organization of which they are a part. These attributes include name values, address and contact information, and a relationship to other entities with which a person may be associated, such as department.

A variety of document classes as well as actions exist for security. Some of the key documents of concern include security plans and security policies. These classes contain details around the various locations and assets within the plant and the needs for performing security tours or rounds to validate the status of these entities, as well as the various requirements related to the security of an NPP, many of which come through various forms of regulation. Security officer role is a child of personnel role class that may be assumed by any person who performs this duty. Additionally, a security role class also exists under organizational role so that security departments or organizations within an NPP can inherit this group of attributes.

4.2.9 Condition Reporting System Data

The condition-reporting-system data include the condition-report document. A condition report documents any observed condition that needs to be reported, often relating to an asset’s condition or performance, but may also apply to procedures and processes. Condition reports may be submitted by individuals within the plant when some deviation from normal behavior or state is observed.

Condition reports also track whether a condition has been resolved and trend codes to help staff understand long-term trends in reported conditions. Many of the requirements for this class were solvable through relationships to other classes like asset, system, and process. Additionally, the location for some asset for which a condition report is being provided was important. The report makes use of the object property “located at” to create a relationship to entities such as area, building, elevation, and room. Additional data properties were added, such as classification code, condition significance, and immediate action taken.

4.2.10 Work Orders

A work order is generated by a work request. A work request is created when it is deemed necessary for work to be performed on some asset. The generated work order contains the information on the work to be performed and references other classes, including a work procedure, which is the primary source for the instructions of work to be performed as well as who is responsible for the work. The work-order class includes attributes related to the asset on which work was performed, the location, performance information, when the work was performed, the status and results of the work, etc.

A work order is a process that occurs within an NPP and is therefore reliant on a structure for process modeling. Other processes include scheduling, security tours, surveys, etc., some of which have already been discussed in previous data sources. In order to fully understand how such process is generally represented, research was performed on standards such as the UML, Systems Modeling Language (SysML) (Balmelli 2007), and Business Process Modeling Notation (Grosskopf et al. 2009). Additionally, a previous effort targeting the work process in Al Rashdan and Agarwal 2019 was studied. Although modeling processes and activities in an ontology are limited by the types of content within the ontology (classes, relationships, and properties), an approach was developed that was found sufficient for modeling the desired processes at this stage. This solution incorporates the use of the LML action class along with two subclasses: planned and actual action. Additionally, a pair of classes called planned and actual transition hold information related to the process transition. The details of this approach were discussed earlier in Section 4.1.1. The approach will be validated in future work.

4.2.11 System Engineer’s Notebooks

Engineer notebooks may be realized within various nuclear plants in different ways, including the use of manual paper processes, spreadsheet software, or some other digital tool. Regardless of the various engineer notebook states and methodologies within the plant, the ontology contains a common structure that can be used for digital storage and exchange of these data. Similar to the idea of components and equipment within physical assets, data structures were necessary both at a document level and for the repeated items within an engineer’s notebook. When measurements or recordings are taken by an engineer, these values are sent to their most appropriate locations, either objects associated directly to an asset or some other document created for that type of data. However, notes taken by an engineer that do not match some existing data source need a structure to use. The engineer notebook class was created as a child of the document class, and uses the common pattern of referencing another class for repeated use. The class referenced is the note class, which allows for storing notes and logging information related to who created the note and when.

A relationship to the quality class was added to the engineer notebook class for linking the desired measurements with those within quality—for example, the engineer is recording a measurement, such as voltage. Relationships from the engineer notebook to the artifact and database classes allow for connecting this class with other documents or drawings which might be referenced or used within the engineer notebook.

4.2.12 Schedule

The schedule data are composed of the action and transition classes (discussed in Section 4.1.1). The action class may be thought of as equivalent to the idea of an activity, a term commonly used in various scheduling tools. The activity structure is the heart of any schedule. It includes information on who is related to or performing the activity, what activity may have occurred previously, and what activity is coming next, as well as the various time information related to the start, duration, and end of the activity. Within DIAMOND, an action may be decomposed by other actions; thus, the action class constitutes the objects for a whole activity such as “repair pump,” as well as the individual steps within that activity like “bolt a nut.” There are a number of relationships surrounding these action and transition classes.[c]

A schedule follows the same format of using action and its subclasses along with planned-transition and actual-transition classes (previously discussed). Planned actions generate planned transitions which, in turn, generate other planned actions, and these classes are used for the creation of some schedule. When the schedule is being executed or has been executed, the actual-action and actual-transition classes may be utilized to hold details around the performance of the individual execution of that schedule.

A schedule can also make use of the time-entry class for historical needs, as well as a scheduler role (to be used across persons or organizations who fill this function) and document-centered classes containing the information for some project within the plant. This includes classes like project overview and project issue, which hold additional details around a project such as budget and issues or problems encountered.

4.2.13 Logistics

The logistics data contain classes and properties associated with inventory, purchase orders, etc. Logistics data involve a large number of document-centered classes. As a result, properly representing logistics data and ensuring correct placement of these classes and relationships between them is demanding. For example, a request for quote is a document class. In addition to requiring some high-level properties around the document itself, such as due date, the primary information within the document is found in the class reference to the “request for line item” class. Beyond the structure of the documents themselves, the relationships between them should also be modeled. In this example, a request for quote

[c] It can be useful to view the Usage tab within Protégé when viewing a class or property to see what other entities are using it.

could generate a purchase order. The purchase order, in turn, should be referenced by the “incoming receipt” class. This chain of relationships between classes required significant attention and consideration.

The inventory of the plant is also represented in logistics data. Examples of these classes include the inventory-stock record class, which contains a warehouse code, stock code, and the quantity of that stock code. Inventory bin item and inventory item record are other classes which work to structure the inventory of the NPP correctly.

4.2.14 Clearance Orders

Clearance orders data have to do with the creation and execution of clearance orders, including tagging in and out equipment. This includes classes for approval, temporarily removing an equipment from service, or restricting access to assets before work is performed. When a tag is created, information such as what assets are affected, who will perform the work, and the dates of the work are entered.

When a physical tag is printed, to be placed in the proper area to mark an asset as “tagged out”, it is represented by classes for both the clearance-tag document as well as a type of physical asset that is placed within the plant, which is represented through the sign class, a child of equipment with a relation to the clearance-tag label. This sign class has attributes such as the size of the sign, location, and reference to the label which contains the information that will be presented on the sign. Before a clearance tag can be created and work performed, a clearance order must be created that dictates what work is to be performed. This class exists under document and contains, in addition to unique properties around the planned scope of work, references to the clearance tags that will be created and the equipment which is affected by a related clearance process. The clearance process could be realized through the tag in or tag out process entities.

4.2.15 Vendor and Plant Documentation

The vendor and plant documentation involved several trivial classes, such as plant drawings, manuals, etc. It was determined that a document management system class was necessary to keep track of documents or drawings. This class represents a database and is, therefore, a child of the database class. This class holds basic information related to the database and a relationship to the information-entity class, allowing it to reference any classes (the documents and drawings which the system will track) within it. Properties such as the edition or revision of some document were necessary for several of these classes. A document status-change record is referenced by the plant documentation class, which allows for easy tracking of changes to some documentation. Document number, name, status, description, user, and date are all included fields in this document status-change record class. Additionally, a document type class allows for the classification of a document which can be used to define metadata properties for the document.

4.2.16 Industry Operating Experience

The industry operating-experience data are intended to capture data related to information shared across the nuclear industry or within an individual plant. Large amounts of this documentation stem from regulation requirements around the need to share certain types of events that occur at an NPP. Classes were designed around the types of entities and organizations by which operating-experience events are classified. Event notifications and licensee event-report classes were created as part of this effort. Attributes, including descriptions, flags, dates, identification numbers, and references to other documents, make up a part of the properties of these classes. While regulation and proprietary needs may dictate what information is shared, industry operating experience should expand to provide sufficient structure for detailing the various events that can be used to help the industry progress and learn through sharing knowledge across plants and organizations.

5. SUMMARY AND FUTURE WORK

This work created the seed model and ontology to enable seamless data integration for the nuclear industry. The result is DIAMOND. This report described the survey of literature performed and the selected models and ontologies that were incorporated into DIAMOND. Three key elements were identified and used: classes, object properties (relationships), and data attributes. These elements were incorporated into an OWL file using an open-source tool called Protégé. Annotations were used for creating data profiles. The development process of DIAMOND coupled a model perspective, which structured the data in a hierarchical tree, and a data-source perspective, which populated the structure according to actual NPP data sets. Key examples of DIAMOND elements were described in this report with the aim to illustrate the development process and rationale behind it. The DIAMOND OWL file is planned for open-access release in 2020 and includes the details of the developed model. The scope of future work is:

- Increase the fidelity of DIAMOND in collaboration with multiple industrial entities. This will be through engaging industry partners (both commercial and non-commercial) to solicit feedback and contribute directly to the model. This will also be performed through targeted data-sources analysis and use cases.
- Incorporate unstructured data into DIAMOND. Due to the outdated model of operations of NPPs in the United States, unstructured data are common data forms at NPPs and need to be incorporated.
- Create an interactive development environment for open access to enable the industry to collaboratively develop the model. This is needed to ensure that any modification to DIAMOND is traceable, verifiable, and undoable if it is foreseen to have negative effects.
- Expand the scope of DIAMOND to include the design aspect of reactors in collaboration with the DOE Versatile Test Reactor program, the manufacturing aspect in collaboration with DOE Transformational Challenge Reactor program, and other applications, as deemed necessary to benefit the nuclear industry.

6. REFERENCES

Al Rashdan, A. and Agarwal, V. (2019). A Data Model for Nuclear Power Plant Work Packages. *Nuclear Technology, Special section on Big Data for Nuclear Power Plants, Vol.* 205, pp. 1053-1061.

Al Rashdan, A.Y. and St Germain, S.W. (2018). *Automation of Data Collection Methods for Online Monitoring of Nuclear Power Plants. INL/EXT-18-51456,* Rev 0. Idaho National Lab. (INL), Idaho Falls, ID (United States).

Al Rashdan, A.Y., Krome, C.J., St Germain, S.W. and Rosenlof, J. (2019). *Method and Application of Data Integration at a Nuclear Power Plant. INL/EXT-19-54294,* Rev 0. Idaho National Lab. (INL), Idaho Falls, ID (United States).

Arp, R., Smith, B. and Spear, A. (2015). *Building Ontologies with Basic Formal Ontology.* Mit Press, 2015.

Balmelli, L. (2007). An overview of the systems modeling language for products and systems development. *Journal of Object Technology, Vol.* 6(6), pp.149-177.

Basic Formal Ontology Team (2019). Basic Formal Ontology (BFO). Retrieved August 2019 from: https://github.com/BFO-ontology/BFO.

Bennett, S. (2008). How is SOI Different from Traditional Integration? Retrieved August 2019 from https://s-bennett.com/2008/05/20/how-is-soi-different-from-traditional-integration/.

Cassidy, P. (2010). Overview of the COSMO Ontology Project. Retrieved August 30, 2019 from http://www.micra.com/COSMO/COSMOoverview.doc.

Department of Defense (2010). The DoDAF Architecture Framework. Version 2.0, U.S. DoD Deputy Chief Information Officer, Washington, DC. Retrieved from: https://dodcio.defense.gov/Portals/0/Documents/DODAF/DoDAF_v2-02_web.pdf.

EPRI (2015). *Common Information Model Primer*. Electrical Power Research Institute, 3002006001.

Falconer, S. (2013). OntoGraf. Retrieved August 2019 from: https://protegewiki.stanford.edu/wiki/OntoGraf.

Grosskopf, A., Decker, G. and Weske, M., (2009). *The process: business process modeling using BPMN*. Meghan Kiffer Press.

Guizzardi, G., and Wagner, G. (2004). A Unified Foundational Ontology and some Applications of it in Business Modeling. In *CAiSE Workshops, Vol.*3, pp. 129-143.

Herre, H., Heller, B., Burek, P. Hoehndorf, R., Loebe, F. and Michalek, H. (2007). General Formal Ontology (GFO): A Foundational Ontology Integrating Objects and Processes. Part I: Basic Principles. *Research Group Ontologies in Medicine (Onto-Med)*, University of Leipzig. Latest intermediate revision: Version 1.0.1, Draft, 14.02.2007.

IDEAS Group (2011). International defense enterprise architecture specification.

IEC (2018). *Framework for energy market communications–Part 301: Common information model (CIM) extensions for markets*. IEC 62325: 301:2018

ISO (2016). *Petroleum, Petrochemical and Natural Gas Industries – Collection and Exchange of Reliability and Maintenance Data for Equipment*, ISO 14224:2016.

ISO (2018). *Industrial Automation Systems and Integration - Integration of Life-Cycle Data for Process Plants Including Oil and Gas Production Facilities - Part 13: Integrated Asset Planning Life-Cycle*, ISO 15926-13:2018.

Jackson, R.C., Balhoff, J.P., Douglass, E., Harris, N.L., Mungall, C.J. and Overton, J.A. (2019). ROBOT: A tool for automating ontology workflows. *BMC Bioinformatics*, *Vol.* 20 (1), pp. 407.

Lexico (2019). Ontology. Retrieved August 2019 from: https://en.oxforddictionaries.com/definition/ontology.

Lifecyclemodeling (2017). Lifecycle Modeling Language (LML) Specification. Retrieved August 2019 from http://www.lifecyclemodeling.org/spec/LML_Specification_1_1.pdf.

Mascardi, V., Cordì, V. and Rosso, P. (2007). A Comparison of Upper Ontologies. In *Woa*, *Vol.* 2007, pp. 55-64.

Masolo, C., S. Borgo, S., A. Gangemi, S.A Guarino, N. and Oltramari, A. (2003). Wonderweb deliverable d18, ontology library (final). *ICT project*, 33052, pp.31.

McMorran, A.W. (2007). *An Introduction to IEC 61970-301 & 61968-11: The Common Information Model*.

Mizoguchi, R. (2010). YAMATO: yet another more advanced top-level ontology. *Proceedings of the Sixth Australasian Ontology Workshop*, pp. 1-16.

Niles, I., and Pease, A. (2001). Towards a standard upper ontology. *Proceedings of the International Conference on Formal Ontology in Information Systems, Vol.* 2001, pp. 2-9. ACM.

OBO Technical WG (2019). Information Artifact Ontology. Retrieved August 2019 from: http://www.obofoundry.org/ontology/iao.html.

Cesare, S. and Partridge, C. (2016). BORO as a Foundation to Enterprise Ontology. *Journal of Information Systems, Vol.* 30 (2), pp. 83-112.

Semantic Arts (2019), Gist, Retrieved August 2019 from: https://www.semanticarts.com/gist/.

Sinclair, P., Addis, M., Choi, F., Doerr, M., Lewis, P. and Martinez, K. (2006). The use of CRM core in multimedia annotation. Retrieved August 2019 from: http://www.cidoc-crm.org/Resources/the-use-of-crm-core-in-multimedia-annotation.

Smith, B. (2019). BFO Basic Formal Ontology. Retrieved August 2019 from: https://basic-formal-ontology.org/users.html.

Stanford University (2019). *PROTÉGÉ*. Retrieved August 2019 from: https://protege.stanford.edu/.

Svátek, V., and Šváb-Zamazal, O. (2010). Entity naming in semantic web ontologies: Design patterns and empirical observations. Retrieved August 2019 from: https://patomat.vse.cz/znal10fi2.pdf.

Terziev, I., Kiryakov, A., and Manov, D. (2005). Base Upper-level Ontology (BULO) Guidance. Deliverable 1.8.1, SEKT project. Retrieved August 2019 from: (https://www.ontotext.com/documents/proton/Proton-Ver3.0B.pdf).

Tzitzikas, Y., Allocca, C., Bekiari, C., Marketakis, Y., Fafalios, P., Doerr, M., Minadakis, N., Patkos, T. and Candela, L., (2013). Integrating heterogeneous and distributed information about marine species through a top level ontology. In *Research conference on metadata and semantic research*, pp. 289-301.

UMBEL (2016). Upper Mapping and Binding Exchange Layer (UMBEL) Specification. Retrieved August 2019 from: http://techwiki.umbel.org/index.php/UMBEL_Specification#Attribution.

W3C (2012a). OWL 2. Web Ontology Language Document Overview. (second edition). Retrieved August 2019 from: https://www.w3.org/TR/2012/REC-owl2-overview-20121211/.

W3C (2012b). W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures. Retrieved August 2019 from: https://www.w3.org/TR/xmlschema11-1/.

W3C (2013). Web Ontology Language (OWL). Retrieved August 2019 from: https://www.w3.org/OWL/.