DETECTING COMMONALITIES IN ASSET MANAGEMENT BUDGET JUSTIFICATIONS

David Lanter, Temple University & CDM Smith

ABSTRACT

This paper explains how an ontology-based object-oriented processing application and database system can be implemented to process asset management work order budget justifications to detect and analyze patterns of commonality of their business cases. Central to this effort is incremental creation of an asset management budget justification ontology, followed by implementation as a global object-oriented data model that supports comparative analysis of budget justifications for individual asset management work orders. Focused on the Operations & Maintenance (O&M) business function, this research suggests a way to achieve a unified standard language and suite of intelligent tools for helping businesses analyze and understand justifications for asset management work order investments within and among an organization's business functions.

ONTOLOGY

In philosophy, ontology is the study of common characteristics of all things with the intent of providing an explanation of the universe through a comprehensive categorization based on their similarities and differences. In artificial intelligence, an ontology is a data structure that organizes knowledge as a semantic network of terminology expressing relationships among concepts conveying meaning with standardized vocabulary for a specialized field. The ontology is used in natural language processing research to guide automatic unsupervised parsing and transformation of text into a data representation of meaning.

To be useful, however, the representation of meaning must enable automated reasoning capabilities that help solve practical problems. The information inherent in asset management work order justifications can be used to build an ontology-based information system (Figure 0) consisting of:

1. A semi-automated parser able to extract meaning from work order justification texts
2. A knowledge representor that transforms and structures this meaning to improve the ontology and persist the meaning into a corresponding object-oriented database
3. A semantic pattern detection and visualization application that queries and processes the work order justification database to find and visualizes salient commonalities and differences that aid understanding.

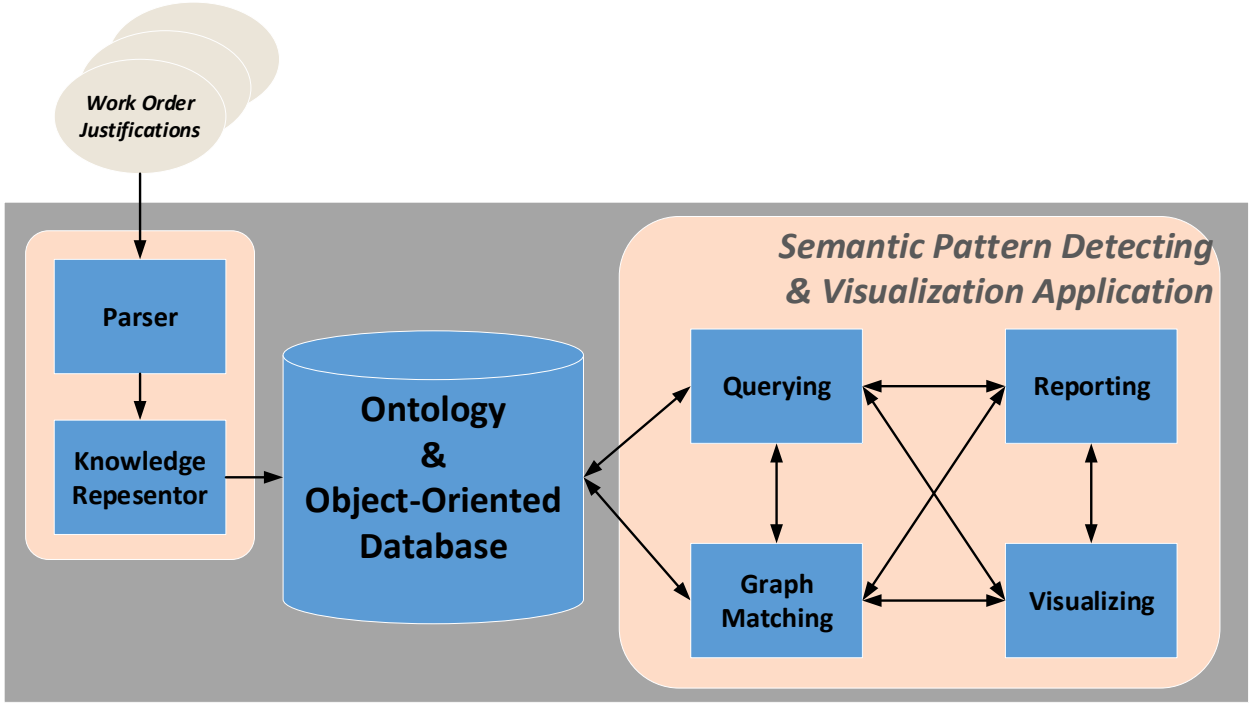

Figure 0. Ontology-based information system.

An ontology is created by consensus to provide an organization with an agreed on formal specification that facilitates communication, reasoning, and sharing knowledge. The organization's commitment to an ontology provides a shared vocabulary used in a consistent manner that assures meaningful communication. The resulting ontology supports development of an enterprise data model, enables automated reasoning and visualizations that provide insights into the business cases expressed in work order justifications and their alignment with business line and organizational objectives, and a way to automate information sharing among networked computers.

TOWARDS A WORK ORDER JUSTIFICATION ONTOLOGY

In an ideal world, an organization would be able to look and find and reuse a previously crafted off-the-shelf ontology (Nguyen, V. 2011, p.36) for automating parsing justification text and reasoning about and detecting commonalities among of the work order budget justifications. An ontology, however, is designed on purpose for a certain application running in a specific context. This makes existing ontologies created by others possibly useful to those who lack knowledge of a subject area. The generic concepts and language expressed in the ontologies of others, however, are not likely well matched to a well-established business' concepts, terminology, values and objectives of each of its business areas.

Use of an externally defined ontology in an automated parser would result in processing that results in failed determinations of valid commonalities in work order justifications (i.e. false positives and false negatives). This would be due to ontological mismatch between the language and concepts encoded within the off-the-shelf ontology and the meaning expressed in the justification written by the organization's work order justification author. That is, mismatch would result from use of different terms representing the same concept and the same terms for different concepts, and concepts expressed in the justification completely missing from the off-the-shelf ontology.

For example, many of the fourteen published ontologies analyzed by Sinha and Dutta (2020) included semantic representations of environmental contexts (i.e. water bodies, watersheds, flood plains, and floods), instruments and sensors. While one ontology of the fourteen included electrical assets, none of the other ontologies' examples included the wide-range of flood control assets which are central to the Operations and Management (O&M) work order budget justifications of flood management organizations.

Large well-established businesses in a particular infrastructure sector (CISA 2020) have likely developed, improved and evolved their own shared conceptualizations, terminology, objectives and values over decades (and in a few cases over centuries) as their experts established the domain of infrastructure asset management of their sector. These conceptualizations and terminology are expressed in work order justifications of asset management expenditures. They can be formalized within a work order justification ontology. This would address problems of synonyms, homonyms and omissions in terminology. It would also provide a basis for developing automated reasoning capabilities able to identify similarities and patterns (Reul, Q.H. 2012) in justifications written by organizational staff which often are found in subsumed (i.e. subset) and disjoint relationships among concepts expressed in specialized language of O&M work order justifications. An O&M work order ontology will also provide a basis for comparison and learning from the ontologies of other critical infrastructure asset management experts and researchers.

This research is focused on the design of a prototype O&M work order budget justification ontology and prototype computer application capabilities for analyzing, finding, visualizing and aiding explanation and understanding of commonalities and differences among O&M work order justifications. The resulting ontology, parser, and corresponding object-oriented analytical application and database would enable insights into the state of an organization's infrastructure, as well as detecting and removing redundancies, conflicts and inefficiencies; and improving coordination and synergies among related work orders and the larger projects they may be part of.

ONTOLOGY DESIGN

Abstraction is the principle tool of ontology design. The designer selects important characteristics and properties of objects and ignores others determined to be not relevant. This cognitive process helps the designer understand, classify and model reality. Ontology design utilizes three non-overlapping primitive abstraction mechanisms: classification, aggregation, and generalization.

Classification is used to define a single concept as a class of real-world objects characterized by common properties. It makes explicit the IS-MEMBER-OF semantic relationship existing among a set of similar objects.

Aggregation is used to define a new class from a set of other classes that represent its constituent parts. It makes explicit the WHOLE-PART or IS-PART-OF relationship between a composite object and the set of its required and optional component objects. The IS-PART-OF relationship is a logical "AND" relationship between the composite object and its component parts.

Generalization is used to define a subset relationship between elements of two or more classes. It makes the IS-A relationship between a superclass (i.e. generalization) and its subset classes (i.e. specializations). The IS-A relationship is a logical "OR" relationship between the instance of the superclass and the alternative specializations. The properties defining the characteristics of the generic superclass are inherited by all the specialized subclasses.

Building a machine processible ontology for analytical reasoning about work order justifications is a gradual process of building and integrating conceptualizations of infrastructure assets and their business cases and lifecycles. This activity enables the designer's perception of the reality of the informational domain of each business-line is progressively enriched and refined as the ontology is incrementally developed. This process is typically organized using strategies based on structured transformations. 'Top-down' and 'bottom-up' are basic strategies which are used within the more pragmatic and efficient 'mixed' approach. The top-down strategy refines abstract concepts into more concrete ones. Bottom-up, in contrast, creates abstract classifications, aggregations, and generalizations from concrete classes of objects.

Top-down Design

Top-down design produces a final ontology through structured decomposition. That is, a single concept is transformed into a more detailed specification. Systematic application of the top-down design primitives adds new details to the original conceptualization. The design process ends when all asset life-cycle and business-line work-order justification concepts have been included as details in the resulting ontology. All concepts are present at each step in a pure top-down approach. That is, no new abstract concepts are considered. Only additional detail is added to the starting concepts in the ontology. This is only possible, however, for a designer possessing an a priori high-level understanding of the asset life-cycle and business-line work-order justification concepts.

The top-down design primitives each apply to a single concept and produce a more detailed description. The following primitives are used to transform a starting ontology into a more detailed one:

- Refine a class of objects into a relationship between two more classes
- Refine a class into a generalization hierarchy or a subset
- Split a class into a set of independent classes. The effect of this primitive is to introduce new classes, not to establish relationships or generalizations among them
- Refine a relationship into two (or more) relationships among the same classes
- Refine a relationship into a path of classes and relationships. Applying this primitive corresponds to recognizing that a relationship between two concepts should be expressed via a third concept, hidden in the previous representation
- Refine a class by introducing properties
- Refine a class by introducing a composite property
- Refine a simple property either into a composite property or into a group of properties

Bottom-up design

Bottom up design introduces conceptualization on the application domain that were not captured at any level of abstraction by the previous version of the ontology. This approach is the reverse to that of top-down design. The bottom-up primitives introduce new concepts and properties previously missing from the ontology. This approach is a simple one where one part of the problem is attacked at a time. It results in a sequence of gross restructuring as complex intermediate design ontologies are integrated. The bottom-up primitives are applied to a set of elementary concepts to combine them and build up more complex ones. The bottom-up primitives are:

- Generate a new class to represent a concept with specific properties not expressed in the ontology
- Generate a new relationship between existing classes
- Create a new object that is a generalization for previously defined classes

- Generate a new property and incorporate it in a previously defined class
- Create a composite property and incorporate it into a previously defined class

Mixed Design Strategy

The mixed design strategy is useful when the concepts forming the domain are complex. It makes use of the best parts of both the top-down and bottom-up approaches. Mixed design begins with a top down approach to produce a skeleton ontology. Based on an understanding of basic relationships among important concepts expressed in the information domain gained from prior work, the designer begins partitioning the concepts into subsets. The partitioned concepts are examined and the most important are placed within the skeleton ontology and systematically linked to integrate the concepts within the partitions. The utility of the resulting skeleton ontology is in its expression of the underlying structure and organization of the informational domain. A well-designed skeleton ontology serves as a core providing ease to subsequent elaboration and evolution of the ontology through bottom-up integration of new conceptualizations.

The resulting ontology provides semantic structure for understanding concepts of the domain. It can also support development of an input parser able to recognize and extract the expression of the concepts in texts. The ontology can serve as a model to follow for implementing a corresponding object-oriented schema for an application to use in processing instance data extracted by the parser, persisting within a database, and analyze to solve problems including detecting and analyzing commonalities and differences and identifying and helping explain patterns within work order justifications.

ONTOLOGY FOR O&M WORK ORDER JUSTIFICATIONS

The work order object types and relationships presented in this paper represent a preliminary working hypothesis of concepts and terminology central to O&M work order justifications. A skeleton ontology for the O&M work order justifications may be formulated from the following general conceptualization:

A justification for a work order's budget typically includes a description of a ***Solution*** to resolve an ***Issue(s)*** affecting a infrastructure ***Asset(s)*** and characterized by a ***Symptom(s)*** that is needed to mitigate or facilitate a negative or positive ***Impact(s)***.

The elements structured within an ontology are referred to as subjects or objects and relationships. A classification of objects typically expressed in an O&M work order justification include: Asset(s), Component(s), Issue(s), Symptom(s) Impact(s), and Solution(s). These are defined in Figure 1.

Asset	An infrastructure asset
Component	A component (or subunit) of an asset
Issue	Condition affecting performance of an asset. (If the affect on performance is negative, the issue is either a defect or deficiency. If the affect is positive, however, the issue may be an opportunity to improve performance of the asset.)
Symptom	A physical indication of a condition of defect or deficiency, or opportunity for improvement
Impact	A negative consequence likely to happen as a result of the issue's effect on an asset, or a positive consequence (benefit) likely to happen as a result of resolving the issue
Solution	Solution proposed to resolve a asset issue and improve the asset's performance

Figure 1. Example Subjects and their definitions.

OBJECT-ORIENTED IMPLEMENTATION

The subjects/objects expressed in the relationship predicates of the ontology developed for parsing the text of O&M work order justifications can also be expressed as an object-oriented data model (i.e. schema). This schema can be implemented to:

1. Store work order justification instance data as objects in working memory of a computer application to support inferencing, analysis and display at runtime
2. Store work order justification instance data as objects in a database to persist for reuse within the application at later times

Object-oriented applications and databases provide mechanisms for efficiently creating, processing, and querying composite-component and generalization-specialization relationships.

Composite-Component Relationships

In aggregation a "whole-part" or "a-part-of" association is established using a logical AND relationship between a composite object and its component objects. Aggregation supports recursive transitive composite-component relationships among constituent objects. That is, a composite work order justification object has its component objects, each of which may in in turn have its own component objects. Anyone of these components may be composites with their own components, and so on.

The semantics of composite reference are further refined on basis of whether an object is part of only one object or more than one object. This leads to two types of composite reference: exclusive or shared. An exclusive composite reference from an object X to another object Y means that Y is part of only X. A shared composite reference from X to Y means that Y is a part of X and possibly other objects. Shared composite references can be established to merge and unify justifications based on commonalities detected in their assets, issues, symptoms, impacts, or solutions.

The World Wide Web Consortium's (W3C) Web Ontology Language (OWL) and Resource Description Framework (RDF) provide specifications for creating metadata models and general software methods for encoding, querying, interpreting, and using ontologies. RDF will be used here to describe the use of "Relationship Predicates" for specifying relationships connecting subjects and objects in ontologies. This is similar to the classical conceptual modeling approaches such as Chen's Entity-Relationship (Chen 1976), Object-Oriented Design (Booch 1991, Rumbaugh et al. 1991) and Unified Modeling Language (UML) for encoding relationships among entities and objects in data models and schemas.

RDF makes statements about relationships between concepts in the form: **subject-predicate-object,** which is known as **triples**. In an RDF triple, a subject denotes a concept. Object denotes a related trait, aspect, characteristic or property of the subject. The object can be a subject in another RDF triple. Predicate expresses the relationship between the subject and its object. This is illustrated in Figure 2.

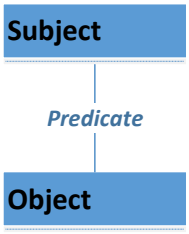

Figure 2. Predicate expresses a relationship between a subject (i.e. concept) and a related object (i.e. trait, aspect, characteristic or property of the subject.)

An object in one predicate may be the subject in another predicate relationship. This is illustrated in Figure 3.

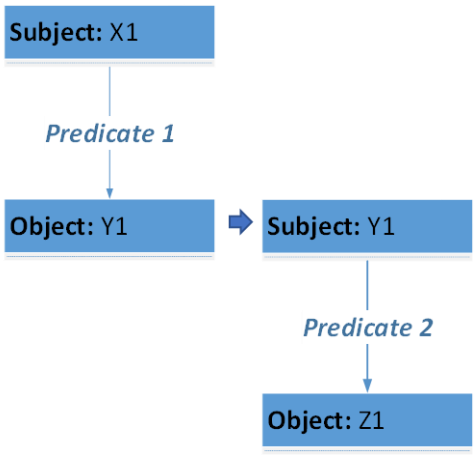

Figure 3. Illustration of an object in one Predicate 1 being the subject in Predicate 2.

A sequence of predicates can be used to expresses transitive relationships among concepts. Navigating across transitive relationships enables associative inferencing. For example, Figure 4 illustrates the transitive association: If Z1 is a property of Y1 as established by Predicate 2, and Y1 is a property of X1 via Predicate 1, then Z1 is also a property of X1.

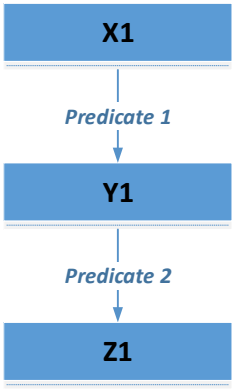

Figure 4. Predicate 1 associates X1 with Y1, Predicate 2 associates Y1 with Z1, and navigation across both predicates enables transitive association of X1 and Z1 through Y1.

A collection of RDF statements can express an ontology as a labeled directed multi-graph of associations known as a semantic network. The semantic network is an artificial intelligence knowledge representation which is both human and machine readable. Semantic networks support intuitive visual displays and automated inferencing algorithms based on network traversal.

Three general relationship predicates can be used to associate Asset, Component, Issue, Symptom, Impact, and Solution as subjects and objects to form a central skeleton for a O&M work order justification ontology on which to build up details. These are: “Has a”, “Affected by”, “Has risk of”, “Characterized by”, and “Mitigated by” and are defined in Figure 5.

"Has a"

Subject (**Asset**), Predicate (**Has a**), Object (**Component**)

"Affected by"

Subject (**Asset**), Predicate (**Affected by),** Object (**Issue**)
Subject (**Component**), Predicate (**Affected by),** Object (**Issue**)

"Has risk of"

Subject (**Issue**), Predicate (**Has risk of**), Object (**Impact**)

"**Characterized by"**

Subject (**Issue**), Predicate (**Characterized by**), Object (**Symptom**)

"Mitigated by"

Subject (**Issue**), Predicate (**Mitigated by**), Object (**Solution**)

Figure 5. General relationship predicates for connecting the subjects and objects found in O&M work order justifications.

The basic elements for a prototype O&M work order justifications skeletal ontology are subjects and objects illustrated as labeled boxes in Figure 6 and relationship predicates depicted with as labeled links connecting the subjects and objects in an aggregation to form a Work Order Justification object.

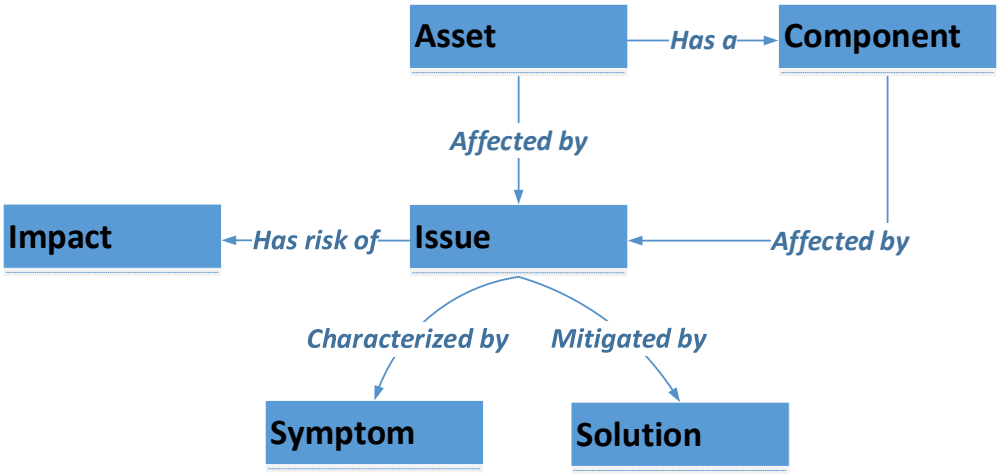

Figure 6. Central concepts (i.e. subjects/objects and relationships) aggregated to form an O&M work order justification.

Generalization-Specialization Relationships

Generalization is useful for modeling similarities and differences among concepts and the object data classes used to represent them in an application's memory and persistent datastore. The generalization, or class being refined, is sometimes referred to as the superclass, and each refined specialized version is a subclass. Generalization establishes a logical OR relationship been the more refined subclasses.

A generalization is sometimes called an "Is-A" semantic relationship because each instance of a subclass is also instance of the superclass. Properties and operations common to a set of subclasses are attached as 'generalized' features to the superclass. These features are shared through inheritance by each subclass. Each subclass inherits the features of its superclass(es) and refines or specializes them by adding their own specific properties and operations. The inheritance of generalized features is transitive across an arbitrary number of generations of subsequent subclass refinements of the original superclass. As a direct result of transitive closure, an instance of a subclass is simultaneously an instance of all its ancestor classes in a generalization hierarchy. That is, all ancestor class attributes and operations apply to the subclass instances.

Generalization-specialization relationships enable further elaboration of the subject/object classes and semantic relationships that comprise the skeletal schema to provide an organizing framework for structuring properties pertaining to work order justifications and storing the values for the properties. Analysis of O&M work order justifications identifies opportunities to further elaborate the Solution component of the skeletal work order justification ontology and schema with semantics and specialized subclasses for expressing and storing unique property values common to: Evaluation work orders, Repair work orders, Replacement work orders, Improvement work orders, and Augmentation work orders.

Below are RDF relationship primitives for adding the "Is-A" relationship to augment the skeletal ontology for expressing an OR relationship that differentiates among different types of work order justification solutions:

- Subject (**Solution**), Predicate (**Is a**), Object (**Evaluation**)
- Subject (**Solution**), Predicate (**Is a**), Object (**Repair**)
- Subject (**Solution**), Predicate (**Is a**), Object (**Replacement**)
- Subject (**Solution**), Predicate (**Is a**), Object (**Improvement**)
- Subject (**Solution**), Predicate (**Is a**), Object (**Augmentation**)

Examples of O&M work order justifications found in work orders of public works utilities often include alternate solutions illustrated in Figure 7 and which can be read "…a solution offered by a work order 'is a' evaluation, repair, replacement, improvement, or augmentation; and an evaluation 'is a' Regulatory Compliance evaluation, Hydraulic Analysis evaluation, or an evaluation of Alternative Options."

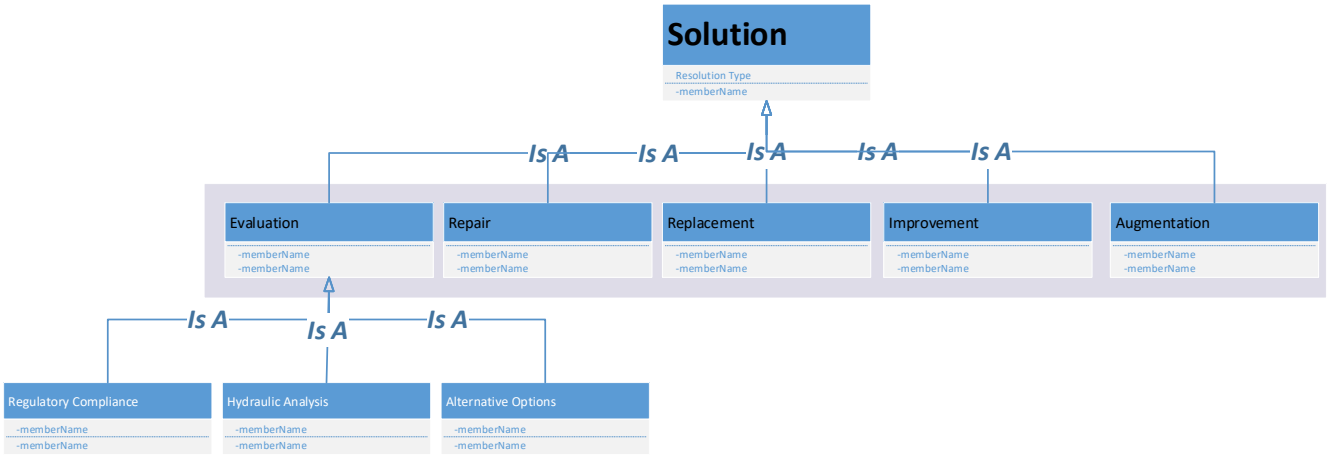

Figure 7. The generalized Solution superclass and its specialized subclasses for storing attributes unique to Evaluation, Repair, Replacement, Improvement, and Augmentation work orders.

Below are two RDF relationship predicates for expressing semantics of defect and deficiency issues:

- Subject (**Issue**), Predicate (**Is a**), Object (**Defect**)
- Subject (**Issue**), Predicate (**Is a**), Object (**Deficiency**)

Figure 8 illustrates the Issue generalization and two subclasses for storing different information pertaining to defects versus deficiencies.

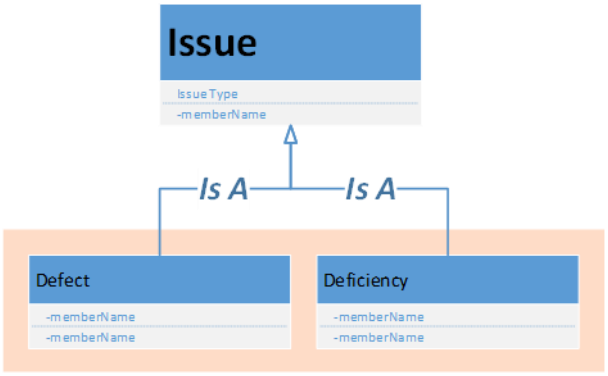

Figure 8. The generalized Issue superclass and its specialized subclasses for storing different properties for describing Defects versus Deficiencies.

Below are two RDF relationship predicates for different types of assets referenced in work order justifications:

- Subject (**Asset**), Predicate (**Is a),** Object (**Geographic Feature**)
- Subject (**Asset**), Predicate (**Is a),** Object (**Facility – Constructed Asset**)

Figure 9 illustrates two general types of assets: geographic features and constructed facilities.

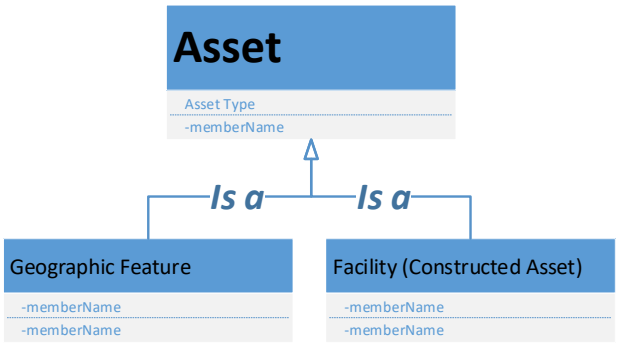

Figure 9. The generalized Asset superclass and its specialized subclasses for storing different properties of Geographic Features and Facilities (Constructed Assets).

Figure 10 illustrates a work order justification ontology expressed as a Unified Modeling Language (UML) data model ('schema') based on aggregation and generalization relationships. The UML schema captures and structures semantics for serving as a:

1. Ontology able to support parsing justification texts
2. Schema able to support development of
    a. A database for storing justification datasets, and
    b. An application object able to process justification datasets to detect commonalities and patterns among work order justifications, and for developing a database for storing the justification data

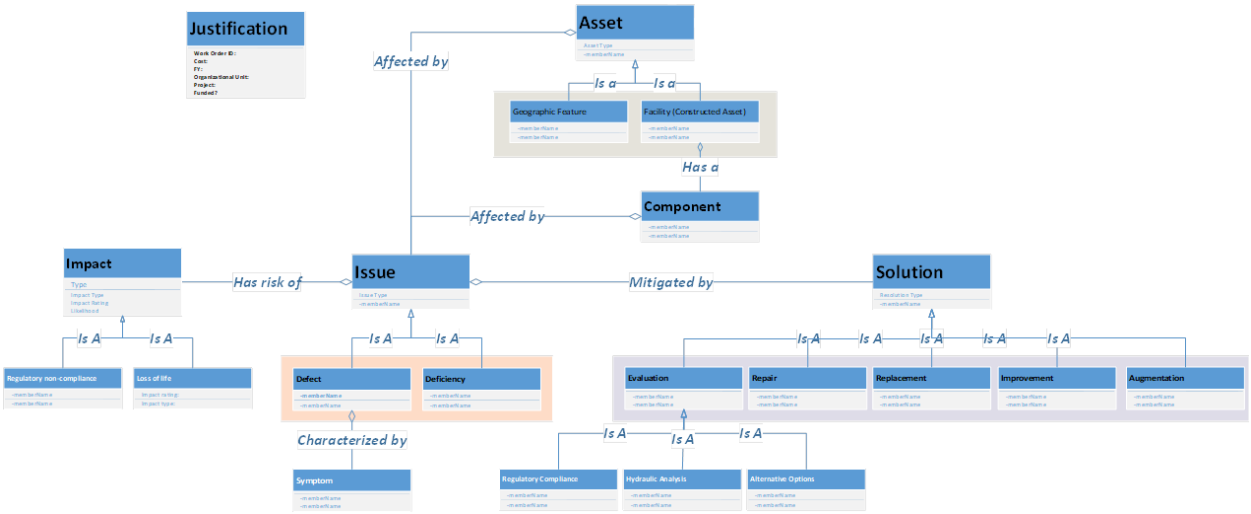

Figure 10. A prototype work order justification ontology expressed as a Unified Modeling Language (UML) schema based on aggregation (displayed with diamonds) **and generalization** (displayed with triangles) **relationships.**

WORK ORDER JUSTIFICATION EQUIVALENCE AND SIMILARITY

The O&M work order justification ontology can be expressed as a skeletal database schema illustrated in Figure 10. Implementation as an object-oriented database schema enables combined use of schema navigation and graph matching in an application capability for automatically comparing the instance data of two work order justifications to determine if they are equivalent or similar. Determining ontological equivalence and similarity among work order justification instance datasets is made possible by a schema graph matching capability that examines and detects equivalence and similarity in their Assets, Components, Impacts, Issues, and Solutions.

That is, two justifications are equivalent if their:
Assets are equivalent + **Issues are equivalent** + **Impacts are equivalent** +**Solutions are equivalent**

The work order justification knowledge representation, describes the semantic relationships among Assets, Component, Issues, Symptoms, Impacts, and Solutions justifying proposed O&M investments denoted by S. It is a logical description of the work order justification object-oriented database. It consists of nodes representing work order justification object types $\underline{O}$, and a set of semantic link types (i.e. relationships) $\underline{R}$:

$$S = (\underline{O}, \underline{R})$$

Let O_i denote a justification object type, $\underline{O} = \{O_i: i = $ Asset, Component, Issue, Symptom, Impact, Solution, Geographic Feature, Facility (Constructed Asset), Evaluation, Repair, Replacement, Improvement, Augmentation, Regulatory Compliance, Hydraulic Analysis, and Alternative Options$\}$

Semantic relationship types, $\underline{R}$, link these objects to structure the conceptualizations expressed in a work order justification. Let R_j denote a link type, $R = \{R_j: j =$ Has a, Affected by, Has risk of, Characterized by, Mitigated by, Is a$\}$.

Each R_j is defined as a set of ordered pair of objects O_{j1} and O_{j2} such that $R_j = (O_{j1}, O_{j2})$ where $R_j \in \underline{R}$ and $O_{j1}, O_{j2} \in \underline{O}$.

$R_{\text{Has a}} = \{(O_{\text{Asset}}, O_{\text{Component}})\}$
$R_{\text{Affected by}} = \{(O_{\text{Asset}}, O_{\text{Issue}}), (O_{\text{Asset}}, O_{\text{Component}})\}$
$R_{\text{Has risk of}} = \{(O_{\text{Issue}}, O_{\text{Impact}})\}$
$R_{\text{Characterized by}} = \{(O_{\text{Issue}}, O_{\text{Symptom}})\}$
$R_{\text{Mitgated by}} = \{(O_{\text{Issue}}, O_{\text{Solution}})\}$
$R_{\text{Is a}} = \{(O_{\text{Solution}}, O_{\text{Evaluation}}), (O_{\text{Solution}}, O_{\text{Repair}}), (O_{\text{Solution}}, O_{\text{Replacement}}), (O_{\text{Solution}}, O_{\text{Improvement}}), (O_{\text{Solution}}, O_{\text{Augmentation}}), (O_{\text{Asset}}, O_{\text{GeographicFeature}}), (O_{\text{Asset}}, O_{\text{Facility(ConstructedAsset)}}), (O_{\text{Evaluation}}, O_{\text{RegulatoryCompliance}}), (O_{\text{Evaluation}}, O_{\text{HydraulicAnalysis}}), (O_{\text{Evaluation}}, O_{\text{AlternativeOptions}}), \}$

TRANSITIVE LINK OPERATOR

Algorithms for traversing links in a work order justification knowledge representation provide a basis for access the data visualizing and analyzing the rational for a budgetary investment expressed in a justification instance. A transitive link operator (Kunii 1990, p. 18) can be defined to start at an object instance of a justification instance and recursively identify and navigate all semantic relationships to identify data records for object instances expressing the meaning of the justification.

Let o_i be an instance of a given justification object type O_i and r_i be an instance of a given justification relationship type R_i, then *All-related-objects(*o_i*)* can be defined as a transitive link operator that:

1. Set list of *related-objects* to null
2. Identify each relationship r_i in $\underline{R}$ that o_i is part of in either of the roles O_{j1} or O_{j2}, if r_i is null return *related-objects*
    a. Otherwise
        i. Traverse linkage of each relationship r_i connected to o_i
        - Append the associated o_{j2} or o_{j1} as instantiated in the justification by r_i to *related-objects*
        - Recursively apply 2 to the associated o_{j2} or o_{j1} as instantiated in the justification by r_i

PROPERTIES OF JUSTIFICATION OBJECTS

Each object structured within the work order justification knowledge representation may be stored in working memory of the parser and analytical processing application or persisted in the database along with its associated properties and values according to its type: i.e. Asset, Component, Issue, Impact, Symptom, Solution, Geographic Feature, Facility (Constructed Asset), Evaluation, Repair, Replacement, Improvement, Augmentation, Regulatory Compliance, Hydraulic Analysis, and Alternative Options.
Associated with O_i in $\underline{O}$ is an ordered list of properties P_i, $P_i = \{P_{i1}, P_{i2}, \ldots, P_{ik\ k=f(i)}\}$.

For example,

O_{Asset},A_{Asset} = (ObjectID, AssetType, Synonyms, Homonyms, AssetDatabase, AssetID, …)
$O_{Component}$,$A_{Component}$ = (ObjectID, ComponentType, Synonyms, Homonyms, ComponentDatabase, ComponentID, …)
O_{Issue},A_{Issue} = (ObjectID, IssueType, Synonyms, Homonyms, …)
$O_{Symptom}$,$A_{Symptom}$ = (ObjectID, AffectedArea, Observation, Synonyms, Homonyms, Details, Type, …)
O_{Impact},A_{Impact} = (ObjectID, ImpactType, Synonyms, Homonyms, ImpactRating, ImpactSource, Likelihood, LikelihoodSource, …)
…

Let p_{im} denote a value of P_{im}, then an instance p_i is a given justification object type P_i, which can be expressed as $p_i = (p_{i1}, p_{i2}, \ldots, p_{ik})$.

Combining these metadata with the semantic relationship links described above results in a metalevel representation of work order justifications. The representation can be used for an ontology information system: as an ontology for the parser, and as a knowledge representation for a semantic pattern detecting and visualization application for finding and explaining commonalities, and as a schema for an object-relational database to persist work order justifications.

Semantic network representation coupled with graph matching has been demonstrated as a solution for finding commonalities and differences in apparently disparate expressions of analytic reasoning (Lanter 1994). These techniques provide a basis for automatically comparing work order budget justifications, and provides a basis for determining equivalence, similarities and differences in their logic.

WORK ORDER JUSTIFICATION EQUIVALENCE

Data equivalence exists in any portions of two or more justifications that represent the same types and properties of objects structured within the same kinds of relationships.

The search for equivalence focuses on commonality among knowledge representations within two domains:

1. Properties of objects
2. Relationships between objects

Equivalence of justification object instances is based on their equivalent property values, which can be denoted:

$$o_{i'} \equiv o_{i''}$$

iff

$$\forall P_{ik} \in P_i \wedge p_{i'k} = p_{i''k}$$

Equivalence of relationship instances is based on equivalence of their objects in the object roles they play, which can be denoted:

$$r_{i'} \equiv r_{i''}$$

iff

$$\forall R_{jw} \in R_j \wedge o_{j,1'} \equiv o_{j,1''} \wedge o_{j,2'} \equiv o_{i,2''}$$

WORK ORDER JUSTIFICATION SIMILARITY

Determination of equivalence in justifications, however, can be confounded by conflicts in words used to label the concepts expressed in the objects. Such conflicts make it necessary to search for similarity in metadata representations of the different realities and differences in metadata representations of the same reality expressed by authors of work order justifications (Batini et al., 1992). For example, synonyms exist when the same issues, impacts or solutions are referred to by different names. Homonyms exist when the same name is used for different issues, impacts, or solutions. The use of standard issue, impact and solution classification schemes provides a theoretical basis for ensuring that such synonym and homonym conflicts do not occur when comparing justifications. One possible approach to matching synonyms and homonyms used between different justifications is the classification 'crosswalk'. For example, a classification crosswalk can be developed to enable matching similar categories of issues, impacts or solutions that may be found in issues, impacts, or solutions written by different justification authors.

Justification similarity exists in any portions of a set of justifications that represent similar types and properties of objects structured within the same kinds of relationships. Similarity of justification object instances is based on their similar property values, which can be denoted:

$$o_{i'} \approx o_{i''}$$

iff

$$\forall P_{ik} \in P_i \wedge p_{i'k} \approx p_{i''k}$$

Similarity of relationship instances is based on similarity of their objects in the object roles they play, which can be denoted:

$$r_{i'} \approx r_{i''}$$

iff

$$\forall R_{jw} \in R_j \wedge o_{j,1'} \approx o_{j,1''} \wedge o_{j,2'} \approx o_{i,2''}$$

ANALYSIS OF COVARIATION OF WORK ORDER JUSTIFICATIONS

The skeletal ontology and object-oriented schema described in this research provides a straightforward basis for supporting graphical query functions for analyzing and understanding commonalities and patterns in work order justifications. In addition to using schema navigation to traverse and evaluate objects matching across AND relationships of composite/component objects OR relationships of generalization/specialization objects can also be examined and matched for equivalence or similarity.

This kind of graph matching can start at any of the Issue, Impact, Solution, and Asset nodes of a set of justifications being compared. Examination of the resulting specializations of work order justification components can reveal insightful covariation of similarities and differences in otherwise disparate datasets. Queries from the following classes can be interactively concatenated in response to displays of available work order justification data to narrow down the desired justifications.

1. Display distribution of justifications across Assets and their properties
2. Display distribution of justifications across Components and their properties
3. Display distribution of justifications across Issues and their properties
4. Display distribution of justifications across Symptoms and their properties
5. Display distribution of justifications across Impacts and their properties
6. Display distribution of justifications across Solutions and their properties
7. Display distribution of justifications across Assets, Components, Symptoms, Impacts and Solutions and their properties

For example, justifications can be queried based on a property of a component object such as "type":

- A query on an **asset's type** will return work order justifications proposed for that type of asset and enable display, subsequent graph matching for equivalence/similarity determination, querying and analysis of the covariation of all:
  - Issues affecting that type of asset
  - Impacts resulting from issues affecting that type of asset
  - Symptoms of issues affecting that type of asset
  - Solutions for resolving the issues and resolving the conditions affecting that type of asset described in the justifications

- A query on an **issue type** will return work order justifications proposed for that type of issue, and enable display, subsequent graph matching for equivalence/similarity determination, querying and analysis of the covariation of all:
  - Assets with that type of issue
  - Symptoms of that type of issue
  - Impacts of that type of issue
  - Solutions to that type of issue
  - described in the justifications

- A query on a **symptom type** will return work order justifications proposed for that type of symptom, will enable display, subsequent graph matching, querying and analysis of the covariation of all:
  - Assets with that type of symptom
  - Issues associated with that type of symptom
  - Impacts associated with that type of symptom
  - Solutions that quell that type of symptom
  - described in the justifications

- A query on an **impact type** will return work order justifications proposed for that type of impact, and enable display, subsequent graph matching, querying and analysis of the covariation of all:
  - Assets associated with that type of impact
  - Issues that may result in that type of impact
  - Symptoms associated with that type of impact
  - Solutions that mitigate that type of impact
  - described in the justifications.

CONCLUSION

This paper described an approach for developing an ontology-based object-oriented processing application and database system able to learn how to parse and process work order justifications to detect and analyze patterns of commonality of their business cases. The stepwise construction of a global object-oriented work order justification ontology/data model was illustrated. Logic was provided for analyzing the semantic structure of work order justifications and automatically detecting commonalities necessary for comparative analysis.

Future implementation in a pilot project focused on Operations & Maintenance (O&M) work order budget justifications offers a way to evaluate the benefits of unified standard language and suite of intelligent tools for analyzing and increasing understanding of business cases for and asset management work orders.

REFERENCES

Batini, C., Ceri, S., & Navathe, S. B. (1992). *Conceptual Database Design An Entity-Relationship Approach*. The Benjamin/Cummings Publishing Company, Inc.

Booch, G. (1991). *Object-Oriented Analysis With Applications*. The Benjamin/Cummings Publishing Company.

Chen, P. (1976). The Entity-Relationship Model - Toward a Unified View of Data, *ACM Transactions on Database Systems*. 1(1), 9–36.

Critical Infrastructure Sectors. (2020, March 24). Cybersecurity & Infrastructure Security Agency. Retrieved August 28, 2020 from https://www.cisa.gov/critical-infrastructure-sectors

Kunii, H. S. (1990). *Graph Data Model and Its Data Language*. Springer-Verlag.

Lanter, D. P. (1994). Comparison of Spatial Analytic Applications of GIS. In Editors: W. K. Michener, J. W. Brunt & S. G. Stafford (Eds.), *Environmental Information Management and Analysis: Ecosystem to Global Scales* (pp. 413-425). Taylor & Francis.

Nguyen, V. (2011). *Ontologies and Information Systems: A Literature Review*. DSTO-TN-1002, Defense Science and Technology Organisation, Commonwealth of Australia

Reul, Q. H. (2012). *Role of Description Logic Reasoning in Ontology Matching*, [Unpublished Doctoral dissertation]. University of Aberdeen.

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. & Lorensen, W. (1991). *Object-Oriented Modeling and Design*. Prentice-Hall, Inc.

Sinha, P. K. & Dutta, B. (2020) A Systematic Analysis of Flood Ontologies: A Parametric Approach, *Knowledge Organization*, 47(20), 138-159.

Dr. David Lanter is Assistant Professor of Practice in Management Information Systems at the Fox School of Business at Temple University where he directs the school's graduate program in Information Technology Auditing and Cyber Security. He was vice president of information management services at CDM Smith where he designs technical and cybersecurity solutions for federal, state, regional and municipal agencies.