A Design for Additive Manufacturing Ontology

Mahmoud Dinar[1]
G. W. Woodruff School of Mechanical Engineering,
Georgia Institute of Technology,
Atlanta, GA 30332
e-mail: mdinar3@gatech.edu

David W. Rosen
G. W. Woodruff School of Mechanical Engineering,
Georgia Institute of Technology,
Atlanta, GA 30332

Design for additive manufacturing (DFAM) gives designers new freedoms to create complex geometries and combine parts into one. However, it has its own limitations, and more importantly, requires a shift in thinking from traditional design for subtractive manufacturing. There is a lack of formal and structured guidelines, especially for novice designers. To formalize knowledge of DFAM, we have developed an ontology using formal web ontology language (OWL)/resource description framework (RDF) representations in the Protégé tool. The description logic formalism facilitates expressing domain knowledge as well as capturing information from benchmark studies. This is demonstrated in a case study with three design features: revolute joint, threaded assembly (screw connection), and slider–crank. How multiple instances (build events) are stored and retrieved in the knowledge base is discussed in light of modeling requirements for the DFAM knowledge base: knowledge capture and reuse, supporting a tutoring system, integration into CAD tools. A set of competency questions are described to evaluate knowledge retrieval. Examples are given with SPARQL queries. Reasoning with semantic web rule language (SWRL) is exemplified for manufacturability analysis. Knowledge documentation is the main objective of the current ontology. However, description logic creates multiple opportunities for future work, including representing and reasoning about DFAM rules in a structured modular hierarchy, discovering new rules with induction, and recognizing patterns with classification, e.g., what leads to "successful" versus "unsuccessful" fabrications. [DOI: 10.1115/1.4035787]

Keywords: ontology, design for additive manufacturing, design representation

1 Introduction

Design for additive manufacturing (DFAM) has tremendous potential in freeing designers from the limitations of conventional manufacturing, despite its own limitations. Creating complex geometries that are difficult to produce with subtractive manufacturing, and consolidating multiple parts or fabricating parts together in an assembled state to reduce manufacturing time and cost are two examples of the freedoms that additive manufacturing (AM) offers. Seizing these benefits cannot be achieved unless designers shift their thinking from traditional design for subtractive manufacturing. This is challenging, even for experienced designers. Unlike traditional manufacturing techniques where process planning is fairly established, there is a lack of formal and structured guidelines on benefits and limitations of AM, as well as how designs should be modified to accommodate those benefits and limitations. Designers will need to overcome the design for manufacturability fixation, associated with traditional manufacturing which is based on removing material from larger blocks, and decomposing a design concept into multiple parts in an assembly.

Additionally, there are a variety of AM technologies developed by different vendors with different capabilities and constraints (e.g., surface finish, material cost, and minimum bead size). There is a need for a system that can guide designers through the pros and cons of implementing each process for a specific design feature, and configuring process parameters such as part orientation during fabrication. In some DFAM cases, designers realize that part designs can be changed to accommodate process limitations without compromising the intended functionality of the design. In other cases, designers may configure process parameters to satisfy design parameters. An example of the former case is to use dispersed hemispheres instead of a continuous profile in the design of a screw assembly. An example of the latter is when fabricating a living hinge (revolute joint), the direction of fabrication should be normal to the plane that is congruous to the circular features. DFAM guidelines should encapsulate these tradeoffs.

Though there is a growing attention toward (AM) technologies, industrial applications are still limited. The reasons mirror the issues explained above. There is a lack of references that suit the industry. The literature on the application of AM processes is not structured in a way to help industries in making decisions about whether additive or traditional manufacturing techniques are more appropriate for a given design. In the absence of quantifiable metrics that facilitate determining success of implementing AM, or making comparisons among different AM processes, industries will struggle in adopting AM technology. A design guidance system for AM, therefore, will help industries in acquiring DFAM techniques and developing DFAM skills.

The first step in realizing the tentative DFAM guide system is developing a knowledge base. Such a base can emerge from identifying DFAM needs, collect design guidelines, rules, and case studies. This information can then be synthesized into a cohesive knowledge base. One source of information can be experimental data from fabricating sample parts. Existing databases and catalogs of vendors about processes, machines, and materials are other sources.

The knowledge will serve a few purposes. First, it will enable documenting and storing existing knowledge about AM process and DFAM that is rather ad hoc. Second, it forms a basis for a coaching or tutoring system to teach DFAM to novices, or a recommendation system to support experts or industry practitioners. Third, it can be integrated into a conventional CAD tool.

To operationalize the knowledge base, we propose a formal DFAM ontology represented in OWL/RDF. There are a few reasons to choose an OWL ontology over a relational database. One is that ontologies are superior in modeling knowledge compared to relational databases. The structure of relational databases is

[1]Corresponding author.

Contributed by the Design Engineering Division of ASME for publication in the JOURNAL OF COMPUTING AND INFORMATION SCIENCE IN ENGINEERING. Manuscript received August 31, 2016; final manuscript received January 10, 2017; published online February 16, 2017. Assoc. Editor: Yong Chen.

geared toward improving performance in data retrieval, not necessarily corresponding to a natural expression of a phenomenon. Ontologies focus on expressiveness (richer model) instead of performance [1]. Another reason is that a formal ontology is more pertinent to tutoring systems [2]. Concept maps [3] and their node-link representation which is similar to a formal OWL ontology have been used in many tutoring systems. Finally, the OWL formalism facilitates integration to web technology which raises the potential outreach of the DFAM guide system. Besides, implementation will be easier as there are many resources developed for Semantic Web, for which the standard OWL language was created. The ontological framework has a logical foundation which supports various types of reasoning that lead to knowledge discovery as well as information retrieval.

To summarize, these three purposes, i.e., repository for AM processes and DFAM knowledge, basis for a tutoring/recommendation system for DFAM, and basis for a CAD tool necessitate the following requirements for the DFAM knowledge base: (1) representing domain knowledge (e.g., what materials are implemented on a machine; what are process parameter limits on a machine) and experiential knowledge (similar to benchmark studies, e.g., feature sizes that were fabricated with a specific AM process, or surface finish as a function of overhang angle); (2) support reasoning with common inference engines; and (3) easy to translate to data structures used in common CAD tools. In the rest of this paper, we first review existing knowledge on DFAM and few relevant knowledge representations in design. The approach toward developing the ontology and the details of the data model are explained, followed by a discussion of some of the modeling issues and their implications. The paper concludes with some of the future research directions.

2 Review of Past Work

2.1 Knowledge Bases of Design for Additive Manufacturing. AM technologies have matured over the past couple of decades despite the more recent attention they have gained. However, there is limited work on developing a comprehensive and coherent AM knowledge base. Most researchers have conducted experiments to find capabilities and limitations of specific AM processes. Knowledge is expressed in guidelines that state quantitatively what has resulted in successful fabrications. Benchmark parts have been studied for AM processes such as powder bed fusion (PBF) and material extrusion (MEX), e.g., see Refs. [4] and [5]. These studies typically investigate the accuracy and repeatability of fabricating a variety of features, e.g., overhanging beams as well as quality characteristics such surface finish.

There have been efforts in collating information from different studies into a more comprehensive catalog-like form than single benchmarks. Kranz et al. [6] developed design guidelines for laser AM for TiAl6V4, in a series of recommendations with examples of favorable and unfavorable configurations. Although such recommendations are helpful, they are not clearly categorized, and relations among design and process parameters are also missing. This is a major caveat in benchmarking studies. Important issues arising from how design and process parameters influence one another are left out. Examples of such issues include:

- How does the resolvability of holes and thin walls vary with orientation and the thickness of the surrounding part?
- What is the best way to fabricate a snap fit or a living hinge in a selective laser sintering part?
- How much clearance should be provided between a shaft and a bore when fabricated in an assembled state?

Seepersad et al. [7] conducted a preliminary study that responds to some of these issues. They fabricated a set of benchmark parts using polymer PBF, and set guidelines on feature resolution, i.e., what is allowable in a design feature, e.g., a circular hole, as a function of other design and process parameters, e.g., plate thickness and part orientation. Such knowledge helps designers in dimensioning and tolerancing parts much more precisely. However, more work needs to be done on other design features, e.g., rotating shafts inside bores. In addition, more knowledge is needed to be formally acquired, especially expert knowledge from everyday practitioners and crowd-sourced efforts, considering that some of the data, e.g., life, might be available long after fabrication.

AM knowledge can also be described in an almost purely analytical form. Nelaturi et al. [8] used mathematical morphology to find regions in the geometric model of a part that are prone to build failure as they fall below minimum recommended layer thickness values. The geometrical model is decomposed into a set of volumes (slices) with layer thickness as height and closed polygons as area. The analysis of a part forms a printability map which is used to compute design corrections locally.

As mentioned above, an important aspect of AM knowledge is the relations among design features, design parameters, and process parameters. Similar to feature catalog of Kranz et al. [6] for a specific material and process type, Adam and Zimmer [9] developed a catalog of design rules for different design features and parameters, using PBF and MEX. The rules were derived from tabulated experimental data for different design features such as overhang, showing whether fabrication was successful or not (a class attribute with values "OK" and "Destroyed") with respect to design parameters, e.g., overhang length. Even though the catalog makes it possible to compare fabrication success of a design feature using different AM processes, there is a missed opportunity in formalizing this knowledge in an easily searchable way.

AM knowledge bases so far have been mostly ad hoc, geometry-centric, process-specific, and informal. To be more useful in guiding designers, an AM knowledge base needs to accommodate empirical data, represent relations among entities (ideally, representing parametric and/or mathematical expressions of how some entities are related), and have a structure that is easy to query, i.e., easy to understand and facilitate search for different entities (process type, design feature, design parameter, process parameter, etc.). These requirements relate not only to the content of the knowledge base but how it is represented which we review next.

2.2 Representations of Design Knowledge. Various knowledge representation methods and forms have been implemented in engineering design and for different purposes. Some are used for representing product data, some for process data, and some for both. They also are used at different abstraction levels and different stages of the design process. Ontological frameworks have been often implemented in describing cognitive processes and conceptual design; examples are think maps [10] and problem maps [11] which were inspired by knowledge modeling with concept maps [3]. Standard formalisms are rarely used. Wölkl and Shea proposed the application of SysML [12] for conceptual design, though they conceded that the process is trivial.

Embodiment design and detailed product data have long been the main application of information and knowledge modeling in engineering design, especially since the advent of CAD/CAM tools, though most representations are for information modeling purposes. Yet, researchers have proposed representation schemes for purposes such as data transfer, knowledge reuse, and design automation. Summers et al. [13] created the design exemplar. The schema uses a node-link representation in bipartite graphs to facilitate pattern matching and change propagation. Design exemplars require a complete and explicit description of design data in two distinct sets. They call one set entities and the other constraints. However, it is difficult to make a clear distinction between the data type of each set. For example in modeling a gear, "Circle" and its instance "pitch_circle_1" as well as "Parameter" and its instance "pitch_radius_1" are entities; "Radius" (an abstract entity) and "Equation Eq_a" (an instance) are constraints. Design rules are defined with if-then statements. Sen et al. [14] have

extended the design exemplar in consistency checking of function models by using the Pellet reasoners over an OWL ontology. They define a grammar with description logic rules such as "A material flow can carry one or more flows of subtypes energy and signal, but not material."

The need for a formal knowledge representation of AM to guide designers has been addressed. Jee et al. [15] have proposed a modular representation for describing design rules for AM. The schema consists of three categories of geometric, process, and material primitives. Three types of modules are also defined. One is a composition of primitives, e.g., in the relation between undercut angle and overhang feature dimension. The second and the third types are compositions of primitives and modules, and modules with other modules, respectively. They present examples of rules in this structure with a case study. The description is bottom-up however, i.e., modules seem to emerge from the definition of observed experimental events rather than domain knowledge.

To summarize, AM knowledge has been documented often as results of benchmark studies. The relations among different design and process parameters, an important component of AM knowledge, are not represented in a coherent manner. Previous knowledge representation schemas in engineering design serve different objectives that cover specific steps in the design process and rarely DFAM. They also are not easily scalable and their development is trivial since most applications are limited to lab use, and also because they seldom utilize the growing paradigm of the Semantic Web, its established OWL/RDF formalism, and available tools such as Protégé.

3 Approach

Review of past work showed that there is a need for a DFAM knowledge base that guides designers in understanding the capabilities and limitations of various AM processes, how it is different from traditional manufacturing, and how the process and design affect each other. The benefits of a formal representation and their absence in existing DFAM knowledge bases were also described. To formally express DFAM knowledge, an ontology was developed using the Protégé tool.

Protégé and OWL are based on description logics (DL) [16], a subset of first-order logic which was created to add formal logic to knowledge representation in semantic networks [17]. DL has distinctive features that are associated with monotonic logic including the open-world assumption and excluding negation as failure; in other words, lack of knowledge does not imply falsity. It is also possible to implement inferences based on first-order logic (sometimes considered subsuming DL), e.g., by answer set programs [18] which in turn supports modular rule compositions.

3.1 Ontology Development Process. To create a DFAM ontology, a scope of content and objectives of the tentative ontology were identified. To meet the requirements outlined in the introduction, the target ontology should capture knowledge on different types of AM processes, how various design features can be fabricated, and what parameters lead to success or failure in fabrication. The knowledge base should support different levels of abstraction, i.e., it should express domain knowledge as well as empirical knowledge gathered from benchmark studies. To develop the ontology the following steps were taken:

(1) Enumerate the entities (e.g., "Design_feature," "Part_orientation").
(2) Define possible object properties in a triple of seminatural language sentences with properties (e.g., "Design_feature isComposedOf Design_feature," "Design_parameter parametrizes Design_feature").
(3) Define subclasses of entities (e.g., "Design_feature > Assembly > Thread").
(4) Populate with instances.
(5) Evaluate model and iterate through the previous steps.

3.2 The Protégé Tool for Authoring OWL Ontologies. The Protégé tool [19] is a graphical user interface for creating ontologies based on the OWL/RDF language of the Semantic Web. As a DL language, it basically deconstructs knowledge of a concept into a collection of triple sets in the form of *subject-predicate–object*. An object in one triple can be the subject of another triple which leads to a graph of inter-related nodes with edges that describe a property connecting the nodes. In Protégé, the *predicate* is actually called an *object property* where *subject* of the RDF triple is the "domain" of the object property, and *object* is its "range." Another interpretation which may be used in translating relational database records to OWL/RDF triples is that *subject* is an entity identifier or a row ID of a table, *predicate* is an attribute name or column name of a table, and *object* is the attribute value.

Attributes of an entity are defined by "data property" with the same triple structure: entity is the "domain", and the "range" is a data type such as string or float. Instances [of entities] are added under "individuals" in Protégé. The attributes that define the entity are added as "data property assertions." When adding an assertion, a "data type" can also be specified regardless of whether or not a data type was assigned to the "data property" in defining the attribute. This is one of the ways for checking consistency in the ontology with a reasoner.

SPARQL is the language for querying OWL/RDF ontologies [20]. One of its strengths is that it makes it possible to query one's ontology, together with other ontologies and datasets from multiple online and remote sources. Combined with the ability to merge ontologies in Protégé, knowledge reuse is an inherent feature of this approach whether the knowledge base is domain dependent and closed, is general and public knowledge such as what is found in DBPedia [21], or a synthesis of both. The fundamental key of the representation is the concept of triples which can capture various relations, e.g., "class_A relates_to/affects class_B," "class_A subClassOf class_B," or "Instance_A is_a class_A." Therefore, the general form of a condition statement of a SPARQL query is written with three variables as {?x ?y ?z}. The variables can be substituted with constants/determinants to narrow the search results. For example, the following SPARQL query looks in an ontology named DFAM and returns all the machines that implement the polyvinyl alcohol (PVA) material:

```
PREFIX DFAM: <http://address/ontology.owl#>
SELECT * FROM
{DFAM:PVA DFAM:isImplementedOn?machine}
```

4 The DFAM Ontology

We have followed the process that was outlined in Sec. 3.1 to create a DFAM ontology and examine it in a real application. This section starts with a detailed description of the knowledge model of the proposed DFAM ontology. A case study with three design features is described. The features are shown, and relevant entities, attributes, and relations among them are identified and formalized in Protégé. The knowledge that goes in the fabrication of these design features is modeled and information about multiple instances is retained.

4.1 Structure of the Ontology. The DFAM ontology was created with a set of high-level entities which were enumerated based on the important knowledge that should be expressed in a general fabrication scenario. In this scenario, a fabrication event occurs when an AM process, machine, and material are chosen, process parameters are set, and design parameters that affect the design feature are identified. A distinction is made between design feature (which is about design intent regardless of method of manufacturing it) and manufacturing feature. For example, an overhang is a manufacturing feature. A designer does not necessarily (and should not) design a feature considering that it will be above a critical angle with the orientation of fabrication. Therefore, the

Table 1 Relations (object properties) among high-level entities in the DFAM ontology (reading direction →↑)

	AM event	Manuf. feature	Design feature	Design param.	Machine	Material	Process param.	Process type	Ref.
AM event			**isEmbodiedBy**	isLimitedBy	isControlledBy		isControlledBy		**isProvidedBy**
Manuf. feature			realizes	isLimitedBy					
Design feature	embodies	**isRealizedBy**	**isComposedOf**	isParametrizedBy	**isBuiltOn**	**isCreatedFrom**	controls		
Design param.	**limits**	**limits**	**parametrizes**	**isComposedOf**	isLimitedBy		isLimitedBy	isLimitedBy	
Machine	**controls**		builds	**Limits**		implements	**limits**	implements	
Material			creates	**Limits**	**isImplementedOn**			**limits**	
Process param.	controls		isControlledBy	**Limits**	isLimitedBy		**isComposedOf**		
Process type					**isImplementedOn**	isLimitedBy			
Ref.	provides								

high level entities in the DFAM ontology are AM event, Manufacturing feature, Design feature, Design parameter, Process parameter, Machine, and Material, as well as a Reference entity which points to the source of the information (e.g., a benchmark study in the literature).

Next, the object properties, i.e., the relation between any two of the high level entities, are defined. An active or passive verb that appropriately defines the relation is used. Similar relations use common names, e.g., *isImplementedOn* relates machine and material to an AM process. Table 1 shows the object properties identified in the current DFAM ontology. For each pair of entities that are related, one direction (in bold font) is defined with domain and range in Protégé, and the other direction is linked to the reverse with the "Inverse Of" property. The justification for the appropriateness of the chosen names or the direction is not relevant to the sufficiency of the ontology at this stage and therefore is not discussed further.

The relation between an entity and its parent or child in OWL does not imply a composition or aggregation. It means being a type of a superclass. An *isComposedOf* object property was defined to accommodate such relations for design features (resembling a product architecture or an assembly navigator in a CAD model), and design and process parameters (qualitatively showing parametric relations).

The subclasses of the high level entities were defined in the next step of developing the DFAM ontology. The high level entities were expanded as much as possible. For the purposes of the ontology outlined before, one major modeling decision was to have as few data properties as possible in the ontology. For example, in expanding design parameters, instead of being an attribute (data property) of bore-shaft assembly, Radial clearance is the subclass of circular entity which itself was the subclass of Geometric size parameters. Figure 1 shows the hierarchical class structure of the entities in the DFAM ontology. The list shown is simplified and some of the entities are hidden to save space. Attention should be paid to the class structure rather than specific entity names.

The AM event consists of two disjoint subsets: an event either is a build failure or success, and a cleaning failure or success. A design feature is an assembly or a part. The *isComposedOf* object property allows defining an assembly as a composition of parts since the object property is inherited from the superclass design feature. Design parameter has two subclasses: geometric size parameter and orientation parameter which is about the reference datums on the feature. Geometric size parameters are divided into categories of similar types such as angular or spatial parameters. Make and model of a machine is the basis for naming subclasses of Machine. As it will be described in examples of instantiation of

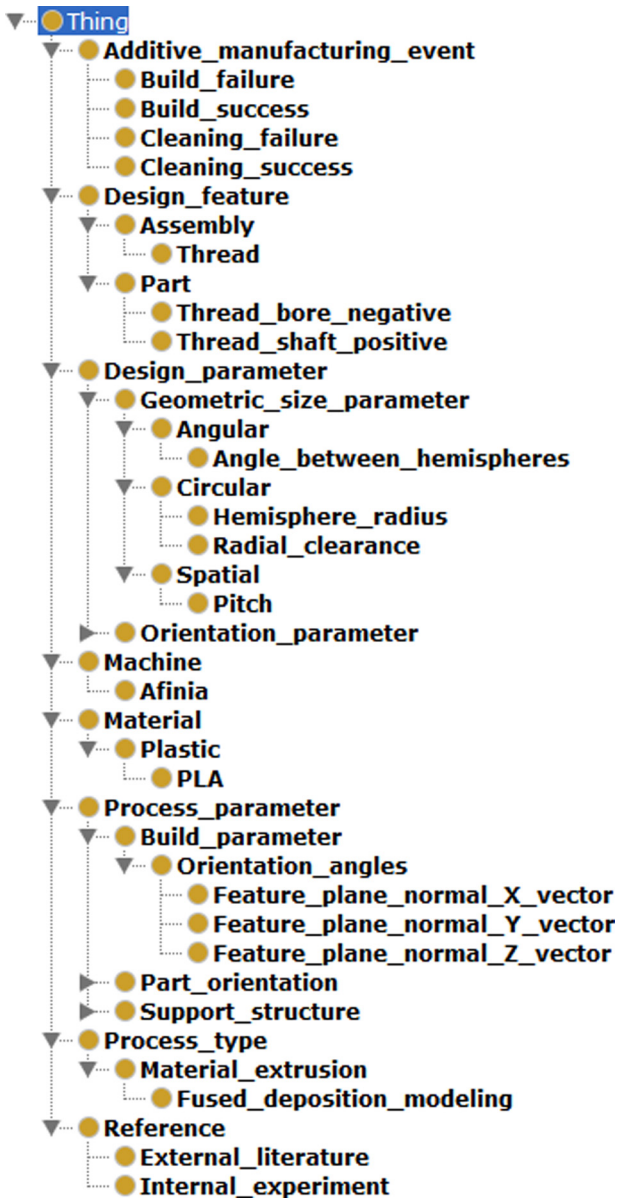

Fig. 1 The hierarchical entity structure of the DFAM ontology

events and data population in the ontology, the machine name as an entity rather than an instance allows storing different fabrications (AM events) with the same machine even if all other parameters remain the same. In such a case only the instance of the event is different and the same machine instance is related to the event. Similarly, material is decomposed based on family (plastic or metal) and type/substance. Process parameter consists of build parameter, Part orientation, and support structure. Build parameter is defined by three components of the vector normal to the feature plane of interest in fabrication. Another entity at the same level, Part orientation, can be used for specifying whether the feature plane is parallel, perpendicular, or at an angle to the build orientation. While the former classification shows a numeric parametric relation, the latter provides a qualitative understanding of the part orientation. The purpose of the ontology is knowledge expression and characteristics such as uniqueness are not critical; in OWL, a commonly used descriptor of an entity is "Equivalent to" which is used to show two entities (often merged into an existing ontology from different sources) refer to the same thing. Finally, Process type shows a classification of AM technologies, and Reference shows whether the knowledge is based on our experiments or comes from other sources in the literature of practice. An exhaustive list of the entities in the DFAM ontology cannot be given but the structure remains the same for storing knowledge of additional features. For the case study which will be described in Sec. 4.2, note that Assembly has a subset {Revolute joint, Slider crank} and part has {Hole, Pin}. Common design parameter subclasses are possible when appropriate, e.g., radial clearance is a design parameter for both threaded assembly and revolute joint assemblies.

Fig. 2 Sample models and fabricated design features

4.2 Case Study Design Features. The objective of this research at the current stage is to show how DFAM knowledge can be modeled in the OWL/RDF formalism. Therefore, a few design features were selected with general criteria such as being a common feature in the design of actual parts and benefiting from the freedoms that AM provides. For example, a revolute joint is a common design feature that appears in the design of many products, and can be fabricated in a single step without the need for assembly using an AM process. Two other design features have been chosen: a threaded assembly (screw connection) and a slider–crank mechanism, see Fig. 2. The design features have been characterized analytically and experimentally. Parametric CAD models of these joint features have been developed and several MEX samples have been fabricated. Based on the experiments, limits and additional relationships are derived for the parametric CAD models which will also be a part of the knowledge modeled in the ontology. It can be seen that DFAM of the threaded assembly replaces a continuous profile on the shaft with a set of dispersed hemispheres along the thread spiral.

4.3 Instantiation and Population of Data. To illustrate how knowledge was captured in the DFAM ontology, assertions about instances shown in Sec. 4.2 were added to the knowledge base. For the first example, information about a revolute joint is partially shown in Fig. 3 as a network graph highlighting the composition of part features and relation to design parameters. Entities (shown with circles) and relations among them capture experiential abstract knowledge, while individuals or instances (shown with diamonds) store experimental knowledge about actual fabrications. This graph only shows experiential knowledge of design features compositions. Similar knowledge about design parameters is captured in the DFAM ontology but is not shown here. It can be seen that a network graph representing all the experiential

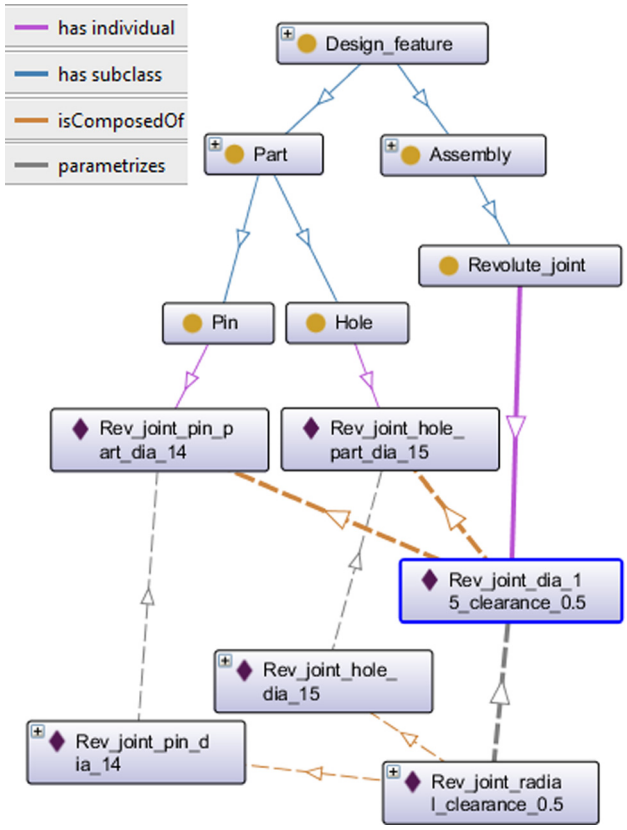

Fig. 3 Network graph of the revolute joint entities and instances

Table 2 Knowledge expressed in the ontology with triple sets in OWL/RDF formalism

Subject	Object property	Object
Thread assembly	isCreatedFrom	PLA
PLA	**subClassOf**	Plastic
Thread assembly	isBuiltOn	Afinia model X
Thread assembly	isComposedOf	Thread bore
Thread assembly	isComposedOf	Thread shaft
Hemisphere radius	Parametrizes	Thread shaft
Radial clearance	Parametrizes	Thread shaft
Radial clearance	**value**	1 mm
Thread assembly	isControlledBy	Part orientation
Part orientation	isComposedOf	Feature plane normal Z vector
Feature plane normal Z vector	**value**	1

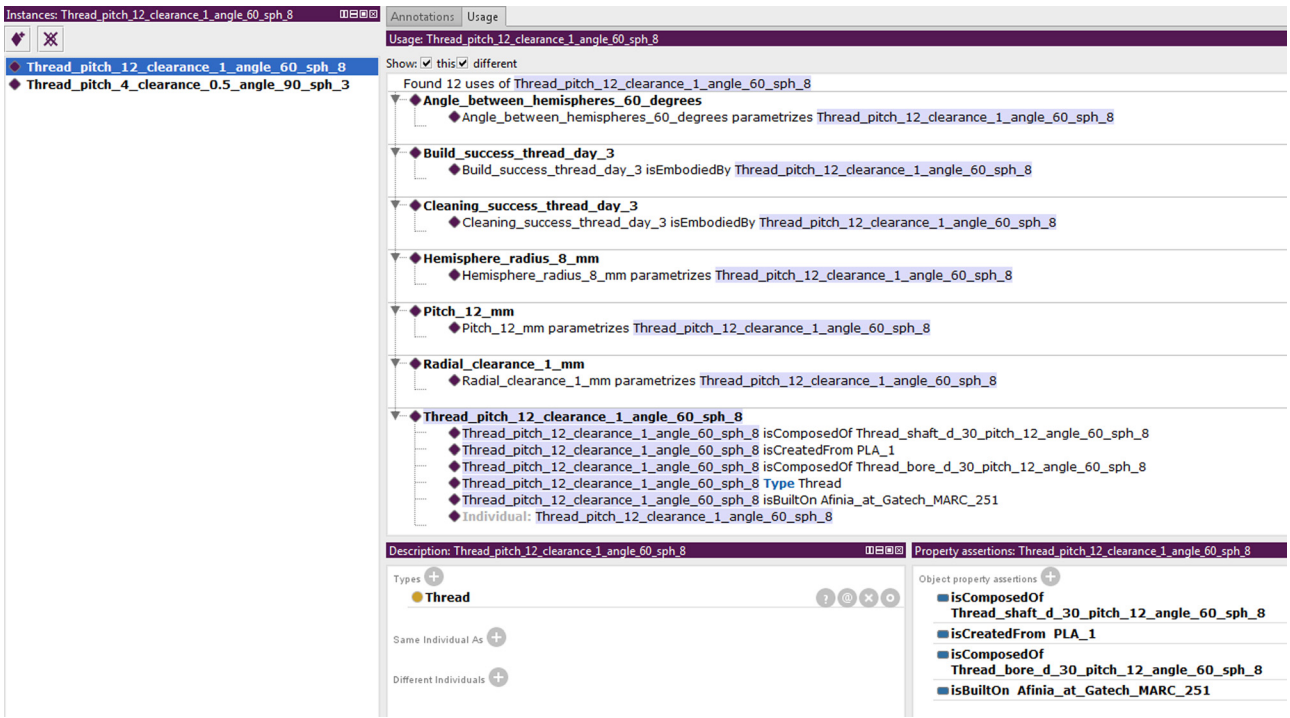

Fig. 4 Showing usage of all entities and attributes of an instance in the Protégé tool

and experimental knowledge of just one feature can be difficult to communicate visually in a limited space. Alternative textual representations, e.g., triple predicates, can be effective in describing knowledge stored in the DFAM ontology.

Consider another example of the experimental features; fabricating a threaded assembly (screw). The threaded assembly is an assembly part composed of a bore and shaft. Hemisphere radius and radial clearance are two of the design parameters. Based on the definitions of object properties given previously, some of the relations which should be expressed are exemplified in Table 2 in a set of triples, the building block of an OWL/RDF formal ontology. The structure is easily readable. The properties in bold font are standard OWL terms.

From the names of the subjects and objects, it can be seen that this formalism is powerful in expressing domain knowledge as well as experimental knowledge of actual events in the same structure. It also makes it easier to add new knowledge or modify legacy data compared to conventional relational databases which have a strict separation of data model and data content. Results of actual fabrications of ongoing experiments are being added to the knowledge base in the DFAM ontology. Figures 4 and 5 depict snapshots of the Protégé tool which show threaded assembly

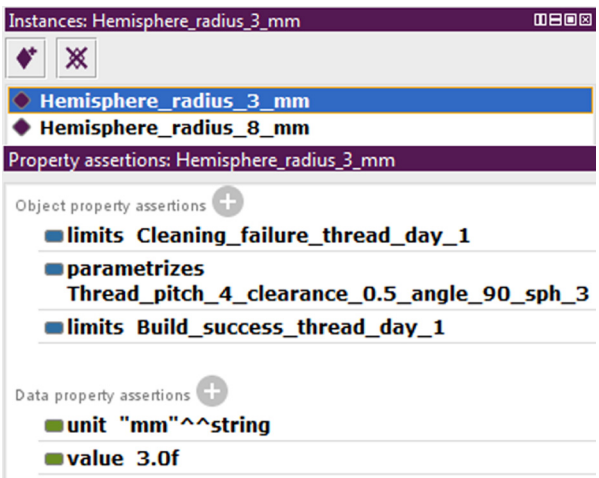

Fig. 5 Storing parametric data with assertions

fabrication data resulted in successful building and cleaning in one instance and successful building but cleaning failure in another. Once an instance is added, the object property assertions are chosen to store the relation between the subject and object in the triple.

Instances can be multiple builds of the exact same part, so naming of instances of all entities other than the AM event entity can be common. The name should capture the combination of defining design parameters. This enables capturing data for fabrications from a crowd source and examining whether similar settings have similar fabrication results. It also makes the ontology more readable and easy to understand which can be useful with respect to one of the purposes of the ontology as a basis for a tutoring system.

5 Discussion

Three requirements were identified for the DFAM ontology. It should represent domain and experiential and experimental knowledge, facilitate reasoning associated with descriptive logics, and be a basis for or be easy to integrate into a CAD tool. This section discusses how the proposed DFAM ontology can satisfy these requirements. As this is an ongoing work, the discussion is limited to the case study. The choice of the OWL formalism and the Protégé tool provides use of inference engines, and the XML structure of OWL can be the means for integration into a CAD system. In addition, knowledge modeling in ontologies is a satisficing problem. Ontologies are often examined for sufficiency with respect to competency questions [22]. Therefore, in a few examples we demonstrate scenarios where the DFAM ontology can be used to help designers with respect to the requirements for the knowledge base.

Guiding designers occur at different levels. In a general scenario, a designer might need to know what AM process is implemented on a machine, or the different types of material that can be used on that machine. In another scenario, the designer might want to find the machines that use a particular material such as PVA. Scenarios can be more specific about the limitations or conditions of DFAM. Designers might be interested in knowing the design parameters that limit success of a design feature in an AM process. These scenarios can be rewritten as competency questions such as:

- Which material is used on machine X?
- What is the minimum wall thickness that can be built on the machine X?
- What AM processes are capable of fabricating a revolute joint with a radial clearance less than 0.8 mm?

The fundamental key of the representation is the concept of triples which can capture various relations, e.g., "class_A relates_to/affects class_B," "class_A subClassOf class_B," or "Instance_A is_a class_A." Therefore, the general form of a condition statement of a SPARQL query is written with three variables as {?x ?y ?z}. The variables can be substituted with constants/determinants to narrow the search results. For example, the following SPARQL query looks in the DFAM ontology and returns the different types of material that are used on Machine_X, corresponding to the first competency question: SELECT?material WHERE {?material DFAM:isImplementedOn DFAM:Machine_X}. The last competency question requires a more complex query as shown in Fig. 6.

As mentioned, one requirement for the DFAM knowledge base is to support reasoning in common CAD tools. Using the slider–crank mechanism as an example, a usage scenario is presented that illustrates the importance of this type of requirement. In the near future, it is likely that CAD software will support manufacturability analysis with respect to AM processes. In designing the slider–crank, the designer needs to know the proper clearance to use in the revolute and sliding joints. If the AM process and material are selected, queries to the knowledge base could retrieve

SPARQL query:

```
PREFIX dfam: <http://www.semanticweb.org/.../dfam-ontology-3#>

SELECT ?process_type ?radial_clearance
WHERE
{
 ?radial_clearance dfam:limits dfam:Cleaning_success
 ?process_type dfam:limits ?radial_clearance
 ?radial_clearance dfam:parametrizes dfam:Revolute_joint
 FILTER(?radial_clearance<"0.8")
}
```

process_type	radial_clearance
FDM_on_Afinia	Radial_clearance_0.5_mm

Execute

Fig. 6 SPARQL query for a competency question

the smallest clearance associated with any other instances that are related to the same process and material. Similarly, if a small mechanism is desired, the designer will want to determine the minimum feature size capability of an AM machine so that the mechanism links and bearing features are not too small; querying a machine's minimum feature size can be accomplished readily. Finally, a query to find the minimum overhang angle associated with a specific AM machine could be used to generate support structures on all part surfaces that are below that overhang angle for a given build orientation.

Another formalism that supports reasoning with OWL ontologies is semantic web rule language (SWRL) [23]. SWRL rules are conjunctions of predicates in the precedent of the rule that lead to a consequent predicate. The predicates are the subjects, object, and data properties in the ontology with variable names substituting instances. For example, the following rule assigns wall features whose thickness fall below a threshold of 1 mm to a class called Thin_wall which can reflect on manufacturability issues (process capability):

$$\text{Wall}(?\text{wall}) \wedge \text{Wall\_thickness}(?\text{w\_t}) \wedge \text{parametrizes}(?\text{w\_t}, ?\text{wall}) \wedge \text{hasValue}(?\text{w\_t}, ?\text{thickness}) \wedge \text{swrlb:lessThan}(?\text{thickness}, 1.0) \rightarrow \text{Thin\_wall}(?\text{wall})$$

The description logic (DL) formalism of OWL creates some opportunities in knowledge modeling over conventional relational databases. While in traditional databases events are recorded as a collection of values on attributes, ontologies consist of relations among all atoms, and atoms can be any of class or attribute names, and instance names of classes or values of attributes. The all-in-one formalism extends information retrieval suitable for feature-based machine learning to knowledge retrieval and relational learning [24].

Besides being more expressive with human-interpretable explanations [24,25], first-order logic (which subsumes DL) can also be applied to reason with modular rules, e.g., by translating OWL into Prolog as done in Ref. [26]. Even though modular reasoning rules can be written in OWL (with the SWRL language), first-order logic rule authoring languages like Prolog are more powerful and concise especially in handling negations. For example, to write a rule that finds all unsuccessful build events for fabricating threaded assemblies, a DL rule should exclude a conjunction of all other design features which is not only trivial, but requires updating the rule every time a new design feature is added to the ontology. The predicate logic representation also makes it possible to clearly separate facts and rules, as well as structuring a modular and compositional rule hierarchy. Consider the examples given in Fig. 7. The facts, denoted with lowercase, correspond to instances. Generalization rules compose different conditions (variables in uppercase) to describe a more general condition which is more likely to be a part of another rule, e.g., all

```
% Rule 1: unsuccessful threads
:- ~thread(Design_feature).
:- unsuccessful(AM_event).
% Rule 2
radial_clearance_fabrication_threshold(Threshold):-
building(AM_event,success), minimum(Radial_clearance,Threshold).

% Generalization rules
% Gen. rule: unsuccessful events
unsuccessful(AM_event):-
event(AM_event,Design_feature), cleaning(AM_event,failure), building(AM_event,failure).

% Gen. rule: all threads are design features
design_feature(Design_feature):- thread(Design_feature).

% Facts
building(build_success_thread_day_3,thread_pitch_12_clearance_1_angle_60_sph_8).
thread(thread_pitch_12_clearance_1_angle_60_sph_8).
isComposedOf(thread_pitch_12_clearance_1_angle_60_sph_8,thread_shaft_d_30_pitch_12_angl
e_60_sph_8).
parametrizes(radial_clearance_1_mm,thread_pitch_12_clearance_1_angle_60_sph_8).
value(radial_clearance_1_mm,1).
```

Fig. 7 Example of modular reasoning rules with Prolog

threaded assemblies are design features. Depending on what the rule is supposed to discover, generalization rules or specific statements can be used which lead to flexibly define different rules. In the example given, rule 1 returns all unsuccessful AM events (which uses a generalization rule) for threaded assemblies; the Horn clause with the negation ∼thread(Design_feature) filters all design features that are not threaded assemblies. Rule 2 finds the minimum radial clearance for successful builds (regardless of whether the cleaning process led to success or failure). For brevity, many intermediate statements and rules are not included in the example, e.g., the function for finding a minimum value. Yet, it can be seen how the logical formalism facilitates reasoning with modular rules.

6 Future Work

The proposed DFAM ontology is capable of storing domain and experiential knowledge, retrievable for guiding designers in a tutoring system and as a basis of a CAD tool. There are several directions to improve on the existing ontology. First is expanding the scope by adding knowledge for more design features from our own experiments or previous benchmark studies. Some criteria can be used for selecting design features with potential for increased benefits from AM. They are part consolidation, weight reduction, structural strength, geometrical complexity, service life, and production volume. Design features can be evaluated against these criteria in light of traditional design and manufacturing practices and limitations, e.g., many traditionally designed parts have a large portion of mass that is orders of magnitude below ultimate strength levels since traditional manufacturing practice does not always afford removing such idle mass. Such knowledge should be a part of the guide system, e.g., as a set of measures of goodness.

Implementing the DFAM ontology in a tutoring system and evaluating its benefits is another direction. This is intimately related to the reasoning capabilities required for presenting guidelines to designers. Using logical induction methods such as inductive logic can lead to discovering new rules after adding more empirical knowledge to the ontology. Other machine learning approaches for general classification have potential to draw patterns from larger data sets.

Acknowledgment

This study was supported by Grant 4053.001 from America Makes. An earlier version of this paper was presented at the ASME IDETC/CIE 2016 conference in Charlotte, NC. We thank the reviewers of the conference for their constructive comments.

References

[1] Uschold, M., 1998, "Knowledge Level Modelling: Concepts and Terminology," Knowl. Eng. Rev., **13**(1), pp. 5–29.

[2] Zouaq, A., and Nkambou, R., 2010, "A Survey of Domain Ontology Engineering: Methods and Tools," *Advances in Intelligent Tutoring Systems*, R. Nkambou, J. Bourdeau, and R. Mizoguchi, eds., Springer, Berlin, pp. 103–119.

[3] Novak, J. D., and Gowin, D. B., 1984, *Learning How to Learn*, Cambridge University Press, Cambridge, UK.

[4] Ippolito, R., Iuliano, L., and Gatto, A., 1995, "Benchmarking of Rapid Prototyping Techniques in Terms of Dimensional Accuracy and Surface Finish," CIRP Ann.-Manuf. Technol., **44**(1), pp. 157–160.

[5] Mahesh, M., Wong, Y. S., Fuh, J. Y. H., and Loh, H. T., 2004, "Benchmarking for Comparative Evaluation of RP Systems and Processesnull," Rapid Prototyping J., **10**(2), pp. 123–135.

[6] Kranz, J., Herzog, D., and Emmelmann, C., 2015, "Design Guidelines for Laser Additive Manufacturing of Lightweight Structures in TiAl6V4," J. Laser Appl., **27**(S1), p. S14001.

[7] Seepersad, C. C., Govett, T., Kim, K., Lundin, M., and Pinero, D., 2012, "A Designer's Guide for Dimensioning and Tolerancing SLS Parts," Solid Freeform Fabrication Symposium, Austin, TX, pp. 921–931.h

[8] Nelaturi, S., Kim, W., Rangarajan, A., and Kurtoglu, T., 2014, "Manufacturability Feedback and Model Correction for Additive Manufacturing," ASME Paper No. DETC2014-34222.

[9] Adam, G. A. O., and Zimmer, D., 2014, "Design for Additive Manufacturing—Element Transitions and Aggregated Structures," CIRP J. Manuf. Sci. Technol., **7**(1), pp. 20–28.

[10] Oxman, R., 2004, "Think-Maps: Teaching Design Thinking in Design Education," Des. Stud., **25**(1), pp. 63–91.

[11] Dinar, M., Danielescu, A., MacLellan, C., Shah, J., and Langley, P., 2015, "Problem Map: An Ontological Framework for a Computational Study of Problem Formulation in Engineering Design," ASME J. Comput. Inf. Sci. Eng., **15**(3), p. 031007.

[12] Wölkl, S., and Shea, K., 2009, "A Computational Product Model for Conceptual Design Using SysML," ASME Paper No. DETC2009-87239.

[13] Summers, J. D., Shah, J. J., and Bettig, B., 2004, "The Design Exemplar: A New Data Structure for Embodiment Design Automation," ASME J. Mech. Des., **126**(5), pp. 775–787.

[14] Sen, C., Summers, J. D., and Mocko, G. M., 2013, "A Formal Representation of Function Structure Graphs for Physics-Based Reasoning," ASME J. Comput. Inf. Sci. Eng., **13**(2), p. 021001.

[15] Jee, H., Lu, Y., and Witherell, P., 2015, "Design Rules With Modularity for Additive Manufacturing," Solid Freeform Fabrication Symposium, pp. 1450–1462.

[16] McGuinness, D. L., and Wright, J. R., 1998, "Conceptual Modelling for Configuration: A Description Logic-Based Approach," Artif. Intell. Eng. Des. Anal. Manuf., **12**(04), pp. 333–344.

[17] Baader, F., Horrocks, I., and Sattler, U., 2008, "Description Logics," *Handbook of Knowledge Representation*, F. Van Harmelen, V. Lifschitz, and B. Porter, eds., Elsevier, Amsterdam, The Netherlands.

[18] Eiter, T., Ianni, G., Lukasiewicz, T., Schindlauer, R., and Tompits, H., 2008, "Combining Answer Set Programming With Description Logics for the Semantic Web," Artif. Intell., **172**(12–13), pp. 1495–1539.

[19] Musen, M. A., and The Protégé Team, 2015, "The Protégé Project: A Look Back and a Look Forward," AI Matt., **1**(4), pp. 4–12.

[20] Prud'Hommeaux, E., and Seaborne, A., 2008, "SPARQL Query Language for RDF," W3C Recommendation, The World Wide Web Consortium (W3C).

[21] Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., and Bizer, C., 2015, "DBpedia–A Large-Scale, Multilingual Knowledge Base Extracted From Wikipedia," Semantic Web, **6**(2), pp. 167–195.

[22] Uschold, M., and Gruninger, M., 1996, "Ontologies: Principles, Methods and Applications," Knowl. Eng. Rev., **11**(2), pp. 93–136.

[23] Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosof, B., and Dean, M., 2004, "SWRL: A Semantic Web Rule Language Combining OWL and RuleML," W3C (Member Submissions), The World Wide Web Consortium (W3C).

[24] Muggleton, S., De Raedt, L., Poole, D., Bratko, I., Flach, P., Inoue, K., and Srinivasan, A., 2012, "ILP Turns 20," Mach. Learn., **86**(1), pp. 3–23.

[25] Martinez-Cruz, C., Blanco, I. J., and Vila, M. A., 2012, "Ontologies Versus Relational Databases: Are They so Different? A Comparison," Artif. Intell. Rev., **38**(4), pp. 271–290.

[26] Samuel, K., Obrst, L., Stoutenberg, S., Fox, K., Franklin, P., Johnson, A., Laskey, K., Nichols, D., Lopez, S., and Peterson, J., 2008, "Translating OWL and Semantic Web Rules Into Prolog: Moving Toward Description Logic Programs," Theory Pract. Logic Program., **8**(03), pp. 301–322.