Artificial Intelligence for Engineering Design, Analysis and Manufacturing (2010), *24*, 245–257.

doi:10.1017/S0890060409990096

Improved knowledge management through first-order logic in engineering design ontologies

PAUL WITHERELL,[1] SUNDAR KRISHNAMURTY,[1] IAN R. GROSSE,[1] AND JACK C. WILEDEN[2]

[1]Department of Mechanical and Industrial Engineering, University of Massachusetts, Amherst, Massachusetts, USA
[2]Department of Computer Science, University of Massachusetts, Amherst, Massachusetts, USA

(RECEIVED May 8, 2008; ACCEPTED April 21, 2009)

Abstract

This paper presents the use of first-order logic to improve upon currently employed engineering design knowledge management techniques. Specifically, this work uses description logic in unison with Horn logic, to not only guide the knowledge acquisition process but also to offer much needed support in decision making during the engineering design process in a distributed environment. The knowledge management methods introduced are highlighted by the ability to identify modeling knowledge inconsistencies through the recognition of model characteristic limitations, such as those imposed by model idealizations. The adopted implementation languages include the Semantic Web Rule Language, which enables Horn-like rules to be applied to an ontological knowledge base and the Semantic Web's native Web Ontology Language. As part of this work, an ontological tool, OPTEAM, was developed to capture key aspects of the design process through a set of design-related ontologies and to serve as an application platform for facilitating the engineering design process. The design, analysis, and optimization of a classical I-beam problem are presented as a test-bed case study to illustrate the capabilities of these ontologies in OPTEAM. A second, more extensive test-bed example based on an industry-supplied medical device design problem is also introduced. Results indicate that well-defined, networked relationships within an ontological knowledge base can ultimately lead to a refined design process, with guidance provided by the identification of infeasible solutions and the introduction of "best-case" alternatives. These case studies also show how the application of first-order logic to engineering design improves the knowledge acquisition, knowledge management, and knowledge validation processes.

Keywords: Engineering Design; First-Order Logic; Knowledge Management; Ontology; Semantic Web Rule Language

1. INTRODUCTION

As engineering projects have become larger and more detailed, industry has become increasingly reliant on distributed design processes. This trend, along with advancements in computer technology, has resulted in a shift away from traditional design notebooks to more computational knowledge management systems. Such systems are often required to support the challenges of managing the rich information in distributed environments created by both geographical and organizational dispersions. Within these distributed environments, proper knowledge management has the potential to maintain the integrity of the design process by providing an engineer with a more complete understanding of a design (Erickson et al., 1997; Morris, 1998).

Capable of providing formalized, adaptable architectures, ontologies have become a popular means for managing and distributing knowledge bases. In recent works, the authors have explored these ontological knowledge management techniques, leading to the development of frameworks for engineering analysis and engineering design optimization (Grosse et al., 2005; Witherell et al., 2006). These frameworks of ontologies successfully shared and managed both engineering analysis and optimization knowledge, although their application in engineering design revealed some understandable shortcomings. As with most knowledge management systems, in the absence of built-in validation mechanisms, these ontologies allow knowledge to be incorrectly instantiated and improperly used. Moreover, the large amount of knowledge instantiation associated with developing rich ontological knowledge bases presents a formidable task.

This paper presents methods for helping maintain the reliability of information within an ontological knowledge base in addition to assisting in the knowledge instantiation process,

Reprint requests to: Sundar Krishnamurty, Department of Mechanical and Industrial Engineering, 160 Governor's Drive, University of Massachusetts, Amherst, MA 01003-2210, USA. E-mail: skrishna@ecs.umass.edu

thereby demonstrating the applicability and usefulness of first-order logic. To this end, first-order logic is used to address three key issues in knowledge management that ontologies by themselves do not:

1. corroborating knowledge instantiations,
2. maintaining consistency during the knowledge instantiation process, and
3. minimizing redundancy in the knowledge instantiation process.

Distributed design requires the contribution of many, including experienced and inexperienced, new and old. Instantiating and utilizing uncorroborated knowledge can be detrimental to the design process. When capturing and reusing information, the underlying conditions of design content (e.g., modeling assumptions necessary for dimensional reductions in models, assumptions necessary for feature suppressions, etc.) are not always identified or understood, often becoming foregone conclusions, or accepted without consideration. As designs evolve and changes mount, ensuring transparency of, and satisfaction of, underlying conditions for models (such as engineering analysis models and manufacturing models) is critical for knowledge reuse. Therefore, identifying and understanding such conditions early in the knowledge capturing process is vital for ensuring the integrity of the design process.

Another aspect of knowledge instantiation, ensuring consistency, is critical when navigating, retrieving, or operating on knowledge in an instantiated knowledge base. Inconsistent knowledge can lead to disastrous results (Euler et al., 2001). To address this problem, this work presents a comprehensive approach that enables knowledge bases to maintain consistency by utilizing ontology domain concepts in combination with the ability to compare values across domains.

Knowledge instantiation within large knowledge frameworks can become progressively more time-consuming, yet repetitive, with design information often sharing common values. For instance, two optimization models based on the same product may share related design and analysis models, the same design parameters, and the same objectives. Relationships between properties can be created and asserted to simplify and speed up the instantiation process. Such relationships have the capability to transfer known values, thus avoiding unnecessary repetition. These facilitating methods ease the task of creating similar knowledge while minimizing the possibility of human error.

The following sections present an overview of logic in engineering design with particular emphasis on the use of ontologies and Horn clauses within the Semantic Web, followed by a detailed description of improved knowledge management using first-order logic. Then, the unique features of the resulting ontological tool for supporting the engineering design process (OPTEAM) are highlighted with the help of two case studies and the results are discussed.

2. LOGIC IN ENGINEERING DESIGN

2.1. Expert systems

Logic has been previously adopted in one form or another by many in the engineering community, most notably in the development of expert systems. First adopted by the artificial intelligence community, expert systems emerged with the development of early rule-based systems such as Mycin (Buchanan & Shortliffe, 1984), an expert system used for diagnosing infectious blood diseases in the early 1970s. Engineering adaptations soon emerged, including those by Gottlob and Nejdl (1990), Shephard et al. (1990), Turkiyyah and Fenves (1996), and Becker and Kaepp (1997). In the engineering design community, Brown (1985) proposed the Design Specialists and Plans Language for developing "routine" designs within an expert system. In these systems, the application of logic often resulted in unsuccessful attempts to "automate" portions of the product development process, such as designing products "at the push of the button."

Despite being pursued heavily early on, it was realized expert systems such as these often encountered problems, for example, noise resulting from certainty factors and leading to infeasible results. This led to some improvements in expert system design, for instance, here Bayesian statistics were introduced to address uncertainties (Spiegelhalter et al., 1993). Consequently, over time, many have come to abandon the expert system approach, realizing that in most cases it is not yet feasible to achieve desirable solutions without significant human input. Although this may no longer hold true as technology advances, it is indeed a limitation currently recognized (Prasad, 2004; Rogers, 2004). Without the necessary human factor, logic-driven design automation often produces infeasible or unusable results, leading many researchers to reevaluate their approach.

In general, the effectiveness of an expert system is measured by two main components (Dos Santos & Mookerjee, 1991): the *knowledge base* and the *control strategy* that affects the processing order of the knowledge base. The separation of these two components is essential when differentiating an expert system from a knowledge-based framework. The development of a "control strategy" for processing the knowledge base became a major obstacle in the advancement of expert systems, leaving the knowledge base the focus of many new approaches. This perspective forms the basis for the work presented in this paper.

2.2. Ontologies and knowledge-based engineering

Because knowledge bases are an essential part of expert systems, much of their advancement was achieved in tandem with the development of expert systems. Early works with the development of knowledge bases in engineering focused on product knowledge representation, including work by deKleer and Brown (1983), Iwasaki and Chandrasekaran (1992), Alberts and Dikker (1992), Henson et al. (1994),

https://doi.org/10.1017/S0890060409990096 Published online by Cambridge University Press

Goel, Bhatta, and Stroulia (1996), Goel et al. (1996), Qian and Gero (1996), Ranta et al. (1996), and Umeda et al. (1996). These works laid the groundwork for formally representing product information, such as the high-level divisions of product information representation into form, function, and behavior adopted in the NIST Design Repository Project (Szykman et al., 1998) and again in the design Repository developed at the Missouri University for Science and Technology (Bohm et al., 2006). The core-level knowledge representation consisting of objects and relationships adopted by the NIST Design Repostitory (Szykman et al., 2000) corresponds to a fundamental concept of a formal information structure that, when instantiated, represents knowledge about a particular domain that can then be operated upon by computers and humans alike (Grosse et al., 2005). Works such as these highlight the advantages offered through the formal representation of knowledge in product design, specifically when pertaining to ontologies.

More recently, Bullinger et al. (2005) demonstrated the benefits of capturing knowledge using ontologies. Although noting the importance of capturing knowledge to create a more user-friendly design environment, they also acknowledge the possible advantages accompanying the ability to operate on ontological knowledge bases. Bullinger et al. recognize that the full power of ontological-based knowledge management resides in support of not only knowledge acquisition but also the ability to analyze and reason on this knowledge. Bullinger and colleagues point out that "By describing the typical methods of an application domain as well as the associated requirements appropriately a software agent (SWRL reasoner) can be used to recommend use of methods or materials." The present paper extends the work of Bullinger et al. by exploring methods for utilizing captured knowledge.

Researchers at the University of Michigan adopted logic to facilitate interoperability during product development. Patil et al. (2005) propose a Product Semantic Representation Language (PSRL) to enable semantic interoperability across application domains using mathematics and corresponding reasoning. PSRL is based upon DARPA Agent Markup Language (DAML) + Ontology Inference Layer (OIL), precursors to the Web Ontology Language (OWL; www.w3.org/TR/owl-features/). Patil et al. (2005) note that description logics (DLs) such as DAML + OIL may not be able to completely represent all relevant design knowledge and that more powerful first-order logic may be best suited for full and accurate representation.

Collaborative work by Wayne State University, Chonnam National University, and University of Pittsburgh resulted in the implementation of Semantic Web Rule Language (SWRL) rules within a knowledge base for assembly design (Kim et al., 2006). This implementation uses the expressivity of the Semantic Web and SWRL to partially automate portions of the assembly design process through logical assertions. Kim et al. (2006) illustrate the effectiveness of capturing assembly process knowledge using ontologies and domain concepts as opposed to traditional data syntax. Kim et al. (2006) also illustrate how ontologies and SWRL facilitate collaboration in the assembly design process. Using their ontology, classifications are made within the assembly design process and otherwise unspecified constraints are inferred.

Extending their initial work, Kim et al. (2006) developed an information-sharing paradigm, called semantic assembly design modeling, to facilitate product development collaboration. In this paradigm, particular constraints are defined that must be satisfied during the assembly design process and then SWRL rules are able to imply or assert other constraints not initially identified. In this manner, they are able to facilitate the knowledge acquisition process with assertions using first-order logic. While acknowledging the necessity for capturing design rationale, or "higher level" knowledge, Kim did not propose any methods for operating on it.

At the University of Massachusetts Amherst, preliminary work with engineering analysis and optimization knowledge resulted in the adoption of ontologies and the development of two separate knowledge-capturing tools, Ontology for Engineering Analysis Models (ON-TEAM) and Ontology for Optimization (ONTOP). ON-TEAM (Grosse et al., 2005; Witherell et al., 2006), the initial application, provides engineers with the ability to capture both "higher" and "lower" level engineering analysis model knowledge, such as modeling assumptions and parameter values, respectively. ONTOP was subsequently created to facilitate engineering design optimization. ONTOP provides engineers the ability to quickly identify a feasible method for a given design optimization problem and enables the user to instantiate and store relevant optimization modeling knowledge.

3. A BRIEF INTRODUCTION TO FIRST-ORDER LOGIC

Multiple languages used by the artificial intelligence community provide full or almost full expressivity of first-order logic, including F-Logic (Kifer & Lausen, 1989), CARIN (Levy & Rousset, 1998), and KIF (Genesereth & Fikes, 2001), among others. Each of these languages brings its own unique abilities when creating and operating on a knowledge base. The DL-based OWL, however, has emerged as a leading implementation language for developing ontological knowledge frameworks. Primarily because its Semantic Web ties, OWL allows knowledge to be easily shared over the Web. OWL uses three different expressive sublanguages, OWL Lite, OWL DL, and OWL Full. Although all are based on DL, OWL DL most closely corresponds to it. OWL DL corresponds with SHOIND-n DL, a fragment of classical first-order logic (Tsarkov et al., 2004).

Considered a subset of first-order logic (Borgida, 1996), DL (Fig. 1) is a language used in knowledge representation to express concepts and concept hierarchies. DL provides a method for representing the terminological knowledge of an application domain in a formal, structured manner. What makes DL attractive, however, is both its decidability and tractability.

https://doi.org/10.1017/S0890060409990096 Published online by Cambridge University Press

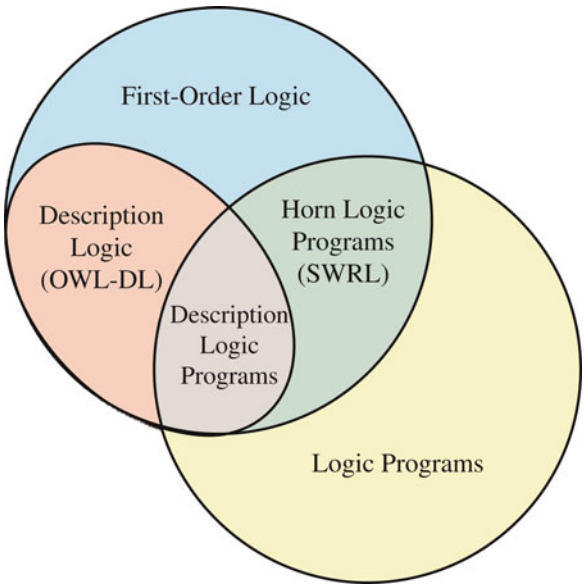

Fig. 1. An illustration of logic types based on Grosof et al. (2003). [A color version of this figure can be viewed online at journals.cambridge.org/aie]

The decidability of DL allows consistency checks to be performed on ontologies, assuring they are well formed. The tractability, or computational tractability of DL, refers to the ability to perform automated reasoning on the knowledge base using realistic computing resources and reasonable amounts of time.

One of the most attractive aspects of OWL is its ability to be extended by the SWRL (Horrocks et al., 2005). SWRL, introduced by a W3C (http://www.w3.org/) proposal to increase the expressivity of the Semantic Web, extends OWL both syntactically and semantically. Although other first-order languages may possess more expressivity than SWRL, they do not provide the same level of practicality provided by SWRL's complimentary nature to OWL and inherent ties to the Semantic Web. Developed from Rule ML (Wagner et al., 2003), the SWRL extension of OWL creates a much more expressive language than either OWL or Horn clauses individually.

SWRL is based on Horn clauses that allow for such relationships as "if–then" statements. A Horn clause is defined as a clause with at most one positive literal (Horn, 1956), thereby any number of if–then conditions leads to only one conclusion. Horn clauses are important in theorem proving in that the conjunction of two Horn clauses is a Horn clause. A simple example of a Horn clause is the following:

$$(p \wedge q) \Rightarrow s$$

This reads as if p and q, then s, or given a p, and given a q, the conclusion s can be drawn. Such inferences are essential in providing additional functionalities to an OWL knowledge base, as SWRL allows further conclusions to be drawn based on existing knowledge. The expressive power of SWRL also allows "existentials" to be expressed in the head of a rule, thus extending beyond the expressive power of Horn clauses (Tsarkov et al., 2004).

SWRL provides "built-in" capabilities, which gives the language additional expressivity independent of the Horn-like rules. These built-ins include mathematical functions such as multiply (swrlb:multiply), divide (swrlb:divide) and comparisons such as greater than (swrlb:greaterThan) or less than (swrlb:lessThan). Other built-ins include such everyday information as date and time, as well as string operators such as *match* and *contains*.

While providing a means of increasing expressivity, SWRL's Horn-like rules also potentially impair OWL-DL's DL-based decidability. An ontology is considered undecidable when classes exist in which membership cannot be decided by an algorithm (www.nist.gov/dads/HTML/undecidableLanguage). Schmidt-Schaub's (1989) simulations of role values maps are an example of how the expressiveness of SWRL can lead to undecidability. Such scenarios create problems when employing traditional reasoners to check ontology consistency. This potential lack of decidability, however, is avoided by using relatively simple implementations of SWRL. To ensure decidability, a translation approach using first-order logic may also be used (Tsarkov et al., 2004).

Figure 2, the Semantic Web Stack, shows the roles of OWL and SWRL. OWL, built semantically on the Resource Description Framework (RDF) and RDF schema, provides the ontology language (upper orange stack). This language captures the concepts of ontologies within the infrastructure of the Semantic Web. On top of the "Ontology" stack lie "Rules" and "Logic framework." These additional layers of the Semantic Web stack are expressed using SWRL. OWL is used to formally represent knowledge associated with the design process, but SWRL provides the ability to conduct logical inferencing on this knowledge.

As the Semantic Web infrastructure continues its growth, the number of supporting applications, such as reasoners and inference engines, also continues to grow. OWL reasoners are tools mainly utilized to ensure the consistency and classify the instances of an OWL ontology. Inference engines

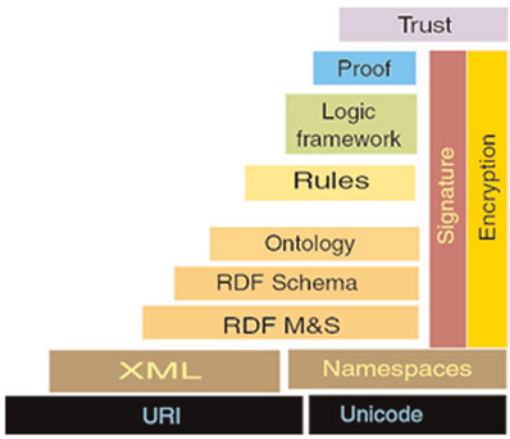

Fig. 2. The Semantic Web Stack from Tim Berners-Lee presentation for the Japan Prize, 2002. [A color version of this figure can be viewed online at journals.cambridge.org/aie]

https://doi.org/10.1017/S0890060409990096 Published online by Cambridge University Press

provide the ability to draw conclusions and make assertions based on SWRL rules. There are currently several alternatives for implementing SWRL rules within an OWL knowledge base. Some OWL reasoners, such as RacerPro (Haarslev et al., 2004), Pellet (Sirin et al., 2004), and Hoolet (Tsarkov et al., 2004), among others, have extended their capabilities to support SWRL inferencing. Ontological development tools such as Protégé (Gruber, 1994; Noy et al., 2001; Gennari et al., 2003) and Swoop (Kalyanpur et al., 2005) have provided reasoning and inference capabilities to their established knowledge bases. Protégé has built-in support for SWRL using the SWRLTab and JESS rule engine (Friedman-Hill, 2003), and Swoop has integrated the reasoner Pellet. The availability of tools such as these allow for the practical creation of an ontological knowledge base for engineering design that is facilitated by first-order logic.

4. LOGIC-BASED METHODS FOR IMPROVED ONTOLOGICAL KNOWLEDGE MANAGEMENT

4.1. Insights into developing intelligent frameworks

The application of Horn rules provides additional expressivity and generates many new possible applications for an ontological knowledge base. For example, first-order logic now allows the knowledge acquisition process to be guided by identifying uncorroborated and inconsistent knowledge, as well as facilitated by inferring and asserting values when instantiating a knowledge base. During the development of these methods, two learned insights helped better provide the foundation for the efficient development of intelligent ontological knowledge bases:

1. When capturing abstract knowledge, providing deliberately structured frameworks allows such knowledge to be most proficiently employed.
2. When dealing with multiple distributed frameworks, it is prudent to designate a location for asserting instances that have been inferred as unsubstantiated or inconsistent knowledge.

To address the first insight, the difference must first be understood between lower and higher level knowledge instantiations. Lower level knowledge includes very "basic" information, such as values of parameters and constraints. Higher level knowledge includes the more abstract knowledge, such as assumptions and idealizations. Higher level knowledge is traditionally stored only as text strings, resulting in only human-interpretable knowledge. However, by capturing these text strings as separate individual instantiations, higher level knowledge can be made not only human interpretable but also machine interpretable. This can be achieved by tightly modeling these instances of higher level knowledge within specific ontology domains, allowing inherent associations to be made with the knowledge. This leads to an increase in more logic-friendly object-type knowledge and enables effective operation on higher level knowledge. For instance, consider modeling assumptions. In developing a model, many types of assumptions may be made, such as on the geometry, on the loading, or on material properties. Distinguishing between these types of assumptions is important when corroborating knowledge during the knowledge instantiation process. The ability to capture higher level knowledge in domains leads to a much richer source of information than would capturing a conglomerate of text strings.

The second insight addresses complications that arise when developing an intelligent knowledge framework in the open world of the Semantic Web. When exclusively using reasoners and OWL, restriction classes are used to classify types of knowledge. Therefore, when developing a knowledge base for strictly reasoning on restriction classes, it is advantageous to create large amounts of classes and separate different types of knowledge. This paper proposes that scenarios often exist when it is more practical, if not necessary, to use a Horn rule to classify a knowledge instantiation while maintaining the integrity of a knowledge framework, as opposed to creating additional restriction classes. However, because the characteristics of SWRL, knowledge instantiations cannot simply be "reclassified." This means that an instance of knowledge within an OWL framework cannot simply be moved from one class to another using SWRL. Human input is therefore necessary to reclassify knowledge. To support human input, the introduction of an umbrella class provides a means for reclassifying knowledge. This umbrella class, such as a "Violations" class, contains asserted knowledge instantiations that do not comply with developed rules. The umbrella class simply becomes an additional superclass to an asserted instance. The umbrella class concept is meant for assisting in guiding and validating the knowledge capturing process more than facilitating knowledge acquisition, and thus, becomes an important concept when developing an "intelligent" knowledge base to support the engineering design process.

4.2. Development of intelligent methods

In contrast to those described in Section 2, the methods proposed in this paper present a unique approach to the application of logic in engineering design. This approach does not automate processes and leave important decisions to computer algorithms, but instead serves as a facilitator to the engineering design process. The introduction of logical inferencing to explicit engineering design and optimization knowledge frameworks provides capabilities beyond those achieved in earlier works that captured knowledge through ontologies.

In Section 1 we noted that first-order logic could be used to address three key issues in knowledge management that ontologies alone do not address:

1. corroborating knowledge instantiations,
2. maintaining consistency during the knowledge instantiation process, and

https://doi.org/10.1017/S0890060409990096 Published online by Cambridge University Press

3. minimizing redundancy in the knowledge instantiation process.

These three issues will now be addressed using methodologies discussed in this paper.

The first to be discussed is how to address the issue of corroborating knowledge instantiated within a distributed framework. One way of corroborating knowledge, especially lower level knowledge, within an OWL framework is with the use of SWRL built-ins. Built-ins, such as "swrlb:greaterThan" or "swrlb:lessThan," allow for comparisons of instantiated values. For instance, values of parameters can be constrained by comparing them with limitations imposed by SWRL rules. When a SWRL limitation is breached, the responsible value is then asserted into an umbrella class. The extensive library of SWRL built-ins allows for many such comparisons. The corroboration of higher level knowledge is a little less straightforward. Though still based on the ability of inferencing to compare instances, corroboration of higher level knowledge also depends on the explicitness of the domains used to capture the higher level knowledge, such as the assumptions example given in Section 4.1. As the domains used to capture higher level knowledge become increasingly explicit, the identified uncorroborated knowledge becomes more specific. The amount of achievable knowledge corroboration ultimately depends on the extensiveness of the knowledge framework.

The second issue, maintaining knowledge consistency, is achieved using both OWL and SWRL. Using OWL, property values can be restricted to come from only identified classes. This is useful for achieving such objectives as ensuring only continuous parameters are used when applying a continuous optimization technique. However, class structure does not always allow for such restrictions, and such restrictions are not always desired. For instance, consider the task of assigning a unit to a model parameter. Were the units ontology structured to distinguish between types of measurement, such as length or mass, instances belonging to the same class of such an ontology may include both meter and foot. Even if a second ontology were developed to distinguish between English and metric units, this would not solve the problem of distinguishing between millimeter and meter. SWRL has the ability to address situations such as this by comparing instance values. Were one model parameter to use the unit millimeter, while another were to use the unit meter, SWRL has the ability to identify that different instances of units were used. Once it has been determined that one unit value was not the same instance as another unit value, the parameters are inserted into the umbrella class so the situation can be rectified and unit consistency can be insured.

Finally, the third issue identified, minimizing redundancy in the knowledge instantiation process, involves utilizing the inference capabilities of SWRL. As explained in the Section 1, these methods are most useful when basing a new knowledge instantiation on an existing one. Domains in an ontological framework often share many of the same properties. The creation of a property for the purposes of identifying when there is redundant information in a new knowledge instantiation allows an engineer to choose when to transfer knowledge from one instance to another. The value of such a "based_on" property decides when and what existing knowledge is to be passed. This method is available only when the existing knowledge instantiation shares at least one of the same properties as the new knowledge instantiation.

Now that the methodology for addressing the identified issues has been explained, the development of a tool based on this methodology will be described in Section 5.

5. IMPLEMENTATION: DEVELOPMENT OF OPTEAM

The integration of ON-TEAM and ONTOP laid the groundwork for a linked knowledge base that incorporates optimization models, analysis models, and geometric models. Conjoining domains to form a common knowledge base allows for a much more inclusive framework for the facilitation of both engineering design optimization and engineering analysis. By adding Horn rule functionality to the ontological knowledge base, the tool OPTEAM was developed, an intelligent and extensive product design and development knowledge base with knowledge easily stored and readily accessible using ontologies.

OPTEAM was developed and implemented in the ontological development tool Protégé. Protégé provides an easy to use graphical interface for manipulating both OWL and SWRL. Protégé's open source code and plug-in capabilities allow for the development of independent tools, such as the automatic tech report generator (Kanuri, 2007), which allows an engineer to recapture much of the time spent instantiating modeling knowledge by automatically generating technical reports from the captured knowledge. Although developed in Protégé, OPTEAM's foundation of the Semantic Web's OWL and SWRL allow it to exist independently.

OPTEAM facilitates the engineering analysis and design processes by asserting values of similar properties across classes and instances. Creating an analysis model based upon a geometric model is an example of this. In the design environment, it is often the case that an analysis model is based upon a geometric model, usually a computer-aided design (CAD) model. Because of this dependency, much of the knowledge between the two may also be shared, including geometry, modeling assumptions, and idealizations. By recognizing this dependency and using a "based_on" property, logical relationships assert the pertinent knowledge within a new knowledge instantiation.

To help successfully guide the knowledge instantiation process, as well as identify uncorroborated knowledge within an existing knowledge base, a "Violations" class was created within the OPTEAM framework. When necessary, relationships assert one or more instances into the "Violations" class, identifying when uncorroborated knowledge has been created

https://doi.org/10.1017/S0890060409990096 Published online by Cambridge University Press

and providing the engineer with the opportunity to address any "concerns" OPTEAM may have. Although "Violations" instances are automatically inserted, decisions on how to handle these instances are left to the engineer. A property of this "Violations" class is to identify which property value of the instance created the violation. Another property is able to identify what other instances of knowledge caused the violation. This method is the basis for identifying uncorroborated knowledge instantiated within the OPTEAM framework due to design modifications or changes to preexisting knowledge. For example, analysis or optimization models may contain instances of knowledge that are affected by a modification of a dimension within a CAD model.

To support the ability to operate on higher level knowledge concepts, taxonomies of idealizations, idealization justifications, and assumptions have been created. Proceeding on the notion that all idealizations are based on assumptions, explicit rules are used to determine whether an idealization is justified. Because all idealizations are based on assumptions, assumptions possessed by models are compared with assumptions required by an idealization. When a model no longer supports all assumptions required by an idealization, the idealization becomes invalid, and the model is asserted into the "Violations" class. Thus, when models are created, they support any assumptions made by original idealizations. As modifications are made, model assumptions and idealizations are altered, but the requirement that all idealizations are supported by assumptions continues to be enforced.

Similar rules are applied to recognize when a model's limitations have been reached, providing a unique method for model management. These limitations are identified not by what a model is unable to support, but by what a model does and will support. In the "open-world" framework of OWL, and its lack of support for negation as failure, this is very important. The ability to realize when a model limitation has been reached can guide an engineer in his or her decisions throughout the design process.

Many rules have been and are being developed and implemented in support of OPTEAM. Table 1 defines some of the applications of SWRL rules and their relevance. The rules are numbered and will later be referenced when describing the example instantiation of an I-beam. These rules were all created to operate on the developed framework, independent of specific knowledge instantiations.

Table 1. *Examples of implemented SWRL rules*

Rule Application	Description	SWRL Example
1. Populating a library of instances	Automatically populate a library of models, images, etc.	ModelA(?*y*) ⇒ LibraryofModelA(?*y*)
2. Unit consistency	Identifies unit inconsistencies	Parameter(?*x*) ∧ isConstrainedBy(?*x*, ?*y*) ∧ hasUnits(?*x*, ?*z*) ∧ hasUnits(?*y*, ?*a*) ∧ differentFrom(?*z*, ?*a*) ⇒ Violation(?*x*)
3. Associating models	Associate models through a common model or models.	Model(?*x*) ∧ Modelof(?*x*, ?*y*) ∧ Model(?*z*) ∧ Modelof(?*z*, ?*y*) ⇒ hasAssociatedModel (?*x*, ?*z*) ∧ hasAssociatedModel(?*z*, ?*x*)
4. Propagation of properties	Propagate properties of a child model to a parent model.	Submodel(?*x*) ∧ Model(?*z*) ∧ hasParameter(?*x*, ?*y*) ∧ hasParentModel(?*x*, ?*z*) ⇒ hasParameter(?*z*, ?*y*)
5. Creation of supporting knowledge	Create knowledge that is implied by the creation of an instance.	ConstrainedModel(?*x*) ∧ hasVariable(?*x*, ?*y*) ∧ isConstrainedBy(?*y*, ?*z*) ⇒ hasConstraint(?*x*, ?*z*)
6. Identifying constraint violations	This rule example sets an upper limit on a variable.	Constraint(?*x*) ∧ hasValue(?*x*, ?*y*) ∧ Parameter(?*z*) ∧ hasValue(?*z*, ?*a*) ∧ Greater/ Less(?*x*, *Greater*) ∧ isConstrainedBy(?*z*, ?*x*) ∧ swlb:lessThan(?*y*, ?*a*) ⇒ Violation(?*x*)
7. Calculation of objective value	Find the current value of an objective function such as $f(x) = 3x - 5y$.	ObjectiveFucntion(?*x*) ∧ Parameter1(?*x*, ?*z*) ∧ hasValue(?*z*, ?*a*) ∧ Parameter2(?*x*, ?*b*) ∧ hasValue(?*b*, ?*c*) ∧ Parameter3(?*x*, ?*d*) ∧ hasValue(?*d*, ?*e*) ∧ Parameter4(?*x*, ?*f*) ∧ hasValue(?*f*, ?*g*) ∧ swrlb:multiply(?*h*, ?*a*, ?*c*) ∧ swrlb:multiply(?*i*, ?*e*, ?*g*) ∧ swrlb:subtract(?*y*, ?*h*, ?*i*) ⇒ hasValue(?*x*, ?*y*)
8. Creation of new models based on existing ones	This is an example of some of the knowledge that may be passed when using existing models as templates for other models.	Model(?*x*) ∧ Modelof(?*x*, ?*y*) ∧ IntendedFor(?*x*, ?*d*) ∧ hasParameter(?*x*, ?*a*) ∧ hasAssociatedModel(?*x*, ?*b*) ∧ NewRevision(?*x*, ?*z*) ⇒ Model(?*z*) ∧ Modelof(?*z*, ?*y*) ∧ hasParameter(?*z*, ?*a*) ∧ hasAssociatedModel(?*z*, ?*b*) ∧ IntendedFor(?*z*, ?*d*)

6. CASE STUDIES

6.1. Engineering case study 1: The design of an I-beam

The instantiation of a knowledge base for the design and optimization of an I-beam, shown in Figure 3, is used to demonstrate how the addition of logical operators eases much of the knowledge capturing process, while also providing guidance. This example exploits many of the rules illustrated in Table 1. In this example, the initial problem statement was to minimize the cross section of an I-beam subject to deflection and stress constraints, as seen in Figure 3. The units used here are English, and the length is measured in inches and pressure in pounds per square inch.

https://doi.org/10.1017/S0890060409990096 Published online by Cambridge University Press

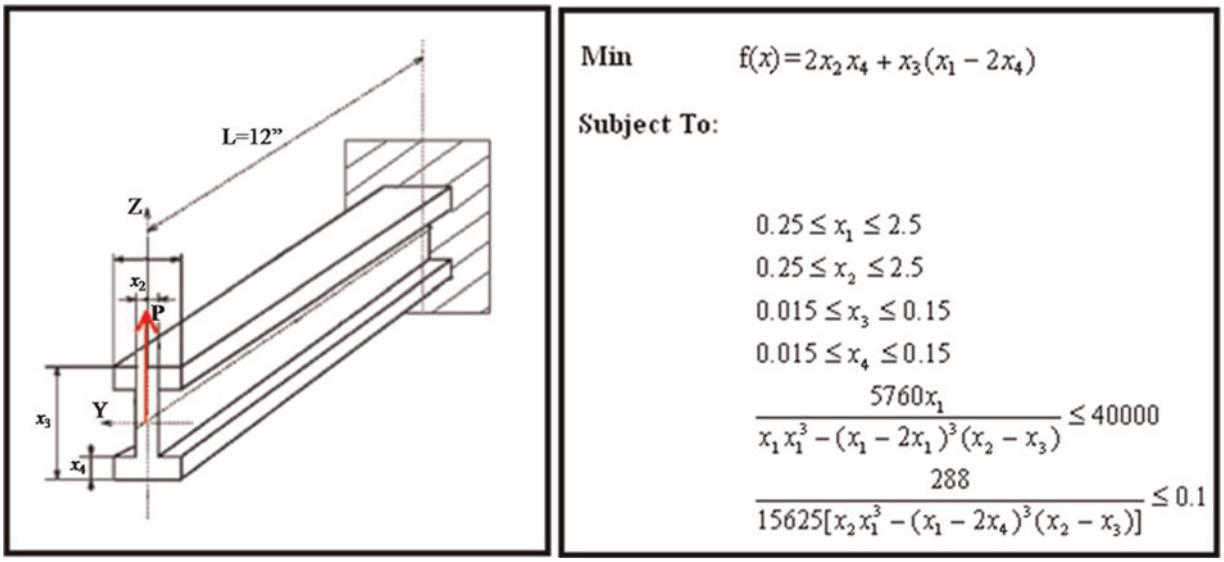

Fig. 3. Optimizing and I-beam subject to constraints. [A color version of this figure can be viewed online at journals.cambridge.org/aie]

The knowledge capturing process for the I-beam began with the simple instantiation of a product instance. This product instance served as the root instance during the development of the I-beam, with all related knowledge either directly or indirectly linking to it. This initial step was followed by the development of a CAD model of an I-beam while capturing the knowledge involved in creating its geometry. For a simple I-beam problem, the lower level knowledge was limited to values of thickness, height, width, and length. The higher level knowledge included a brief description of the model, as well as any idealizations made, such as the neglecting of weld geometry during the modeling process.

After creating the geometric representation of the I-beam, the next step was to develop the initial analysis problem. Using a strictly OWL knowledge framework, this step would require navigating through the ontology and beginning a fresh instantiation of an analysis model, in this scenario a finite element analysis model. However, using a rule similar to rule 8 (the exact properties and classes used may vary) from Table 1 in the OPTEAM framework, much of the knowledge instantiation was automated because the analysis model was based on the existing geometric model. For instance, they share the same related models, are based on the same initial product, share many of the same parameters, and share properties such as who the model is intended for. Automatically instantiating common parameters and their values insured the analysis model was an accurate representation of the geometric model. The automatic instantiation of this knowledge reduced the amount of time required to create a successful knowledge instantiation.

After creating the basic instantiation of the finite element analysis model using the shared attributes, model-specific information was added. This knowledge consisted of any higher level knowledge including assumptions, such as the negligible effect of the beam welds, and resulting idealizations, such as the suppression of the welds. Again, OPTEAM facilitated the knowledge instantiation process. In this scenario associations between analysis, geometric and optimization models were made using rule 3 from Table 1. Rules 1 and 5 also facilitated the knowledge instantiation by passing common values. Rule 2 guided the knowledge gathering process by providing unit consistency checks.

OPTEAM used a SWRL rule similar to rule 8 when creating a new instance of an optimization model. Based on the results of the initial analysis, the initial parameters were set for the optimization of the I-beam. These parameters included the geometrically related parameters, the material property parameters, and the initial conditions set forth by the analysis results, such as stress and deflection. These initial conditions were important in determining appropriate optimization methods. For this example, the objective was to minimize the cross section of the beam. Although OWL was used to ensure the proper optimization method was chosen between continuous or discrete based on the classification of the optimized parameters, a SWRL rule was used to ensure a constrained method was used rather than an unconstrained method based on the existence of constraints in the given problem.

The I-beam optimization problem was subjected to both stress and deflection constraints. Although OWL offers the ability to capture information about these constraints within the knowledge base, SWRL has the ability to flag violated constraints. Using SWRL rules similar to rule 6 in Table 1 and SWRL built-ins, relationships were defined between parameter values and constraint values. These relationships identified when a constraint was violated by asserting an instance of the I-beam optimization model into the "Violations" class while identifying what constraint was violated. This again demonstrates how SWRL adds a semblance of intelligence to an OWL knowledge base.

https://doi.org/10.1017/S0890060409990096 Published online by Cambridge University Press

The I-beam example highlighted a selection of the rules offered by OPTEAM. Together, these rules provide a significant improvement over OWL frameworks in knowledge management and capturing capabilities while also simplifying much of the process. An example of what SWRL rules look like when implemented in Protégé is seen Figure 4, showing the implementation of SWRL rules used in the I-beam knowledge instantiation.

6.2. Engineering case study 2: The design of a pediatric left ventricular assist device (PVAD) impeller

The topological optimization of a PVAD impeller is used to further exhibit the capacity of OPTEAM. Unlike the I-beam example, however, the PVAD impeller example concentrates more on the operation on higher level knowledge.

The PVAD impeller example was introduced (Witherell et al., 2006) to demonstrate the comprehensive knowledge that can be captured by an OWL-based framework. Many assumptions and idealizations led to the final topological optimization of the PVAD impeller. Beginning as a three-dimensional (3-D) impeller with a sophisticated blade design, the blades were deemed to have a negligible effect on the overall stress experienced by the impeller. Therefore, their complex geometry was suppressed. This suppression became a model idealization. The idealization of blade suppression resulted in an axisymmetrical analysis model where one did not exist before. This symmetry allowed the creation of a two-dimensional (2-D) model that accurately represented the behavior of the 3-D model in the context of the analysis objectives. This 2-D model was then used to run both stress and modal analyses on the PVAD impeller, as well as a topological optimization.

OPTEAM possesses the ability to not only capture the knowledge associated with the 2-D representation of the PVAD impeller but also identify when the knowledge is no longer valid. The initial idealization of the PVAD impeller was the suppression of the impeller blades. For this suppression to be made, it was assumed that the current blade structure had a negligible effect on the intended modal and stress analyses. However, if a computational fluid dynamics analysis were run, blade suppression would be detrimental to acquiring accurate flow results. In such a case, the blade suppression would no longer be valid. Furthermore, if a design change was made, such that the size and shape of the blades were altered, then blade suppression may no longer be an appropriate idealization for either the stress or modal analyses. Although a single engineer carrying out an analysis may recognize when an idealization becomes invalid, such occurrences will become increasingly difficult to identify in a distributed environment, such as the Semantic Web. OPTEAM provides a platform addressing such situations.

In the presented design scenario of the PVAD impeller, an existing impeller design was gently modified so it could be

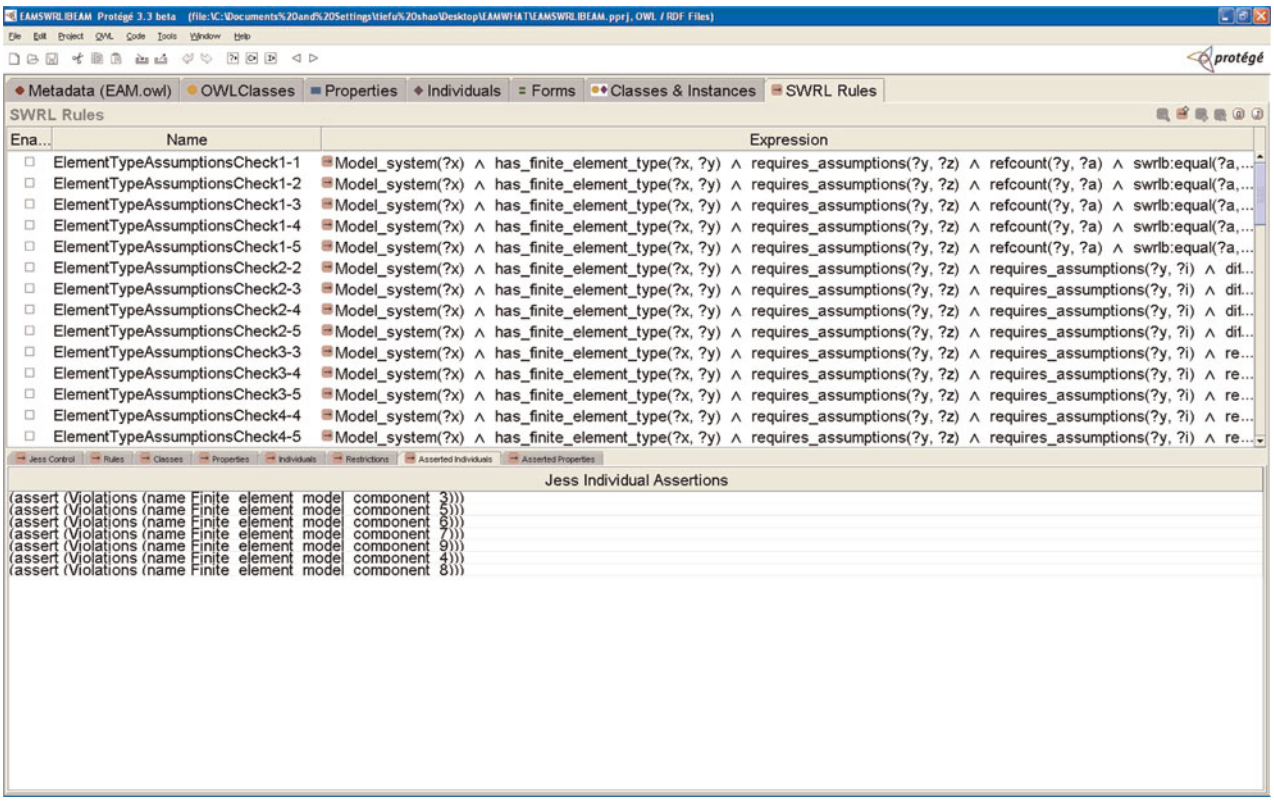

Fig. 4. Implemented SWRL rules in SWRL Tab. [A color version of this figure can be viewed online at journals.cambridge.org/aie]

https://doi.org/10.1017/S0890060409990096 Published online by Cambridge University Press

used in a new yet similar environment. The modifications to the design of the impeller included a mild increase in the size of the blades. The new environment also introduced a slightly increased flow rate experienced by the impeller. An existing finite element model was copied when beginning the new stress analysis. However, because of the blade alterations, the new finite element model could no longer support the assumptions of "Negligible blade mass" and "Negligible flow rate." The new model now had a new list of assumptions.

The 2-D finite element model of the impeller required an "Impeller axisymmetry" idealization to be made during the creation of this impeller. The existence of the "Impeller axisymmetry" idealization required the "Impeller blade suppression" idealization, because of the complex geometry of the blade. Therefore, all assumptions required for the "Impeller blade suppression" idealization were also necessary for an "Impeller axisymmetry" idealization to be made. The removal of the "Negligible blade mass" and "Negligible flow rate" assumptions triggered a SWRL rule that stated that all assumptions required to make an idealization must also be present in the model, otherwise a violation occurred. Because the "Negligible blade mass" and "Negligible flow rate" assumptions were no longer made, OPTEAM was able to identify that the "Impeller blade suppression" idealization made by this model was no longer applicable. Therefore, it was able to deduce that the "Impeller axisymmetry" idealization was also no longer valid. Because these idealizations were no longer supported, the 2-D axisymmetrical impeller model was asserted into the "Violations" class, as seen in Figure 5.

Figure 6 is a screen shot of an instance of an asserted violation of a 2-D axisymmetrical impeller model. It is seen that the instance belongs to two classes, the "Finite_element_model_component" class and the "Violations" class. The "Violations" class property values show that OPTEAM was able to identify that a violation has occurred and what assumptions and idealizations were no longer valid. This awareness helps ensure the integrity of the design process during its analysis model development phase. However, if it were deemed that the model supported these assumptions and idealizations, the violated assumptions would be added to the model assumptions, and the asserted violation could be removed.

7. DISCUSSION

This paper demonstrated how the application of first-order logic can be used to guide and facilitate the knowledge acquisition process. Other advantages also exist. The methods presented in this paper are not intrinsic to their demonstrated implementations; they can be adapted to additional applications throughout the engineering design process as well. For instance, logic can identify when a certain decision may be beneficial over another, such as choosing applicable techniques in an optimization problem, and identifying what advantages one technique may provide over another.

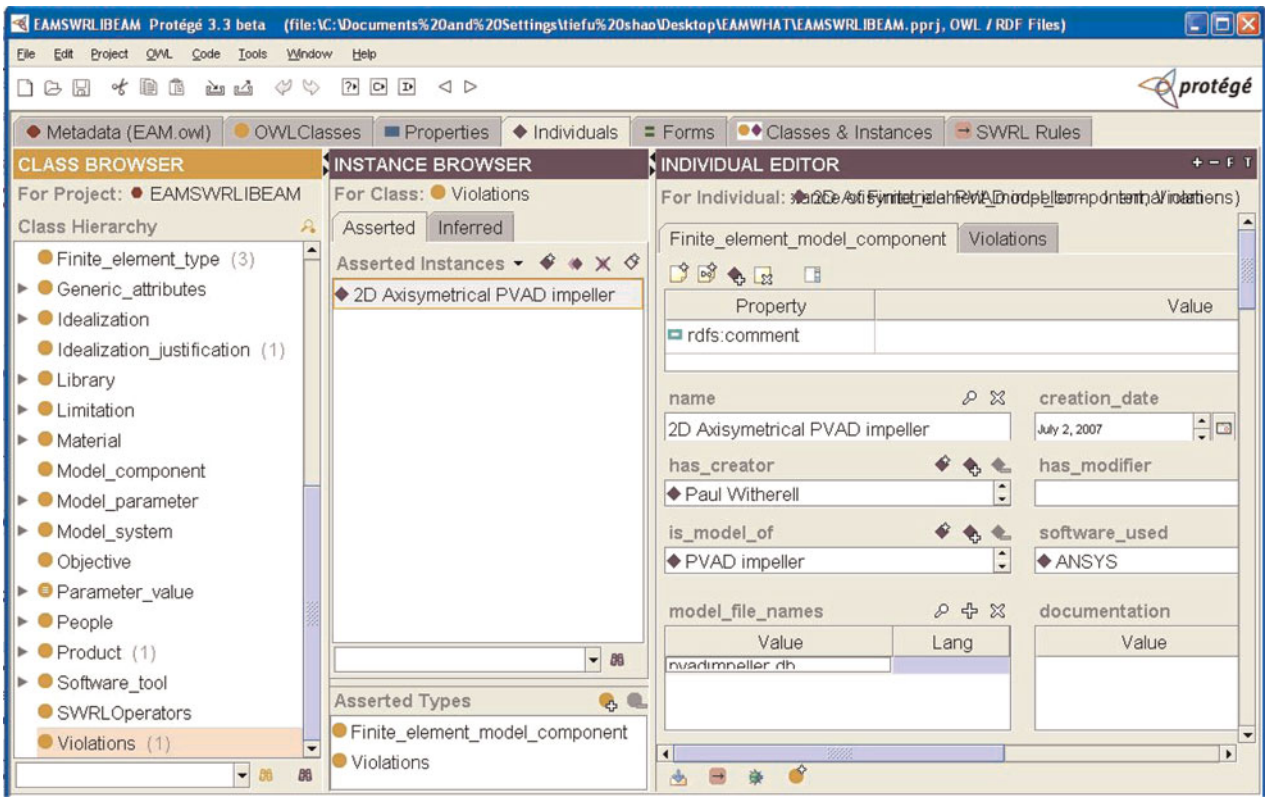

Fig. 5. Asserted violation. [A color version of this figure can be viewed online at journals.cambridge.org/aie]

https://doi.org/10.1017/S0890060409990096 Published online by Cambridge University Press

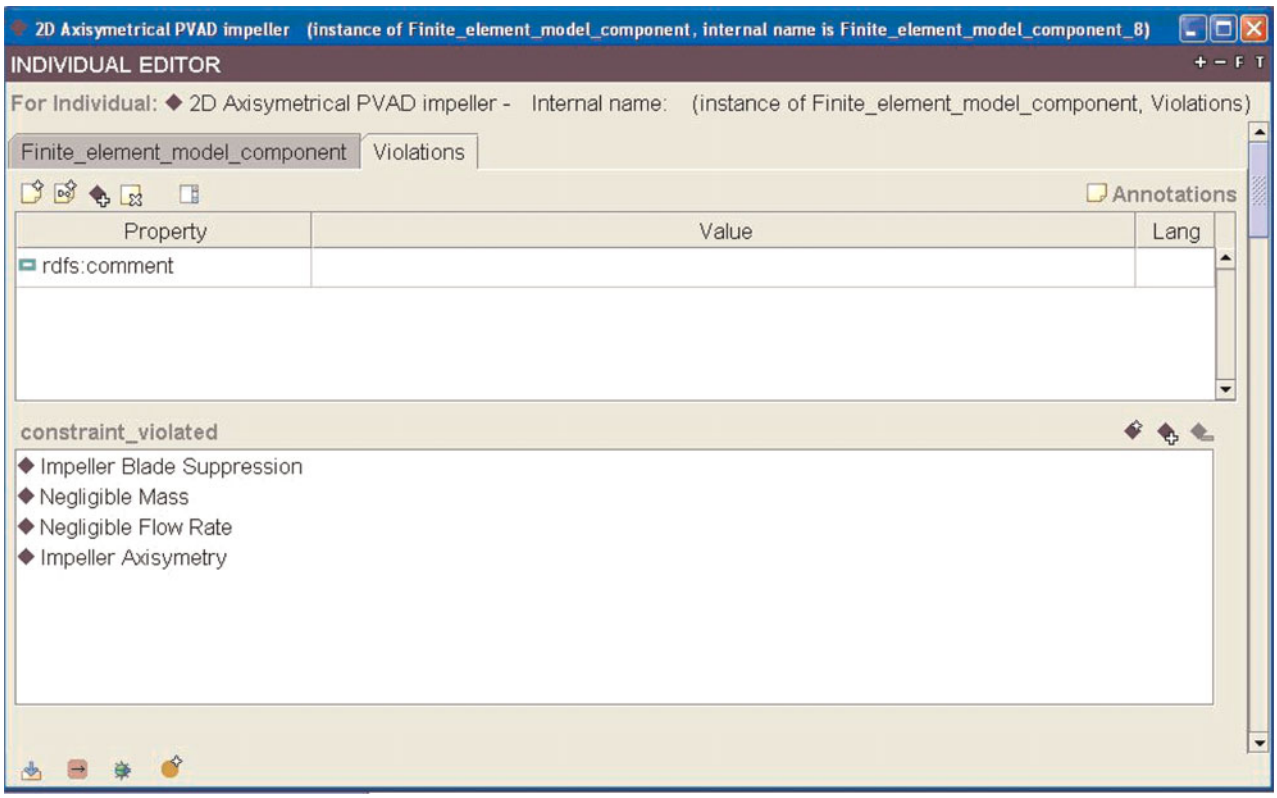

Fig. 6. Violations properties. [A color version of this figure can be viewed online at journals.cambridge.org/aie]

For a specific example, a gradient-based method may provide an accurate result but require a significant amount of computational costs, whereas a direct search method may sacrifice some accuracy but at a fraction of the costs. Other examples include choosing the most effective element/mesh combination in a finite element problem, or perhaps identifying the most computationally friendly model. For example, an I-beam meshed with beam elements may possess a greater accuracy than it would with a solid tetrahedral element model when subjected to bending stresses. These are decisions an engineer often faces during the product development process, and rules can assist the engineer in his or her decision by not only presenting alternatives but also the related design implications during the design process.

Until now, the use of Horn logic has focused on inferencing between classes and class properties of an ontological framework. However, the ability to inference on specific instances of ontologies, thus providing the ability to curtail large proliferations of classes in an ontological framework, has not been addressed. For example, consider an electromechanical circuit system. Here, if the system's electronics (such as a circuit board) required an ontology for each component, a specialized framework approach would quickly become difficult to manage. An alternative approach is to operate on *instances* of assemblies and assembly components, thus capturing the same knowledge without requiring the creation of specialized ontologies, creating relationships among instance properties instead. This scenario still allows for the use of rules to track how altering parameter values (i.e., length, width, thickness, or diameter) may affect the integrity of the resulting design, without the need of a complex framework.

Another significant outcome of this work lies in the approach taken to overcome the many challenges presented by the Semantic Web when developing a dynamic knowledge base. Although OWL's and SWRL's connections with the Semantic Web provide many unique benefits, there are also associated drawbacks. For example, the open-world nature of the Semantic Web and its only partial expressivity of first-order logic can make many otherwise mundane logical inferences difficult. A definitive example, as well as one of the hardest challenges to overcome, is the nonexistence of negation as failure. This means that while assertions are allowed when required conditions are met, the same cannot be said for when these conditions are not met. To address this challenge a "Violations" class was created, turning this particular limitation into an advantage and providing a unique method for introducing knowledge discrepancies to engineers. Advances in semantic systems such as reasoners and rule engines and the development of corresponding languages promise to offer significant enhancements to the currently available capabilities of Semantic Web-based distributed knowledge frameworks.

This work presented methods for expanding the traditional purposes of knowledge-based engineering to serve a larger role in the design process. Beyond providing a structured knowledge framework, first-order logic was able to establish both influential and dependent relationships within a suite of

https://doi.org/10.1017/S0890060409990096 Published online by Cambridge University Press

well-defined engineering domain ontologies. These relationships provided an intelligent aspect to a well-formed knowledge base, allowing the knowledge base to advance beyond a static repository. Although it may be argued that existing CAD tools offer similar capabilities to manage basic data, such as parameters and constraints, such rules are native to a single environment, and are unable to be executed across systems. In the developed ontological framework, the abstraction of domain knowledge allows relationships to be created between more complex entities beyond simply numerical values, as shown by the case studies. In short, the addition of rules provides a unique dynamic attribute unseen in the traditional engineering knowledge base, creating an environment that not only supports collaboration but also the corroboration of knowledge.

8. SUMMARY

This paper introduced the use of first-order logic subsets implemented in the form of the Semantic Web's OWL and SWRL to enhance the knowledge acquisition and knowledge management capabilities of ontologies. The developed methods, based on Horn rules, allow for a substantial reduction in the possibility of human error during the design process by the identification of uncorroborated and inconsistent knowledge that may otherwise go unnoticed. Specifically, they addressed three key issues in knowledge management that ontologies by themselves do not address: corroboration of knowledge instantiations, retained consistency during the knowledge instantiation process, and minimization of redundancy in the knowledge instantiation process. This work resulted in the engineering knowledge management tool OPTEAM. Findings from its application to two case studies show that logic-based methods offer a structured approach to enhance knowledge acquisition, knowledge management, and knowledge validation techniques in engineering design ontologies.

ACKNOWLEDGMENTS

This material is based on work supported by NSF Grant 0332508 and by industry members of the NSF Center for e-Design.

REFERENCES

Alberts, L.K., & Dikker, F. (1992). Integrating standards and synthesis knowledge using the YMIR ontology. In *Artificial Intelligence in Design* (Gero, J.S., & Sudweeks, F., Eds.), pp. 517–534. Boston: Kluwer Academic.

Becker, B.J., & Kaepp, G.A. (1997). BDS: a knowledge-based bumber design system. *1997 ASME Design Engineering Technical Conf.*, Paper No. DETC97/CIE-4272, Sacramento, CA.

Bohm, M.R., Stone, R.B., Simpson, T.W., & Steva, E.D. (2006). Introduction of a data schema: the inner workings of a design repository. *Proc. ASME IDETC/CIE.*

Borgida, A. (1996). On the relative expressiveness of description logics and predicate logics. *Artificial Intelligence 82(1–2)*, 353–367.

Brown, D.C. (1985). Capturing mechanical design knowledge. *Proc. ASME Int. Computers in Engineering Conf.*, Boston.

Buchanan, B.G., & Shortliffe, E.H. (1984). *Rule Based Expert Systems: The Mycin Experiments of the Stanford Heuristic Programming Project*. Reading, MA: Addison–Wesley.

Bullinger, H.J., Warschat, J., Schumacher, O., Slama, A., & Ohlhausen, P. (2005). *Ontology-Based Project Management for Acceleration of Innovation Projects*. LNCS, Vol. 3379. New York: Springer.

de Kleer, J. & Brown, J.S. (1983). Assumptions and ambiguities in mechanistic mental models. In *Mental Models* (Genter, D., & Stevens, E.L., Eds.), pp. 155–190. Hillsdale, NJ: Erlbaum.

Dos Santos, B.L., & Mookerjee, V. (1991). Towards optimal expert system design, *Proc. 24th Hawaii Int. Conf. Systems Sciences*.

Erickson, D.M., Brown, D.R., Hwang, K., Pan, Y., & Daga, A. (1997). A framework for cooperating engineering knowledge agents. *1997 ASME Design Engineering Technical Conf.*, Paper No. DETC97/CIE-4299, Sacramento, CA.

Euler, E.E., Jolly, S.D., & Curtis, H.H. (2001). The failures of the Mars Climate Orbiter and Mars Polar Lander: a perspective from the people involved. *Proc. Guidance and Control 2001*, Paper No. AAS 01-074. Springfield, VA: American Astronautical Society.

Friedman-Hill, E. (2003). *Jess in Action*. Greenwich, CT: Manning Publications.

Genesereth, M., & Fikes, R. (2001). *Knowledge Interchange Format Version 3.0 Reference Manual Technical Report*, Logic Group Report Logic-92-1, Stanford University. Accessed at http://logic.stanford.edu/kif/Hypertext/kif-manual.html

Gennari, J., Musen, M.A., Fergerson, R.W., Grosso, W.E., Crubézy, M., Eriksson, H., Noy, N.F., & Tu, S.W. (2003). The evolution of Protégé: an environment for knowledge-based systems development. *International Journal of Human–Computer Studies 58(1)*, 89–123.

Goel, A., Bhatta, S., & Stroulia, E. (1996). KRITIK: an early case-based design system. In *Issues and Applications of Case-Based Reasoning to Design* (Maher, M., & Pu, P., Eds.). Mahwah, NJ: Erlbaum.

Goel, A., Gomez, A., Grue, N., Murdock, J.W., Recker, M., & Govindaraj, T. (1996). Explanatory interface in interactive design environments. In *Artificial Intelligence in Design* (Gero, J.S., Ed.). Boston: Kluwer Academic.

Gottlob, G., & Nejdl, W. (1990). *Proc. Expert Systems in Engineering, Principles and Applications, Int. Workshop*. LNCS, Vol. 462. New York: Springer.

Grosse, I.R., Milton-Benoit, J.M., & Wileden, J.C. (2005). Ontologies for supporting engineering analysis models. *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing 19(1)*, 1–18.

Grosof, N.B., Horrocks, I., Volz, R., & Cecker, S. (2003). Description logic programs: combining logic programs with description logic. *Proc. 12th Int. Conf. World Wide Web WWW2003*, pp. 48–57, Budapest, Hungary, May 20–24.

Gruber, T., & Olsen, G. (1994). An ontology for engineering mathematics. *Proc. 4th Int. Conf. Principles of Knowledge Representation and Reasoning* (Doyle, J., Torasso, P., & Sandewall, E., Eds.), pp. 258–269. San Mateo, CA: Morgan Kaufmann.

Haarslev, V., Möller, R., & Wessel, M. (2004). Querying the Semantic Web with Racer + nRQL. *KI-04 Workshop on Applications on Description Logics.*

Henson, B., Juster, N., & de Pennington, A. (1994). Towards an integrated representation of function, behavior and form, computer aided conceptual design. *Proc. 1994 Lancaster Int. Workshop on Engineering Design* (Sharpe, J., & Oh, V., Eds.), pp. 95–111. Lancaster: Lancaster University EDC.

Horn, A. (1956). On sentences which are true of direct unions of algebras. *Journal of Symbolic Logic 16*, 14–21.

Horrocks, I., Patel-Schneider, P., Bechhofer, S., & Tsarkov, D. (2005). OWL rules: a proposal and prototype implementation. *Journal of Web Semantics 3(1)*, 23–40.

Iwasaki, Y., & Chandrasekaran, B. (1992). Design verification through function and behavior-oriented representations: bridging the gap between function and behavior. In *Artificial Intelligence in Design* (Gero, J.S., Ed.), pp. 597–616. Boston: Kluwer Academic.

Kalyanpur, A., Parsia, B., Sirin, E., Cuenca-Grau, B., & Hendler, J. (2005). Swoop: a Web ontology editing browser. *Journal of Web Semantics 4(1)*. doi:10.1016/j.websem.2005.10.001

Kanuri, N. (2007). *Ontologies and methods for interoperability of engineering analysis models (EAM's) in an e-design environment.* Master's Thesis. University of Massachusetts Amherst.

Kifer, M., & Lausen, G. (1989). F-logic: a higher-order language for reasoning about objects, inheritance, and scheme. *Int. Conf. Management of Data*, pp. 134–146.

Kim, K., Yang, H., & Manley, D. (2006). Assembly design ontology for service-oriented design collaboration. *Computer-Aided Design and Applications 3(5)*, 603–613.
Levy, A.Y., & Rousset, M.C. (1998). Combining Horn rules and description logics in CARIN. *Artificial Intelligence 104(1–2)*, 165–209.
Morris, K.N. (1998). Agent support for collaborative design, *1998 ASME Computers in Engineering Conf.*, Paper No. DETC98/CIE-5551.
Noy, N.F., Sintek, M., Decker, S., Crubezy, M., Fergerson, R.W., & Musen, M.A. (2001). Creating Semantic Web contents with Protege-2000. *IEEE Intelligent Systems 16(2)*, 60–71.
Qian, L., & Gero, J.S. (1996). Function–behavior–structure paths and their role in analogy based design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 10(4)*, 289–312.
Patil, L., Dutta, D., & Sriram, R. (2005). Ontology-based exchange of product data semantics. *IEEE Transactions on Automation Science and Engineering 2(3)*, 213–225.
Prasad, B. (2004). Knowledge driven automation. *Enterprise Engineering Systems, ParTech 2004*.
Ranta, M., Mäntylä, M., Umeda, Y., & Tomiyama, T. (1996). Integration of functional and feature based product modeling—the IMS/GNOSIS Experience. *Computer-Aided Design 28(5)*, 371–381.
Rogers, J. (2004). Getting the most gains out of knowledge-based engineering—Parker Aerospace Experiences. *2004 Annual Conf. TechniFair.*
Schmidt-Schauß, M. (1989). Subsumption in KL-ONE is undecidable. *Proc. 1st Int. Conf. Principles of Knowledge Representation and Reasoning KR '89* (Brachman, R.J., Levesque, H.J., & Reiter, R., Eds.), pp. 421–431. Los Altos, CA: Morgan Kaufmann.
Shepar, M.S., Bachmann, L., Georges, M.K., & Korngold, E.V. (1990). Framework for reliable generation and control of analysis idealizations. *Computer Methods in Applied Mechanics and Engineering 82(1–3)*, 257–280.
Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., & Katz, Y. (2004). Pellet: a practical OWL-DL reasoner. *3rd Int. Semantic Web Conf. ISWC2004*.
Spiegelhalter, D., Dawid, A., Lauritzen, S., & Cowel, R. (1993). Bayesian analysis in expert systems. *Statistical Science 8(3)*, 219–247.
Szykman, S., Sriram, R.D., Bochenek, C., & Racz, J. (1998). *The NIST Design Repository Project. Advances in Soft Computer-Engineering Design and Manufacturing*. London: Springer–Verlag.
Szykman, S., Sriram, R.D., Bochenek, C., Racz, J.W., & Senfaute, J. (2000). Design repositories: engineering design's new knowledge base. *Intelligent Systems and Their Applications 15*, 48–55.
Tsarkov, D., Riazanov, A., Bechhofer, S., & Horrocks, I. (2004). Using vampire to reason with OWL. *3rd Int. Semantic Web Conf.*
Turkiyyah, G.M., & Fenves, S.J. (1996). Knowledge-based assistance for finite element modeling. *AI Applications in Civil and Structural Engineering 11(3)*, 23–32.
Umeda, Y., Ishii, M.,Yoshioka, M., Shimomura, Y., & Tomiyama, T. (1996). Supporting conceptual design based on the function–behavior–state modeler. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 10*, 275–288.
Wagner, G., Tabet, S., & Boley, H. (2003). MOF-RuleML: the abstract syntax of RuleML as a MOF model. *Integrate 2003, OMG Meeting*, Boston.
Witherell, P., Krishnamurty, S., & Grosse, I.R. (2006). Ontologies for supporting engineering design optimization. *Journal of Computing and Information Science in Engineering 7(2)*, 141–150.

Paul Witherell is a third-year PhD student in the Department of Mechanical and Industrial Engineering at the University of Massachusetts Amherst. He received his BS and MS degrees in mechanical engineering from the University of Massachusetts Amherst in 2004 and 2006, respectively. Mr. Witherell's research interests are in the area of product design, with a focus on knowledge representation in the product development process. Much of his work is based on the development of ontologies and semantic methods to support and facilitate the many aspects of the product development process.

Sundar Krishnamurty is an Associate Professor of mechanical and industrial engineering at the University of Massachusetts Amherst. He received his BS in civil engineering from IIT-Kanpur in 1982, his MS in civil engineering from the University of Pennsylvania in 1984, and his PhD in mechanical engineering from the University of Wisconsin–Madison in 1989. Dr. Krishnamurty is currently the site Director for the NSF Industry/University Cooperative Research Center (NSF-I/UCRC) for e-Design and Realization of Engineering Products and Systems. His research focuses on a wide range of fundamental and applied research topics in design, including Semantic Web ontologies, decision-based design, design innovation, and therapeutic design.

Ian R. Grosse is an Associate Professor in mechanical engineering and Director of the Intelligent Modeling, Analysis, and Design Laboratory at the University of Massachusetts. He was also a former site Director for the NSF Center for e-Design and Realization of Engineered Products and Systems. He received his BS from Cornell University in 1979 and his MS and PhD from Virginia Polytechnic Institute and State University in 1983 and 1987, respectively. Dr. Grosse's research interests include the application of Semantic Web technologies for modeling, capturing, and sharing engineering design knowledge in a distributed environment and finite element modeling and analysis of biological systems in comparative biology and anthropology.

Jack C. Wileden is a Professor in the Department of Computer Science and Director of the Convergent Computing Systems Laboratory at University of Massachusetts Amherst. He has been on the faculty of the University of Massachusetts Amherst since 1978. He received a BA in mathematics and MS and PhD in computer and communications sciences from the University of Michigan in 1972, 1973, and 1978, respectively. Dr. Wileden's research interests are programming languages and interoperability. His work is primarily directed toward producing tools, techniques, and formal foundations to support development and evolution of maximally seamless systems comprising interoperating components.