

# An Ontology-Based Feature Recognition and Design Rule Checker for Engineering

Luis Enrique Ramos García<sup>1</sup>, Alexander Garcia<sup>2</sup>, John Bateman<sup>1</sup>

<sup>1</sup> University of Bremen, FB 10

Cartesium, Enrique-Schmidt-Strasse. Bremen, Germany

[bateman@uni-bremen.de](mailto:bateman@uni-bremen.de), [s\\_7dns7r@uni-bremen.de](mailto:s_7dns7r@uni-bremen.de)

<sup>2</sup> University of Arkansas for Medical Sciences

Little Rock, Arkansas, USA

[alexgarcia@gmail.com](mailto:alexgarcia@gmail.com)

**Abstract.** This paper describes the design and implementation of an Ontology-Based System for Features Recognition and Design Rules Checking in the domain of sheet-metal engineering using Semantic Web technologies. The system was implemented by means of the Protégé Application Programming Interface (API), a rule engine and a reasoner. Using ontology in the core of the system enabled the representation of manufacturing rules and Automatic Features recognition. Rules and Queries were not hard coded in the program, giving the system a high level of maintainability and reusability. Features were classified as general and specific, easing the work of classifying newer features as they appear given their previous classification. Most common sheet metal features referred to in the literature were recognized by the system.

**Keywords:** Semantic Web, OWL, SWRL, SQWRL, CAD, CAM, CAPP

## 1 Introduction

Sheet metal parts are elements commonly used in several products manufactured in modern industry; aerospace, automotive, appliances, machine tools, etc. are only some of these. Although various software tools are available for making digital designs for such metal parts, trying to use those designs as input for process planning and manufacturing is not straightforward. The majority of standards for Computer Aided Design (CAD) do not represent certain features that are needed when manufacturing. For instance, declaring when a circle corresponds to an inner edge or an outer edge has significant consequences during manufacturing although not necessary when displaying designs. Facilitating the interoperability across the CAD-CAM (Computer Aided Manufacturing) process should make it possible for designers to select optimal manufacturing conditions. For instance, to choose a sheet thickness that would prevent crashes when punching holes.

In order to achieve such interoperability, manufacturing features must be extracted from CAD files. Extracting these features entails identifying certain patterns from the CAD files that are to receive additional manufacturing-relevant enrichment. This information about features is used *a posteriori* to determine the machining tools and

manufacturing processes required to manufacture a given design [1]. Feature information can also then be used to pre-check designs in order to detect production rules violations. If these violations are not detected in the early stage of design, the life cycle of its development increases; raising both production costs and time to market [2]. Manual recognition of the targeted features is not a viable alternative; Automated Features Recognition (AFR) is thus required. However, even nowadays AFR is not fully integrated with CAD software tools. It is applicable only to parts with relatively simple geometry and still requires human intervention to obtain the features identified. Moreover, such techniques are generally supported only on expensive CAD software tools that are beyond the reach of small-scale industry [3].

To address these shortcomings of AFR from sheet metal designs and design checking, in this paper we propose an ontology-based framework that facilitates interoperability across the entire CAD-CAM process. Our system integrates both a CAD and a feature ontology; these ontologies were written in OWL (Web Ontology Language) and represent 2D primitives as lines, arcs and circles in the CAD ontology and as edges, slots, tab and holes in the features ontology. In order to provide rules and query support, OWL was complemented with SWRL [4] and SQWRL [5]. The result is an ontology-based system that allows us to automatically extract the most common manufacturing features referred to in the literature.

The paper is organized as follows. We begin with related work in Section 2. Section 3 then describes the components in the architecture of the implemented system. Section 4 shows the implementations and some results of the evaluation on a sample design and Section 5 concludes with an outlook for future work.

## 2 Related work

### 2.1 AFR & Design Advisory Systems

Henderson & Anderson [6] reported on early experiences with the use of production rules that were not hard-coded into the system using Prolog, while Meeran & Pratt [7] proposed to extract features from 2D drawings to generate process planning and machining instructions from CAD. The latter approach was also based on logic programming, adopting Prolog as the implementation language. These authors mentioned that Prolog presented certain limitations when dealing with trigonometric functions.

Other authors, such as Vasilakis [8] and Krifa *et al* [9], did not use logic programming. Rules were represented instead as a set of nested IF *<list of conditions>* THEN *<hypothesis><list of actions>*. In the case of Vasilakis [8], when a design rule was violated the system showed the rule violated but did not suggest any corrective action. Radhakrishnan [2] integrated a rule checker within a design adviser; this approach identifies distance violations amongst features. The general tendency in these contributions was the development of feature recognition and design advisory systems with hard-coded rules.

In summary, two main classes of approach can be identified from the literature. On the one hand there are approaches based on logic programming. Here, rules are separated from code, facilitating interoperability, shareability and maintainability; these approaches have generally used Prolog as implementation language. On the other hand, there are also approaches relying on procedural programming, commonly using C and C++ as implementation languages. As these are not rule-based, maintainability and share-ability are limited. Babic *et al* [10] consider that emerging approaches should be hybrids.

## 2.2 Ontology for CAD

The limitations of CAD software tools have motivated researchers to develop frameworks that overcome those limitations and some of these have already adopted ontological approaches. Vasilakis & Andersen [11] and Krifa *et al* [12], for example, developed ontologies for the Standard for the Exchange of Product model data (STEP). Here, Krifa *et al* [12] acknowledged the limitation of STEP and accordingly developed OntoSTEP. OntoSTEP is based on OWL precisely because of its support of logic reasoning and inference mechanisms.

Approaches addressing the problem of interoperability across CAD standards have been explored by Ghafour *et al* [13], Sun & Ding [14] and Ramos [15]. Ghafour *et al.* proposed a common Design Features Ontology written in OWL and one domain ontology from each specific standard, an approach similar to that of Sun & Ding. Ramos focused on an ontology for CAD primitives populating a CAD ontology. In addition, Grüninger & Delaval [16] developed an ontology for the cutting process of sheet metal parts; this approach was proposed as a theoretical step from which the planning process could be supported. Franke *et al* [17] developed an ontology-based tool for quality control in CAD design. They aimed to verify cases in which overlapping occurred in order to determine if this constituted a design mistake. Their architecture included their own software tool (OntoDMU) and the system HETS [18].

## 2.3 Sheet Metal Features and Rules for Manufacturing

Defining features in the manufacturing domain is not straightforward [1]; a feature is understood here as a part of a mechanical design that has a specific functionality and that is semantically significant within the manufacturing process. Farsi & Arezoo [19] classified sheet metal features according to whether those features were internal or external; internal features are holes and external features are different types of notches. Radhakrishnan *et al* [2] classified rules into two types: intrafeature rules and interfeature rules. Intrafeature rules are related only to the feature itself and its constraints, indicating minimum values of its dimensional parameters. Interfeature rules describe restrictions across features and contours. In Fig. 1 some of the most common features and rules found in the literature are described. The figure includes features names and some of their most important parameters. The indicated rules are related directly with the production process. It is necessary to take such rules into account in order to provide the necessary structural strength to the design's features. Improperly formed features will contribute to waste of material in the finished products due to buckle, cracks or breaks.

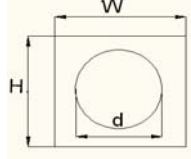
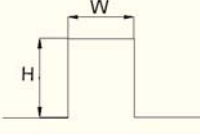
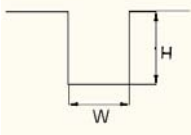
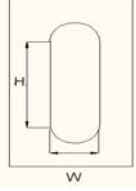
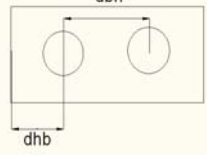
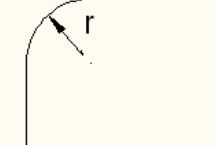
 <p><b>Circular Hole</b></p> <p> <math>d &lt; H</math> &amp; <math>d &lt; W</math>  <math>d &gt; T</math> &amp; <math>d &gt; 2.5mm</math> </p> <p> <i>T</i> = Thickness  <i>d</i> = diameter  <i>H</i> = High  <i>W</i> = Wide </p>	 <p><b>Protrusion or Tab</b></p> <p> <math>W &gt; 1.5T</math> &amp; <math>W &gt; 0.5</math>  <math>\frac{H}{W} &gt; 5</math> </p>
 <p><b>Notch or slot</b></p> <p> <math>W &gt; 1.5T</math> &amp; <math>W &gt; 0.5mm</math>  <math>\frac{H}{W} &gt; 5</math> </p>	 <p><b>Oblong holes</b></p> <p> <math>d &lt; H</math> &amp; <math>d &lt; W</math>  <math>d &gt; T</math> &amp; <math>d &gt; 2.5mm</math> </p>
 <p><b>Distance to holes</b></p> <p> <math>dbh &gt; T</math>  <math>dbh &gt; 1.5mm</math>  <math>dbh &gt; T</math>  <math>dbh &gt; 2T</math>  <math>dbh &gt; 0.8mm</math> </p> <p> <i>dbh</i>: distance between holes  <i>dhb</i>: distance hole border </p>	 <p><b>Corner</b></p> <p><math>r \geq 0.7 T</math></p>

Fig. 1. Manufacturing features with their constraints

### 3 An Ontology-Based AFR and Design Checker system

The approach developed here builds on the state of the art we have seen and takes this further, relying on a modular architecture that makes use of the Protégé API. In the following subsection we describe the approach's components and the way in which they interact with each other. Although the organization and classes of the two ontologies are often comparable, it is important to see that they reflect fundamentally different *kinds* of ontological information. To the extent that similarities are found, then this renders the subsequent mapping process more straightforward, but there is

no requirement that the ontologies correspond precisely. Since the identification criteria for the CAD and the features ontologies are quite distinct—the former relating to descriptions of design, the latter to manufacturability—it is both theoretically and architecturally cleaner to maintain them separately. We return to this important consideration and the issues of perspectives below.

### 3.1 A CAD Ontology

The first component to be discussed is the CAD ontology. During the development of this ontology several CAD formats, such as DXF [20] and IGES [21], were considered. The resulting CAD ontology is under continued revision and update. Fig. 2 illustrates the current state of the CAD Ontology: <http://bit.ly/q8baMm>.

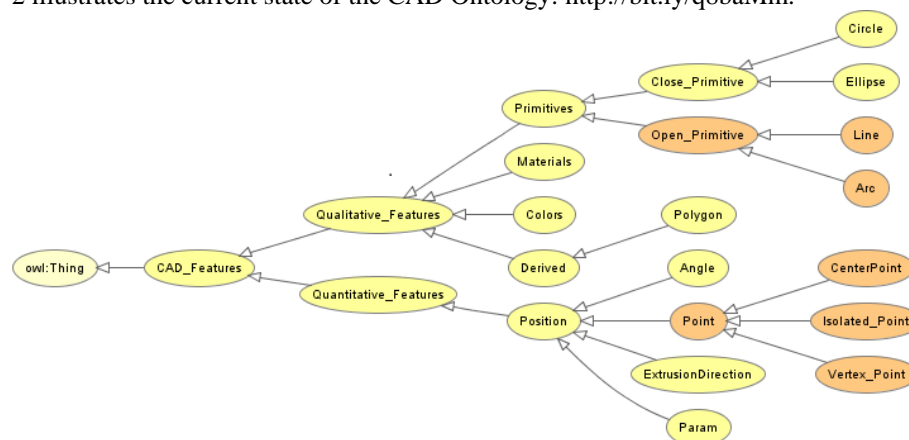


Fig. 2. CAD Ontology, <http://bit.ly/q8baMm>

As is illustrated in the figure, we divided *CAD\_Features* into the two subclasses: *Qualitative\_Features* and *Quantitative\_Features*. Among the former, *Primitives* can be either *Open* or *Closed*. Under these classes, *Line*, *Arc*, *Circle* and *Ellipse* are understood as *Primitive* elements that are present in any CAD standard that provides at least a 2D representation. *Derived* elements are then *Qualitative\_Features* that may vary and are standard dependent; they appear as a combination of certain primitive elements. For example, a *Vertex* appears when two primitive lines are connected at one end. Definition of concepts, such as *Vertex*, is dimensional-space dependent.

### 3.2 Ontology of Features

For the development of the ontology of features, available at <http://bit.ly/pyQGEE>, we considered information such as that shown in Fig. 1. An extract of the resulting ontology is presented in Fig 3. There are two main concepts: these are *Shape\_Features* and *NonGeometry*. *Shape\_Features* was divided further into *Qualitative\_Features* and *Quantitative\_Features*, analogously to the top-level distinction in the CAD ontology. *Named\_Feature* involves features found in the literature that can be precisely defined and identified using their properties or attributes. *Atomic\_Feature* is then a *Named\_Feature* consisting of exactly one

identifiable element which has meaning in a manufacturing domain. These elements can have *Open* or *Closed* edges.

*Composed\_Features* include features that appear as a combination of *Atomic\_Features*. *Partial\_Composed* includes identifiable features that need to be part of a whole to be consistent. *Total\_Composed* covers a feature that exists as a whole. *Partial\_Composed* features have to be part of a *Total\_Composed* feature. *hasPart* and *isPartof* were included following a standard mereological [22] point of view and its terminologies and definitions. Properties such as *properWidth*, *properHeight*, *properCircularHole* were included as part of SWRL rules. When these rules are evaluated, results are bound to true or false. If results are all true, the design passes its evaluation; otherwise the design is flagged as needing to be checked.

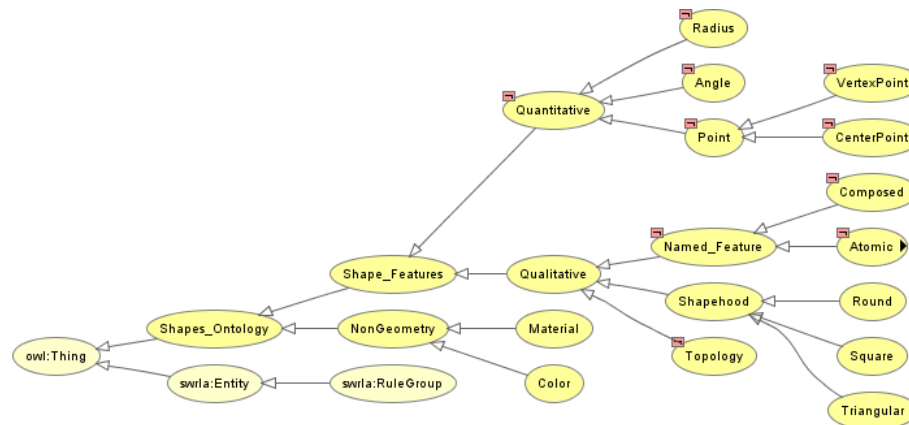
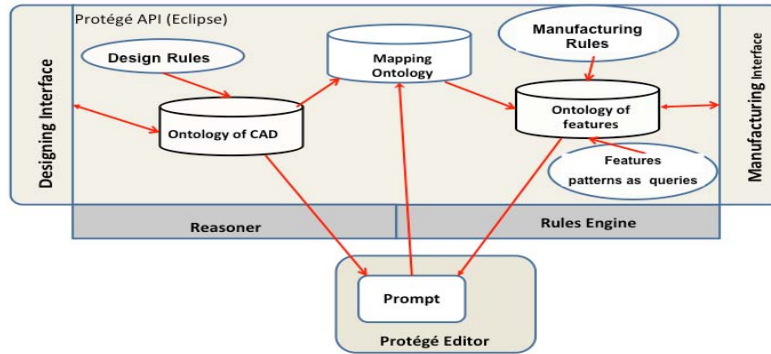


Fig. 3. Sheet features Ontology, <http://bit.ly/pyQGEE>

### 3.3 Mapping two domains of interpretation

One of the main issues of AFR is the level of abstraction involved in the manufacturing domain [23]. On the one hand, designers make designs from the functionality and usability point of view, with its own constraints and rules. On the other hand, manufacturing engineers evaluate a design from the manufacturing point of view, considering factory constraints. It is for this reason that, in general, quite distinct standards, formats and tools have been developed. In the two domains, two different interpretations of the requirements take place for the *same object*, so an interoperability channel between them is needed. This channel should facilitate the interchange of information about products as well as the evaluation of designing and manufacturing constraints.

In Fig. 4, we introduce the framework implemented for dealing with this situation. Knowledge transfer between domain ontologies has been facilitated by means of a third, mapping ontology, which keeps track of the related concepts of both domains. This mapping ontology was developed using the Prompt plug-in [24]. Given that in many cases the difference between the domains can be reduced to



**Fig. 4** System Architecture

terminological issues (factoring out often implicit identity criteria), we first made an internal comparison [25]. This consisted of comparing concepts based on attributes and properties. The ontology of CAD and the ontology of features were used as inputs and a mapping ontology was the output. Mapping was semi-automatic because the algorithm did not detect all relationships; human intervention was needed to complete the process. This preliminary mapping already gave us the necessary information to generate individuals of the features ontology and then to transfer the knowledge about individuals and their properties from the ontology of CAD to the ontology of features in a workable fashion.

### 3.4 Feature Patterns as Queries

Features were recognized by querying the features ontology with SQWRL. These queries were performed at two levels of abstraction. A first set of queries was implemented to facilitate a general classification of features and a second set was executed to facilitate specialized classification. These two sets of queries reduced the impact of not extracted or unidentified features referred to in the literature [26]. Within the first set of queries, all given *atomic features* were classified as *is part of*, a general pattern. Subsequently, having assigned all *atomic features* to some general pattern, we proceeded to identify specific patterns. If some atomic feature was not classified as *is part of* some specific pattern, then a new specific pattern had to be added.

For example, a protrusion pattern is formed by three connected edges having a specific slope. The query for extracting this pattern was expressed as follows:

```
Open_Primitive(?l1) ^ Slope(?l1, "-0.0") ^
Open_Primitive(?l2) ^ Slope(?l2, "Infinity") ^
Open_Primitive(?l3) ^ Slope(?l3, "+0.0") ^
hasendpoint(?l1, ?p1) ^ hasstartpoint(?l2, ?p1) ^
hasendpoint(?l2, ?p2) ^ hasstartpoint(?l3, ?p2) °
```

```
sqwrl:makeSet(?s, ?l3)
→ sqwrl:select(?s, ?l1, ?l2, ?l3)
```

The number of sets obtained and their composition allows the generation of composed features. Once the query has been executed, composed features are related to atomic features by means of the *has-part* property.

### 3.5 Design and Manufacturing Rules Checking

Although AFR research is receiving considerable attention, a recognized feature still does not indicate anything about its quality. This aspect is fundamental in order to reduce the cost of manufacturing a given design [27]. To this end, two sets of SWRL were implemented to check manufacturing rules. The first set allowed us to enrich the features ontology with the necessary engineering information about features. For instance, the width of a *Partial Composed* feature was calculated with the following rule:

```
PartialComposed(?pc) ∧ hasEdge(?pc, ?e) ∧ slope(?e,
"0.0") ∧ hasEndPoint(?e, ?p2) ∧ hasYcoordinate(?p2,
?y2) ∧ hasStartPoint(?e, ?p1) ∧ hasYcoordinate(?p1,
?y1) ∧ swrlb:subtract(?v1, ?y2, ?y1) ∧ swrlb:abs(?v1) →
width(?pc, ?v1)
```

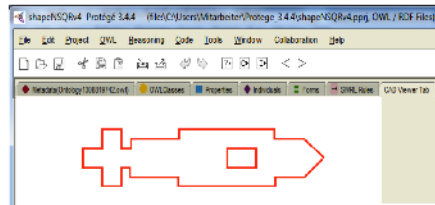
The second set of rules was implemented for checking the design. For instance, given the thickness of a given material, a minimum width is mandatory to make its manufacturability feasible. For some materials the width must be 1.5 times greater than the given thickness. This rule was expressed in SWRL as follows:

```
PartialComposed(?pc) ∧ width(?pc, ?w) ∧ heigh(?pc, ?h)
∧ thickness(?pc, ?t) ∧ swrlb:multiply(?v1, ?t, 1.5) ∧
swrlb:greaterThan(?w, ?t) → proper_width(?pc, true)
```

### 3.6 Reasoning framework

The reasoning capabilities of the Semantic Web framework gives us the possibility of classifying instances of classes against the ontologies of our application. Our reasoning was done in two steps. Firstly, using SWRLTab [28], rules and queries were edited, implemented and debugged. JESS [29], a rule engine for JAVA platforms, has been included in Protégé for enabling this tab. Each OWL Model and its corresponding SWRL rules were loaded into the JESS engine. The JESS engine was then run and the resulting inferences were uploaded into the OWL model. Thus, the amount of knowledge encoded in the OWL model was augmented. Results of querying the models were also considered for creating new individuals. This result was finally compared with the design in order to determine if the result was consistent with it. Here we used Pellet [30] for the automatic verification of the consistency of the OWL model and for classifying the instantiated features.





This is the CAD viewer Protégé plug-in. The CAD file is parsed, the ontology is instantiated and the corresponding figure is displayed

Warnings:

```

WARNING: [OWLFrameStore] - DefaultOWLNamedClass(
http://www.owl-ontologies.com/Ontology1316006675.owl#Polygon)
INFO [AWT-EventQueue-0] (ABox.java:1541) - Consistency not(
http://www.owl-ontologies.com/Ontology1316006675.owl#Isolated\_Point) for 2 individuals [
http://www.owl-ontologies.com/Ontology1316006675.owl#Point\_48,
http://www.owl-ontologies.com/Ontology1316006675.owl#Point\_40]
INFO [AWT-EventQueue-0] (ABox.java:1628) - Consistent: false Tree depth: 0 Tree size: 67 Time: 9
INFO [AWT-EventQueue-0] (ABox.java:1541) - Consistency not(
http://www.owl-ontologies.com/Ontology1316006675.owl#Isolated\_Point) for 1 individuals [
http://www.owl-ontologies.com/Ontology1316006675.owl#Point\_40]
INFO [AWT-EventQueue-0] (ABox.java:1628) - Consistent: false Tree depth: 0 Tree size: 67 Time: 8
INFO [AWT-EventQueue-0] (ABox.java:1541) - Consistency not(
http://www.owl-ontologies.com/Ontology1316006675.owl#Isolated\_Point) for 1 individuals [
http://www.owl-ontologies.com/Ontology1316006675.owl#Point\_48]
INFO [AWT-EventQueue-0] (ABox.java:1628) - Consistent: false Tree depth: 0 Tree size: 67 Time: 7
  
```

 A screenshot of the Protégé console window showing a list of warning and information messages. A blue arrow points from the 'Warnings:' text above to the first warning message in the console. The messages include warnings about consistency and consistency checks for specific classes and individuals.

The reasoner signals an inconsistency whenever a general feature violates a manufacturing rule. Our system generates the corresponding warnings, informing about the presence of proper and improper features of each kind. If any warning is given as the system is running, there is the possibility to make the necessary corrections in the design or in the extracted features.

Fig. 5. Sample shape

Secondly, using the SWRL API [31], the OWL Reasoning API [32] and the Protégé API, a prototype was developed. *SWRLJessBridge*, *SWRLQueryAPI* and *ProtegePelletJenareasoner* were also used in this implementation; the overall architecture is presented in Fig 4. Rules and the results of running the rules were loaded into models at run time. Queries were made to the OWL models, and the results were again imported so as to add new knowledge into the OWL models. Finally, the reasoner verified the consistency of these ontologies and classified instanced features. Resulting from this process we obtained an OWL model of features containing all the necessary information pertaining to recognized features with their quality; our model also includes non-identified features.

## 4 Implementation and Results

The implementation was tested with several designs of shapes. In Fig. 5 one of these is displayed by means of a CADViewer plug-in for Protégé [15]. Our first evaluation of the design consisted of diagnosing the topological correctness of the model and verifying the connectivity of edges and vertex [33]. For this verification, we defined *Isolated\_Point* as a subclass of *Point*. After populating the CAD ontology, the reasoner was invoked. If instances of *Isolated\_Point* are found then the design is

considered as inconsistent; such a case is illustrated in Fig. 6. As soon as the problem was corrected, the system continued with the extraction of features and their validation.

The features described in Fig. 1 were recognized and extracted into the features ontology. Manufacturing rules were run on the OWL model adding newer knowledge. This new knowledge was then used by the reasoner to classify features into sets of proper and improper features.

## **5 Conclusions and future work**

AFR is a research and application area devoted to bridging the gap between design and manufacturing. Most of the approaches used in AFR are divided into procedural and declarative approaches. We have developed an Ontology-Based AFR and Design Rules Checker System that combines the advantages of both approaches. An ontology of CAD and an ontology of features are the fundamental components of our system. The necessary interoperation between both ontologies was achieved by means of a mapping ontology generated in a semi-automatic manner using the Prompt Protégé plug-in.

SWRL was used to perform engineering calculations in order to add appropriate semantics to the features ontology. SQWRL was used for modeling feature patterns commonly referred to in the literature. Sheet metal features were extracted as the result of queries applied to the features ontology. SQWRL queries and SWRL rules were written at several levels of abstraction in order to make a progressive identification of features possible. By using such combination, SQWRL and SWRL, we have identified a significant number of features.

In this paper we have demonstrated that complex rule of mechanical and manufacturing engineering processes can be expressed using OWL and SWRL. Similarly, we have demonstrated that complex engineering information can be retrieved from the ontology using the SQWRL language. These rules and queries were also used for inferring knowledge and driving appropriate decision-making information. New features, rules and patterns can be integrated into our system by means of standard ontology editors. In the near future, we plan to integrate a recommendation ontology. Thus, as soon as a design violation is found, a recommendation will be presented to the user. We will also investigate how to deal with intersecting mechanical features.

The resulting set of recognized features could be used as an input in Computer Aided Process Planning (CAPP) systems, from which process planning will be obtained. We will continue this line of development as a Semantic Web CAPP system, focusing on the accuracy of the kind of planning generated with the Semantic Web Technologies taking into consideration other aspects of the engineering environment and investigating scalability.

## References

1. Cayiroglu, I.: A new method for machining feature extracting of objects using 2D technical drawings. *Comput. Aided Des.* 41, 1008-1019 (2009).
2. Radhakrishnan, R., Amsalu, A., Kamran, M., Nnaji, B.O.: Design rule checker for sheet metal components using medial axis transformation and geometric reasoning. *Journal of Manufacturing Systems.* 15, 179-189 (1996).
3. Kumar, S., Singh, R.: Trends and Developments in Intelligent Computer Aided Design of Progressive Dies. *AMR.* 6-8, 241-248 (2005).
4. Horrocks, I., Patel, P.F.-S., Boley, H., Tabet, S., Grosz, B., Dean, M.: SWRL: A Semantic Web Rule Language Combining OWL and RuleML, <http://www.w3.org/Submission/SWRL/#7.1>, (2004).
5. Amar Das, M.O.: SQWRL: a Query Language for OWL. 6th International Workshop on OWL: Experiences and Directions (OWLED 2009). Vrije Universiteit Amsterdam, Chantilly, VA, United States (2009).
6. Henderson, M.R., Anderson, D.C.: Computer recognition and extraction of form features: A CAD/CAM link. *Computers in Industry.* 5, 329-339 (1984).
7. Meeran, S., Pratt, M.J.: Automated feature recognition from 2D drawings. *Computer-Aided Design.* 25, 7-17 (1993).
8. de Sam Lazaro, A., Engquist, D.T., Edwards, D.B.: An Intelligent Design for Manufacturability System for Sheet-metal Parts. *Concurrent Engineering.* 1, 117 -123 (1993).
9. Soman, A., Padhye, S., Campbell, M.I.: Toward an Automated Approach to the Design of Sheet Metal Components. *AI.* 17, 187-204 (2003).
10. Babic, B., Nesic, N., Miljkovic, Z.: A review of automated feature recognition with rule-based pattern recognition. *Computers in Industry.* 59, 321-337 (2008).
11. Andersen, O., Vasilakis, G.: *Building an Ontology of CAD Model Information.* Springer Berlin Heidelberg (2007).
12. Krüger, S., Barbau, B., Fiorentini, X., Sudarsan, R., Sriram, R.: OntoSTEP: OWL-DL Ontology for STEP, [http://www.nist.gov/customcf/get\\_pdf.cfm?pub\\_id=901544](http://www.nist.gov/customcf/get_pdf.cfm?pub_id=901544).
13. Ghafour, Abdul, S., Ghodous, P., Shariat, B., Perna, E.: An Ontology-based Approach for Procedural CAD Models; Data Exchange. *Proceeding of the 2006 conference on Leading the Web in Concurrent Engineering: Next Generation Concurrent Engineering.* pags. 251–259. IOS Press, Amsterdam, The Netherlands, The Netherlands (2006).
14. Sun, L.-juan, Ding, B.: Ontology-based Semantic Interoperability among Heterogeneous CAD Systems. *Information Technology Journal.* 9, pp. 1635 - 1640 (2010).
15. Ramos, L.: *Ontological CAD Data Interoperability Framework.* Presented at the SEMAPRO 2010, Florence, Italy Octubre (2010).
16. Grüninger, M., Delaval, A.: A First-Order Cutting Process Ontology for Sheet Metal Parts. *Proceeding of the 2009 conference on Formal Ontologies Meet Industry.* pp. 22-33. IOS Press (2009).
17. Franke, M., Klein, P., Schröder, L.: *Ontological Semantics of Standards and PLM Repositories in the Product Development Phase.* Proc. 20th CIRP Design Conference 2010. Alain Bernard.
18. Mossakowski, T., Maeder, C., Lüttich, K.: *The Heterogeneous Tool Set, Hets,* (2007). In *TACAS 2007* (2007), O. Grumberg and M. Huth, Eds., vol 4424 of LNCS, Springer, pp. 519 - 522.
19. Farsi, M., Arezoo, B.: *Feature Recognition and AND Design Advisory System for Sheet Metal Components.* Presented at the International Advanced Technologies Symposium, Turkey Mayo 13 (2009).
20. Autodesk: DXF Reference, [http://images.autodesk.com/adsk/files/acad\\_dxf1.pdf](http://images.autodesk.com/adsk/files/acad_dxf1.pdf), (2009).

21. U.S. Product Data Association, U.S.P.D.A.: Initial Graphics Exchange Specification 5.3, [http://www.uspro.org/documents/IGES5-3\\_forDownload.pdf](http://www.uspro.org/documents/IGES5-3_forDownload.pdf), (1997).
22. Varzi, A.: Mereology, <http://plato.stanford.edu/entries/mereology/>.
23. Shah, J.J., Rogers, M.T.: Functional requirements and conceptual design of the feature-based modelling system. *Computer-Aided Engineering Journal*. 5, 9-15 (1988).
24. Noy, N.F., Musen, M.A.: The PROMPT suite: interactive tools for ontology merging and mapping. *International Journal of Human-Computer Studies*. 59, 983-1024 (2003).
25. Euzenat, J.: State of the art on ontology alignment, <http://www.starlab.vub.ac.be/research/projects/knowledgeweb/kweb-223.pdf>, (2004).
26. Marchetta, M.G., Forradellas, R.Q.: An artificial intelligence planning approach to manufacturing feature recognition. *Computer-Aided Design*. 42, 248-256 (2010).
27. Li, X., Yoo, S.B.: Integrity validation in semantic engineering design environment. *Computers in Industry*. 62, 281-291 (2011).
28. Martin, O.: SWRLTab, <http://protege.cim3.net/cgi-bin/wiki.pl?SWRLTab>, (2011).
29. Friedman-Hill, E.: the Rule Engine for the JavaTM Platform. Sandia National Laboratories (2008).
30. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. *Web Semant*. 5, 51-53 (2007).
31. O'Connor, M., Nyulas, C., Shankar, R., Das, A., Musen, M.: The SWRLAPI: A Development Environment for Working with SWRL Rules. *Proceedings of the International Workshop on OWL: Experiences and Directions (OWLED 2008)* (2008).
32. Protege-OWL Reasoning API, <http://protegewiki.stanford.edu/wiki/ProtegeReasonerAPI>, (2009).
33. Tanaka, F., Kishinami, T.: STEP-based quality diagnosis of shape data of product models for collaborative e-engineering. *Computers in Industry*. 57, 245-260 (2006).