**Proceedings of DETC'05**
**2005 ASME Design Engineering Technical Conferences**
**Long Beach, California, USA, September 24–28, 2005**

**DRAFT DETC2005-85310**

# CONCEPTUAL DESIGN KNOWLEDGE MANAGEMENT AND THE SEMANTIC WEB

**Joseph B. Kopena** **Christopher D. Cera** **William C. Regli**[*]
{tjkopena, cera, regli}@cs.drexel.edu
Geometric and Intelligent Computing Laboratory
Department of Computer Science Drexel University
Philadelphia, Pennsylvania 19104
http://gicl.cs.drexel.edu/

## ABSTRACT

*The early stages of engineering design are critical, as the decisions made at this point have the most impact on the final product. However, little software is available to support engineers during the initial, conceptual design phase. In addition, at this and all other stages of design, engineers are increasingly tasked with utilizing unwieldy collections of data such as databases of legacy designs and catalogs. This work addresses both of these issues. A conceptual design interface with several advancements crucial to industrial deployment is developed and used to aid design. Among these are provisions for real-time collaboration and security. A representation of mechanical devices based on intended function is developed and used by the conceptual design interface to capture design semantics. This representation is defined using a description logic, enabling automated reasoning. The descriptions created using the conceptual design interface can thus be employed to annotate designs, create search queries, and to organize collections of designs. Further, this work incorporates Semantic Web technology, enabling conceptual design knowledge to be published and accessed effectively on the World Wide Web. New applications of design repositories are made possible by this but new issues must be investigated and addressed, as discussed here.*

## 1 INTRODUCTION

Although much software is available and employed in industry to support detailed geometric design, tolerancing, simulation, process planning, and other aspects of product development, there remains little support for working at the abstract, conceptual level of design semantics. However, the advantages of providing such support have fostered a great deal of research in this area. *Conceptual design* interfaces for engineers working in the early stages of design are a prominent line of work in computer-aided design (CAD). Similarly, annotation tools for capturing *design semantics*—the intended function and behavior of elements in a device and their rationale—have received much research attention but are not widely used in industry.

Conceptual design is the process of developing a high-level product design. Abstract ideas are generated and evaluated against product requirements and each other. Detailed design proceeds according to that seen as most favorable. This early stage in product development is critical, as the solution path chosen at this point is the largest determinant in the product's final design, cost, and success.

The importance of capturing design semantics begins during the early conceptual design stages and continues throughout the life-cycle of a product. Recorded output from early conceptual design serves to guide and direct the entire design proces. Similarly, rationale for choices made at each level of design must be recorded to guide more detailed development. This information also has great value in enabling later troubleshooting and revision, re-use in variant design, and inclusion into a larger design.

If captured appropriately, design knowledge as recorded by software could also be used in a variety of applications beyond recording the design process. In particular, this knowledge could be used in the management of collections of engineering de-

[*]Also of the Department of Mechanical Engineering

signs. Much of an engineer's time is spent examining large collections of corporate legacy data for designs which may be modified to solve new problems, or searching through catalogs in search of components to be incorporated into a design [1]. Support for these tasks in commercial environments is typically very limited, consisting of browsing manually-managed, sparse categories based on product line or searching on keywords in associated design documentation or attributes.

However, design knowledge—the semantics and concepts behind a design, captured by an annotation and conceptual design interface—can be used in a *design repository* [2,3] to provide enhanced versions of these services. Design repositories are an evolution of design databases which apply techniques and methods developed in the knowledge representation community to capture and reason on collections of designs. This knowledge can be used to enable such capabilities as search based on complex device descriptions, classifying devices into sophisticated categorization schemes, and automatically developing categorization schemes for a given set of designs, organizing the collection. All of these can be used in managing engineering design knowledge, providing improved support for variant design and other tasks.

These goals are closely aligned with that of the Semantic Web—the capture, exchange, utilization, and management of large collections of disparate knowledge. New possibilities for design repositories are raised by the Semantic Web. The ability to annotate Web content with machine interpretable knowledge will enable the easy publication of a wide variety of data—text, CAD, images, simulations—with associated design semantics, improving access for human and machine consumption. This will enhance collaboration as well as software services. Based on a shared, common syntax and representational semantics provided by Semantic Web technology and standards, design knowledge may be more easily exchanged between applications and users. In addition, it facilitates the incorporation of design knowledge into World Wide Web content, allowing for sophisticated domain-specific or more general Web search engines and portal sites to use that information. For example, catalogs with embedded design semantics may be harvested and used in advanced, design repository-based web portals.

In this work the three major components of such a system are developed: a conceptual design/annotation interface, a representation of design semantics, and reasoning mechanisms based on that representation. Figure 1 depicts the interaction between these components. The annotation/conceptual design interface is used to record design semantics, captured in a form defined by the representation. This information can then be used as input to a repository and other reasoning systems to provide a variety of different services.

The components developed in this work feature several prominent aspects. As design is increasingly performed by geographically and organizationally distributed teams, it is essential that secure, collaborative design environments are provided at all stages of design. The conceptual design and annotation interface developed in this work is therefore designed as a collaborative environment and includes several novel security mechanisms. Design semantics are captured via a representation based on engineering function. This representation is defined in a description logic, providing a formal semantics which enables automated reasoning. In addition, the representation makes use of Semantic Web technology, in particular the W3C Ontology Web Language (OWL)[1]. This provides a standard description logic language with which design knowledge captured in this form can be shared among many reasoners and users. The description logic inference upon which the repository reasoning engine is based provides for a great deal of expressiveness while maintaining favorable computational properties. These logics also provide for new classes of reasoning and applications in this domain. In addition, as it is based on description logic semantics and not specifically the domain of engineering design, the reasoning at the core of the repository is applicable to other domains as well.

This paper is organized as follows: The following section briefly reviews work related to each of these components. The approaches taken in this work to developing these components as well as prototype implementation is described in Section 3. System operation is briefly shown in Section 4. Some challenges and interest points are presented in Section 5, followed by concluding remarks in Section 6.

## 2 RELATED WORK

This section reviews related work in collaborative conceptual design and design representation for repository-oriented reasoning.

### 2.1 CONCEPTUAL DESIGN

Approaches to conceptual design fall roughly into two categories: *functional modeling* and *sketching and layout*. Functional modeling [4,5] is rooted in traditional mechanical design. These systems focus on the attributes of and relationships between design elements. Typically these are modeled and presented using an attributed graph, restricted grammar, or tables. Sketch and layout-based conceptual design [6–8] focus more on the general shape and configuration of elements in the design. A common technique is to provide for generating three-dimensional designs from two-dimensional sketches, facilitating rapid generation of approximate shape and configuration. An extensive survey on the state-of-the-art in conceptual design can be found in [9].

The conceptual design interface employed in this work is a combination of these two approaches. Properties and connections between design elements is captured in a functional modeling fashion but are associated with abstract or detailed geometry.

[1]`http://www.w3.org/TR/owl-features/`

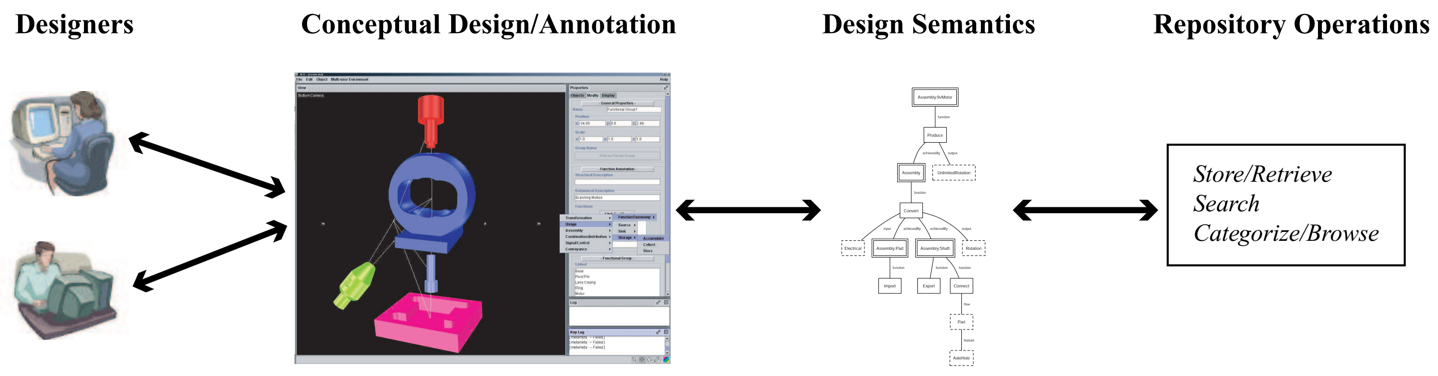


Figure 1. INTERACTION BETWEEN COMPONENTS IN MANAGING DESIGN KNOWLEDGE.

In this way, knowledge of both the function and form of a design may be captured and evolved over time.

Additionally, the conceptual design interface developed in this work addresses several issues necessary for such applications to transition from research to industry. This interface is built on a network architecture in order to support collaborative real-time design. It also includes several novel security mechanisms for protecting design knowledge during collaboration between participants with varying security levels.

## 2.2 ASSEMBLY REPRESENTATIONS

Most familiar to engineers are the representations of devices used in Computer Aided Design (CAD) packages. At the core of these are the solid models of the device's components. Upon these are placed constraints in the form of analytic geometry equations describing the motions present in the mechanism. However, such equations provide little basis for reasoning beyond simulation through constraint solving. In addition, current CAD does not typically capture abstract information such as function in any form suitable for automated reasoning or even efficient human use. There is also little support for devices which operate in multiple domains, e.g. with electrical and mechanical components.

Also familiar to many engineers are representations of function as in [10]. Many representations have been developed to explicitly capture the functions present in an assembly [11–13]. Such representations typically capture function information at various levels of abstraction and across multiple domains, but typically lack the formal framework to support automated reasoning.

Qualitative physics and logic-based representations [14–18] attempt to define the semantics of assemblies in more abstract manners than the geometric equations employed in CAD in order to provide for richer inferences. However, these systems typically do not address many types of inference associated with design repositories, such as determining similarities between devices. In addition, the expressiveness of the languages used often incurs significant cost in terms of computability and tractability.

## 3 SYSTEM OVERVIEW AND COMPONENTS

This section provides an overview of the approach taken in this work to engineering design knowledge management based on design semantics along with a description of a prototype implementation. The following sections outline each of the three core system components shown in Figure 1: the conceptual design/annotation interface with which the designer interacts; the representation used to capture the designer's input; and the reasoning mechanisms which implement the repository's services.

## 3.1 CONCEPTUAL DESIGN: FACADE

The approach taken to conceptual design in this work has three aspects: the core capabilities for conceptual design and semantic annotation, a network architecture to support a collaborative environment, and security provisions for collaboration across organizations and access privileges. These have been implemented inside the FACADE (Framework for Access-control in Computer-Aided Design Environments) system, a multi-purpose CAD architecture, as a collection of modules which may be used or removed as needed. FACADE's implementation has been designed for maximum portability across all platforms. It has been tested and simultaneously developed in the Solaris/SunOS, Linux, and Windows 2000 operating systems. Core functionality is in C++. Some Java components are also used, connected via the Java Native Interface (JNI) standard. Multi-threaded portions of the framework uses Posix Threads (pthreads) under Unix derivatives and Windows Threads under Windows.

The most basic module is a 3D model viewer based on OpenGL and VRML that supports standard camera navigation operations and the ability to view models using different shading algorithms or as a wireframe. Security provisions have been implemented as a compile-time option integrated into this module. A novel access-control mechanism for geometric data called *role-based viewing* [19] has been employed in this work to protect sensitive aspects of the design.

This mechanism is based on a familiar role-based access control framework, but provides several extensions specific to viewing three dimensional data, such as suppressing features of the geometry when a viewer does not have adequate permissions. It also provides several alternatives to this “all-or-nothing” approach, specifically the ability to simplify and distort the geometry. Such a technique is often used in industry, albeit manually. The contribution of this technique is that design details are hidden through automatic simplification of the geometry, allowing for hiding elements entirely or providing partial information based on the viewer’s privileges.

On top of these core viewing capabilities is built a lite design module which supports several basic tasks: selection of a part, assembly, or other selectable entity; applying affine transformations; setting opacity; decomposing a part into multiple parts; manipulating control points on parametric surfaces (eg. bezier patches, splines, and NURBS); and grouping parts and assemblies into abstract assemblies. This subsystem is a prerequisite for most of the other modules. For example, the security module already described requires this module in order to provide an interface for authorized users to assign roles and security parameters for later viewing by other users.

Collaborative design is supported by a client-server architecture which provides for real-time multi-user viewing and editing. A network module in the client interacts with a FACADE server that acts as an authoritative source for approving client requests. The server works with the security module to provide role-based views to clients which do not have permission to manipulate or view a model or its semantic features at full resolution. Consistency is maintained throughout all connected clients by sending rejection messages whenever a requested operation causes a conflict. Both “thin” and “fat” clients are supported by the server. Thin clients rely on the server to produce an image of the current geometry, which is transmitted using the Remote Frame Buffer (RFB) protocol [20]. Fat clients transmit and receive transformation requests and resulting geometry to and from the server using a FACADE-specific protocol. Network code uses BSD-style sockets under Unix-based derivatives and Winsock2 under Windows.

At the focus of this paper is the module supporting conceptual design and semantic authoring. Three dimensional sketch functionality is provided by the ability to insert primitive shapes into the design or load pre-designed components to be used in creating an assembly. These may then be altered using the previously described lite design module. Annotation of detailed designs is enabled by the ability to load pre-defined geometry, for example geometry exported from another CAD product.

Design semantics may be associated with the design through functionality for attributing and linking design elements with function and flow information, as described in the following section. FACADE extracts annotation relations and attributes from OWL ontologies which have been retrieved from the World Wide Web and processed by an OWLJessKB reasoner incorporated into this module. Annotations may be saved along with geometry or exported to an OWL file and published to the Web.

## 3.2 REPRESENTATION: FUNCTION AND FLOW SIGNATURES

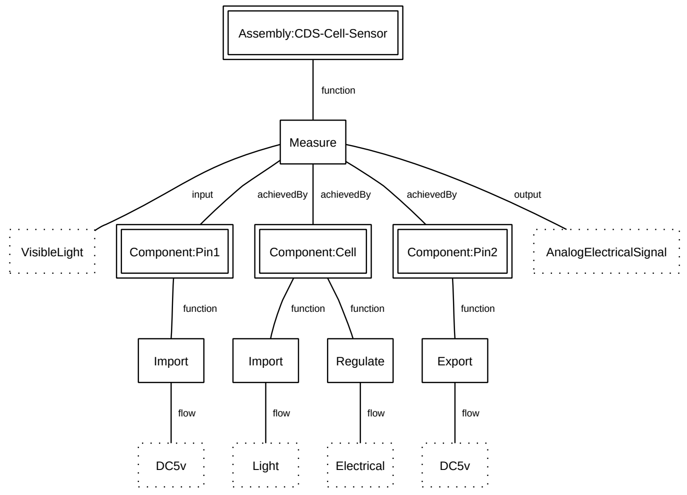
Design semantics are captured in this work by *function and flow signatures* [21], a representation based on current research in functional modeling [11–13]. This representation explicitly captures the function of a device—the goals it achieves and tasks it performs. Functions may be decomposed into subgoals which subassemblies or components of the device must accomplish. Functions operate on flows which generally represent materials, signals, and energy. Examples include moving items down a conveyor belt, transmitting a control signal, and reducing a rotational speed.

These signatures most resemble the diagrams of [11] but differ in that this work defines these signatures inside a formal framework, using a description logic. In this way automated reasoning can be performed on instances of this representation. This contrasts with most work in this area, the representations of which are largely suitable solely for human consumption or simple keyword searching.

Figure 2 gives a model under this representation for a common type of light sensor, a cadmium sulfide cell. These sensors are photoresistors, the presence of light decreases the resistance encountered by electricity flowing across the cell. This information can be represented by a function and flow signature as in Figure 2(a). By interpreting the signatures as a set of objects and relations, they can be represented in a description logic model as in Figure 2(b).

This representation does not rigorously and unambiguously capture the semantics of mechanisms. Instead, it provides a language expressive enough to describe and distinguish devices while maintaining efficiency and computability. It is neither so formal as to prevent practical computing, nor so informal as to prohibit automated reasoning [21].

As shown in Figure 3, the representation syntax chosen for this work is the W3C Ontology Web Language (OWL). The core ontology describing the basic objects and relations as well as taxonomies providing an extensive set of function and flow labels are encoded as OWL ontologies and published to the World Wide Web. This makes them accessible to any applications which may need them, such as the semantic authoring interface described in the previous section or non-repository-specific reasoners, such as generic knowledge-based Web search engines. Figure 3(a) provides representative snippets of the ontologies which define this representation. The function and flow model presented in Figure 2 is shown encoded in OWL syntax in Figure 3(b).



(a) Function and flow signature in graphical form.

```
⟨ ⟨ type CDS-Cell-Sensor Assembly ⟩
  ⟨ function CDS-Cell-Sensor node7 ⟩         ⟨ type Cell Component ⟩
  ⟨ function Pin2 node8 ⟩                    ⟨ achievedBy node7 Pin1 ⟩
  ⟨ type node0 AnalogElectricalSignal ⟩      ⟨ type Pin2 Component ⟩
  ⟨ flow node1 node2 ⟩                       ⟨ type node9 Import ⟩
  ⟨ flow node3 node4 ⟩                       ⟨ type node4 Electrical ⟩
  ⟨ function Cell node3 ⟩                    ⟨ function Cell node9 ⟩
  ⟨ type node2 DC5v ⟩                        ⟨ achievedBy node7 Pin2 ⟩
  ⟨ type node5 Light ⟩                       ⟨ flow node9 node5 ⟩
  ⟨ type node6 VisibleLight ⟩                ⟨ flow node8 node10 ⟩
  ⟨ achievedBy node7 Cell ⟩                  ⟨ type node8 Export ⟩
  ⟨ type node3 Regulate ⟩                    ⟨ output node7 node0 ⟩
  ⟨ type node7 Measure ⟩                     ⟨ function Pin1 node1 ⟩
  ⟨ type node1 Import ⟩                      ⟨ type node10 DC5v ⟩
  ⟨ input node7 node6 ⟩                    ⟩
  ⟨ type Pin1 Component ⟩
```

(b) Figure 2(a) in description logic model form.

Figure 2. Function and flow modeling of a Cadmium Sulfide (CDS) Cell, a common photoresistor.

```
<owl:Class rdf:about="&eng;#Artifact" />
<owl:Class rdf:about="&eng;#Function" />
<owl:Class rdf:about="&eng;#Flow" />

<owl:Class rdf:about="&eng;#Assembly">
  <rdfs:subClassOf rdf:resource="&eng;#Artifact" />
</owl:Class>

<rdf:Property rdf:about="#function">
  <rdfs:domain rdf:resource="#Artifact" />
  <rdfs:range rdf:resource="#Function" />
</rdf:Property>

...

<owl:Class rdf:about="&eng;#Function">
  <owl:disjointWith rdf:resource="&eng;#Flow" />
  <owl:disjointWith rdf:resource="&eng;#Artifact" />
</owl:Class>

...

<owl:Class rdf:about="&flow;#AnalogElectricalSignal">
  <rdfs:subClassOf rdf:resource="&flow;#Electrical" />
  <rdfs:subClassOf rdf:resource="&flow;#Signal" />
</owl:Class>
```

(a) Snippets from core ontology and vocabulary extensions.

```
<eng:Assembly rdf:about="#CDSCellSensor">
  <eng:function>
    <func:Measure>
      <eng:input><flow:VisibleLight /></eng:input>

      <eng:achievedBy>
        <eng:Component rdf:about="#Pin1">
          <eng:function>
            <func:Import>
            <eng:flow>
              <flow:DC5v />
            </eng:flow>
            </func:Import>
          </eng:function>
        </eng:Component>
      </eng:achievedBy>

      ...

      <eng:output>
        <flow:AnalogElectricalSignal />
      </eng:output>
    </func:Measure>
  </eng:function>
</eng:Assembly>
```

(b) Portion of CDS Cell representation as given in Figure 2.

Figure 3. Ontologies and data in OWL form.

## 3.3 REASONING: OWLJESSKB

Core reasoning support in this system is provided by OWLJessKB, based on DAMLJessKB [22]. OWLJessKB is a description logic reasoner implemented inside a production system. OWL documents in RDF-XML form are retrieved from the World Wide Web and converted into triples. These are asserted into the knowledge base and rules implementing the OWL semantics are then applied. Applications such as FACADE or a web portal then query the knowledge base to extract inferred knowledge or may apply their own application-specific reasoning inside the production system.

OWLJessKB is a sound description logic reasoner, given that it makes a non-standard closed world assumption. Although this produces some technically incorrect inferences, these tend to be useful in practice as much Semantic Web data does not include specific closure axioms where appropriate. To promote practical run-times, OWLJessKB has been designed as an incomplete reasoner. Uncommon inference rules are not incorporated into the rule base, reducing its size and improving its performance. This means there exist valid inputs which OWLJessKB may not process, but these are rare in practice.

OWLJessKB uses the Java Expert System Shell[2] (JESS) as the underlying production system. The combination of JESS and OWLJessKB provides a solid application development framework for working with Semantic Web ontologies and data, especially in environments well suited for Java development, such as web services and embedded devices.

Standard description logic inferences can be used in a number of repository reasoning tasks. In addition to being interpretable as statements in a description logic model, the function and flow signatures presented in this paper may also be interpreted as concept descriptions. In this way, they may be used to manually create a categorization against which devices in the repository are sorted. Search functionality may be similarly implemented, treating the query as a concept description and classifying device instances against that definition. Description logic *subsumption* may be used to manually construct a categorization in a bottom-up, generally more intuitive fashion. Less standard inferences may also be applied, for example induction of the *least common subsumer* of sets of classes may be used to automatically create a categorization from a collection of devices. This has the potential to be a powerful information management tool once techniques have been developed to address several issues, e.g. the density of the generated hierarchy. Repository capabilities provided by these inferences and their relationship to traditional databases is discussed further in [21].

The Semantic Web introduces several new capabilities to design repositories. In particular, the ability to markup device-related Web content with the representation outlined in this paper makes it straightforward to publish a wide variety of data—text, CAD, images, simulations—with associated design semantics to any number of repositories and other reasoning engines. Within engineering design, this makes it easier for loosely coupled design teams to collaborate, for example by identifying redundant design efforts. Another application is for part catalogs and similar materials to be marked up on corporate websites. That information may then be aggregated by a portal site providing an interface to many vendors. With a repository running at the core of the site, large amounts of data may be collected, organized, and utilized in more robust and sophisticated ways than under current practice.

## 4 SCENARIO

Figure 4 gives an example of using this system to query a repository of designs. In Figures 4(a)–4(c) the designer creates a three-dimensional sketch of a simple sensor. This sketch is then annotated with the desired functions and flows, as in Figures 4(d) and 4(e). Figure 4(f) depicts the function and flow signature as exported from the conceptual design interface. The reasoner then converts the signature into a class description, shown in Figure 4(g). Members of this class description, those designs which match the signature, are then returned to the designer as the results of the query.

## 5 DISCUSSION

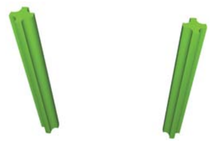
The following briefly outlines a few issues raised in the development of this system, particularly in regard to the Semantic Web.

**Effective User Interfaces.** There exist many efforts to develop domain neutral, generic annotation tools for the Semantic Web. For many domains and applications, such as engineering design, these will be unsuitable. Users are uncomfortable with knowledge representation, the process is too slow, the data too complex. However, it is advantageous to uncouple these tools from specific ontologies and increase their applicability. FACADE currently loads its labels from OWL ontologies. Use of a core ontology requires implementation of an annotation module compliant with and supportive of its structure. If the tool incorporated more reasoning, it would be possible to leverage and support ontologies in a more dynamic fashion. The challenge would be to maintain flexibility while providing an interface that is specific or adaptive enough to remain relevant and effective. Additionally, the difficulty faced in hiding the details of knowledge representation will increase with the expressiveness of the underlying languages.

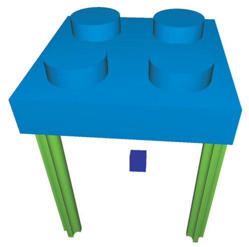
**Knowledge Association and Linking.** A large problem faced in developing FACADE was the mechanisms by which to connect Semantic Web-encoded knowledge with traditional data. Currently, the association is maintained at the artifact level by names shared between the functional modeling and objects of the VRML geometry upon which FACADE primarily operates. Less clear is how to maintain this connection at other levels and to other media—e.g. between a function and a feature, with several associated CAD models. Also not obvious is how best to store and distribute media while maintaining links to associated material. Fortunately, general techniques for addressing these issues will be applicable across a range of domains and applications.

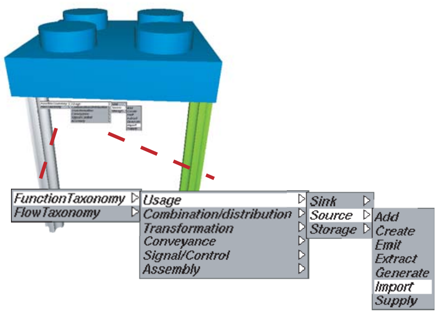
**Expressiveness, Computation, and Development.** The bulk of current work on the Semantic Web employs relatively constrained logical foundations, most frequently description logics. These place substantial limits on expressiveness in favor of computability and tractability. Such logics can not capture even min-

[2]`http://herzberg.ca.sandia.gov/jess/`

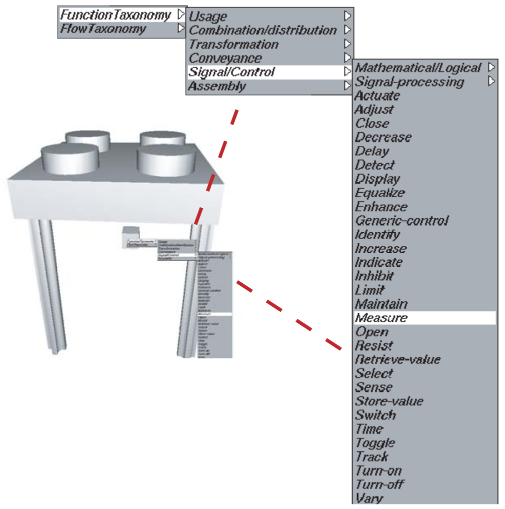

(a) The designer places generic shapes to represent the sensor leads.

(b) A plate is placed to stand in for the sensor body.

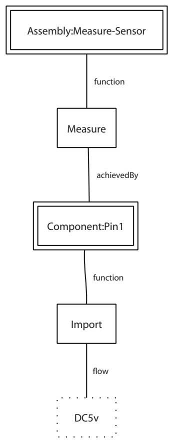

(c) All three components are grouped into an assembly, represented by a box.



(d) One lead is annotated as achieving the *Import* function.



(e) The assembly is annotated as achieving the *Measure* function.



(f) The function and flow signature is exported from the annotations.

$$\textit{Measure-Sensor} \equiv \textit{Assembly} \sqcap \exists \textit{function}.[\textit{Measure} \sqcap \exists \textit{achievedBy}.[\textit{Component} \sqcap \exists \textit{function}.[\textit{Import} \sqcap \exists \textit{flow}.\textit{DC5v}]]].$$

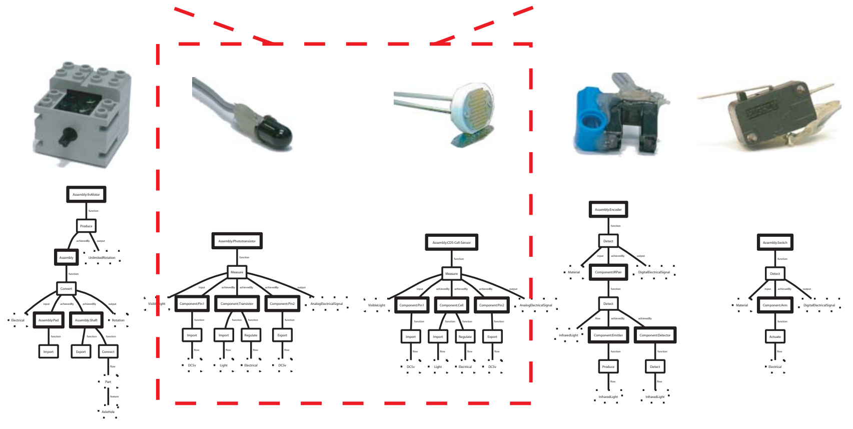

(g) The signature is converted to a class description against which designs are classified.

Figure 4. An example of creating a query for electronic sensors which measure a flow.

imal domain semantics in engineering. The challenge then is to maximize the benefits of accepting these limitations. For example, function and flow signatures do not rigorously and unambiguously capture design semantics. Instead, they provide a language expressive enough to describe and distinguish devices while maintaining efficiency and computability. They are neither so formal as to prevent practical computing, nor so informal as to prohibit automated reasoning. Importantly, they are also well suited to the application—description logics offer a variety of inferences and abilities well suited to design repositories, such as classification and the least common subsumer.

**Syntax and Canonical Form.** Prominent among limitations of function and flow signatures and similar representations is that it is possible to create distinct models of the same design. That these are not equivalent will be missed due to the lack of true semantics. However, in addition to being difficult to develop, ontologies strong enough to overcome this will probably face significant practical and theoretical complexity problems. An approach to addressing this issue might be to employ hierarchical representations with varying levels of semantics for use in different tasks. For example, rich semantics may be used to determine canonical or simpler form for given input, which may then be submitted to more complex processes such as insertion into the repository.

A related issue stems from the fact that although there is recognition in industry of the benefits of applying advanced techniques to engineering design, there is also a great deal of inertia. To overcome this, bridges to legacy and classical applications will have to be developed as knowledge-based tools are brought on line. One approach, taken by the Process Specification Language Project (PSL)[3], is to develop grammars for canonical forms associated with certain classes and relations. Adapters and other software may then correctly parse much valid data via the grammars rather than extensive reasoning.

**Tractability and Specialized Reasoning** These techniques raise issues regarding the validity of software which cannot accept all valid inputs, particularly of concern in the diverse environments of the World Wide Web and the agent-populated Semantic Web. Such tradeoffs are similar to those made in restraining the logical foundations of Semantic Web, and highlight other possibilities. Although popular for their decidability and fast reasoning in practice, many of all but basic description logics are theoretically intractable. In addition, even fast implementations may be overcome by applications such as real-world repositories, which may consist of thousands to millions of parts.

The approach to this problem under investigation in this work is to indirectly apply the semantics of the representation, such as the aforementioned conversion to canonical form. In addition, the correctness of special-purpose algorithms may be verified via comparison against the ontology. Importantly, the semantics of the representation may guide the development of and provide meaning to specialized algorithms and techniques, e.g. ontology-optimized categorization techniques based on the least common subsumer.

[3]An ontology of manufacturing processes—`http://nist.gov/psl/`.

## 6 CONCLUSIONS

Supporting the early stages of engineering design requires conceptual design tools with provisions for practical use, such as security and real-time collaboration. It also requires the ability to capture both the form of the design and the underlying functional semantics. Properly captured, such knowledge can be used in a variety of manners, from recording design rationale to working with large collections of engineering data. In addition, technology and results from the developing Semantic Web may ease this process and enable new applications. This paper has introduced work addressing each of these areas, and is hoped to contribute to the introduction of such capabilities to industry.

## ACKNOWLEDGMENT

This work supported in part by National Science Foundation (NSF) CAREER Award CISE/IIS-9733545 and Office of Naval Research (ONR) Grant N00014-01-1-0618. Additional support by Honeywell FM&T, AT&T Labs, Bentley Systems and Lockheed Martin, Naval Electronics and Surveillance Systems. All opinions, findings, and conclusions expressed in this material are those of the author(s) and not necessarily those of the supporting organizations.

## REFERENCES

[1] Ullman, D. G., 1997. *The Mechanical Design Process*. McGraw-Hill, Inc. ISBN 0-07-065756-4.

[2] Szykman, S., Bochenek, C., Racz, J. W., Senfaute, J., and Sriram, R. D., 2000. "Design repositories: Engineering design's new knowledge base". *IEEE Intelligent Systems*, **15**(3), May/June, pp. 48–55.

[3] Szykman, S., Sriram, R. D., and Regli, W. C., 2001. "The role of knowledge in next-generation product development systems". *ASME Transactions, the Journal of Computer and Information Science in Engineering*, **1**(1), March, pp. 3–11.

[4] Serrano, D., and Gossard, D., 1992. "Tools and techniques for conceptual design". In *Artificial Intelligence in Engineering Design: Design Representation and Models of Routine Design*, C. Tong and D. R. Sriram, eds., Vol. I. Academic Press, 1250 Sixth Ave, San Diego, CA, ch. 3, pp. 71–116. ISBN 0-12-660561-0.

[5] Al-Hakim, L., Kusiak, A., and Mathew, J., 2000. "A graph-theoretic approach to conceptual design with functional perspectives". *International Journal of Computer Aided Design*, **32**(14), December, pp. 867–875. Special issue on Conceptual Design.

[6] Qin, S., Wright, D., and Jordanov, I., 2000. "From on-line sketching to 2d and 3d geometry: a system based on fuzzy knowledge". *International Journal of Computer Aided Design*, **32**(14), December, pp. 851–866. Special issue on Conceptual Design.

[7] Lipson, H., and Shpitalni, M., 2000. "Conceptual design and analysis by sketching". *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing (AIEDAM)*, **14**(5), November, pp. 391–402.

[8] Eggli, L., Hsu, C. Y., Bruderlin, B. D., and Elber, G., 1997. "Inferring 3d models from freehand sketches and constraints". *Computer-Aided Design*, **29**(2), February, pp. 101–112.

[9] Wang, L., Shen, W., Xie, H., Neelamkavil, J., and Pardasani, A., 2001. "Collaborative conceptual design–state of the art and future trends". *Computer-Aided Design*, **34**(13), pp. 981–996.

[10] Pahl, G., and Beitz, W., 1996. *Engineering Design—A Systematic Approach*, 2nd ed. Springer, London, UK.

[11] Szykman, S., Racz, J. W., and Sriram, R. D., 1999. "The representation of function in computer-based design". In ASME Design Engineering Technical Conferences, 11th International Conference on Design Theory and Methodology, ASME, ASME Press. DETC99/DTM-8742.

[12] Hirtz, J., Stone, R., McAdams, D., S, S., and Wood, K., 2001. "Evolving a functional basis for engineering design". In ASME Design Engineering Technical Conferences, 13th conference on Design Theory and Methodology, ASME, ASME Press. DETC01/DTM-21688.

[13] Kirschman, C., Fadel, G., and Jara-Almonte, C., 1996. "Classifying functions for mechanical design". In ASME Design Engineering Technical Conferences, 8th conference on Design Theory and Methodology, ASME, ASME Press. DETC96/DTM-1504.

[14] Faltings, B., 1990. "Qualitative kinematics in mechanisms". *Artificial Intelligence*, **44**, pp. 89—119.

[15] Kuipers, B., 1984. "Commonsense reasoning about causality: Deriving behavior from structure". *Artificial Intelligence*, **24**, pp. 169—204.

[16] Forbus, K., 1984. "Qualitative process theory". *Artificial Intelligence*, **24**, pp. 85–168.

[17] Subramanian, D., and Wang, C., 1993. "Kinematic synthesis with configuration spaces". In Proceedings of Qualitative Reasoning 93,, pp. 228–239.

[18] Joskowicz, L., and Neville, D., 1996. "A representation language for mechanical behavior". *Artificial Intelligence in Engineering*, pp. 109—116.

[19] Cera, C. D., Kim, T., Han, J., and Regli, W. C., 2004. "Role-Based Viewing in Secure Collaborative CAD". *Accepted to the Journal of Computer Aided Design*.

[20] Richardson, T., and Wood, K. R., 1998. The RFB Protocol version 3.3. Protocol Specification, July.

[21] Kopena, J. B., and Regli, W. C., 2003. "Functional modeling of engineering designs for the semantic web". *IEEE Data Engineering Bulletin*, **26**(4), December, pp. 55–62.

[22] Kopena, J. B., and Regli, W. C., 2003. "DAMLJessKB: A tool for reasoning with the semantic web". In International Semantic Web Conference, Springer Verlag, pp. 628–643.