2007

Design and Innovative Methodologies in a Semantic Framework

Rui Fernandes
University of Massachusetts Amherst

Ian R. Grosse
University of Massachusetts Amherst, grosse@ecs.umass.edu

Sundar Krishnamurty
University of Massachusetts Amherst, skrishna@ecs.umass.edu

Jack C. Wileden
University of Massachusetts Amherst, wileden@cs.umass.edu

Follow this and additional works at: http://lib.dr.iastate.edu/edesign_conf

Part of the Industrial Engineering Commons, and the Programming Languages and Compilers Commons

Recommended Citation
http://lib.dr.iastate.edu/edesign_conf/11

This Conference Proceeding is brought to you for free and open access by the Center for e-Design at Iowa State University Digital Repository. It has been accepted for inclusion in Center for e-Design Proceedings by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digireps@iastate.edu.
Design and Innovative Methodologies in a Semantic Framework

Abstract
Significant expenditure and effort is devoted to the never ending search for reduced product development lifecycle time and increased efficiency. The development of Semantic Web technologies promises a future where knowledge interchange is done seamlessly in open distributed environments. This paper illustrates how Semantic Web technologies in their current state of development can be effectively used to deploy an infrastructure supporting innovation principles and the engineering design processes. A mechanical design was chosen to model the initial phase of a design project using semantic ontologies. This included a set of design requirements, creating a functional model, and making use of the Theory of Inventive Problem Solving (TIPS). The ontology development strategy is built on a combination of larger domain knowledge ontologies and simple process ontologies. Linked user requirements, engineering design, and functional modeling ontologies facilitated the application of TIPS through a set of semantic rules to generate design recommendations. The developed semantic knowledge structure exemplifies a practical implementation of a functional model which served as a record of the design process and as a platform from which to gain additional usefulness out of the stored information.

Keywords
Design

Disciplines
Industrial Engineering | Programming Languages and Compilers

Comments
DESIGN AND INNOVATIVE METHODOLOGIES IN A SEMANTIC FRAMEWORK

Rui Fernandes, Ian Grosse, Sundar Krishnamurty
University of Massachusetts Amherst
Department of Mechanical and Industrial Engineering
220 ELAB
Amherst MA 01003
rpfernan@student.umass.edu, grosse@ecs.umass.edu, skrishna@ecs.umass.edu

Jack Wileden
University of Massachusetts Amherst
Department of Computer Science Box 3461
Amherst MA 01003
jack@cs.umass.edu

ABSTRACT
Significant expenditure and effort is devoted to the never ending search for reduced product development lifecycle time and increased efficiency. The development of Semantic Web technologies promises a future where knowledge interchange is done seamlessly in open distributed environments. This paper illustrates how Semantic Web technologies in their current state of development can be effectively used to deploy an infrastructure supporting innovation principles and the engineering design processes. A mechanical design was chosen to model the initial phase of a design project using semantic ontologies. This included a set of design requirements, creating a functional model, and making use of the Theory of Inventive Problem Solving (TIPS). The ontology development strategy is built on a combination of larger domain knowledge ontologies and simple process ontologies. Linked user requirements, engineering design, and functional modeling ontologies facilitated the application of TIPS through a set of semantic rules to generate design recommendations. The developed semantic knowledge structure exemplifies a practical implementation of a functional model which served as a record of the design process and as a platform from which to gain additional usefulness out of the stored information.

INTRODUCTION
The development of knowledge infrastructure technologies has the potential to provide engineers with tools and processes to access and operate on high level engineering knowledge. Distributed methods for accessing and modeling engineering knowledge in an open cooperative framework have been developed, yet practical applications have proven to be difficult and elusive. The goal of creating a global knowledge architecture relating information from different domains within an open architecture poses many challenges which range from the ability to search large distributed knowledge structures to the need to satisfy a wide range of applications. At a smaller scale and within focused applications the benefits of the Semantic Web are still relevant, but many of the concerns do not play a dominant role.

Engineering design innovation is driven by the experience and creativity of the designer to solve specific problems. Experience provides a pool of conceptual analogies from which the designer derives possible design concepts. Creativity gives the designer the ability to apply non-obvious concepts to address design issues. Much like the study of the game of chess where a player gains an advantage by studying historical games and understanding different strategies and how they relate to specific game situations, a designer can gain an advantage by having a large pool of design concepts and problem solving methodologies to draw upon. Unlike chess where there is a well established and accepted terminology for describing situations, engineering design has relied on textual descriptions and the associated lack of precision. General design taxonomies such as the Functional Basis [11], TIPS [3], Object-Oriented Representation [7], and application specific taxonomies such as ship hull design [17], manufacturing and design [6], assembly design [12] provide the necessary basis for creating a set of ontologies [2] to unambiguously describe engineering designs. From these ontologies a semantic knowledge infrastructure can be created to provide designers access to a virtual repository of information directly related to design issues. Such a repository would allow designers to query a large pool of knowledge for potential design solutions [23].

In this paper we show that the functionality of the Semantic Web can be harnessed to apply innovation methods to engineering design knowledge. First we discuss the
development of a strategy for creating a semantic infrastructure such that it works with the existing semantic tools and protocols. The infrastructure developed is based on a series of small application focused ontologies and larger ontologies based on established taxonomies. The resulting knowledge structure provides a platform for designers to document designs, and create functional models. A series of semantic rules allow designers to apply innovation methods to functional models and generate design options. To finalize, the potential of this infrastructure is demonstrated with an example design project, which includes a set of design requirements, a functional model, and a series of generated design recommendations.

**SEMANTIC INFRASTRUCTURE**

The design knowledge was modeled using the Semantic Web infrastructure, which “is based on the idea of having data on the Web defined and linked such that it can be used for more effective discovery, automation, integration, and reuse across various applications” [8]. The Semantic Web knowledge representation protocols, Web Ontology Language (OWL), and Semantic Web Rule Language (SWRL) provide a way for representing knowledge as a series of related terms which are machine operable [4]. An ontology as defined within the context of the Semantic Web is a formal representation of the relationship among terms [4]. Using OWL taxonomies can be represented and shared across the internet such that others can make use of the represented knowledge. The ability to link to external ontologies combined with the ability of computers to operate on the stored knowledge promises direct access to richer and more meaningful information.

Both OWL and SWRL are text-based languages that make use of the same technologies used throughout the internet for accessing web pages. OWL uses Extensible Markup Language (XML) as its basic syntax and Resource Description Framework (RDF) to describe the relationship between terms. SWRL is an extension to OWL. It provides a way to implement rules that operate on OWL knowledge representations. All OWL and SWRL development in this project was carried out using Protégé, an ontological development tool developed by Stanford Medical Informatics at the Stanford University School of Medicine (http://protege.stanford.edu).

The complexity associated with modeling general purpose information such as a generic product ontology is one of the major challenges which will need to be overcome before the full potential of the semantic web is realized [9]. The challenges of creating semantic representations range from creating base representations allowing interoperability across domains [14], searching large distributed knowledge bases [5], to integrating separate ontologies [15]. Unlike centralized information systems where transactions can be clearly delimited the state of information is not necessarily available requiring more complex reasoning [8].

For this project an approach was chosen to balance complexity and feasibility. Enough detail, or complexity, is needed for rich representations, yet large knowledge structures can be unwieldy and difficult to implement. Specific attention was paid to the volatility of the knowledge representations. Highly volatile ontologies subject to regular modifications and can be difficult to integrate into larger knowledge frameworks, while more static ontologies can be relatively large without imposing development and usability barriers. A full description of the field of engineering design requires many taxonomies, ranging from generic concepts such as units systems, to taxonomies dedicated to the design of a single product family. It is not possible to include every possible taxonomy in a semantic infrastructure as it would quickly become unwieldy and too complex. Yet over time designers are likely to need a great number of taxonomies in order to create full descriptions of design scenarios.

The solution chosen for this project is a framework based on dynamic links between ontologies rather than the integration of ontologies into a single large ontology. The ontologies were grouped into two categories, process ontologies and domain knowledge ontologies. Process ontologies define and control the engineering design process; they are more specific to individual organizations, and they cover topics such as people, design process, and project management. Process ontologies, like processes in general, change over time from organization to organization and are not likely to be shared with external entities. Domain knowledge ontologies define the concepts used to describe and define scenarios. They represent areas of knowledge such as units systems, functional models, and engineering analysis. Domain knowledge ontologies are less volatile and less application specific than process ontologies. The separation of domain knowledge from process modeling ontologies creates focused knowledge representations, allowing the sharing of knowledge regardless of the chosen process model. The flexibility inherent in this knowledge modeling approach fosters the organic growth of a distributed semantic knowledge infrastructure.

**THE ONTOLOGIES**

The goal of the ontologies developed for this project was to model the engineering design process with emphasis on applying innovative methodologies. Two domain knowledge ontologies related to design and innovation were developed for this project. The development of the TIPS and Functional Basis ontologies, followed defined taxonomies and care was taken to keep the representations general without implied applications, maximizing the ability to share knowledge. The Functional Basis [11] taxonomy was chosen to model device functionality as there are a variety of methodologies [13, 19, 21] which operate on Functional Basis models. The Function Basis class structure provides a good starting point for developing a semantic web ontology. TIPS was chosen as the innovation method to operate on the design knowledge as it readily allows for mapping into a semantic ontology [24]. Further, TIPS methods for generating design suggestions can be readily implemented using semantic rules. Three minimalist process ontologies- People, Project, and Design were developed...
specifically for the application at hand, keeping development effort and complexity to the minimum while capturing enough information to provide a sufficiently complete representation of the design process.

The Functional Basis (Figure 1) ontology is an OWL representation based on the Functional Basis taxonomy described in the National Institute of Standards and Technology (NIST) technical note 1477 [11]. It provides a consistent methodology for decomposing and describing functional behavior based on material, energy, and signal flows, and functions that act on the flows. In addition to the class structure directly based on the Functional Basis taxonomy whose root classes are Flow and Function, classes were added in order to define functional models. The classes added are Functional Model, Operation, and Input Flow (Figure A1) which are used to build functional models. The goal of these classes is to create individual functional models which must have inputs, outputs, and perform at least one operation. The Functional models can also be nested in order to create sub-models or operation groupings. Operations are the basic building blocks of the functional models. They have input and output flows and perform one function. Input flow is a “bookkeeping” class, containing both a flow and its source(s) which allows the tracking of flows from one operation to another. An output flow class is not necessary as it would contain redundant information. The Functional Basis ontology is a pure class structure comprised of over 200 classes and no instances.

Figures 2 through 4, 6, and A1 through A5 are representations of the semantic class structure produced using the Protégé ontology authoring software package. The boxes represent classes and the arrows represent the parameters defining class relations. The class boxes make use of two formats: showing the class properties (Figure 2), or in an abbreviated format (Figure 4) where only the class names are shown. When shown, the class properties display the variable name, variable type, and cardinality. If more than one value can be associated with a variable an asterisk is present along with the variable type name.

The TIPS ontology (Figure 2) is based on the Theory of Inventive Problem solving methodology [3] and is used to generate suggestions based on design compromises associated with the design representations. TIPS relates a series of 40 principles of invention to design contradictions in order to solve the contradictions. Figure 2 shows the class structure of the TIPS ontology, where the Design Contradiction class is composed of 2 engineering parameters, one improving and one worsening feature, and one or more principles of invention. The Principle of Invention class contains one or more suggestions. The TIPS ontology, in contrast to the Functional basis ontology, is a simple knowledge structure with 4 classes but containing over 1600 instances.

The processes ontologies are largely driven by the needs of its users. The People ontology (Figure A2) addresses the need to link activities to people. It is comprised of basic identification and role information. Additional properties can be assigned to different classes of users as the need arises. The Project ontology (Figure A3) deals with project identity, requirements, timelines, and tasks. Note that the Project ontology makes use of properties defined in the people class. This linking is what allows the sharing of ontologies and the application of external knowledge structures.

The Design ontology (Figure 3) serves as the hub for the
framework. It is the only ontology which is accessed directly, as all the other ontologies are accessed on a read-only basis and are served from a web server. The Design ontology links to all the other ontologies used in this project. It provides certain degree of flow control through enforceable information dependencies and contains the SWRL rules used to implement TIPS. The Design class has a basic text description, one or more requirements as defined in the Project ontology, an optional associated design, an optional geometric model, optional documentation, design parameters, design contradictions, and one or more functional models. The Design parameter class represents important design metrics used in the evaluation of designs. It is composed of a text description, and links to relevant requirements and functional model. The Geometric Model and Documentation classes link to external documentation and geometric models. The Design Contradiction class behaves differently from other classes in this framework. In Figure 4 we see that although the Design Contradiction class is defined locally within the Design ontology. It can be assigned as a property of a remote class. In this case it becomes a property of Functional Model and Operation classes of the Functional Basis ontology.

The extensibility of Semantic Web is one of its most powerful features. It allows developers to go beyond just linking to external ontologies by enabling them to actively expand them for their purposes. In this project the Functional Basis ontology does not have a design contradiction class. Yet within our Design ontology a design contradiction is defined as a property of the Functional Model and Operation classes of the Functional Basis ontology. Without this functionality the process of implementing the TIPS methodology would require significantly more effort than what was required for this project.

Once a design contradiction property is added to functional models a rule is defined such that it compares any instance of a design contradiction with the TIPS design contradictions stores in the TIPS ontology if the design contradiction property is defined.

---

**Figure 3.** Class structure of the Design ontology.

<table>
<thead>
<tr>
<th>Description</th>
<th>String</th>
</tr>
</thead>
<tbody>
<tr>
<td>description</td>
<td></td>
</tr>
<tr>
<td>has_design_contradiction</td>
<td>has_functional_basis_model</td>
</tr>
<tr>
<td>has_design_parameter</td>
<td></td>
</tr>
<tr>
<td>has_requirement</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 4.** The Design contradiction class links Design, Functional Basis, and TIPS ontologies.
contradiction matches TIPS design contradiction it copies the relevant design principles along with the design suggestions from TIPS and adds them to the local ontology. This required one simple rule:

\[
\begin{align*}
\text{Design\_contradiction}(?x1) \\
\text{worsening\_feature}(?x1, ?x3) \\
\text{improving\_feature}(?x1, ?x2) \\
\text{TIPS:Design\_contradiction}(?x4) \\
\text{TIPS:worsening\_feature}(?x4, ?x3) \\
\text{TIPS:improving\_feature}(?x4, ?x2) \\
\text{TIPS:principle\_of\_invention}(?x4, ?x5) \\
\rightarrow \text{principle\_of\_invention}(?x1, ?x5)
\end{align*}
\]

This rule translates to: If a local Design Contradiction exists with worsening feature A and improving feature B and there exists a TIPS Design Contradiction with worsening feature A, improving feature B, and principle of invention C then the local Design Contradiction must also have principle of invention C.

Empirical study has shown that ontological models of engineering design should address four major classifications: design process, product, functions, and issues [1]. Design process classification deals with the structure of the design process and its implementation. Product classification addresses the product composition: assemblies, components, and subassemblies. The functional classification deals with delineating how a product should work. Issues represent implicit and explicit requirements, or design constraints. The semantic framework developed covered the necessary classifications to establish a complete representation of the design process. Certain aspects are more fully developed than others which represent the emphasis of this project on innovation methods. The concept of a product classification is present in the sense that design efforts have a set of requirements which are linked to the functional model and ultimately lead to product features and geometric representations. The design process methodology is inherent in the way the information is constructed; product features depend on the design representations, which contain the functional model, which depend on the requirements which are contained within the project definition. The Project Ontology addresses the design constraints with a generic requirements class which can be further expanded into requirements specifications.

**RESULTS AND DISCUSSION**

In order to test the developed semantic framework the Functional Basis model of a cordless screwdriver developed by Stone et al. [22] was chosen. The model was entered into the semantic framework using Protégé. Figure 5 shows a data entry form in Protégé displaying the Design class instance for the cordless screwdriver. Figure 5 also shows how this knowledge structure can be used to annotate the context under which the design takes place.

Figure 6 shows a diagram of the screwdriver model, with the cordless screwdriver functional model being composed of 4 submodels: power supply, electric motor, chassis, and drive mechanism. The functional model of the power supply is
shown in Figure 7. Unlike a cordless screwdriver model composed of submodels, the power supply is composed of the root building blocks of functional models, operations. The development of the functional model can provide the designer with a new perspective on the problem at hand and ensures that solutions developed are consistent with functional specifications. The structure of models and submodels provides the designer with the ability to experiment with different functional groupings. In the case of the cordless screwdriver the power supply provides a good example on how different functional groupings can lead to different designs where the charging circuitry can be considered part or the power supply, or as an external charger. Figure A4 shows a diagram of the store electricity operation with associated inputs and outputs.

The functional models along with the design parameters can provide the designer with significant insights into the design problem at hand. In the example shown, one of the design parameters likely to play a major role in innovative solution is mass. Using this insight along with the TRIZ methodology a series of design contradictions were defined. These contradictions were then compared to the TRIZ reference table using SWRL rules generating design recommendations (Figure 8). The contradiction created was based on the cordless screwdriver functional model and the weight design parameter. The contradiction (Figure 8) has an improving parameter “Adaptability” and a worsening parameter “Weight”. This contradiction relates to the ability of a cordless screwdriver to drive screws at different speeds and torque levels, which typically requires an electric motor capable of high levels of torque at different speeds. This leads to larger, heavier and less efficient motors. The design suggestions (Figure 9) associated with the dynamics principle of invention seem to have good potential for addressing the problem. The dynamics design principle in this problem would likely mean the introduction of a gear box which would allow the user to switch operating range without the need of an electric motor capable of delivering adequate torque over a very wide speed range.

Another facet of the developed framework is the ability to trace knowledge. As an example the maximum weight requirement defined within the project ontology is linked from the design instance as well as from the instance of design parameter (Figure A5). This is a useful feature in the evaluation of the potential impact of specification changes by simply searching for design information dependent on the affected specification.

The process of developing a functional model did not provide the designer ready-to-use design contradictions. The TIPS terminology does not have good correlation with the Functional basis taxonomy nor does it correspond to a cohesive taxonomy, rather the terms used have a more descriptive role. This could be seen as a failure, yet the functional models and design ontology provided a detailed description and a clear conceptualization of the problem at hand. If the TIPS terminology exactly matched the Functional description, a computer program could process the functional model and the generated design recommendations as a result of a deterministic process may lack the creativity needed for innovation. One of the major benefits of creating design contradictions to generate design suggestions is the introduction of new concepts into the design process, as such the manual generation of design contradictions might be the most effective method to achieve this.

Figure 7. Power supply functional model data entry form.

Figure 8. Input form in Protégé for the Design contradiction class showing the design recommendation resulting from the application of TIPS.
SUMMARY AND FUTURE WORK

As a proof of concept exercise the results from this project are encouraging. The approach of mixing large domain knowledge ontologies with small focused process ontologies proved viable. Large domain ontologies were developed and deployed without specific customizations. Using custom process ontologies we were able to extend the domain ontologies to fit the purpose of this project. Much work still needs to be done for this embryonic concept to develop into a useful design methodology. Functional models and TIPS have provided a foundation for further work, but increasing the knowledge architecture to include richer models and additional problems solving methodologies will lead to more power innovation tools. The framework developed for this project needs to be expanded so that design problems can be analyzed using more than one innovation methodology and to include design behavior analysis. Potential candidates for expanding innovation or problem solving methodologies beyond TIPS are Transformation Principles [18, 19], and semantic patent representations [20], both provide methods for generating design ideas and are based on well define taxonomies. Behavior models play an important role in the design process and as such this area will need to be included in future development. Increasing the number of domain knowledge ontologies, such a materials and process ontologies, is another area where growth is needed. A combination of expanded domain ontologies along a wider variety of innovation methodologies would provide the framework needed to tackle larger and more interesting design challenges.

The greatest gain for a design engineer resulting from this semantic framework is the ability to rapidly manipulate and analyze the stored knowledge using a variety of different methodologies. Like the example of the chess player presented at the beginning of the paper, the ability to quickly evaluate and analyze different scenarios can provide a substantial advantage.

ACKNOWLEDGEMENTS

This material is based on work supported by the National Science Foundation under Grant No. 0332508 and by industry members of the NSF Center for e-Design.

REFERENCES


ANNEX A

FIGURES

Figure A1. Detail view of the Functional Model classes in the Functional Basis ontology

Figure A2. Class structure of the People ontology
Figure A3. Class structure of the Project ontology. Note: The arrows are linking parameters between classes; instances are in red; and class names with an asterisk are allowed multiple values.

Figure A4. Diagram of the class instances defining the store electricity operation.
Figure A5. The Maximum weight requirement is referenced from both the cordless screwdriver instance of the Design class and from Mass instance of the Design parameter class. This allows the information to be referenced where it is needed without redundancies. Note: The arrows are linking parameters between classes; instances are in red; and class names with an asterisk are allowed multiple values.