Artificial Intelligence for Engineering Design, Analysis and Manufacturing (2007), *21*, 137–154. Printed in the USA.
Copyright © 2007 Cambridge University Press 0890-0604/07 $25.00
DOI: 10.1017/S0890060407070199

Ontology-based design information extraction and retrieval

ZHANJUN LI AND KARTHIK RAMANI

Purdue Research and Education Center for Information Systems in Engineering, School of Mechanical Engineering, Purdue University, West Lafayette, Indiana, USA

(RECEIVED October 25, 2005; ACCEPTED July 25, 2006)

Abstract

Because of the increasing complexity of products and the design process, as well as the popularity of computer-aided documentation tools, the number of electronic and textual design documents being generated has exploded. The availability of such extensive document resources has created new challenges and opportunities for research. These include improving design information retrieval to achieve a more coherent environment for design exploration, learning, and reuse. One critical issue is related to the construction of a structured representation for indexing design documents that record engineers' ideas and reasoning processes for a specific design. This representation should explicitly and accurately capture the important design concepts as well as the relationships between these concepts so that engineers can locate their documents of interest with less effort. For design information retrieval, we propose to use shallow natural language processing and domain-specific design ontology to automatically construct a structured and semantics-based representation from unstructured design documents. The design concepts and relationships of the representation are recognized from the document based on the identified linguistic patterns. The recognized concepts and relationships are joined to form a concept graph. The integration of these concept graphs builds an application-specific design ontology, which can be seen as the structured representation of the content of the corporate document repository, as well as an automatically populated knowledge base from previous designs. To improve the performance of design information retrieval, we have developed ontology-based query processing, where users' requests are interpreted based on their domain-specific meanings. Our approach contrasts with the traditionally used keyword-based search. An experiment to test the retrieval performance is conducted by using the design documents from a product design scenario. The results demonstrate that our method outperforms the keyword-based search techniques. This research contributes to the development and use of engineering ontology for design information retrieval.

Keywords: Design Information Extraction; Design Information Retrieval; Design Semantics; Ontology; Reuse

1. INTRODUCTION

Rational mechanical design is the process of materializing the product life cycle information, that is, design information, into a physical prototype. In current practice, textual descriptions and three-dimensional computer-aided design (CAD) models are two of the most prevalent data resources in which the design information is described. Our primary focus is the design information in textual and electronic documents. Design information refers to the product or component to be designed and the design specifications, such as functions, performances, material selections, manufacturing process, environments, and so forth.

Documentation, as a linguistic form, provides the explicit representation of design ideas and concepts accumulated during the design process (Dong, 2006), and therefore, can be reused by other engineers in new designs. Other important roles of documentation include legal issues, patent applications, international standard certifications, and internal practices. Contrarily, because of the increasing complexity of products and the design process, as well as the popularity of computer-aided documentation tools, a large amount of design documents are generated. For example, as reported

Reprint requests to: Karthik Ramani, School of Mechanical Engineering, Purdue University, 585 Purdue Mall, West Lafayette, IN 47907-2040, USA. E-mail: ramani@purdue.edu

https://doi.org/10.1017/S0890060407070199 Published online by Cambridge University Press

by Marsh (1997), there are approximately 40,000 documents produced in the design of a single engine in an aerospace company. Among them, 63% are textual descriptions. The availability of such extensive information resources creates new challenges as well as opportunities for research on how to effectively and efficiently index and retrieve the design documents. This is in contrast to manually indexing the documents using the metadata of the documents, as done by most product data management/product life cycle management (PDM/PLM) systems and general document management tools (Weber et al., 2003). The significance of improving the design information management in industry sectors was also reported in a survey of over 300 companies in two countries (Court et al., 1998). It was found that design engineers spent 20–30% of their time retrieving and communicating design information. Therefore, it is important to minimize such overhead by using effective computer-aided tools.

Design information resources can be classified into external resources and internal resources. Examples of external resources are online patents, journals and magazines, and online catalogs. Internal resources, varying from product specification proposals to final design reports, from drawing notes to engineers' log books, are important constituents of a complete record of a product's evolution. Engineers are dependent on retrieving and using these documents throughout the design process, especially the internal design documents (Court et al., 1994).

Design information retrieval acts as "memory extension" for individual engineers and enables information sharing among them (Ullman, 1997). In these two scenarios, it is assumed that engineers are familiar with the content documented either by themselves or others. Therefore, the retrieved results are expected to match exactly what the users requested. We also perceive the importance of design information retrieval from three further aspects, where queries should be treated in a broader and more abstract manner: design exploration, learning, and reuse. Ahmed and Wallace (2004) discovered that novice design engineers were not always aware of what they needed to know during the design process. By looking through previous design documents, the novice engineers gained experience more rapidly by using the design information retrieval to obtain not only the information they were looking for, but also different design issues related to their queries. The importance of design exploration during the early design stage can best be expressed by the adage, "If you generate one idea, it will probably be a poor idea; if you generate twenty ideas, you may have one good idea." In reality, however, there is a great tendency for engineers to take their first idea and start to refine it toward a final design (Ullman, 1997). Design information retrieval should assist engineers to find the "exact same" with respect to their design concepts as well as ones that are "similar" in the measurement of domain-specific perceptions, such as designs that have similar functions or use the same materials. Semantic ambiguity can often be responsible for the production of novel solutions, whereas replication of the same "meanings" may lead to replication of older solutions. However, it is also true that the ability to retrieve and compare design concepts and their relationships within several parallel contexts can be useful in understanding the implications of certain decisions. Design information retrieval provides engineers with the capability to expand their problems into many candidate solutions before being narrowed to one final decision. It also offers unique insight into past design scenarios and experiences, which are known as design reuse, and have been receiving increased attention from both the research and industry sectors (Sivaloganathan, 1998).

Despite the benefits, design information retrieval has only met limited success in practice (Busby, 1999). Engineers have always found it hard to locate previous designs for their needs. An empirical study conducted by Marsh (1997) showed that design engineers were reluctant to access documents to which they did not directly contribute. The author was the most likely person to occasionally access documents. The reasons for this are that there is no mechanism for users to be aware of the content, or *semantics* (i.e., concepts and relationships between the concepts), of the design and therefore no mechanism to retrieve it.

Studies by Silberman et al. (2001) in the field of psychology suggested that showing semantic relations encourages people to generate more semantic relations. Guarino et al. (1999) noted that the performance of information retrieval can be improved by using structured and semantics-based representations of documents. In the engineering domain, the structured and semantics-based representation of designs has been studied in fields such as case-based design (CBD), product modeling, and engineering ontology. CBD represents specific designs by using predefined design features such as functions, behaviors, and attribute–value pairs. Although there have been many successful applications (e.g., Sycara & Navinchandra, 1989; Goel et al., 1997), a number of issues still remain unresolved, such as the lack of granularity of the index structure and the subjective definition of the case vocabulary. Also, the case matching based on word pattern matching does not consider enough semantic relations among terms used as indices (Maher & Silva-Garza, 1997). In product modeling (e.g., Szykman et al., 2000; Sudarsan et al., 2005), comprehensive design and development information are recorded by engineers by complying with formalized templates and rules. Engineering ontology (e.g., Olsen et al., 1995; Kim et al., 2003; Kitamura & Mizoguchi 2004) systemizes semantic relations between elements of specific design applications as well as represents the functions and behaviors of design decompositions. Research in product modeling and engineering ontology has made significant progress in establishing complex models as well as standardizing terminologies to describe the details of the design. In many cases, however, establishing the knowledge-sharing agreements or mapping out the design decomposition is potentially more expensive than

https://doi.org/10.1017/S0890060407070199 Published online by Cambridge University Press

the design itself. For example, as pointed out by Stauffer et al. (1987), CAD tools that force engineers to fully develop the functionality as well as the logic diagram of a product during the initial effort produce a reluctance on the part of engineers to use the design tool. Therefore, in our opinion, it is equally important to develop a strategy that is comprehensive and can intelligently extract valuable content from the free-text design document and in which engineers can use natural language to record the design and the design process.

To extract and structure design information from unstructured design documents, one first needs to agree on the kinds of design information that should be identified and represented. This agreement should be based upon an understanding of the information needs of engineers, as well as the characteristics of the design document.

Cognitive studies (e.g., Ahmed & Wallace, 2003; Hmelo-Silver & Pfeffer, 2004) showed that engineers process information at a qualitative level, that is, textual descriptions, rather than at a quantitative level, such as geometry models and parametric data, especially during the early design stages. Baya et al. (1992) developed a framework for understanding the constituents of the design information used by engineers during the design process. These constituents are categorized into "subjects" and "descriptors." Subjects refers to the name of the physical designs at different levels, such as "assembly" and "component." Descriptors refers to the characteristics (mostly qualitative descriptions) of the information that is sought about the subjects, such as "operations" and "locations." An earlier experiment by Kuffner and Ullman (1991) showed that engineers are interested in design information other than that which is contained in drawings and specifications, such as "purpose," "construction," and "operation." Pugh (1997) investigated 32 upper level design issues that should be considered during the design process. A more recent study by Lowe et al. (2000) proposed a decoupled design information taxonomy at the conceptual level. The taxonomy characterizes the technical content of industrial design documents. The subtaxonomies are either domain specific, such as "technical domain" and "contextual attributes," or company specific, such as "product descriptors" and "resource issue descriptors."

All these identified design information descriptions can be abstracted as design concepts and relationships, that is, *design semantics*. Examples of the design concept are phrases such as names of products and components: "reheat burner manifold" and "gear," and their functions: "channel gas" and "rotate." There is a functional relationship between the component concept and its function concept, such as gear having the function of rotate. In contrast, the sentences that describe the different aspects of a design often present a limited set of linguistic patterns as well as a limited scope of lexical terms (Lowe et al., 2000; Ahmed et al., 2005). These terms and patterns can be extracted from the noun phrases, verb phrases, prepositional phrases, and so forth, within one sentence or across sentences. However, base taxonomies are needed to guide such an extraction process.

We describe a framework designed to automatically construct a structured representation model from an unstructured design document to achieve a more effective way of design information retrieval. The purpose is to transform free text descriptions into the semantics-based information units that are lacking in today's engineering information systems. The method uses simplified natural language processing (NLP) techniques to extract the design semantics based on the linguistic patterns identified in the document. A domain ontology is built to represent the domain-specific concepts needed to guide the extraction process. The extracted design concepts and relationships populate a semantics representation of the document, which is structured as well as explicit. We focus on combining the ontology model and NLP technique to extract the design semantics from unstructured, formal, and textual design documents, as well as the algorithm of retrieving the existing designs based on the concepts and relationships in the ontology model.

2. RELATED WORK

The analysis of general unstructured documents has been studied mainly from the following perspectives: in-depth NLP, information retrieval (IR), and information extraction (IE; Glasgow et al., 1998).

The in-depth NLP fully analyzes the input document on both the syntactic and semantic levels. It is usually very complex and expensive. We chose not to take this approach because it does not satisfy the need of our application to focus on the limited domain-specific aspects of the text instead of the general linguistic contexts.

In the classic IR approaches, such as the vector model (Salton, 1989), documents are represented through a list of representative keywords, that is, a vector, defined by the word–document occurrence. Queries are also represented as a vector. The similarity between the document and query is determined by the cosine similarity between two vectors. However, the extracted keywords are unable to represent the desired semantics that many documents contain. For example, the following are three queries through which users want to investigate the designs that

1. lock a car plate with a curvilinear slot sliding along a cylindrical pin in the assembly,
2. have a spur gear assembled with a step shaft using a woodruff key, or
3. have a DC motor with output speed of 100–1000 rpm.

The first query is for a specific design intent or function of a mechanism and its significant components. The second is for the detailed assembly structures. In the last example, the main concern is the exact value of a performance attrib-

https://doi.org/10.1017/S0890060407070199 Published online by Cambridge University Press

uted to a specific component. It is impossible to extract these semantic descriptions accurately by using a few keywords. Keywords are syntactic units that are unable to reflect meanings as well as relationships. Brin and Page (1998) pointed out that approaches using the vector model work well only with small and homogeneous collections such as literature or news releases under a common topic. However, design documents usually describe complex as well as heterogeneous design process and specifications.

IE approaches bring together NLP tools (e.g., syntax analysis) with domain knowledge to extract the concepts in the texts and form a framework based on predefined templates. The domain knowledge can be either formalized as expression patterns by an expert (Hobbs et al., 1996), or learned from a large training corpus (Riloff, 1996). The IE approaches are the closest to our approach regarding the fundamental NLP techniques and the principle of using domain knowledge to assist the extraction process. However, there are two major differences. First, most of the research in IE is on short texts with very restricted topics, such as news of terrorism reports, whereas design documents are more diversified. Therefore, it is necessary to have a more complex and more systematic knowledge representation model to assist the extraction process. Second, in our method, the extracted representation model is also used to index the documents and support the retrieval.

In engineering design, there has been research that aims to analyze unstructured engineering documents for different applications. We classify them into grammar-based parsing, augmented vector model, and classification-based approaches.

Work has been done in parsing drawing notes into a frame-based knowledge representation format by using augmented transition network grammars to extract assembly information for automatic assembly (Tyhurst, 1986). The grammar and vocabulary used in the drawing notes are company specific. Farley (2000) extracted the pieces of equipment and the repair action on them from aircraft maintenance log books. The extracted results are used for case-based retrieval whenever quick maintenance activities are required. The method generates a parsing-tree structure of each record (i.e., a sentence) before the extraction process. This approach relies heavily on domain knowledge models tailored from corporate resources.

Dong and Agogino (1996) proposed an augmented vector model, that is, belief networks, to represent the unstructured design documents. It was built by adding less statistically significant but "causal" correlated terms after the significant terms were identified using vector model-based techniques. The causal relations were constructed by using k-mean clustering and some heuristics. The experiment showed query performance improvement on recall because of the concept expansion based on the augmented representation model. The method is able to find the discriminating terms of each document in the test data, for example, scanned texts of mechanical engineers' handbooks. However, there is still an open question about its performance in processing the majority of design documents.

Assuming a classification structure, either a flat list or a hierarchy of terms for engineering design can achieve better retrieval performance as well as assist the retrieval process. There are some researchers who have tried to build thesauri or taxonomies to classify the documents. Baudin et al. (1992) described the Dedal project. It indexes the electronic design notebooks by using manually generated thesauri, that is, collections of related terms. Building on this work, Yang et al. (2005) attempted to automate the population of the thesaurus by using the vector model-based technique and singular value decomposition. Ahmed et al. (2005) used domain taxonomies to index the documents. The taxonomies were formed from the existing literature as well as interviews with engineers. They also used the vector model-based technique to classify the documents against the terms in the taxonomies. However, the accuracy of the indexing is in question. In the Waypoint system, McMahon et al. (2004) employed user-predefined classification schema and taxonomies to automatically classify the heterogeneous technical documents besides the full-text search. The taxonomy included terms representing the content of the document and metadata of the document. Examples of the terms are product names and task decompositions; examples of the metadata are document formats and document types. The documents were parsed into an inverted file. The associations between the documents and the taxonomy terms were achieved by "constraint-based classification," that is, phrase matching.

In summary, most of the current research does not provide the semantics-based representation structure of the document that satisfies design engineers' needs to retrieve and explore previous designs. In contrast, grammar-based parsing is able to represent the limited set of engineering semantics from very specific engineering documents. However, the deep NLP techniques are not robust or applied to the texts with restricted vocabulary and grammars. In addition, these approaches work with only limited scope of documents. The belief network representation can capture "contextual" information in the document. However, it is not able to identify relations between extracted terms explicitly and accurately. Explicitness is the prerequisite for human consumption, whereas accuracy is critical for retrieval performance. Classification-based approaches take advantage of the domain thesauri or taxonomies to capture concepts from the document, and hence, are capable of improving the precision and recall. With a visible indexing structure, they provide a more intuitive retrieval environment. However, attempts have not been made to capture meaningful relations between concepts and, therefore, the representations are incomplete. Machine-generated taxonomies can alleviate the laborious process of manually building and maintaining taxonomies. However, using automatic taxonomy construction for retrieval often produces unsatisfac-

https://doi.org/10.1017/S0890060407070199 Published online by Cambridge University Press

tory results and is still an open research area (Chung et al., 2002).

Therefore, a more feasible representation for design documents should extract the meanings along with the words from the documents. Our approach focuses on extracting a structured representation from the design document automatically. It represents the document explicitly and succinctly by extracting only the relevant design semantics from the document. The extraction process is assisted by a domain-specific design ontology model. Our approach provides a framework to guide the decoupling and modeling process of the knowledge base from the design documents. In synergy with the domain ontology model, the automatically generated representation structure also acts as the indexing model for the design information retrieval, and therefore provides an environment for design exploration, learning, and reuse.

3. THE PROPOSED APPROACH

3.1. Overview

Given that the input design documents are in correct English, we seek to obtain accurate and concise descriptions of the design semantics automatically from the unstructured and formal design documents, such as technical reports, proposals, and drawing notes. The descriptions are both qualitative and quantitative. Qualitative descriptions characterize the different aspects of the design in an abstract manner. They allow design engineers to design and modify before delving into the details. In contrast, quantitative descriptions provide the details for tasks that qualitative descriptions lack. The complete descriptions allow engineers to understand, evaluate, and reuse previous designs under various conditions.

Automatically extracting design semantics from the document requires recognizing the syntactic structure as well as the semantic meanings of the text. Both linguistic knowledge and domain knowledge are needed to fulfill the inference. This is a nontrivial task even for a simple document with several sentences. In-depth analyzing and fully understanding the syntax as well as semantics is, in practice, very difficult because of the complexities of natural language (Glasgow et al., 1998). However, because we are only interested in the design semantics, which are the specific aspects of the document content, the corresponding linguistic patterns found in the documents are limited. Furthermore, we tackle the domain-specific natural language problem with a general framework, where the methodology as well as the primary modules can be applied to different domains.

To accurately represent the design semantics in a document, we need to extract as much relevant information from the document as possible. Because a concept may be described in more than one sentence and by different words/phrases, it is necessary to analyze the text across sentence boundaries, that is, it must perform reference resolution so that much more information about the concept can be recognized.

Figure 1 shows the architecture of the prototype for design information extraction and retrieval: ontology-based design document analysis and retrieval tool (ODART).

The texts of the input document are processed in two stages. First, at syntax analysis, shallow NLP techniques are used for analyzing the texts in order to form phrases. This refers to tokenizing, part of speech (POS) tagging and disambiguating, and phrase chunking. The linguistic knowledge is represented as the lexicon and syntax rules, which

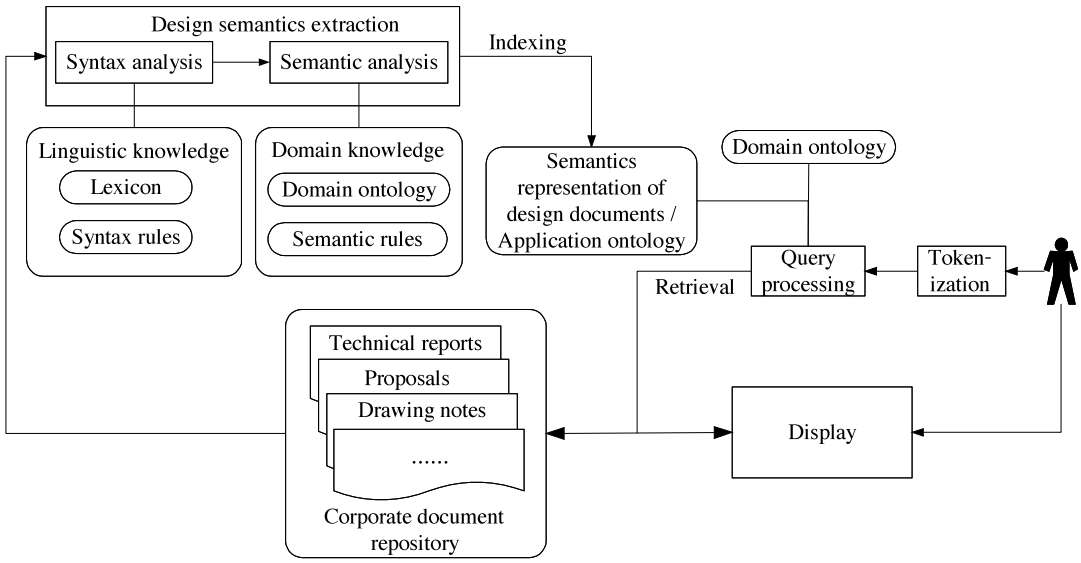


Fig. 1. The system architecture and functional diagram.

https://doi.org/10.1017/S0890060407070199 Published online by Cambridge University Press

are used to assist the syntax analysis. Second, in the semantics analysis, the formed phrases are then matched with the concepts in the domain-specific design ontology, that is, *domain ontology*, and therefore generate concept instances. The relationships between the concept instances are identified based on the position of the phrases in the syntactic structure of the sentence as well as the taxonomy category (in the domain ontology) that the matched concepts belong to. The domain ontology systematizes the fundamental aspects of the design semantics by defining taxonomies organized in a hierarchical structure. These taxonomies define concepts, or "seed words," for recognizing the instance of these concepts, that is, phrases, in the document. The domain ontology is not only used to assist the semantics analysis, but provides a concept space for the mapping from the query keywords to the concepts during query processing. Semantic rules are employed to generate relationships. Semantics analysis includes concept recognizing, reference resolution, and joining.

The concept instances and relationships are joined to form a concept graph (Sowa, 1984). The integration of these concept graphs builds the application-specific design ontology, that is, *application ontology*. It is the semantics-based and structured representation of the design documents as well as the concept–relationship index model of the document. We call the whole process of recognizing and populating the design semantics from the document, *design semantics extraction*, which means using computer-operatable natural language phrases to describe the content of interest in the design document. With this procedure, we reduce the amount of in-depth linguistic analysis and human intervention required for building a large knowledge base from domain-specific documents. The domain ontology is organized in a modular manner, thus reducing the complexity involved in the ontology maintenance and improving the portability as well. An ontology-based retrieval algorithm is developed to support the design information retrieval.

The detailed descriptions of the proposed approach are organized as follows. The ontology representation and acquisition are discussed first. Then we describe the design semantics extraction process, which is developed based on a shallow NLP algorithm and the domain ontology. The application ontology is automatically acquired after the extraction process. We also discuss the retrieval algorithm and the scoring metrics that use the content and structure of the ontology model. In the last subsection, we introduce the empirical studies for the design information retrieval.

3.2. Ontology modeling and acquisition

An ontology is a knowledge representation model to explicitly represent a domain by defining the concepts and various relationships among the concepts (Nirenburg & Raskin, 2004). The ontology model integrates different inference mechanisms within one structure through relationships, to systematize the design information and make it operable by both humans and computers. In our research, the ontology acts as the data structure and the model to systematize the representation of the design semantics. We use a layered design ontology model to assist the semantic extraction from the design document as well as to represent the extracted semantics. Figure 2 shows the ontology representation. The concepts in the application layer are described in terms of the concepts in the domain layer. The application ontology is company specific or even product specific, and therefore is generated from design documents automatically to ensure the validity and dynamics of its contents. The contents of the domain ontology are more generalized through different products and even companies as long as they are in the same engineering domain. For example, all of the mechanical designs deal with the performance and functionalities of the product.

To build the domain ontology, the first step is to identify its scope or themes (Fernández-López et al., 1997). Based on the categories of the important design issues documented by engineers and identified by the aforementioned cognitive studies, we decided to extract the concepts such as the products or components being designed and their functions, performances, material selections, manufacturing processes, and environments (i.e., the environmental objects that the design interacts with). Next, taxonomies under these themes need to be constructed so that the algorithm can recognize a similar phrase in the document according to the concept, that is, the "seed word," defined in the taxonomy.

The domain ontology includes a set of subontologies, that is taxonomies, such as device taxonomy, performance taxonomy, function taxonomy, material taxonomy, manu-

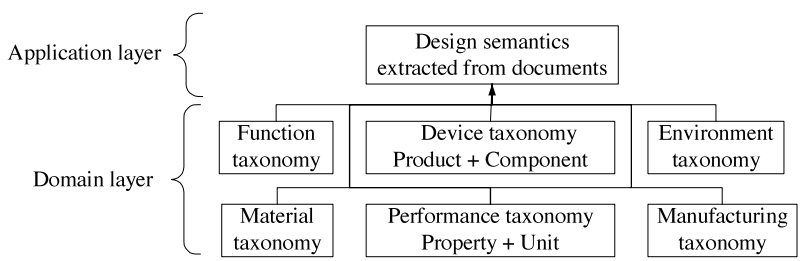


Fig. 2. The layered design ontology model.

https://doi.org/10.1017/S0890060407070199 Published online by Cambridge University Press

facturing process taxonomy, and environment taxonomy. Among them, device taxonomy can be further divided into product taxonomy and component taxonomy, whereas performance taxonomy includes property taxonomy and unit taxonomy as well. Each taxonomy is a hierarchical organization of concepts in an independent subdomain of the design semantics. These concepts are used to assist the semantics extraction from the design documents by matching with the tagged phrases or the "head" word of the phrases. For example, "wheel" is a concept defined in the device taxonomy. It can match with the noun phrases "a wheel" or "the front wheel." In general, each concept contains a list of *lexical terms* such as its abbreviations, acronyms, morphological inflections, synonyms, and itself. The lexical terms are used to calculate the score of the concept against the phrase described in the next section. Recall that these taxonomies also construct a concept space for understanding the user's requests during query processing.

One of the difficulties in building a domain ontology is that it contains both generic concepts and concepts that are specific to a company or a product. This means that the methods and the resources for acquiring different taxonomies need to be studied differently. For example, in general, the naming conventions for device concepts are company specific: the product taxonomy specifies the names of the products a company manufactures, for example, Bosch 13618-2G and F-150, and the upper level concepts that classify the products, for example, power screw driver and truck. The component taxonomy classifies the mechanical and electrical elements that are designed or used by the company, such as gears, suspensions, resistors, and motors. An example of the device taxonomy created from our test document repository is shown in Figure 3. It has upper level concepts, such as a vehicle and mechanical component, and lower level concepts as well. Note that the lexical terms of the concept are not presented in the figure. The device taxonomy should be generated by specific company users through an editing tool, which is part of this system.

The property taxonomy describes the fundamental concepts about physical and geometry properties, such as weight and length, whereas the unit taxonomy classifies the well-established vocabulary of measurement units, such as kilogram and meter. We chose WordNet-2.1 (http://wordnet.princeton.edu/obtain) and customized its categories of "physical property" and "magnitude" to build the property taxonomy. WordNet is a large linguistic resource. It has been adapted for use by many for their ontologies, for example, suggested upper merged ontology (http://www.ontologyportal.org/). It is also applied in the engineering domain for various purposes. For example, it is used by de Vries et al. (2005) as a lexicon to interpret words that engineers use during conceptual design and to seek semantic associations among these words to enhance the design behavior. Guarino et al. (1999) used WordNet to manually generate "lexical conceptual graph" representation of domain-specific documents such as product descriptions for content-based retrieval. The unit taxonomy was acquired from a dictionary of measurement units developed at Exeter University (http://www.ex.ac.uk/cimt/dictunit/dictunit.htm#length). Currently, there are 449 concepts in the performance taxonomy.

The function taxonomy specifies the functional concepts, that is, verbs, which describe the functional purpose or the composing state of the device concept, that is, "sub-

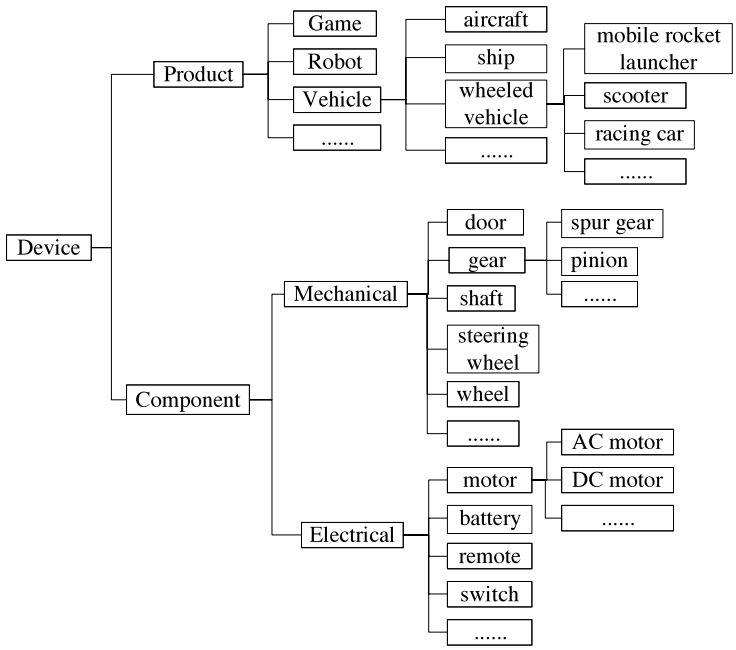


Fig. 3. The device taxonomy.

https://doi.org/10.1017/S0890060407070199 Published online by Cambridge University Press

ject + verb [+ objects]," and the hierarchy of these functional concepts. The function taxonomy includes functional verbs such as "control" and "align," and composing verbs (Levin, 1993) such as "include" and "have." The functional verbs are obtained from the two existing engineering classifications for verbs: function basis (Hirtz et al., 2002) and Collins's mechanical function glossary (Collins et al., 1976). Function basis is a categorization of functions (verbs) and flows (nouns), aiming to capture the function information for a broad variety of engineering designs. Ahmed and Wallace (2003) evaluated the function basis and reported that it covers 94% of verbs used by engineers in describing their designs. Our function taxonomy has a total of 246 verbs.

Material taxonomy and manufacturing process taxonomy classify the concepts of the engineering materials and the manufacturing operations performed in the manufacture of products or components. The concepts were acquired by studying online catalogs (e.g., http://www.matweb.com/search/SearchSubcat.asp), engineering texts (e.g., Ashby, 1999), and engineering handbooks (e.g., Kutz, 2002, 2005). There are 201 concepts in the material taxonomy and 318 in the manufacturing taxonomy.

The environment taxonomy describes objects that may interact with a product or a component of a product, such as human and gas. This taxonomy is based on the flow taxonomy from the function basis and the environment taxonomy by Pugh (1997). It consists of 108 concepts. Figure 4 shows part of the environment taxonomy.

One of the focuses of this paper is the modeling and the automatic generation of the application ontology. Figure 5 shows an example of the extracted design semantics from an input document, that is, the application-specific semantics representation of a toy design project report from an engineering design class. The top part of the figure shows the definition of the application ontology as a hierarchy of classes or design concept instances, such as device, product, component, function, and property. A concept instance is populated whenever there is a phrase recognized and extracted from the design document. It also defines relationships as slots between these concept instances. For example, the has_function is a slot of a device concept instance. It is a relationship between a device concept instance, for example, "DC motor," with a function concept instance, such as "driven." At the bottom, portions of the project report are analyzed to populate the semantics representation, that is, concept instances and relationships, of the document, such as those shown in the middle of the figure. These concept instances of the application ontology are organized as a directed graph. Each subgraph reflects the semantics of one design document. All these subgraphs are constructed under the system root node, called "design entity," to form the application ontology. It directly answers questions about the design and sets the stage for further analysis. Each node of the subgraph represents a concept instance, including product concept instance (PCI), component concept instance (CCI), function concept instance (FCI), property concept instance (PRCI), material concept instance (MCI), manufacturing concept instance (MACI), and environment concept instance (ECI). The arc between nodes represents a relationship. Eight types of relationships are defined in the application ontology: is_a, has_part, has_function, has_object, has_property, has_value, has_material, and has_manufacturing. The details of the definitions can be found in Appendix A.

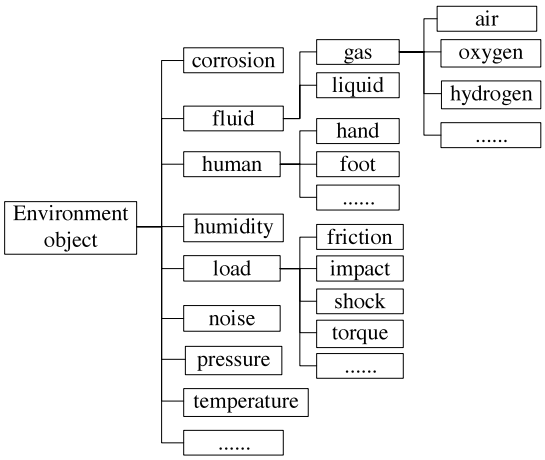


Fig. 4. The environment object taxonomy.

3.3. Design semantics extraction

Our method makes use of syntax analysis, semantics analysis, and the domain ontology to identify the concepts and relationships contained in the documents. It then helps in engineering the application-specific ontology. Figure 6 shows the modules and procedures of the semantics extraction from the design documents.

The lexicon is a list of words with their POS tags (tag_1, tag_2, . . . , tag_n), where tag_1 is the most likely tag for the word, and tag_2 . . . tag_n are other tags of the word not in any particular order. It was originally derived from the Penn Treebank tagging of the Wall Street Journal and the Brown Corpus, with a total of 93,698 words (Brill, 1995). Given the lexicon, syntax rules, and semantic rules, the document analysis algorithm extracts the device, function, performance, and other design semantics from the textual document. The algorithm works as shown in the following subsections.

3.3.1. Tokenizing

The input character stream is parsed into tokens/words and punctuation marks. Sentences are then formed.

3.3.2. POS tagging and disambiguating

Each word is first tagged with its most likely POS as defined in the lexicon. If the word does not have a match in

https://doi.org/10.1017/S0890060407070199 Published online by Cambridge University Press

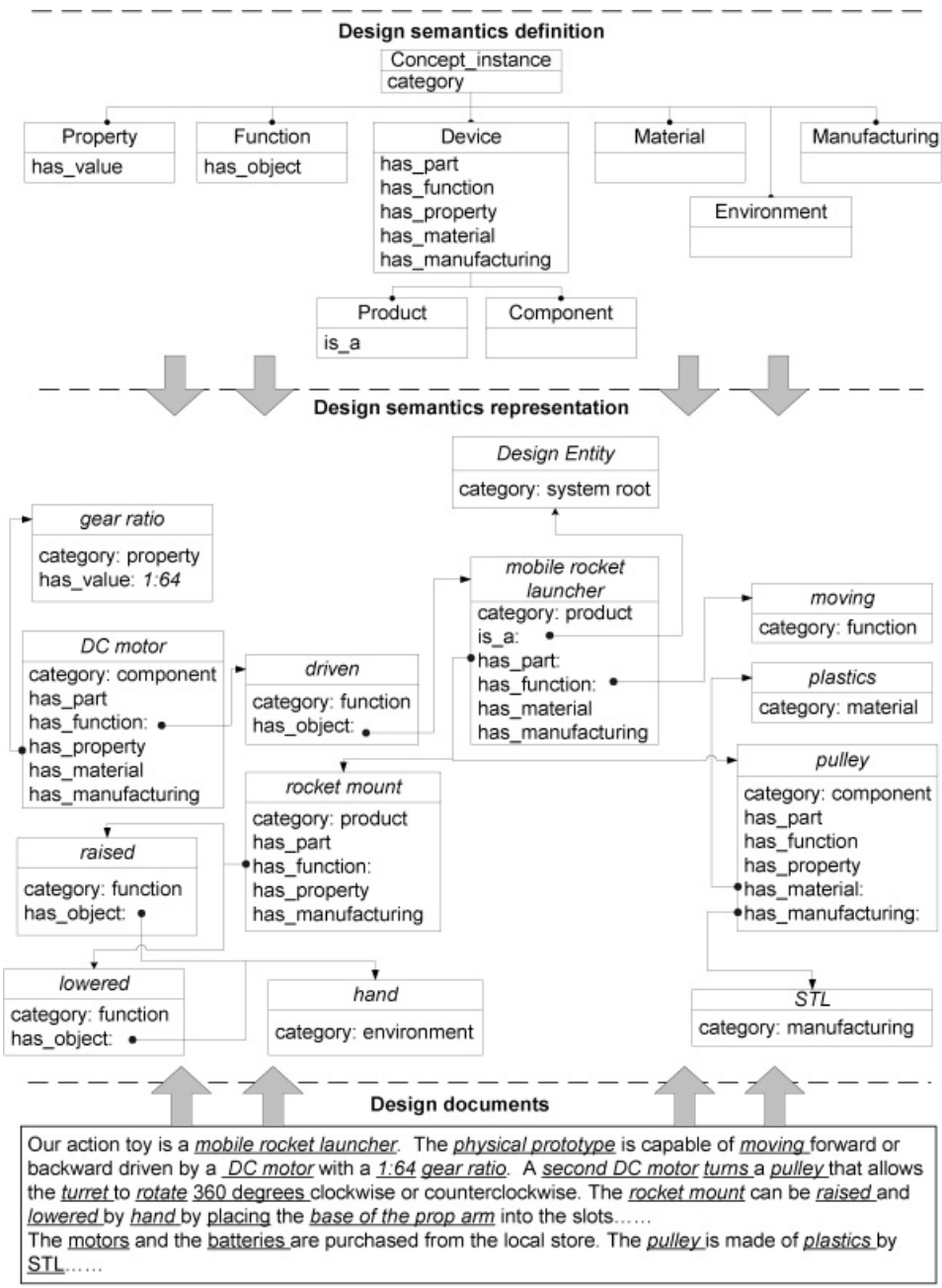


Fig. 5. An overview of the design semantics representation and application ontology. Note that some design concepts are omitted for the sake of clarity.

the lexicon, then the word is assigned an "unknown" tag. To correct the possible mistags, some heuristics for POS disambiguation apply. These heuristics rely on the adjacent words of the word for correction. For example, the heuristic that corrects the tagging of infinitive (TO) from preposition (IN) is "if (IN) to + verb then (TO) to + verb." The "unknown" tags are used for a repeated evaluation of the lexicon, because the corresponding words can be easily pulled out, analyzed, and added into the lexicon for the next tagging. The naming conventions of the tags can be found in the Penn Treebank tag set (Marcus et al., 1994).

3.3.3. Phrase chunking

Group words into noun phrases (NPs), verb phrases (VPs), prepositional phrases, and measurement value phrases based on context-free grammars (Jurafsky & Martin, 2000). For example, one rule is DT ^ NN → NP. This means an NP comprises a determiner followed by a noun. Each VP is

https://doi.org/10.1017/S0890060407070199 Published online by Cambridge University Press

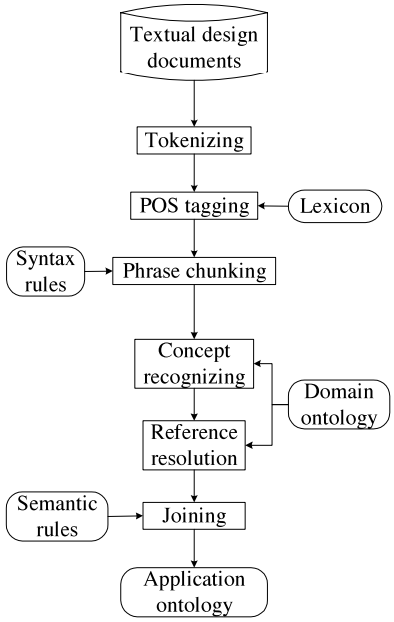


Fig. 6. The modules and procedure of design semantics extraction.

labeled as either active or passive. The prepositional phrase is the preposition (IN) itself.

3.3.4. *Concept recognizing*

The purpose of concept recognizing is to select the most appropriate concept in the domain ontology that matches with a NP or VP. By selecting the concept, a concept instance is generated, with its name being assigned as the string of the phrase. The concept instance is referenced by the corresponding concept in the domain ontology. At the beginning of each matching process, stop words such as auxiliary verbs and articles are removed from the phrases. Breadth first search (BFS) is used to search for concepts in the domain ontology. To achieve the maximum extraction recall, the NPs are compared against all the concepts in the domain ontology, whereas the VPs are matched against the function taxonomy as well as the manufacturing taxonomy. For instance, welding and molding are manufacturing concepts rather than function concepts.

A phrase may match with multiple concepts if they share at least one common word of the phrase. For example, assuming that "a second DC motor" is the phrase to be recognized ("a" will be removed as a stop word), it matches with "motor" concept and its subconcepts, such as "DC motor" and "AC motor" in the device taxonomy (Fig. 3). A concept selection method involving some heuristics has been developed to choose the most appropriate concept by comparing the scores of the matched concepts. Recall that each concept is complemented by a set of lexical terms (T_1, T_2, T_3, . . . , T_i, . . . , T_n), including the concept itself. Each term may consist of multiple words. To calculate the concept score (*Cscore*) for concept C_i, we first calculate the term score (*Tscore*) of all its lexical terms by Eq. (1). Then the *Cscore* is the maximum of all its *Tscores* in Eq. (2). The *Tscore* is calculated based on the number and the position/weight of the matches between the words of the lexical term and the words of the phrase.

$$Tscore_{ij} = \frac{(\#\text{ of words in the phrase } T_j \text{ matches with}) * \sum_{m=1}^{N} \omega_m}{\#\text{ of words in the phrase}}, \tag{1}$$

$$Cscore_i = \max(Tscore_{ij}) \quad 1 \le j \le n. \tag{2}$$

Let us assume a phrase contains N words and m is the index/position of the words in the phrase from right to left, which is also the order of the word-by-word comparison between each term and the phrase, and w_m is 1 if the matched term and phrase contain only one word. Otherwise, to emphasize the matching between head words more than modifiers, w_1 is 0.55 if the rightmost words match and the rest of the matches split the weight of 0.45 equally. Note that w_m is 0 when there are no matches at the corresponding position. The use of the denominator in equation 1 is to normalize the *Tscore* to be in the range of 0 to 1.

For the same example in Figure 3, assuming that the three concepts themselves have the largest *Tscore*, that is, the final *Cscore*, among their lexical terms, the *Cscores* for motor, DC motor, and AC motor will be 0.183, 0.258, and 0.183, respectively. Therefore, the phrase will be indexed as an instance of the DC motor concept.

3.3.5. *Reference resolution*

Reference resolution or "anaphora" is a major issue in natural language understanding at the discourse level (Jurafsky & Martin, 2000). This is because several different phrases can refer to the same sentence constituent. In the design document we have surveyed, the following types of referring expressions are found to be typical.

Definite noun phrases. Definite reference refers to a noun phrase that has already been described in the discourse context. For example, "the motor" may refer to "a DC motor" appearing in the previous sentence. These types of references are solved by tracing back and finding the first match between the head word of the current noun phrase and the head word of the noun phrase in the previous sentence. If the matched phrase has populated its concept instance, the matched concept instance must replace the definite noun phrase.

Generics. Certain general noun phrases always present as references to the product to be designed, for example, "the assembly," "the system," or "the prototype." Therefore, they are substituted by the product concept instance.

https://doi.org/10.1017/S0890060407070199 Published online by Cambridge University Press

3.3.6. Joining

ODART uses concept instances to structure the extracted design semantics through a set of slots/relationships among them. For example, each device concept instance may have relevant concept instances describing its functions, materials, manufacturing operations, properties, part–whole relations, and other environmental objects interacting with it. The joining phase scans the sentences iteratively to generate relationships between concept instances according to the semantic rules defined. In the first scanning, the sentences are scanned to look for property concept instances and their associated values. Each property concept instance is associated with a value, that is, a cardinal number, that may be followed by a measurement unit concept instance by the order of its appearance in the sentence. For example, in the sentence "Overall height of the test rig is 2 meters with a weight of over 10 kilograms," "overall height" and "weight" are property concept instances, whereas "2 meters" and "10 kilograms" are attached to the has_value slot of these two property concept instances, respectively. In the second scanning, the prepositional phrases (e.g., "in" and "of") are checked together with the adjacent concept instances. They may represent the relationships of the has_part, has_function, or has_property, depending on the categories of the adjacent concept instances. Table 1 shows some examples of prepositional phrase analysis. The joining phase scans each sentence again to associate the rest of the concept instances to the corresponding slots of the device concept instances in the same sentence. In the next iteration, the same device concept instances are combined, as well as their slot fillers. The final scan looks for component concept instances that have not been recognized as a part of other device concepts and assigns them to fill the "has_part" slot of the product concept instance. An example of a sentence analyzed is shown in Figure 7. The top portion describes the results after different steps of analysis, whereas the bottom portion illustrates the generated concept instances and their relation to each other through their slots.

After the semantics extraction, only the desired design semantics are extracted. The sentences and paragraphs that do not include the relevant design semantics are discarded. As mentioned before, the extracted concepts and relationships of a document are represented as a subgraph with the product concept instance as the root node. The root node also stores the links to the document for retrieval.

3.4. Query processing

In query processing, ambiguity occurs both syntactically and semantically. The syntactic ambiguity refers to when a given query keyword matches several concepts. For example, the concepts "motor," "DC motor," and "AC motor" all match the keyword "motor." Hence, disambiguation is needed to select the most appropriate concept from the matched ones. At the semantic level, one of the challenges for the current query processing technique is to understand the true meaning of a user's queries. For example, with the query of "find a part which rotates and drives follower up and down," the system should return designs like a cam mechanism even though the word "cam" does not appear in the query or the documents. However, in general keyword-based methods, because the query is treated as a list of independent keywords, the meaning of the query is lost and the retrieval performance is hurt. In our approach, the ontology characterizes the matched concepts and their relationship. The semantics of the query can be recovered at the system representation by using an appropriate concept scoring metric. The metric that measures the semantic closeness between the document and the query is based on an observation from linguistics: the way to disambiguate the meanings of a word in a sentence is by referring to its context such as the adjacent words (Nirenburg & Raskin, 2004).

3.4.1. Scoring metrics

It is assumed that the user's queries are in plain English and request for documents that describe a component or product satisfying the desired design specifications. Examples of queries are "find parts that rotate," "find products having DC motor and linkage," and "find brackets made of plastic and assembled with other parts by snap fit." Key-

Table 1. *Examples of prepositional phrase analysis*

Sentences	Prepositions and Adjacent Phrases	Semantic Rules	Extracted Relationships
The mechanisms show the movement of the gear in the power train to drive the rear wheels.	Gears in the power train	CCI^PP(in)^CCI→has_part (CCI, CCI)	has_part (power train, gears)
The rotation of the cam results in the cam followers and the brake shoes being forced radially outward.	The rotation of the cam	FCI^PP(of)^^CCI→has_function (CCI, FCI)	has_function (cam, rotation)

https://doi.org/10.1017/S0890060407070199 Published online by Cambridge University Press

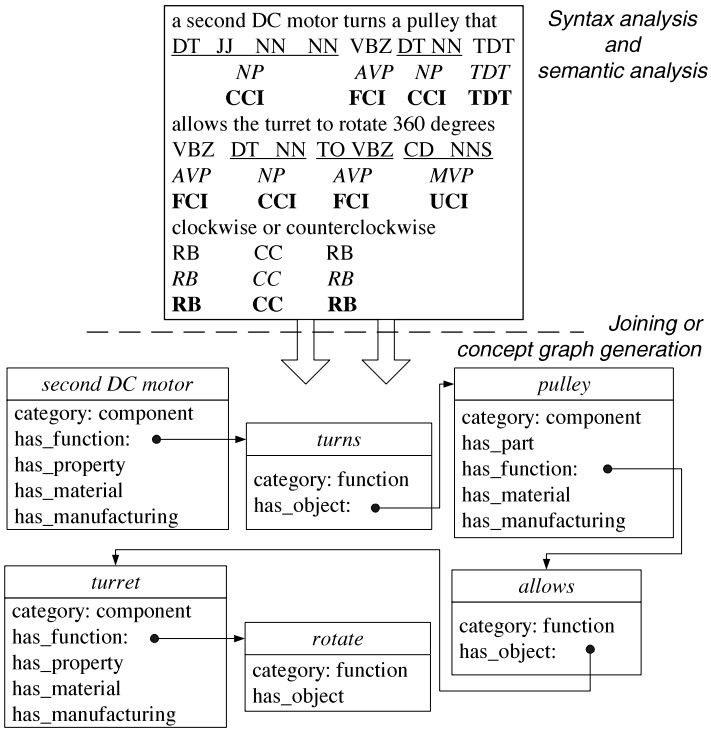


Fig. 7. An example of design semantics extraction from a sentence. The first line of tags indicates the analysis of the sentence after part of speech tagging. The italic tags in the second line and the bold tags in the third line show the analyses of the sentences after phrase chunking and after concept recognizing, respectively. The concept graph is after reference resolution and joining.

words are generated from the text of the query after tokenization and removal of the stop words. They are further analyzed based upon the domain ontology and the application ontology: the domain ontology provides concepts as index keys that are matched by the keywords through the lexical terms of the concepts by BFS. The selected concepts store the direct reference to the corresponding concept instances in the application ontology. Recall that the link to a document is stored with the root node of the subgraph in the application ontology, which comprises concept instances and relationships extracted from the document. Therefore, two metrics are required to find the relevant documents from the query keywords through the domain ontology index and the application ontology index.

$$Tscore_{ij} = \frac{(\#\text{ of keywords in the query } T_j \text{ matches with}) * \sum_{m=1}^{N} \omega_m}{\#\text{ of words in } T_j}, \tag{3}$$

$$Cscore_i = \max(Tscore_{ij}) \quad 1 \le j \le n. \tag{4}$$

Similar to the concept selection metric defined in Section 5, the first metric, the syntactic metric, is to select the concepts from all of the concepts in the domain ontology that match with any of the query keywords. Recall that each concept is represented as a set of lexical terms. The words in each term are compared against the keywords as a whole phrase. Equations (3) and (4) calculate the term score and concept score to select the matched concepts. The index m refers to the position of the words in the term phrase from right to left. Only concepts with a score of 1 are selected. To improve the recall of the retrieval, if a parent concept is selected, its child concepts will also be selected through BFS. We call this process *concept propagation*. However, for the same parent concept, if it has child concepts that also scored 1, these child concepts are selected and the parent concept is discarded. For example, in Figure 3, if the query is "find products having DC motors," the corresponding keywords are "product," "DC," and "motors." The matched concepts are "product," "motor," "DC motor," "AC motor," and "DC powered pump" (under "pump" concept). Their concept scores are 1.0, 1.0, 1.0, 0.275, and 0.075, respectively. Only the concepts "product" and "DC motor" are selected ("motor" is discarded because it is the parent of "DC motor"), and thereof their child concepts.

The second metric, that is, the semantic metric, is to rank the documents that have the concept instances (in the application ontology) referenced by the selected concepts (in the domain ontology). Disambiguation is required to prune the selected concepts that are syntactically appropriate but semantically inappropriate. For example, assume a query is to "find a product that can turn around." The selected con-

https://doi.org/10.1017/S0890060407070199 Published online by Cambridge University Press

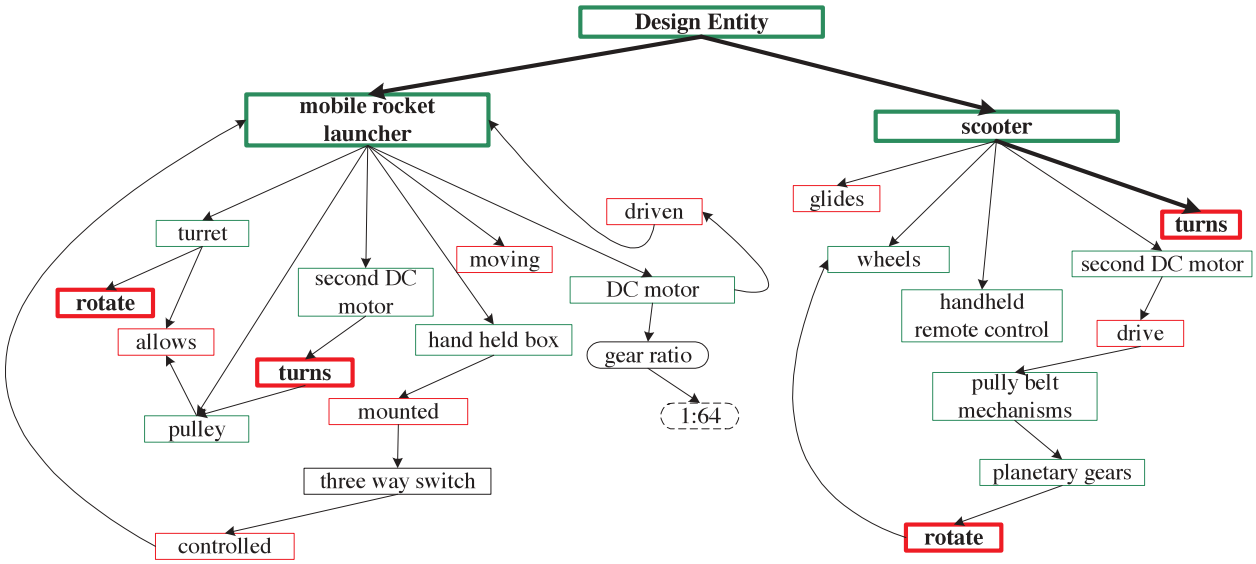


Fig. 8. An illustration of concept pair selection. [A color version of this figure can be viewed online at www.journals.cambridge.org]

cepts are "turn" and all the child concepts of "product," such as "mobile rocket launcher" and "scooter." Figure 8 illustrates the partial semantic representation from the design reports of these two products. Among them, the concept instance of "turn" appears four times and represents functions of three component concepts ("rotate" for "turret," "turns" for "DC motor," and "rotate" for "planetary gears"), and one product concept ("turn" for "scooter"). However, the first three are not product concepts and should not be counted when calculating the ranking of the document. Therefore, it is necessary to use correlated concepts, that is, *concept pair*, instead of separate concepts for disambiguation. The concept pair refers to the desired concept instances being connected by a directed arc in the application ontology. Therefore, the has_function (DC motor, turns), has_function (turret, rotate), and has_function (planetary gears, rotate) are not concept pairs because their component concept instances are not referenced by the selected concept in the domain ontology. Only the has_function (scooter, rotate) is chosen as a concept pair. Note that the root node, that is, "Design Entity," of the application ontology is selected by default in each query processing to ensure that there are concept pairs when queries are for upper level product concepts, for example, "find all vehicles." The pseudocode for finding and counting the concept pairs in a subgraph is shown in Figure 9. Finally, the subgraph, that is, the document that has the largest number of concept pairs, is retrieved first.

3.5. Empirical evaluation

A proof of concept was developed in C++. Figure 10 displays the user interface of ODART. The extracted design semantics are rendered in concept graphs. The pictures of the CAD model are shown only for visualization purposes. They are stored in the same folder with the textual documents. Users can search the design documents by typing a short sentence and read the original document by clicking the button under the corresponding design.

3.5.1. Experimental setup

This research used design reports from a senior engineering design class (ME444, computer-aided design and prototyping) at Purdue University as the test bed. Each team in the class implemented a semester-long project by designing

```
DFS (u) //Start at the root node of the sub-graph
if u is selected concept instance
    if u is PCI
//first time visit the root node of the subgraph
        concept_pair++
        last_node = PCI
    else if u is CCI
        last_node = CCI
        if (last_node == PCI)
            concept_pair++
    else if u is FCI
        last_node = FCI
        if (last_node == PCI or CCI)
            concept_pair++
    else if (last_node == PCI or CCI)
        concept_pair++
end if
for each arc c incident on u do
    if c is unvisited then
        let v be the other endpoint of c
        if v is unexplored
            //still has arc not visited
            DFS(v)
        else
            label c as visited
end
```

Fig. 9. Pseudocode for the concept pair search and calculation.

https://doi.org/10.1017/S0890060407070199 Published online by Cambridge University Press

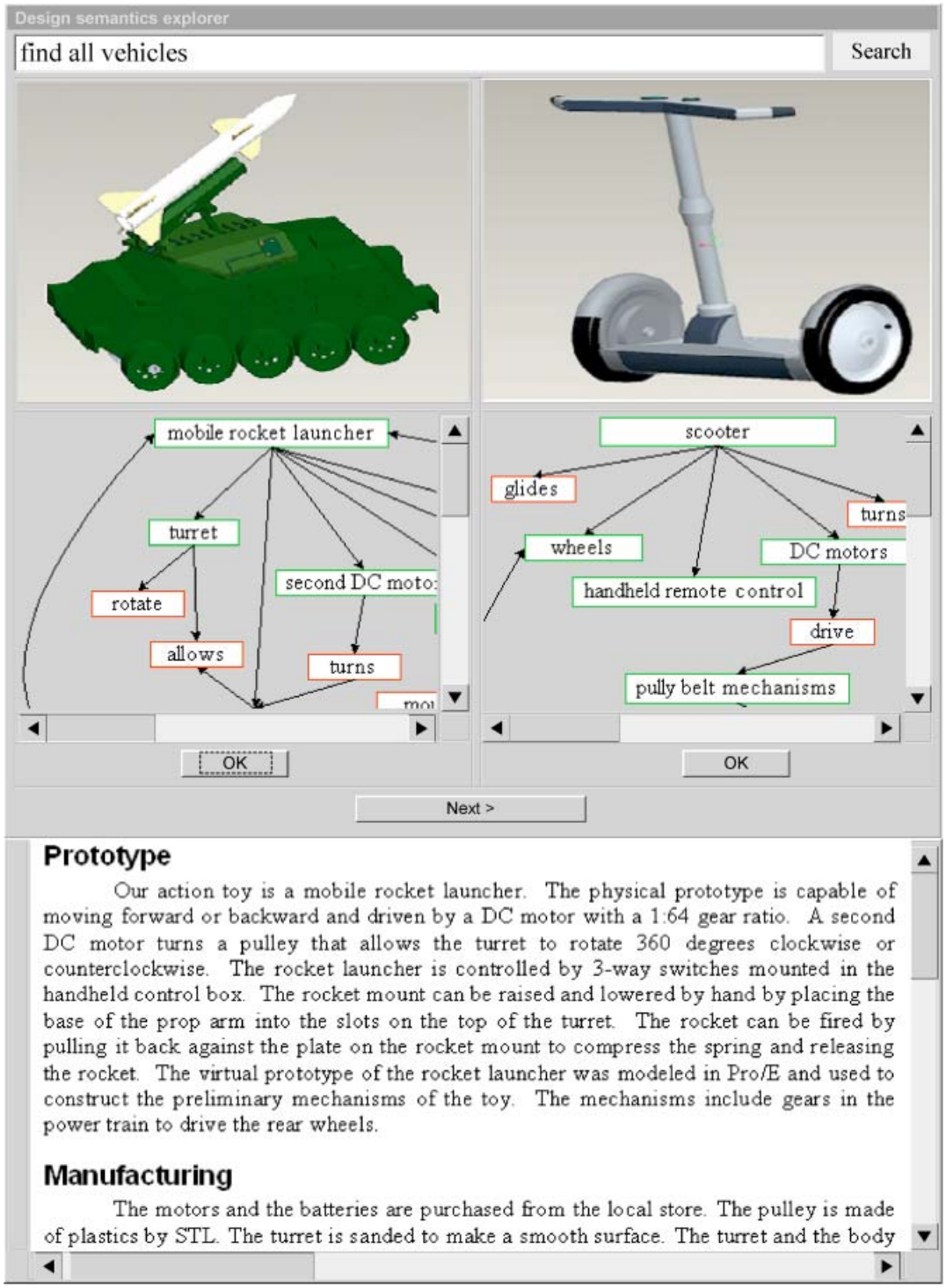


Fig. 10. The system–user interface. [A color version of this figure can be viewed online at www.journals.cambridge.org]

and prototyping a real product, such as an action toy, with some kinematic mechanisms. By the end of the project, teams submitted their final design reports online in a folder along with CAD files and other documents related to their designs. Each report had two major sections describing the physical prototype being designed and manufactured, a summary of the design and manufacturing, and a table for bill of materials (BOM). The summary section is the most information rich and therefore of interest to us. The BOM table has important information about suppliers and costs. It is part of our future research to analyze and extract the design semantics from tables embedded in the design document.

https://doi.org/10.1017/S0890060407070199 Published online by Cambridge University Press

We collected 158 design reports from the classes of the past 3 years. The average length of each summary section was 16.26 sentences and the average number of words in each sentence was 12.37.

The domain ontology model was created using Protégé 3.1 (http://protege.stanford.edu/), an open source ontology editing tool, in which we created the taxonomies as concept hierarchies. The lexical terms of the concept were modeled as the slot attribute of each concept class. It also supports the output of the domain ontology model in multiple formats, such as XML, OWL, and RDF. The domain ontology model was translated into XML script as input to the system.

3.5.2. Experiment for query processing

We employed the standard IR definition of recall and precision (Baeza & Neto, 1999) to measure the retrieval performance:

$$\text{recall} = \frac{\text{number of retrieved relevant documents}}{\text{number of relevant documents}},$$

$$\text{precision} = \frac{\text{number of retrieved relevant documents}}{\text{number of retrieved documents}}.$$

For instance, if 5 documents are retrieved, 3 of which are correct, the total number of documents is 10; then, the recall is equal to 3 of 10, and the precision is equal to 3 of 5. We compared the effectiveness of our method with respect to the most widely used keyword-based technique, the vector model.

Vector model. In the vector model, documents and queries are represented by vectors. Each vector contains a set of words and their weights. The similarity between a query and a document is calculated as the cosine of two vector weights. The weight of each word in the document is calculated based on the normalized term frequency (tf) and inverse-document frequency (idf). Assuming document (D_j) and query (Q_i) have t terms and their associated weights are $w_{i,j}$ and $w_{i,q}$, respectively, for $i = 1$ to t, the equations for calculating the level of similarities between the documents and the query are as follows:

$$\text{sim}(D_j, Q_i) = \text{cosine}(D_j, Q_i) = \frac{\vec{D}_j \cdot \vec{Q}_i}{|\vec{D}_j||\vec{Q}_i|} = \frac{\sum_{i=1}^{t} \omega_{i,j} \times \omega_{i,q}}{\sqrt{\sum_{t=1}^{t} \omega_{i,j}^2} \times \sqrt{\sum_{i=1}^{t} \omega_{i,q}^2}}, \quad (5)$$

$$\omega_{i,j} = f_{i,j} * \text{idf}_i, \quad (6)$$

$$\omega_{i,q} = \left(0.5 + \frac{0.5\text{freq}_{i,q}}{\max_l(\text{freq}_{l,j})}\right) * \text{idf}_i, \quad (7)$$

$$f_{i,j} = \frac{\text{freq}_{i,j}}{\max_l(\text{freq}_{l,q})}, \quad (8)$$

$$\text{idf}_i = \log\left(\frac{+1}{n_i + 1} + 1\right), \quad (9)$$

where $f_{i,j}$ is the normalized frequency of a keyword k_i in a document D_j, idf_i is the inverse document frequency, $\text{freq}_{i,j}$ is the number of times k_i appears in document D_j, $\text{freq}_{i,q}$ number of times k_i appears in the query, $\max_l(\text{freq}_{l,j})$ is the number of times the most popular word in the documents appears, $\max_l(\text{freq}_{l,q})$ is the number of times the most popular keyword in the query appears, N is the total number of documents, and n_i is the number of documents in which k_i appears.

Empirical results. Preliminary tests were conducted to evaluate the effectiveness of using ontology models in improving the performance of retrieving the pertinent design information. Faculty members, graduate students, and senior undergraduate students in mechanical engineering were asked to form queries. They were briefed with the scope of the domain ontology and application ontology prior to query generation. There were a total of 80 queries generated. However, three of them were out of the scope of the ontology models, such as queries about tolerance and geometry modeling method used. Note that the relations between the experience of the subjects and the variation of their queries were not analyzed in this study. This should be a part of future research. The valid sample queries were classified into general queries and specific queries. The general queries were associated with upper level concepts of the device concept taxonomy, either at the product concept level, for example, "search for a vehicle," or at the component concept level, for example, "find products with electrical components." The specific queries were associated with the lower level concepts of the domain ontology. They can be further classified into device queries such as "find products having gear and DC motor," function queries such as "find parts that can rotate," material queries such as "find parts made of plastic," manufacturing process queries such as "show me STL parts," and environment queries such as "find products that have door mechanism moved by hand."

Table 2 shows the results of the average recall and precision for the queries in each category, as well as for all the queries taken together. The results demonstrated that the recall of our ontology-based model outperforms that of vector-model based techniques, especially for general queries. This is because for general queries, more relevant concepts are added to the selected concepts by concept propagation. For example, in the first general query mentioned above, the selected concepts will be expanded from the matched "vehicle" concept to all of its descendent concepts, such as "racing car" and "crane." In the function query example, the selected concept, "rotate," also refers to the concept instance such as "transform torque" and

https://doi.org/10.1017/S0890060407070199 Published online by Cambridge University Press

Table 2. *Comparison of average recall and precision for two retrieval techniques*

Type of Queries	No. of Queries	Recall of		Precision of	
		Ontology Model (%)	Vector Model (%)	Ontology Model (%)	Vector Model (%)
General					
Product	20	100	20	100	100
Component	5	93	7	84	62
Specific					
Device	13	92	80	82	74
Function	11	90	75	92	100
Material	10	92	75	90	82
Manufacturing process	10	88	85	95	85
Environment	8	94	75	85	76
Average		93	60	89	82

"turn," which are the synonyms defined in the lexical terms of "rotate." Therefore, more relevant documents are located by the ontology-based model because of concept propagation and the list of lexical terms. Our approach also achieves better performance with respect to precision because the concept pairs introduce contextual constraints to improve the matching accuracy.

Note that for the retrieved results the vector model is rank based, whereas the ontology-based model is a Boolean retrieval model. To make a reasonable comparison, we decided that in the former case a retrieved document is relevant if it has a similarity score greater than 0.25. The reported precision is when the maximum recall is achieved.

4. CONCLUSIONS AND FUTURE WORK

We have described a framework for design information extraction and retrieval that aims at being effective with respect to the content-bearing phrases encountered in unstructured and textual design documents. The centerpiece of our method is the layered design ontology model, where the application ontology is automatically acquired using a shallow natural language processing technique as well as the taxonomies defined in the domain ontology. We have demonstrated the process used to conceptualize and develop the domain ontology and identified the resources. The domain ontology is organized in a modular structure so it can be extended easily to include taxonomies from other stages of product life cycle, such as customer requirement analysis. By resolving the reference resolution in the design document, the shallow natural language processing technique increases the amount of design information content extracted. The automatically populated application ontology represents each design at the semantic level through the concept graph, which is an explicit as well as succinct representation because only design concepts and relationships are incorporated into it. Users' search intent is detected by using ontology-based query processing: the domain ontology model provides related concepts to augment the recall of the retrieval, and the concept graph structure of the application ontology prunes the irrelevant concepts to improve the precision. The experimental results demonstrate that our ontology-based search outperforms the keyword-based search. Using a sample size of 158 documents, we found that our results improve the average recall by 33% and the average precision by 7%. Future work should focus on measuring performance improvement with larger data sets and in several design environments.

The research suggests that current PDM/PLM systems should take into account the importance of the structured and semantics-based representation from textual design documents, such as technical reports and drawing notes. By extracting dispersed design information and associating it in a structure that is explicit and able to support the inference on a meaningful level, our method has the potential of achieving a more coherent design environment with future PDM/PLM systems.

This research has only begun to explore the possibilities of semantics-based design document analysis for deriving a structured and meaningful model of a design and its application. It represents an initial study in using design ontology for design information representation, extraction, indexing, and retrieval. Because this subject matter in engineering design is an emerging area, future directions are identified below.

First, the most pressing question relates to the cost of maintaining the taxonomies of the domain ontology when there are novel concepts that need to be incorporated. Our current approach integrates a manual ontology engineering tool, that is, the Protégé modeling environment, to tackle this problem. Research in ontology learning and semantic web (e.g., Maedche & Staab, 2001) is addressing this issue by learning domain-specific ontology from a large corpus

https://doi.org/10.1017/S0890060407070199 Published online by Cambridge University Press

of Web pages. We would like to combine their method in order to minimize human intervention in maintaining the domain ontology.

Second, the domain ontology and the data set need to be expanded such that engineers can query the whole product life cycle information. We used the design project report as system input in the prototype. In practice, however, many more free-text documents are generated and stored during the design process, such as project proposals, memos, e-mails, and notes. These documents should also be processed to obtain a complete application ontology that will incorporate more comprehensive design semantics through the product life cycle. To achieve this, the domain ontology has to be extended to include taxonomies for other product life cycle stages. More importantly, the device concepts need to be modeled in a way that can reflect their progressive development histories.

Third, in expanding the current approach to other engineering domains, the same ontology structure as well as the semantics extraction algorithm can be applied. However, the domain ontology needs to be acquired to represent the fundamental concepts of the corresponding domain.

Fourth, for informal design documents, such as engineers' log books, the language used is less likely to comply with the formal documentation format and therefore, it is more difficult to detect linguistic patterns from them.

Fifth, the combination of the ontology-based textual design information extraction and retrieval with shape-based engineering part retrieval has the potential of producing more comprehensive descriptions and achieving more effective search results. We plan to integrate this research prototype with the sketch-based CAD model retrieval system (Pu & Ramani, 2005; http://me98pc26.ecn.purdue.edu/search.aspx).

ACKNOWLEDGMENTS

Our special thanks go to Professor David C. Anderson, School of Mechanical Engineering, Purdue University, for his valuable comments, and to Professor Victor Raskin, Department of English, Purdue University, for his insightful suggestions about ontology and natural language processing. We also acknowledge the support of the 21st Century R&T funds and National Science Foundation Partnership for Innovation Award (NSF-PFI) on ToolingNET. The authors thank the anonymous reviewers for their comments.

REFERENCES

Ahmed, S., & Wallace, K. (2003). Reusing design knowledge. In *Proc. 14th CIRP Design Seminar*, Cairo.

Ahmed, S., & Wallace, K.M. (2004). Identifying and supporting the knowledge needs of novice designers within the aerospace industry. *Journal of Engineering Design 15(5)*, 475–492.

Ahmed, S., Kim, S., & Wallace, K. (2005). A methodology for creating ontologies for engineering design. *Proc. ASME 2005 Int. Design Engineering Technical Conf. Computers and Information in Engineering Conf.*, Long Beach, CA.

Ashby, M.F. (1999). *Materials Selection in Mechanical Design*. Burlington, MA: Butterworth–Heinemann.

Baeza, R., & Neto, B. (1999). *Modern Information Retrieval*. New York: Addison–Wesley.

Baudin, C., Gevins, J., Baya, V., & Mabogunje, A. (1992). Dedal: using domain concepts to index engineering design information. *Proc. 14th Conf. Cognitive Science Society*, pp. 702–707, Anaheim, CA.

Baya, V., Gevins, J., Baudin, C., Mabogunje, A., Leifer, L., & Toye, G. (1992). An experimental study of design information reuse. In *Proc. ASME 4th Int. Conf. Design Theory and Methodology*, pp. 141–147, Scottsdale, AZ.

Brill, E. (1995). Transformation-based error-driven learning and natural language processing: a case study in part of speech tagging. *Computational Linguistics 21(4)*, 543–565.

Brin, S., & Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems 30(1–7)*, 107–117.

Busby, J.S. (1999). The problem with design reuse: an investigation into outcomes and antecedents. *Journal of Engineering Design 10*, 277–296.

Chung, Y.C., Lieu, R., Liu, J., Luk, A., Mao, J., & Raghavan, P. (2002). Thematic mapping—from unstructured documents to taxonomies. *Proc. 11th Int. Conf. Information and Knowledge Management*, pp. 608–610. New York: ACM Press.

Collins, J.A., Hagan, B.T., & Bratt, H.M. (1976). The failure-experience matrix—a useful design tool. *Journal of Engineering for Industry August*, 1074–1079.

Court, A.W., Culley, S.J., & McMahon, C.A. (1994). Information sources and storage methods for engineering data. *ASME Engineering Systems Design and Analysis 5*.

Court, A.W., Ullman, D.G., & Culley, S.J. (1998). A comparison between the provision of information to engineering designers in the UK and the USA. *International Journal of Information Management 18(6)*, 409–425.

Dong, A., & Agogino, A.M. (1996). Text analysis for constructing design representations. *Artificial Intelligence in Engineering 11(1)*, 65–75.

Dong, A. (2006). Concept formation as knowledge accumulation: a computational linguistic study. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 20(1)*, 35–53.

Farley, B. (2000). Extracting information from free-text aircraft repair notes. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 15(4)*, 293–305.

Fernández-López, M., Gómez-Pérez, A., & Juristo, N. (1997). METHONTOLOGY: from ontological art towards ontological engineering. *Spring Symp. Ontological Engineering of AAAI*, Stanford University, CA.

Glasgow, B., Mandell, A., Binney, D., Ghemri, L., & Fisher, D. (1998). MITA, an information-extraction approach to the analysis of free-form text in life insurance applications. *AI Magazine 19*, 59–71.

Goel, A.K., Bhatta, S.R., & Stroulia, E. (1997). KRITIK: an early case-based design system. In *Issues and Applications of Case-Based Reasoning in Design* (Maher, M., & Pu, P., Eds.), pp. 87–132. Mahwah, NJ: Erlbaum.

Guarino, N., Masolo, C., & Vetere, G. (1999). Ontoseek: content-based access to the web. *IEEE Intelligent Systems 14(3)*, 70–80.

Hirtz, J., Stone, R.B., McAdams, D.A., Szykman, S., & Wood, K.L. (2002). A functional basis for engineering design: reconciling and evolving previous efforts. *Research in Engineering Design 13(2)*, 65–82.

Hmelo-Silver, C.E., & Pfeffer, M.G. (2004). Comparing expert and novice understanding of a complex system from the perspective of structures, behaviors, and functions. *Cognitive Science 28*, 127–138.

Hobbs, J.R., Appelt, D.E., Bear, J., Israel, D., Kameyama, M., Stickel, M., & Tyson, M. (1996). FASTUS: a cascaded finite-state transducer for extracting information from natural-language text. In *Finite-State Devices for Natural Language Processing* (Roche, E., & Schabes, Y., Eds.). Cambridge, MA: MIT Press.

Jurafsky, D., & Martin, J.H. (2000). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Upper Saddle River, NJ: Prentice–Hall.

Kim, J., Will, P., Ling, S.R., & Neches, B. (2003). Knowledge-rich catalog services for engineering design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 17(4)*, 349–366.

Kitamura, Y., & Mizoguchi, R. (2004). Ontology-based systemization of functional knowledge. *Journal of Engineering Design 15(4)*, 327–351.

Kuffner, T.A., & Ullman, D.G. (1991). The information request of mechanical design engineers. *Design Studies 12(1)*, 42–50.

Kutz, M. (2002). *Handbook of Materials Selection*. New York: Wiley.

Kutz, M. (2005). *Mechanical Engineers' Handbook, Manufacturing and Management*. New York: Wiley.

https://doi.org/10.1017/S0890060407070199 Published online by Cambridge University Press

Levin, B. (1993). *English Verb Classes and Alternations: A Preliminary Investigation*. Chicago: University of Chicago Press.
Lowe, A., McMahon, C., Shah, T., & Culley, S. (2000). An analysis of the content of technical information used by engineering designers. *Proc. of 2000 ASME Design Engineering Technical Conf.*, Paper No. DETC2000/DTM-14545, Baltimore, MD.
Maedche, A., & Staab, S. (2001). Ontology learning the semantic web. *IEEE Intelligent Systems 16(2)*, 72–79.
Maher, M.L., & Silva-Garza, A.G. (1997). Case-based reasoning in design. *IEEE Expert 12(March–April)*, 34–41.
Marcus, M., Santorini, B., & Marcinkiewicz, M.A. (1994). Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics 19(2)*, 313–330.
Marsh, J.R. (1997). *The capture and utilization of experience in engineering design*. PhD. Thesis. University of Cambridge.
McMahon, C.A., Lowe, A., Culley, S.J., Corderoy, M., Crossland, R., Shah, T., & Stewart, D. (2004). Waypoint: an integrated search and retrieval system for engineering documents. *Journal of Computing and Information Science in Engineering 4(4)*, 329–338.
Nirenburg, S., & Raskin, V. (2004). *Ontological Semantics*. Cambridge, MA: MIT Press.
Olsen, G.R., Cutkosky, M., Tenenbaum, J.M., & Gruber, T.R. (1995). Collaborative engineering based on knowledge sharing agreements. *Concurrent Engineering: Research and Applications 2(3)*, 145–159.
Pu, J.T., & Ramani, K. (2005). A 3D model retrieval method using 2D freehand sketches. *Lecture Notes in Computer Science* (Sunderam, V.S., Albada, G.D.V., Sloot, P.M.A., & Dongarra, J.J., Eds.), Part II, pp. 343–347. Heidelberg: Springer.
Pugh, S. (1997). *Total Design: Integrated Methods for Successful Product Engineering*. Wokingham, MA: Addison–Wesley.
Riloff, E. (1996). Automatically generating extraction patterns from untagged text. *Proc. 13th National Conf. on AI*, pp. 1044–1049.
Salton, G. (1989). *Automatic Text Processing*. Wokingham, MA: Addison–Wesley.
Silberman, Y., Miikkulainen, R., & Bentin, S. (2001). Semantic effect on episodic associations. *Proc. 23rd Annual Conf. Cognitive Science Society* (Moore, J.D., & Stenning, K., Eds.). Edinburgh: Cognitive Science Society.
Sivaloganathan, S., Ed. (1998). *Proc. Engineering Design Conf. 98: Design Reuse. ASME 98*.
Sowa, J.F. (1984). *Conceptual Structures: Information Processing in Mind and Machine*. Reading, MA: Addison–Wesley.
Stauffer, L.A., Ullman, D.G., & Dietterich, T.G. (1987). Protocol analysis of mechanical engineering design. In *Proc. Int. Conf. Engineering Design* (Eder, W.E., Ed.), Vol. 1, pp. 74–85.
Sudarsan, R., Fenves, S.J., Sriram, R.D., & Wang, F. (2005). A product information modeling framework for product lifecycle management. *Computer-Aided Design 37(13)*, 1399–1411.
Sycara, K., & Navinchandra, D. (1989). Integrated case-based reasoning and qualitative reasoning in engineering design. In *Artificial Intelligence in Design* (Gero, J., Ed.). New York: Springer.
Szykman, S., Sriram, R.D., Bochenek, C., Racz, J.W., & Senfaute, J. (2000). Design repositories: Engineering design's new knowledge base. *IEEE Intelligent Systems 15(3)*, 48–55.
Tyhurst, J.J. (1986). Applying linguistic knowledge to engineering notes. In *Proc. Winter Annual Meeting of the American Society of Mechanical Engineers*, pp. 31–136, Anaheim, CA.
Ullman, D.G. (1997). *The Mechanical Design Process*. New York: McGraw–Hill.
Vries, B.D, Jessurun, J., Segers, N., & Achten, H. (2005). Word graphs in architectural design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 19(4)*, 277–288.
Weber, C., Werner, H., & Deubel, T. (2003). A different view on product data management/product life-cycle management and its future potentials. *Journal of Engineering Design 14(4)*, 447–464.
Yang, M.C., Wood, W.H., & Cutkosky, M.R. (2005). Design information retrieval: a thesauri-based approach for reuse of informal design information. *Engineering with Computers 21(2)*, 177–192.

Zhanjun Li is a doctoral student and Research Assistant in the Purdue Research and Education Center for Information Systems in Engineering (PRECISE) at Purdue University. He received an MS from Huazhong University of Science and Technology and a BS from Xi'an University of Technology, both in mechanical engineering. His research interests include engineering information retrieval, ontology engineering, design theory and methodology, and computer-aided design.

Karthik Ramani is a Professor of mechanical engineering and of electrical and computer engineering (courtesy) in the School of Mechanical Engineering and School of Electrical and Computer Engineering (courtesy) at Purdue University and the Director of PRECISE. He earned a BTech from the Indian Institute of Technology, Madras (1985), an MS from Ohio State University (1987), and a PhD from Stanford University (1991), all in mechanical engineering. Dr. Ramani serves on the editorial board of the *Journal of Computer-Aided Design*. His research interests are in geometric computation, shape design and analysis, configuration, constraint representation and solving, and physics as applied to early design.

APPENDIX A

Is_a (PCI, Design Entity): This relationship is only used between the root node of the application ontology, that is, design entity, and the root nodes of the subgraphs, that is, product concept instance. Note that this relationship is generated automatically and implicitly whenever there is a product concept instance extracted from the document.

Has_part (PCI/CCI, CCI): This relationship represents the part–whole between a product concept instance and a component concept instance, or between two component concept instances. For example, "mobile rocket launcher" has_part "DC motor."

Has_function (PCI/CCI, FCI): The relationship refers to the connection between a device concept and one of its functional concepts. For example, the "DC motor" has_function "turn."

Has_object (FCI, ECI/CCI/PCI): The relationship is an extension of the has_function relationship when there exists an "object" in the "subject + verb [+ objects]" function description. Together with the has_function (PCI/CCI, FCI), they represent the interactions between a component and the other device or environment object.

Has_property (PCI/CCI, PRCI): In general, each device concept instance has several property concept instances characterizing its physical and geometrical attributes. For example, the "motor" has_property of "gear ratio."

Has_value (RCI, VI): The numeric value (and its attached unit measurement concept instance) is extracted from the document to fill this slot of a property concept, for example, the "length" has_value "1.2 meters."

Has_material (CCI, MCI): This relationship is between a component concept instance and its material concept instance.

Has_manufacturing (CCI, MACI): This relationship is between a component concept instance and the manufacturing (concept instance) operation used to make it.

https://doi.org/10.1017/S0890060407070199 Published online by Cambridge University Press