Artificial Intelligence for Engineering Design, Analysis and Manufacturing (2007), *21*, 173–192. Printed in the USA.
Copyright © 2007 Cambridge University Press 0890-0604/07 $25.00
DOI: 10.1017/S0890060407070217

Product family design knowledge representation, aggregation, reuse, and analysis

JYOTIRMAYA NANDA,[1] HENRI J. THEVENOT,[1] TIMOTHY W. SIMPSON,[1] ROBERT B. STONE,[2] MATT BOHM,[2] AND STEVEN B. SHOOTER[3]

[1]Department of Industrial and Manufacturing Engineering, Pennsylvania State University, University Park, Pennsylvania, USA
[2]Department of Interdisciplinary Engineering, University of Missouri–Rolla, Rolla, Missouri, USA
[3]Department of Mechanical Engineering, Bucknell University, Lewisburg, Pennsylvania, USA

(Received January 11, 2006; Accepted November 25, 2006)

Abstract

A flexible information model for systematic development and deployment of product families during all phases of the product realization process is crucial for product-oriented organizations. In current practice, information captured while designing products in a family is often incomplete, unstructured, and is mostly proprietary in nature, making it difficult to index, search, refine, reuse, distribute, browse, aggregate, and analyze knowledge across heterogeneous organizational information systems. To this end, we propose a flexible knowledge management framework to capture, reorganize, and convert both linguistic and parametric product family design information into a unified network, which is called a networked bill of material (NBOM) using formal concept analysis (FCA); encode the NBOM as a cyclic, labeled graph using the Web Ontology Language (OWL) that designers can use to explore, search, and aggregate design information across different phases of product design as well as across multiple products in a product family; and analyze the set of products in a product family based on both linguistic and parametric information. As part of the knowledge management framework, a PostgreSQL database schema has been formulated to serve as a central design repository of product design knowledge, capable of housing the instances of the NBOM. Ontologies encoding the NBOM are utilized as a metalayer in the database schema to connect the design artifacts as part of a graph structure. Representing product families by preconceived common ontologies shows promise in promoting component sharing, and assisting designers search, explore, and analyze linguistic and parametric product family design information. An example involving a family of seven one-time-use cameras with different functions that satisfy a variety of customer needs is presented to demonstrate the implementation of the proposed framework.

Keywords: Design Repository; Information Management; Ontology; Product Family

1. INTRODUCTION

Managing product and process data over the total product lifecycle is one of the most critical business processes for many engineering products (Bourke, 1999). Continued pressure to reduce product development time has resulted in an increased focus on methods for representing and storing engineering artifact knowledge in a way that facilitates its retrieval and subsequent reuse (Szykman et al., 2000). Successful product family planning places an even greater requirement on effective information management to exploit the potential of shared assets (Simpson, 2004). By sharing assets such as components, processes, and knowledge across products, companies can efficiently develop a family of differentiated products for a variety of market segments and increase the flexibility and responsiveness of their product realization process (Shooter et al., 2005).

The need for computational design frameworks to support the representation and use of knowledge among distributed designers becomes more critical as product design becomes increasingly knowledge intensive and collaborative (Szykman et al., 2000). In the knowledge management framework proposed in this paper, we capture, reorganize, and convert component design information into a unified

Reprint requests to: Timothy W. Simpson, Department of Industrial and Manufacturing Engineering, 329 Leonhard Building, Pennsylvania State University, University Park, PA 16802, USA. E-mail: tws8@psu.edu

network, called a network bill of material (NBOM), to facilitate both linguistic and parametric design information management for a family of products. The NBOM is encoded as a cyclic, labeled graph using the Web Ontology Language (OWL), an open standard proposed by the Semantic Web group at the World Wide Web consortium (W3C). Collaborative design in heterogeneous and distributed design environments necessitates the use of ontologies as a common communication framework. Ontologies have been developed for many fields to establish common vocabularies and capture domain knowledge, and they have proven to be an advantageous paradigm over recent years (van der Vegte et al., 2002). Gennari et al. (1994) discuss the high payoff of saved effort due to reuse of preexisting knowledge captured in ontologies. The preconceived product family ontologies help designers explore, search, and aggregate design information across different phases of product design as well as across multiple products in a product family. A PostgreSQL database schema has been formulated to serve as a central repository of product design knowledge, capable of housing the instances of the NBOM.

In the next section, we review related literature in product representation and knowledge management. In Section 3 we introduce the knowledge management framework that is proposed in this paper for capturing, organizing, storing, and analyzing information during product family design. An example involving a family of seven one-time-use cameras is given in Section 4 to demonstrate the use of the framework. Section 5 provides closing remarks and discusses future work.

2. BACKGROUND AND RELATED WORK

2.1. Product design information management

A *design* comprises information that can be in many forms (Dixon & Poli, 1999). A *designed artifact* has many different kinds of information associated with it during product realization, starting from highly abstract information in the early phases of design to very detailed information at the parametric phase. As the design evolves, the accompanying information is represented at different levels of abstraction, and Shooter et al. (2000) present a model for the flow of design information to eventually support a semantics-based approach for developing information exchange standards. At the abstract end of the spectrum, Kirschman and Fadel (1998) proposed a taxonomy of elemental mechanical functions that can be used with many decomposition techniques. Iwasaki and Chandrasekaran (1992) focused on the task of design verification using both knowledge of the structure of a device and its intended functions. For more detailed information, Bohm et al. (2005) review commercial software packages that provide a hierarchical decomposition of product structure and their related proprietary and nonproprietary output formats.

The complexity in managing product structure and the associated information increases in a collaborative design environment where the use of different software systems is common, and the task is further complicated by the proprietary nature of much of this information (Hatvany et al., 1993). Saaksvuori and Immonen (2003) discuss the many common output standards that many of these systems use, including the data exchange format, standard for the exchange of product model data, and initial graphics exchange specification, to name a few. Conversion tools among these formats are costly, application-specific, and often lead to further uncertainty about data integrity. The limited design information captured by the individual software systems in proprietary data structures makes it difficult to index, search, and browse design artifacts across the organizational information systems.

The National Institute of Standards and Technology (NIST) is involved in the development of an intelligent design repository based on data language (DL) and a design representation language (Murdock et al., 1997; Szykman et al., 2000). The design repository at the University of Missouri–Rolla (UMR), following NIST's approach toward neutral data exchange, has implemented an extensible mark-up language (XML)-based approach to import and export the product knowledge from the design repository (Bohm et al., 2005). Although the XML representation provides a standard data structure for describing the artifacts, it does not provide the semantics, that is, the meaning of the data structure. As part of the proposed knowledge management framework, we address this limitation by adding a semantic layer using OWL ontologies to the UMR Design Repository that contains more than 100 products (http://function.basiceng.umr.edu/repository). To organize both the linguistic and parametric design information, we use formal concept analysis (FCA), as discussed in the next section.

2.2. FCA

FCA is used for analyzing data and forming semantic structures that are formal abstractions of concepts of human thought (Ganter & Wille, 1999). It borrows its mathematical foundation from order theory, particularly the theory of complete lattices (Gratzer, 1998). In the literature, FCA is used for natural language processing (Priss, 2003), modular information retrieval (Godin et al., 1993), query-based software component retrieval (Lindig, 1995), identification of potential modules in computer programs (Siff & Reps, 1997), design decision making in educational applications (Fernandez-Manjon & Fernandez-Valmayor, 1998), and data analysis and machine learning (Kuznetsov, 2001). Godin and Mili (1993) proposed a formal method for building, maintaining, and refining hierarchies in object-oriented programs using Galois lattices.

The basic notion of FCA is structured around the notion of a *formal context* and a *formal concept*. A *formal context* is a triple $K := (C, P, R)$, where C is a finite set of *objects*,

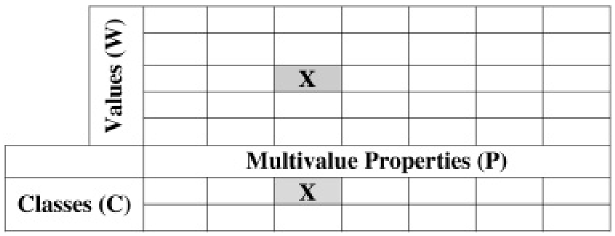


Fig. 1. A multicontext cross table.

P is a finite set of *attributes*, and R is a binary relation between C and P. A *formal concept* of K is represented by the elements of $B(K) := \{(C_1, P_1) \mid C_1 \subseteq C, P_1 \subseteq P, C_1' = P_1, P_1' = C_1\}$; hence, a *formal concept* is a combination of a set of *objects* (called its *extent*) and a set of *attributes* (called its *intent*) such that all objects have all attributes and all attributes belong to all objects. A *relation* R is represented as a subset of the cross product between the finite sets of objects and attributes, that is, $R \subseteq C \times P$. The mappings captured by relation R give rise to a Galois connection (Erné et al., 1991). A *cross table* is used to capture the binary *relation* R between *objects* C and *attributes* P.

A many-valued formal context (C, P, W, R) consists of sets C, P, and W and a ternary relation R between C, P, and W (see Figure 1), that is, $R \subseteq C \times P \times W$ for which it holds that $(c, p, w) \in R$ and $(c, p, v) \in R$ always imply $w = v$. The notation $(c, p, w) \in R$ is read as "the attribute p has value w" for the *object* c. The many-valued attributes are regarded as partial maps from C in W. *Conceptual scaling* (Ganter & Wille, 1999) is the method that is used on many-valued contexts to derive the concept lattice.

The Galois or concept lattice of K is a partially ordered set represented by $\underline{B}(K) = (B(K), \leq)$ with $(C_1, P_1) \leq (C_2, P_2) :\Leftrightarrow C_1 \subseteq C_2\ (\Leftrightarrow P_1 \supseteq P_2)$. Here, $\underline{B}(K)$ is a complete lattice (Gratzer, 1998) for which the infimum (greatest common formal subclass) and supremum (smallest common formal subclass) are given as the following:

$$\bigwedge_{i\in I}(C_i, P_i) = \left(\bigcap_{i\in I} C_i, \sigma\left(\tau\left(\bigcup_{i\in I} P_i\right)\right)\right), \tag{1}$$

$$\bigvee_{i\in I}(C_i, P_i) = \left(\tau\left(\sigma\left(\bigcup_{i\in I} C_i\right)\right), \bigcap_{i\in I} P_i\right), \tag{2}$$

where I is an index set.

A concept lattice is graphically represented using a line diagram known as the Hasse diagram (Ganter & Wille, 1999). The nodes in the Hasse diagram of a context are labeled dually with the objects (below) and attributes (above), and the vertices represent the relationships between the objects. A reduced labeling scheme is often used and is applied in this work as demonstrated in Section 4.

FCA can be used to construct lattice structures for large-scale problems in real time. Let l be the total number of formal concepts, m be the total number of properties, and k the total number of classes in a concept lattice. Then the worst case complexity of the lattice constructing algorithm is generally admitted to be $O((k + m)lmk)$ (Valtchev et al., 2002). Using the algorithm proposed by Nourine and Raynaud (1999) reduces the complexity to $O((k + m)lm)$. Dividing the concepts and forming partial lattices before forming the global lattice structure reduces the complexity to $O((k + m)lm \log m)$ (Valtchev et al., 2002).

There are a number of software tools developed in the last two decades to support FCA in different fields. General Lattice Analysis and Design (Duquenne et al., 2001), Contexts and Implications (Burmeister, 2003), Tools of Concept Analysis (TOSCANA; Groh et al., 1998), and ToscanaJ (Vogt & Wille, 1995; Becker, 2004) are some of the software tools that support FCA. Tilley (2004) reviews these and various other tools available for FCA in more detail.

2.3. Knowledge management and ontologies

The knowledge management community has developed a wide range of technologies and applications as reviewed by Liao (2003). Among these, the Semantic Web is envisioned by W3C as an extension of the current Web to create Web-based knowledgeable systems with various specialized reasoning services systems (Davies et al., 2003). The goal of the Semantic Web is to develop standards and technologies designed to help machines understand more information on the Web and support richer discovery, data integration, navigation, and automation of tasks (Koivunen & Miller, 2001).

Ontologies play an important role in achieving this goal. An ontology consists of a set of concepts, axioms, and relationships that describes a domain of interest. These concepts and the relationships between them are usually implemented as classes, relations, properties, attributes, and values (of the properties/attributes; Daconta et al., 2003). The W3C's Semantic Web initiative proposes a layered approach to a standard Web ontology language, namely, OWL (McGuinness & van Harmelen, 2004). OWL is currently the most expressive semantic markup language for publishing and sharing ontologies on the World Wide Web. Figure 2 summarizes the degree of formality (level of semantic information) stored along with the data by a particular method of information storage, and presents the technological maturity level of OWL and many methods that have preceded it.

Ontologies, like OWL, are well suited for product family design knowledge representation by supporting reasoning outside the transaction context, that is, it avoids a protocol specification to handle standard data format. Instead of trying to capture all the knowledge about a domain or family in a single ontology, OWL promotes distributed ontology development and enables multiple ontologies to be easily linked. OWL ontologies can store the structure of several products in a product family in multiple distributed ontologies to facilitate design information sharing over the Internet. This is also useful for concurrent product family development where the semantics of multiple products need

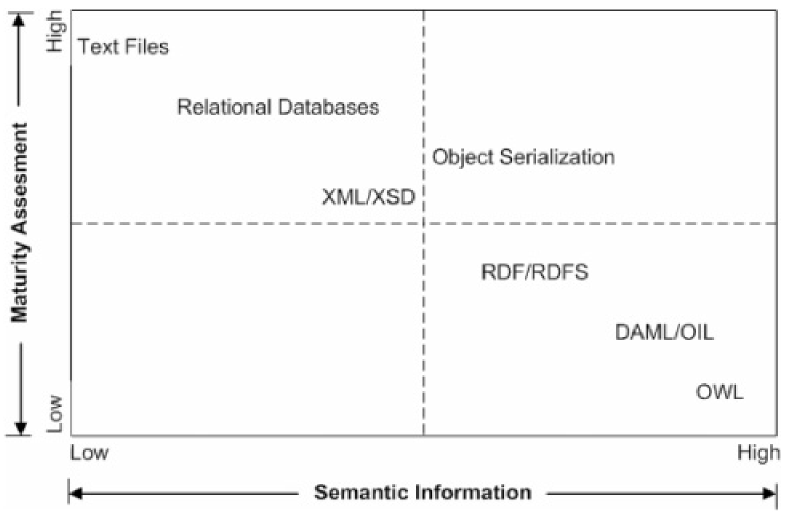


Fig. 2. Methods of persistent information storage.

to be shared across multiple stages of product development. OWL is supported by description logic (Baader et al., 2003), which makes it easier for computers to interpret the semantics without human intervention. Programming packages like Protégé (Noy et al., 2001) can provide a graphical user interface for editing ontologies, and Jena (McBride, 2002) has application programming interfaces for generic OWL manipulation and a rule-based inference engine. Consequently, OWL is central to our knowledge management framework, which is introduced next.

3. KNOWLEDGE MANAGEMENT FRAMEWORK FOR PRODUCT FAMILY DESIGN

To manage the design knowledge associated with a group of related products, we propose the product family knowledge management framework shown in Figure 3. There are three steps to the framework, which aims to organize and analyze both linguistic and parametric product family design information. The first step uses the product family ontology

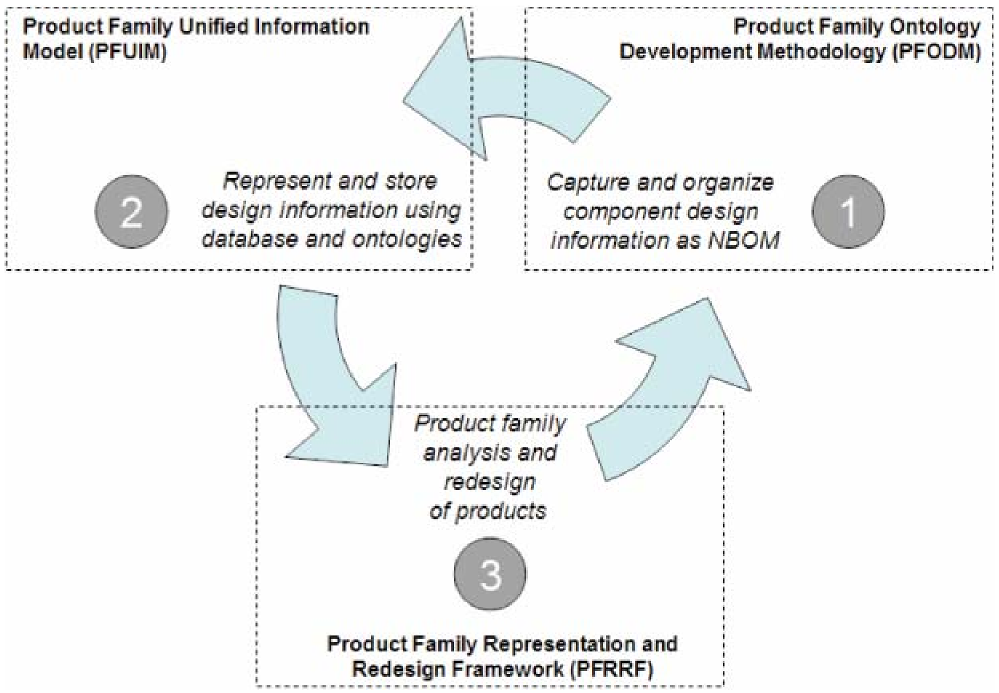

Product Family Unified Information Model (PFUIM)
Product Family Ontology Development Methodology (PFODM)
2
Represent and store design information using database and ontologies
Capture and organize component design information as NBOM
1
Product family analysis and redesign of products
3
Product Family Representation and Redesign Framework (PFRRF)

Fig. 3. The product family knowledge management framework. [A color version of this figure can be viewed online at www.journals.cambridge.org]

development methodology (PFODM) for design knowledge acquisition, graph-based organization, and ontology development (Nanda et al., 2006). The second step employs a database and ontology metadata to represent design knowledge as a graph-based product family uniform information model (Nanda et al., 2005*a*). In the third step, the product family representation and redesign framework (PFRRF) is used for commonality assessment as well as product family redesign (Nanda et al., 2005*b*). The steps are cyclical to ensure and maintain the relevancy of design information in the database (in our case, the UMR Design Repository), and to help designers analyze multiple products within a single product family. The primary contribution in this paper is the integration of these three steps into the framework shown in Figure 3; details on the implementation of each step and their integration follow.

3.1. Step 1: Capture and organize component design information into an NBOM

The first step in the proposed framework is to obtain the necessary data for the product family concerned. This first step is critical to ensure that information is captured and organized in a way such that relevant data can then be represented and stored (see Step 2). The typical way to capture data is using a bill of materials (BOM); hence, if the information is already available through a BOM, then the user can directly reuse the data. If the information is not readily available, dissection of the products in the family is required. We employ the subtract and operate procedure of Otto and Wood (2001) to gather the following information for each part: size, geometry, material, manufacturing process, and assembly/fastening scheme. Production volume and unit cost for each part should also be obtained. If they are not readily available, appropriate methods should be used to estimate them (Boothroyd & Dewhurst, 2002; Ulrich & Eppinger, 2004). In this paper, only high-level features are captured to demonstrate the proposed approach; however, the list can be as refined and detailed as needed for a particular product family. Likewise, for complex large-scale systems, the level of granularity can be varied as needed to build ontologies at the component level, module level, or subsystem level, or any combination thereof, as appropriate for the family being analyzed.

3.1.1. Overview of the database schema to capture linguistic and parametric design data

In this research, we are using an already extensively developed design repository, the UMR Design Repository. It captures design information instances that are useful during product family design. The design information captured by the design repository data schema can be classified into seven main groups: artifact-, function-, failure-, physical-, performance-, sensory-, and media-related information types (Bohm et al., 2005). The database tables are broken up into two separate categories: those that directly store product information and those that are referenced by product storing database tables (Bohm et al., 2006). Figure 4 shows an illustration of the UMR Design Repository schema. The boxes represent data tables, and the arrows represent data relationships. All the repository data tables are represented in this figure with the 13 data storing tables highlighted. A data table makes a reference to another table by an outbound arrow to a particular data table. The tables that store taxonomies and bases are denoted with _type after the table name. Taxonomy and basis storing tables do not reference design data storing tables. Thus, for example, the failure table references the *artifact*, *failure_type* and *failure_rating_type* tables but is referenced by the *failure_data_info* table.

There are several types of media that can be associated with artifacts. Media types can take the form of pictures, graphical functional models, graphical assembly models, two-dimensional CAD files, three-dimensional CAD files, stereo lithographic (.stl) files for rapid prototyping machines, and many others. All the types of media, mentioned and unmentioned, reside in the *media* table of the repository. Instances of media are unique and associated with an artifact, which is captured by the id and *describes_artifact* field. The *data* field in the *media* table is the "large object" pointer for the actual media files of any type.

Information entry occurs within a front-end entry application, whereas information retrieval occurs over the Internet through the UMR Design Repository's Web portal. The current and emerging versions of the repository are built and served by a PostgreSQL (a SQL variant) database (Douglas & Douglas, 2003). There are close to 100 products present in the current PostgreSQL database. In the next section, the specific use of FCA to organize and extract semantic design information is discussed.

3.1.2. Using FCA to organize the database information as a graph

The information stored in the design repository is then organized using FCA to transform stored information into reusable knowledge. FCA is semantically enriched using OWL by applying the PFODM shown in Figure 5 (Nanda et al., 2006). In this methodology, the individual product hierarchies in a BOM are merged to create the product–component *cross table* (see Figure 6), which is the input for FCA. In the product–component *cross table*, products are represented as *objects* and component *instances* as *attributes* with single or multiple values. The product–component cross table can capture both the binary and multicontext relationships between components and products.

A *component* can exist in many *component instances*. A *component* is a part that is used for a certain function or set of functions. Two parts are *component instances* from the same component if they are used for the same functions or set of functions and they differ slightly by size, shape, material, and so forth. The decision to group a number of *component instances* under a particular *component* is mainly made by a designer. For example, consider the two flash

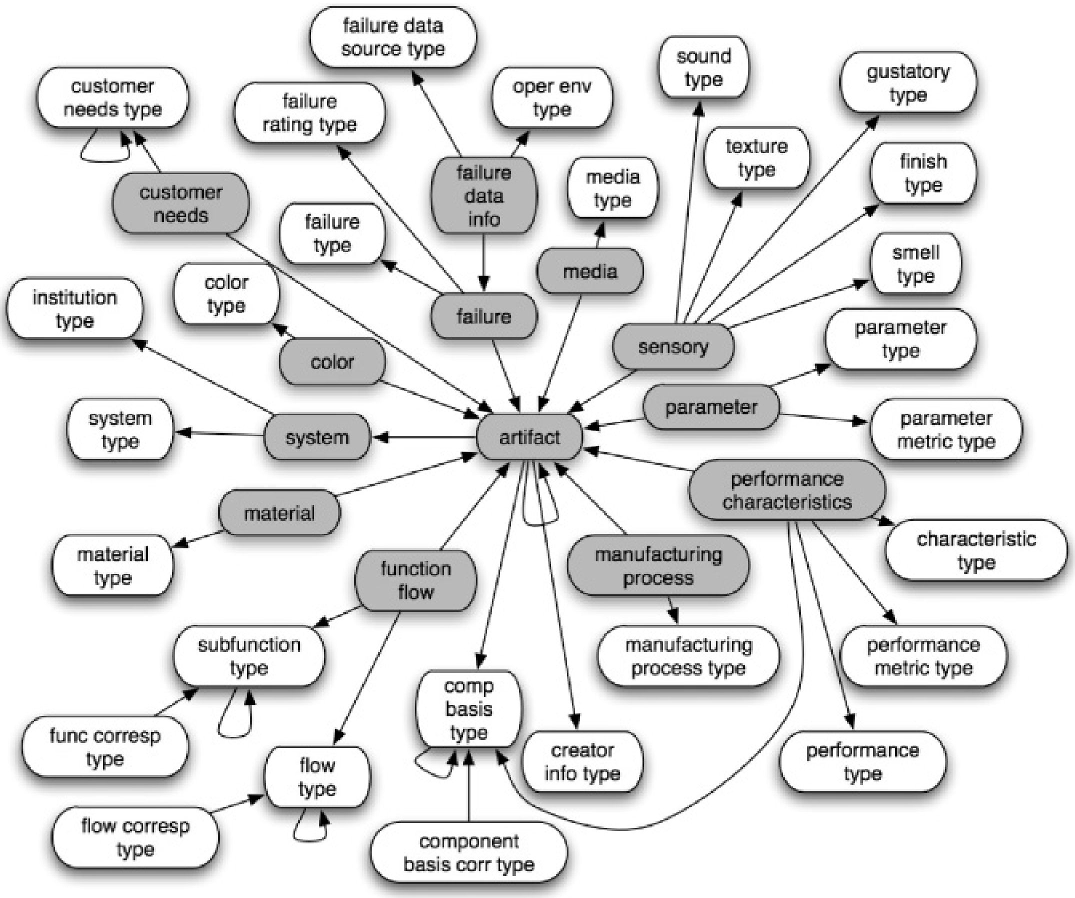


Fig. 4. A graphical view of the University of Missouri–Rolla Design Repository database tables (Bohm et al., 2006).

buttons shown in Figure 7 taken from two one-time-use cameras. They are two different *component instances* (flash button 1 and flash button 2) of the *component* flash button: they provide the same function (turn on the flash) but differ in shape and color. Once these decisions are made, the NBOM can be created as discussed in the next section.

3.2. Step 2: Represent and store product design information

Once the information has been captured and relationships have been created (Step 1), this knowledge is then represented and stored using a product family unified information model (Step 2). This step aims at visualizing relationships that are not captured in traditional product family data information management, and at helping designers navigate through the extensive information collected in Step 1.

3.2.1. *Encoding the NBOM using OWL*

The *many-valued formal contexts* captured in step 1 are used to develop the product family *concept lattice* using FCA, which is then converted into OWL using the PFODM shown in Figure 5 (Nanda et al., 2006). The partial order set of the concept lattice is used to develop the subsumption

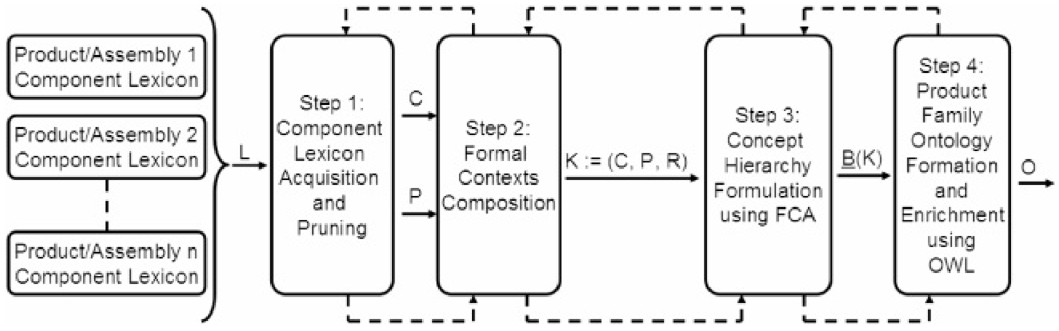

Product/Assembly 1 Component Lexicon
Product/Assembly 2 Component Lexicon
Product/Assembly n Component Lexicon
L
Step 1: Component Lexicon Acquisition and Pruning
C
P
Step 2: Formal Contexts Composition
K := (C, P, R)
Step 3: Concept Hierarchy Formulation using FCA
B(K)
Step 4: Product Family Ontology Formation and Enrichment using OWL
O

Fig. 5. The product family ontology development methodology (PFODM).

Instances		1	1	1	1
		2	2	2	2
		3	3	3	3
		…	…	…	...
		n	n	n	n
		Components			
		Component 1	Component 2	…	Component n
Products	Product 1	1			1
	Product 2	1	2		
	…				
	Product n	2			n

Fig. 6. A product–component multicontext cross table.

hierarchy of the product family ontology. Figure 8 shows the mapping between the products and the component lattices after the ontologies are constructed. Components that are part of a product also have their own lattice structure based on their individual properties. This way, all product and component knowledge is connected as a graph in the knowledge base and stored as multiple OWL ontologies to facilitate browsing, graph-based queries, and ultimately component reuse.

We refer to the resulting concept lattice as the NBOM (Nanda et al., 2005*c*). The NBOM can capture the unique, variant, and common components in a product family as shown in Figure 9. Representing the NBOM in ontologies enables designers to query the knowledge base based on graph pattern matching using querying languages like RDQL (Miller et al., 2002), SPARQL (Clark, 2005), and OWL-QL (Fikes et al., 2003). The NBOM also allows instances of classes that represent the components to be compared against each other, which is useful for product family assessment and redesign (see Step 3).

In addition to the NBOM, the product vector matrix (PVM) and function component matrix (FCM) are used to represent and map different design information structures. Because of unique and variant components present in individual products, each mapping is unique for a particular product in a product family.

The PVM maps the relationships between the functions listed in the function structure model and the weighted customer needs (see Fig. 10). The first column of the PVM lists the product functions (PFs), and the weighted customer needs (CNs) are listed across the top row of the PVM. Because we are only interested in the mapping between design information structures, we do not consider customer needs weights in the PVM at this time. For each function that impacts a particular customer need, a "1" is entered into the cell in the matrix. In Figure 10, the highlighted cell signifies that for product X customer need 1 (CN_1^X) is satisfied by product function 2 (PF_2^X).

FCM maps the relationships between the functions captured in the function structure model and the components listed in the BOM (see Fig. 11). Similar to PVM, for each component that impacts a particular product function, a "1" is entered into the cell in the matrix. In Figure 11, the highlighted cell signifies that for product X component 2 (C_2^X) partially satisfies product function 1 (PF_1^X).

Figure 12 describes the multimodal design representation of a product family using common ontologies. A single ontology is built across the various phases of product family design and is used by each individual product to represent the unique component instances in a product family. The common ontological layer not only ensures data consistency in a given phase but also helps in aggregating information across products in a product family.

The designer can perform an exploratory data analysis in the product family by choosing any single design artifact from a single product and then going back and forth between

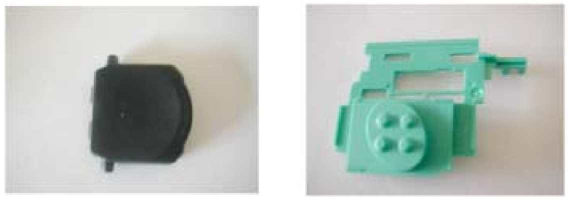


Fig. 7. Two component instances of a camera flash button. [A color version of this figure can be viewed online at www.journals.cambridge.org]

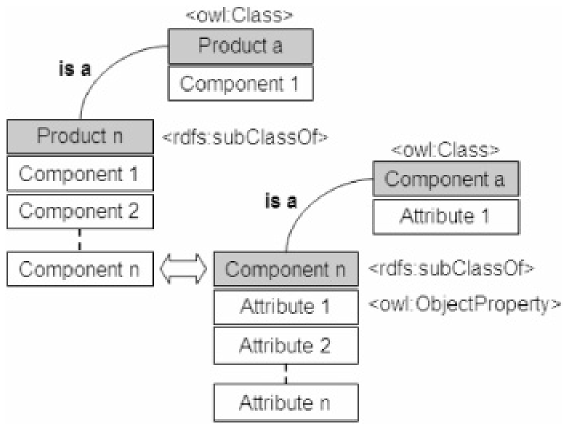


Fig. 8. Product–component–attribute integration.

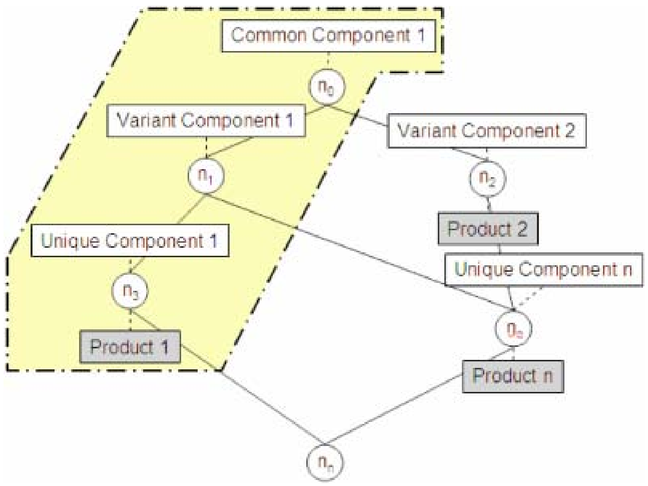
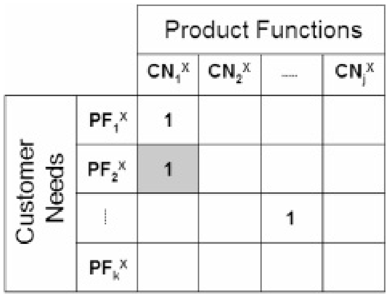


Fig. 9. The networked bill of material (NBOM) lattice structure. [A color version of this figure can be viewed online at www.journals.cambridge.org]

the common ontological layer and the product instance representation. The product family design information aggregation can further be subdivided into two groups: design information aggregation between products of a product family and design information aggregation across different phases of the product realization process. Both are described in more detail in the following.

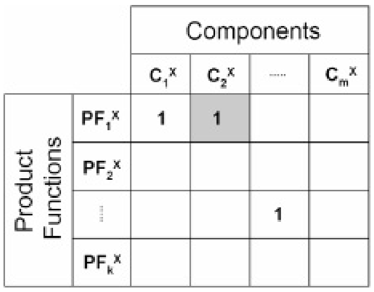
3.2.2. Product family design information aggregation spanning multiple products and across phases

Due to the common ontology layer, each design entity that is part of an individual product is an instance of an OWL class. This conceptual grouping using ontologies helps aggregate information between products within a family. Starting with a single design instance selected by the designer, the information system can query and list all the other instances of that particular class present in the design repository. The information system can also query related classes from the ontology and present them to the designer. For example, if we start with a single product function for product *X*, say PF_1^X, then the system can locate the class for this instance as PF_1^C and list all the other instances of this class present in the system automatically. In addition, when new products are introduced in the product family, they will be represented using these preconceived ontologies making the integration and design analysis process automatic.



		Product Functions			
		CN_1^X	CN_2^X	.....	CN_j^X
Customer Needs	PF_1^X	1			
	PF_2^X	1			
	⋮			1	
	PF_k^X				

Fig. 10. The product vector matrix for an individual product.



		Components			
		C_1^X	C_2^X	.....	C_m^X
Product Functions	PF_1^X	1	1		
	PF_2^X				
	⋮			1	
	PF_k^X				

Fig. 11. The function component matrix for an individual product.

PVM and FCM also capture the relationships between two different design information structures. This mapping helps the system to transparently present design information from across different phases of the product realization process. For example, if we start with a single customer need instance for product *X* CN_1^X, we can easily get the components that satisfy this customer need by moving from customer need to product function to component space using PVM first and then FCM.

3.3. Step 3: Product family analysis and redesign

After the design repository is populated with data for each product (Step 1) and a metadata layer is created using ontologies (Step 2), the product family can be analyzed to identify opportunities to redesign products in the family to improve commonality. In this paper, we use commonality indices for this purpose; they are one of many tools available to support product family analysis and redesign (Simpson, 2004). Commonality indices are reviewed next, followed by methodologies to redesign products in the family using the NBOM.

3.3.1. Using commonality indices to analyze a product family

The primary motivation for product family design is to increase commonality among the products in the family, and several component-based commonality indices have been developed to assess the degree of commonality within a product family. Thevenot and Simpson (2006*a*) provide a detailed comparison between many of these commonality indices and discuss their usefulness for product family redesign. Five such indices are used for illustration purposes in the example in Section 4.3: the *Degree of Commonality Index* (DCI) proposed by Collier (1981), the *Total Constant Commonality Index* (TCCI) from Wacker and Trelevan (1986), the *Commonality Index* (CI) introduced by

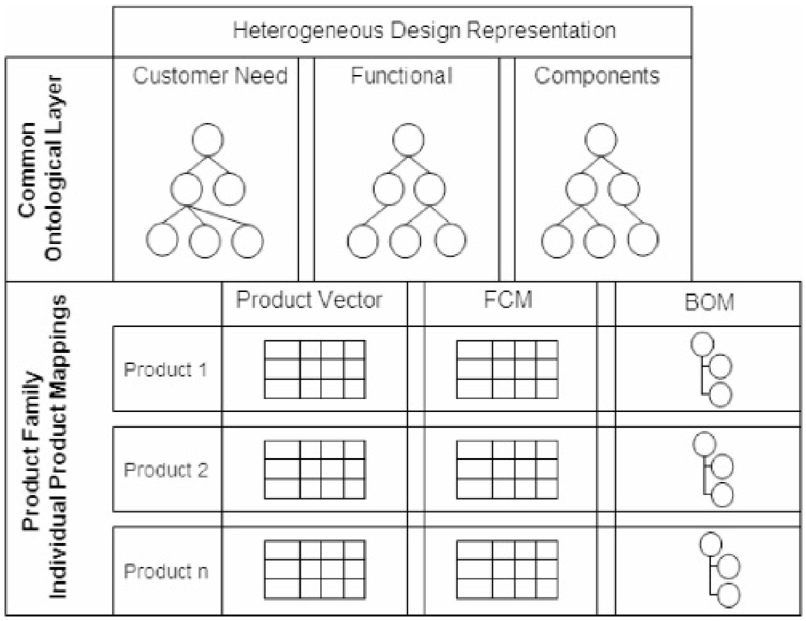


Fig. 12. The model for product family design information aggregation (Nanda et al., 2005*a*).

Martin and Ishii (1997), the *Product Line Commonality Index* (PCI) of Kota et al. (2000), and the *Comprehensive Metric for Commonality* (CMC) developed by Thevenot and Simpson (2006*b*).

3.3.2. Using the graph structure to redesign the product families

Based on the NBOM representation developed in Step 2, we have proposed two approaches to navigate the concept lattice to redesign the product family (see Fig. 13). The first is a *component-based* approach, wherein designers select unique or variant components in a product family and try to make them variant or common to improve commonality in the family. The second is a *product-based* approach, whereby designers select multiple products from a product family and try to increase the commonality between the selected products. In both cases, preference is given to making variant components common over making unique components variant to maximize economies of scale. The steps for implementing both approaches are listed in Figure 13 and details can be found in Nanda et al. (2005*b*) where the resulting PFRRF was first introduced. Its use is demonstrated in the next section, along with the first two steps of the proposed knowledge management framework, with an example involving a family of one-time-use cameras.

4. PRODUCT FAMILY KNOWLEDGE MANAGEMENT: A CASE STUDY

The following sections demonstrate implementation of the three steps that constitute the knowledge management framework to organize and analyze product family design information. For this example, a family of seven one-time-use cameras manufactured by Kodak is used (see Table 1). These cameras are readily available in the market, and offer specific differentiating functions: flash, digital processing, waterproof, black and white, and the Advanced Photo System with switchable format as listed in the table.

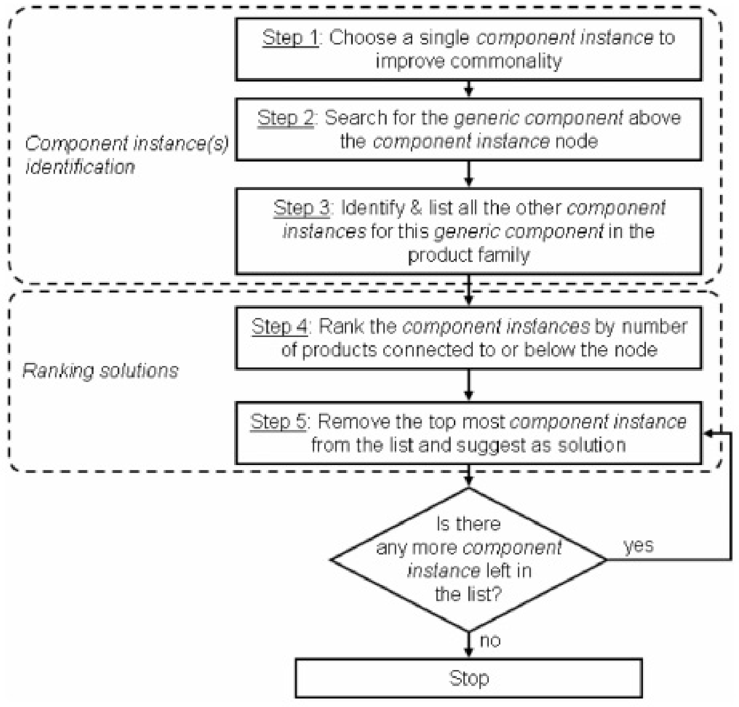
4.1. Step 1: Capture and organize product family design information

To develop the lexicon set to describe the components in the family of one-time-use cameras, they are disassembled to the component level where a component is a part of the camera that cannot be further decomposed into design artifacts. Sections 4.1.1 and 4.1.2 discuss capturing and organizing the design artifact data about the camera family.

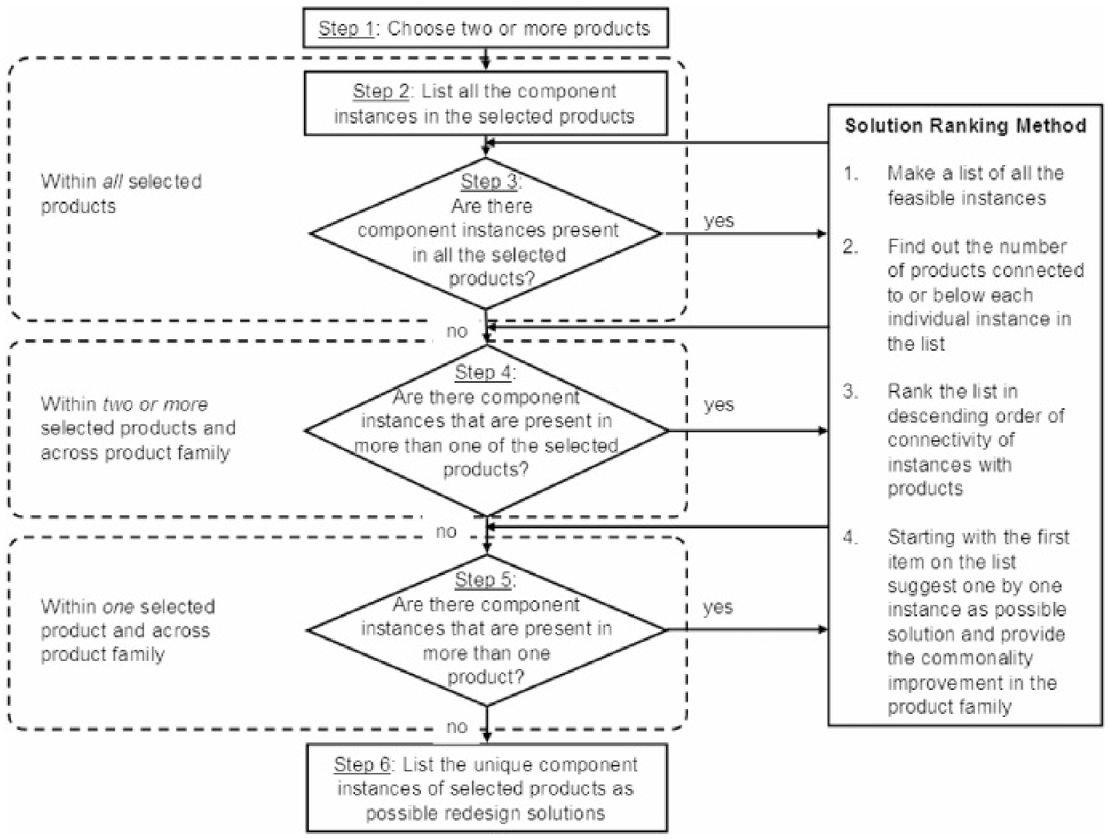
4.1.1. Capturing design information for the camera family

After the products are disassembled, all of the terms associated with the components are collected in a dictionary form, including synonyms for the components. Based on this dictionary of terms or lexicon set, the product family ontology is formalized. For example, the lexicon set for describing the shutter cover for the MAX Outdoor camera can be L_1 := {MAX Outdoor shutter cover, shutter cover of MAX Outdoor, MAX shutter cover}. Figure 14 shows both the linguistic and parametric information associated with a camera shutter cover as stored in the UMR Design Repository.

The linguistic information captured for the Water & Sport and Plus Digital cameras is presented in Figure 15. The


Component instance(s) identification
Step 1: Choose a single component instance to improve commonality
Step 2: Search for the generic component above the component instance node
Step 3: Identify & list all the other component instances for this generic component in the product family
Ranking solutions
Step 4: Rank the component instances by number of products connected to or below the node
Step 5: Remove the top most component instance from the list and suggest as solution
Is there any more component instance left in the list?
yes
no
Stop

(a) Component-based approach for product family redesign


Step 1: Choose two or more products
Step 2: List all the component instances in the selected products
Within all selected products
Step 3: Are there component instances present in all the selected products?
yes
no
Within two or more selected products and across product family
Step 4: Are there component instances that are present in more than one of the selected products?
yes
no
Within one selected product and across product family
Step 5: Are there component instances that are present in more than one product?
yes
no
Step 6: List the unique component instances of selected products as possible redesign solutions
Solution Ranking Method
1. Make a list of all the feasible instances
2. Find out the number of products connected to or below each individual instance in the list
3. Rank the list in descending order of connectivity of instances with products
4. Starting with the first item on the list suggest one by one instance as possible solution and provide the commonality improvement in the product family

(b) Product-based approach for product family redesign

Fig. 13. The product family representation and redesign framework (PFRRF; Nanda et al., 2005*b*).

Table 1. *Summary of one-time-use cameras*

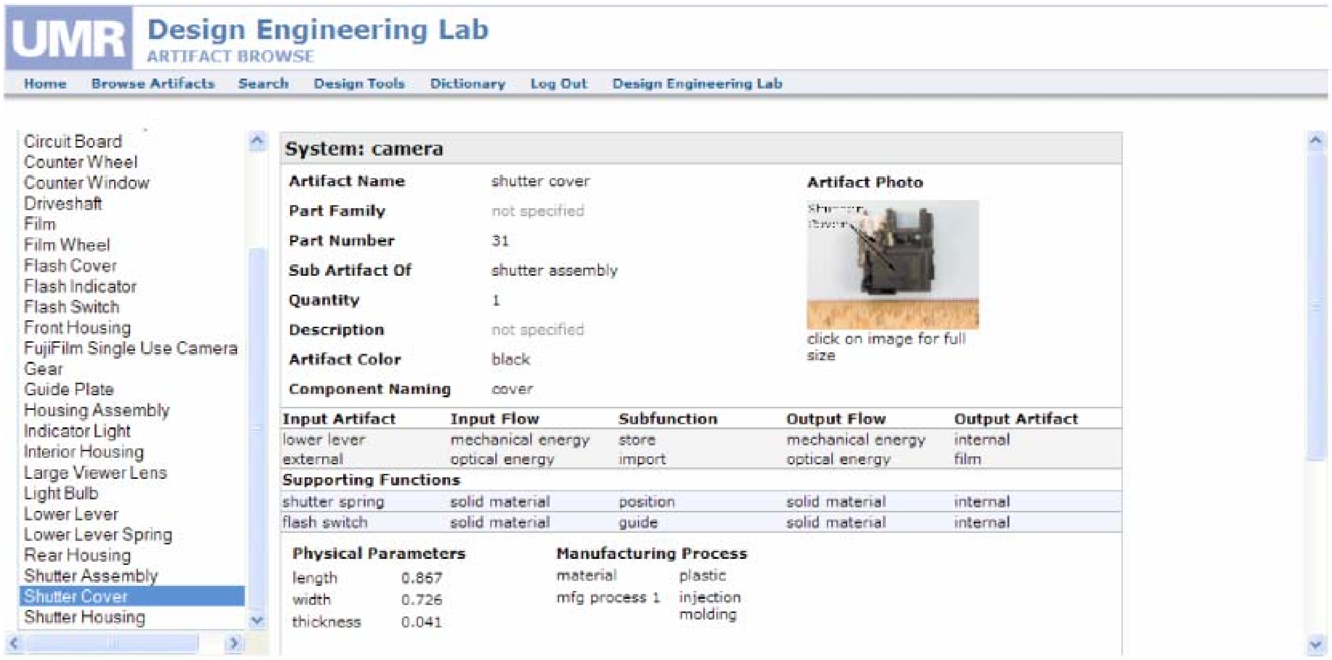
	MAX Outdoor	MAX Flash	Plus Digital	MAX HQ	ADVANTIX Switchable	Black & White	MAX Water & Sport
Film	35 mm color	35 mm color	35 mm color	35 mm color	24 mm color	35 mm black & white	35 mm color
Flash	No	Yes	Yes	Yes	Yes	Yes	No
Waterproof	No	No	No	No	No	No	Yes
Switchable format	No	No	No	No	Yes	No	No
Digital processing	No	No	Yes	No	No	No	No

A color version of this table can be viewed online at www.journals.cambridge.org

linguistic information thus captured is also used to create a dictionary to store the commonly associated keywords. These keywords help reduce the proliferation of synonyms in the design repository. Alternatively, standard terms from the functional basis (Hirtz et al., 2002) and component basis (Kurtoglu et al., 2005) could be used.

4.1.2. Organizing the design information for the camera family

Once the design information is captured, FCA is used to organize the information in a graph structure, which provides the link between all the terms associated with design artifacts in the design repository. To demonstrate the proposed approach, we consider three cameras from the product family: MAX Power Flash, Plus Digital, and Water & Sport. Seven types of components (front cover insert, waterproof front cover, battery, film advance wheel, shutter spring, exposure counter) that are part of the three products are used to illustrate the NBOM. Table 2 shows the components and their relationships with the cameras in a multicontext cross table. Different numbers are used in the cross table to differentiate component types. For example, to represent two types of film advance wheel, the numbers 1 and 2 are used in the cross table.



Circuit Board
Counter Wheel
Counter Window
Driveshaft
Film
Film Wheel
Flash Cover
Flash Indicator
Flash Switch
Front Housing
FujiFilm Single Use Camera
Gear
Guide Plate
Housing Assembly
Indicator Light
Interior Housing
Large Viewer Lens
Light Bulb
Lower Lever
Lower Lever Spring
Rear Housing
Shutter Assembly
Shutter Cover
Shutter Housing

System: camera

Artifact Name	shutter cover
Part Family	not specified
Part Number	31
Sub Artifact Of	shutter assembly
Quantity	1
Description	not specified
Artifact Color	black
Component Naming	cover

Artifact Photo

Input Artifact	Input Flow	Subfunction	Output Flow	Output Artifact
lower lever	mechanical energy	store	mechanical energy	internal
external	optical energy	import	optical energy	film
Supporting Functions				
shutter spring	solid material	position	solid material	internal
flash switch	solid material	guide	solid material	internal

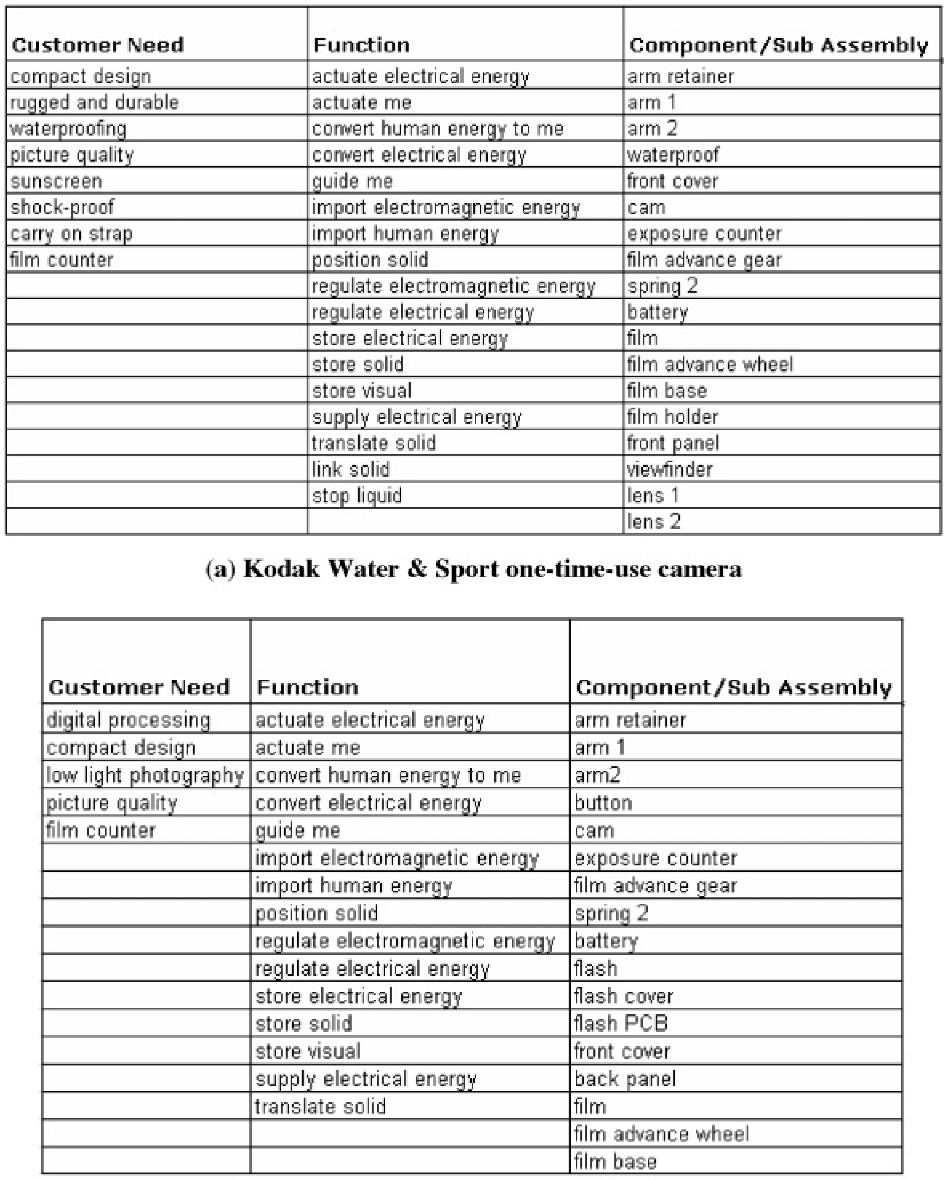
Physical Parameters

length	0.867
width	0.726
thickness	0.041

Manufacturing Process

material	plastic
mfg process 1	injection molding

Fig. 14. Shutter cover details in the design repository. [A color version of this figure can be viewed online at www.journals.cambridge.org]



Customer Need	Function	Component/Sub Assembly
compact design	actuate electrical energy	arm retainer
rugged and durable	actuate me	arm 1
waterproofing	convert human energy to me	arm 2
picture quality	convert electrical energy	waterproof
sunscreen	guide me	front cover
shock-proof	import electromagnetic energy	cam
carry on strap	import human energy	exposure counter
film counter	position solid	film advance gear
	regulate electromagnetic energy	spring 2
	regulate electrical energy	battery
	store electrical energy	film
	store solid	film advance wheel
	store visual	film base
	supply electrical energy	film holder
	translate solid	front panel
	link solid	viewfinder
	stop liquid	lens 1
		lens 2

(a) Kodak Water & Sport one-time-use camera

Customer Need	Function	Component/Sub Assembly
digital processing	actuate electrical energy	arm retainer
compact design	actuate me	arm 1
low light photography	convert human energy to me	arm2
picture quality	convert electrical energy	button
film counter	guide me	cam
	import electromagnetic energy	exposure counter
	import human energy	film advance gear
	position solid	spring 2
	regulate electromagnetic energy	battery
	regulate electrical energy	flash
	store electrical energy	flash cover
	store solid	flash PCB
	store visual	front cover
	supply electrical energy	back panel
	translate solid	film
		film advance wheel
		film base

(b) Kodak Plus Digital one-time-use camera

Fig. 15. Kodak one-time-use camera linguistic information capture.

The cross table in Table 2 is then converted into the NBOM using FCA. The resulting NBOM structure of the product family is shown in Figure 16. The products are shown below the nodes of the NBOM, whereas components are represented on the top of the nodes in the lattice. The formal context and the concept lattice for the one-time-use camera family ontology is formalized using ToscanaJ (Vogt & Wille, 1995; Becker, 2004).

The component cross table is built using attributes for material, color, manufacturing process, and weight for illustration (see Fig. 17). Designers can include as many attributes as necessary to compare components between each other and develop component lattices. The cross table facilitates building of the lattice structure using FCA (see Fig. 18), which is then stored as an OWL ontology for later retrieval and reuse.

4.2. Step 2: Represent and store design information using ontologies

The lattice structure developed in the previous section is next encoded using OWL-DL and enriched using Protégé-2000 (Noy et al., 2001), an ontology editor and a knowledge base editor, with the OWL (http://protege.stanford.edu/plugins/owl/) and the ezOWL plug-ins (http://smi-protege.stanford.edu/svn/ezowl/). Protégé-2000 is one of the most

Table 2. *Kodak product family cross table*

	Front Cover Insert	Waterproof Front Cover	Battery	Film Advance Wheel	Shutter Spring	Exposure Counter
MAX Power Flash			1	1	1	1
Plus Digital	1		2	2	1	1
Water & Sport		1		2	1	1

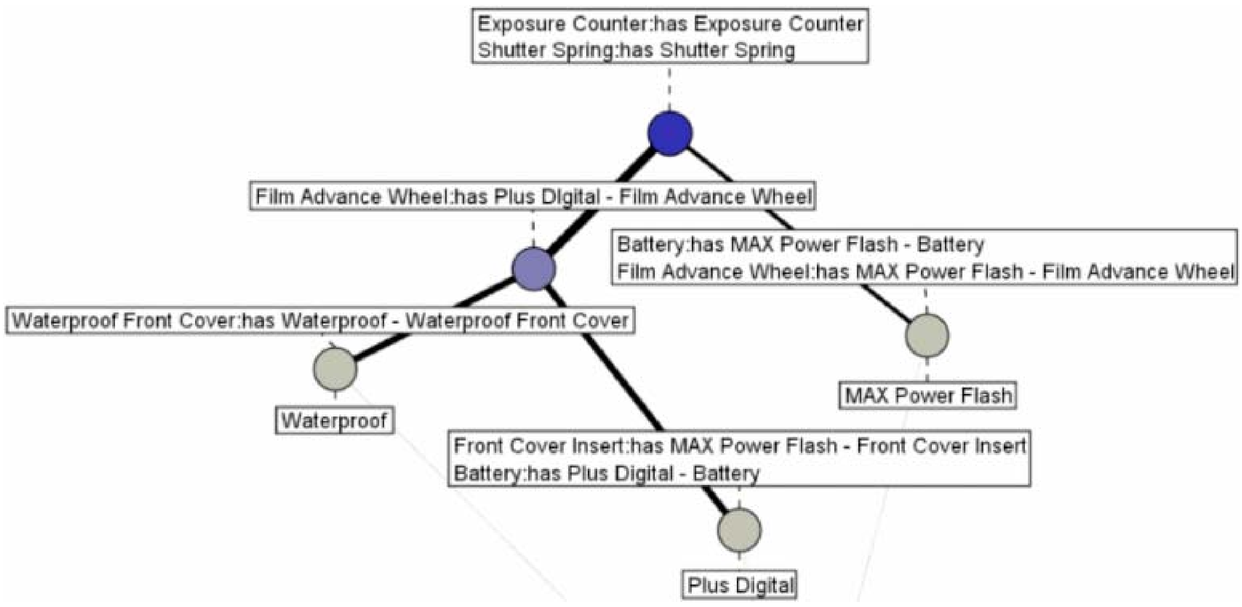
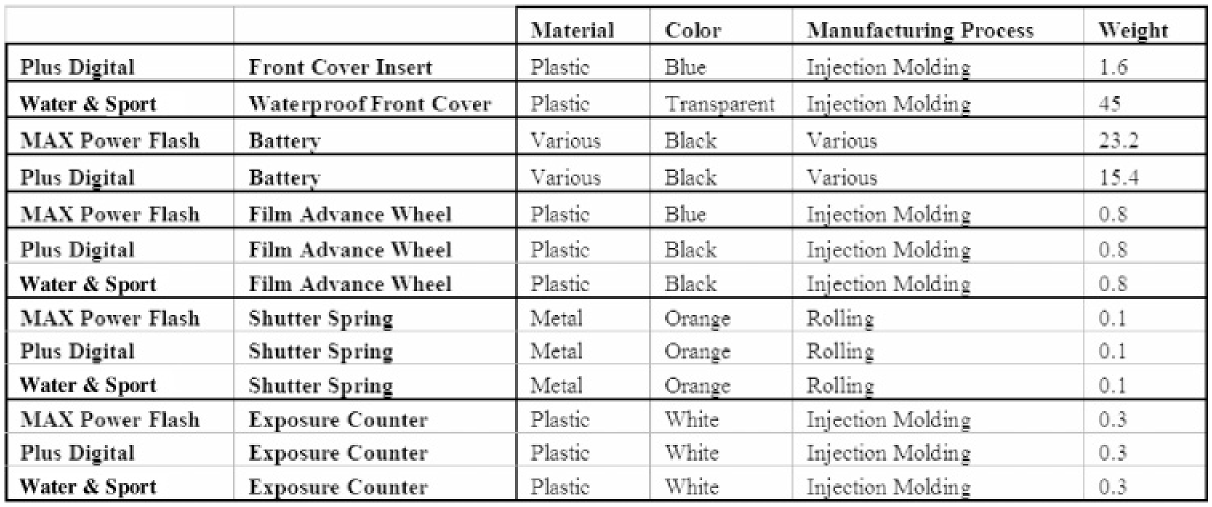

Exposure Counter:has Exposure Counter
Shutter Spring:has Shutter Spring
Film Advance Wheel:has Plus Digital - Film Advance Wheel
Battery:has MAX Power Flash - Battery
Film Advance Wheel:has MAX Power Flash - Film Advance Wheel
Waterproof Front Cover:has Waterproof - Waterproof Front Cover
MAX Power Flash
Waterproof
Front Cover Insert:has MAX Power Flash - Front Cover Insert
Battery:has Plus Digital - Battery
Plus Digital

Fig. 16. The networked bill of material (NBOM) based on the product family lattice structure. [A color version of this figure can be viewed online at www.journals.cambridge.org]



		Material	Color	Manufacturing Process	Weight
Plus Digital	Front Cover Insert	Plastic	Blue	Injection Molding	1.6
Water & Sport	Waterproof Front Cover	Plastic	Transparent	Injection Molding	45
MAX Power Flash	Battery	Various	Black	Various	23.2
Plus Digital	Battery	Various	Black	Various	15.4
MAX Power Flash	Film Advance Wheel	Plastic	Blue	Injection Molding	0.8
Plus Digital	Film Advance Wheel	Plastic	Black	Injection Molding	0.8
Water & Sport	Film Advance Wheel	Plastic	Black	Injection Molding	0.8
MAX Power Flash	Shutter Spring	Metal	Orange	Rolling	0.1
Plus Digital	Shutter Spring	Metal	Orange	Rolling	0.1
Water & Sport	Shutter Spring	Metal	Orange	Rolling	0.1
MAX Power Flash	Exposure Counter	Plastic	White	Injection Molding	0.3
Plus Digital	Exposure Counter	Plastic	White	Injection Molding	0.3
Water & Sport	Exposure Counter	Plastic	White	Injection Molding	0.3

Fig. 17. A product family component cross table.

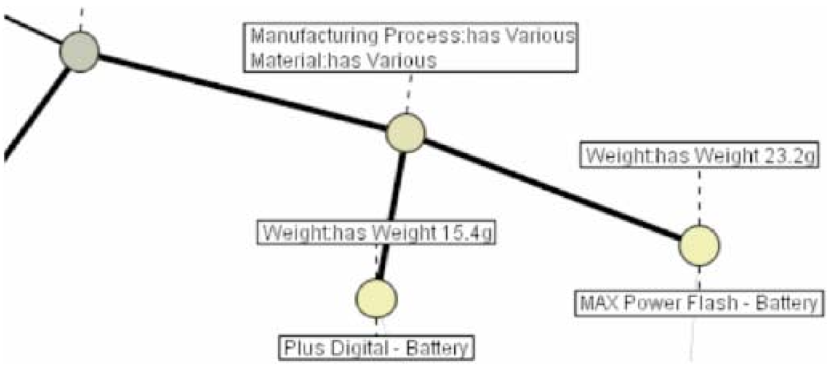


Fig. 18. A component–attribute lattice for batteries in the one-time-use cameras. [A color version of this figure can be viewed online at www.journals.cambridge.org]

popular ontology editing tools as it allows users to construct domain ontologies, customize data entry forms, and enter instances of the ontology (Denny, 2004).

4.2.1. Encoding the NBOM using OWL

The partial order set of the concept lattice developed in Section 4.1.2 is used to develop the subsumption hierarchy of the product family ontology using the PFODM (see Fig. 5). Figure 19 partially shows the ontology structure of the camera family using OWL ontology, which is automatically drawn using ezOWL plug-ins in Protégé-2000. Figure 20 shows the document object model structure of the one-time-use camera product family. The first few lines of the ontology contain the annotation part and specify all the URI references. The complete ontology is available on-line at http://edog.mne.psu.edu/owl/kodak.owl.

As noted in Section 3.2, the PVM captures the relationships between the customer needs and product functions and the relationships between product functions and product components are captured by constructing the FCM. Figure 21 shows the PVM for the Water & Sport camera. For each function that impacts a particular customer need, a "1" is entered into the cell in the matrix; thus, for example, the customer need {(compact design)} is fulfilled by functions ({actuate mechanical energy}, {convert human energy to mechanical energy}, {guide mechanical energy}, {import human energy}). Figure 22 shows the FCM for the Water & Sport camera. Here a "1" is entered into the cell of a matrix for each component that impacts a particular product function. For example, the product function ({guide mechanical energy}) is fulfilled by component ({arm retainer}). These matrices are created for all seven products in the family.

All customer needs, product functions, and information about the components are aggregated in this manner, and a generic ontology is created for the entire product family. After creating the ontologies and the individual instances, the entities between the three types of design information are mapped using PVM and FCM to aggregate information within the family and across different design phases.

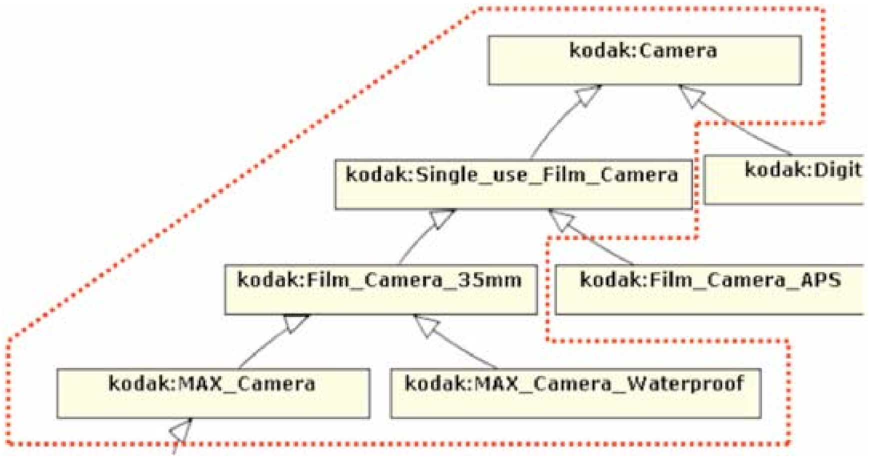
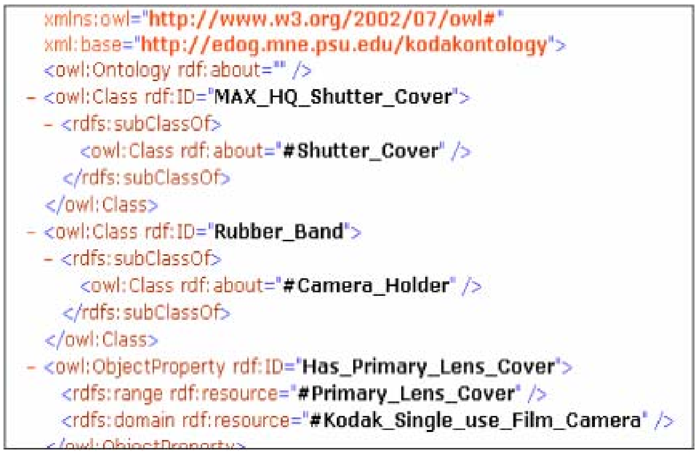


Fig. 19. The camera family ontology. [A color version of this figure can be viewed online at www.journals.cambridge.org]



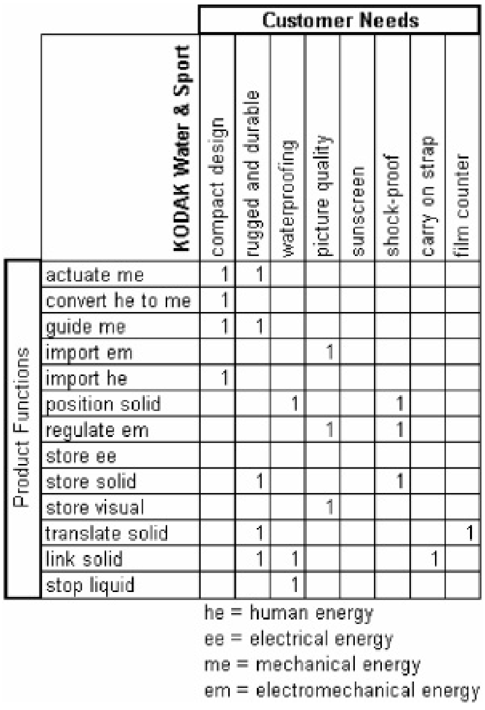
```
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xml:base="http://edog.mne.psu.edu/kodakontology">
   <owl:Ontology rdf:about="" />
 - <owl:Class rdf:ID="MAX_HQ_Shutter_Cover">
   - <rdfs:subClassOf>
       <owl:Class rdf:about="#Shutter_Cover" />
     </rdfs:subClassOf>
   </owl:Class>
 - <owl:Class rdf:ID="Rubber_Band">
   - <rdfs:subClassOf>
       <owl:Class rdf:about="#Camera_Holder" />
     </rdfs:subClassOf>
   </owl:Class>
 - <owl:ObjectProperty rdf:ID="Has_Primary_Lens_Cover">
     <rdfs:range rdf:resource="#Primary_Lens_Cover" />
     <rdfs:domain rdf:resource="#Kodak_Single_use_Film_Camera" />
   </owl:ObjectProperty>
```

Fig. 20. A sample of the Web Ontology Language (OWL) ontology for the cameras. [A color version of this figure can be viewed online at www.journals.cambridge.org]

4.2.2. Information aggregation across cameras and design phases

The common ontological layer in conjunction with PVM and FCM help in automatic design aggregation across phases, and the BOM structure helps design information aggregation across products in a single phase for the designers. For example, let us consider a scenario where the designer is redesigning the {Water & Sport shutter cover}. The system can aggregate all the shutter covers used in the entire product family by first getting the class {Shutter Cover} from which the individual {Water & Sport shutter cover} is an instance and then listing all the other instances of type {Shutter Cover}, that is, ({MAX shutter cover}, {Advantix shutter cover}, {Black and white shutter cover}). The system can also compare the attributes of each shutter cover and present them to the designer. This automatic context-based aggregation of design information can be applied to all of the design entities in every phase of product realization that is represented within the ontology.

The PVM and FCM help aggregate the information across different phases of product design. For instance, the Water & Sport camera customer need {rugged and durable} is satisfied by five product functions ({actuate mechanical energy}, {guide mechanical energy}, {store solid}, {translate solid}, {link solid}). Similarly, the product function {translate solid} is fulfilled by two components ({film advance gear}, {film advance wheel 2}). Combining the common ontologies, PVM, and FCM with a NBOM makes the design information model transparent, flexible, and interoperable across a network of different systems.



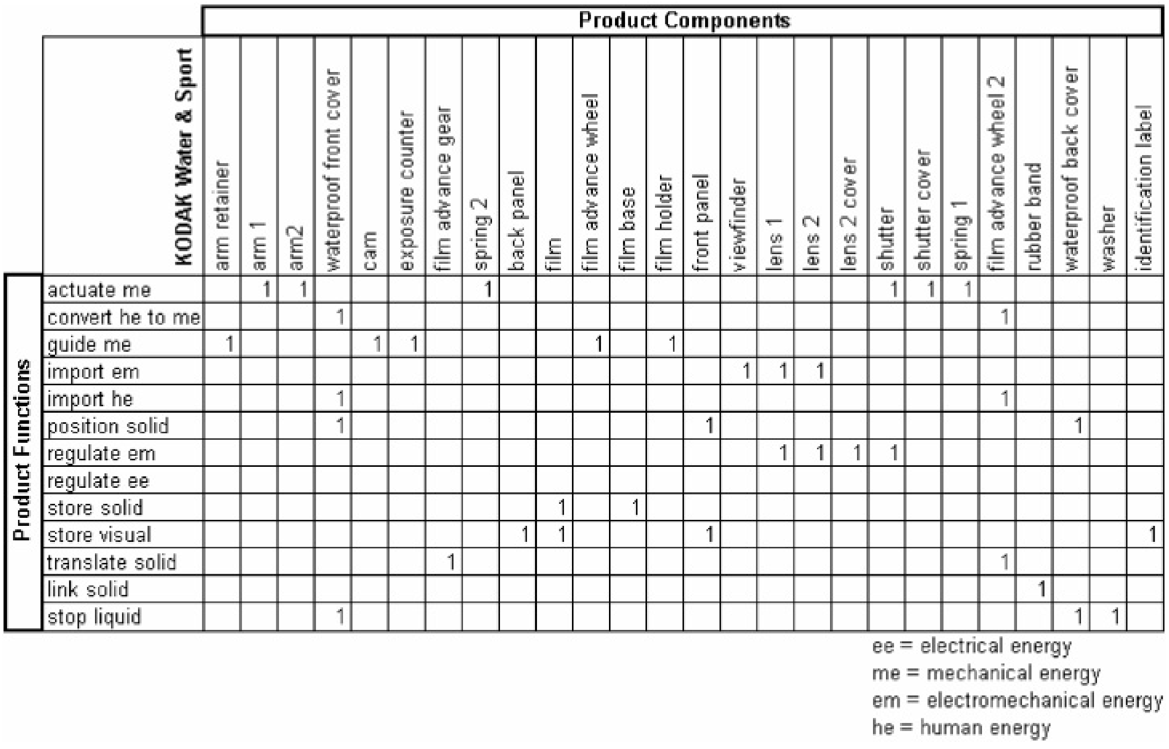
		Customer Needs							
	KODAK Water & Sport	compact design	rugged and durable	waterproofing	picture quality	sunscreen	shock-proof	carry on strap	film counter
Product Functions	actuate me	1	1						
	convert he to me	1							
	guide me	1	1						
	import em				1				
	import he	1							
	position solid			1			1		
	regulate em				1		1		
	store ee								
	store solid		1				1		
	store visual				1				
	translate solid		1						1
	link solid		1	1				1	
	stop liquid			1					

he = human energy
ee = electrical energy
me = mechanical energy
em = electromechanical energy

Fig. 21. The product vector matrix for the Water & Sport camera.

4.2.3. Storing and querying the ontology metalayer

Using the one-time-use camera product family ontology, product designers can explore various types of cameras as well as the components that are part of the camera product family. The one-time-use camera product family ontology can also be queried using semantic queries based on graph pattern matching that can span multiple products across the entire ontology. For example, the RDQL query "SELECT ?classAnyCamera, ?instanceFlash WHERE (?classAnyCamera, ⟨KodakFamily:Has_Flash⟩, ?instanceFlash) USING KodakFamily FOR ⟨http://edog.mne.psu.edu/ontology/kodak#⟩" can query the ontology and list all the cameras and their corresponding flash components (i.e., classes having Has_Flash property) across the product family. In this case, it would return {(Plus_Digital, Flash_Max), (MAX_Flash_Camera, Flash_Max), (MAX_HQ, Flash_Max),



Product Functions	Product Components																									
KODAK Water & Sport	arm retainer	arm 1	arm2	waterproof front cover	cam	exposure counter	film advance gear	spring 2	back panel	film	film advance wheel	film base	film holder	front panel	viewfinder	lens 1	lens 2	lens 2 cover	shutter	shutter cover	spring 1	film advance wheel 2	rubber band	waterproof back cover	washer	identification label
actuate me		1	1					1											1	1	1					
convert he to me				1																		1				
guide me	1				1	1					1		1													
import em															1	1	1									
import he				1																		1				
position solid				1										1										1		
regulate em																1	1	1	1							
regulate ee																										
store solid										1		1														
store visual									1	1				1												1
translate solid							1															1				
link solid																							1			
stop liquid				1																				1	1	

ee = electrical energy
me = mechanical energy
em = electromechanical energy
he = human energy

Fig. 22. The function component matrix for the Water & Sport camera.

(Advantix_Switchable, Flash_APS), (Black_and_White, Flash_Black_and_White)}, which is an array of cameras and their flash components. The semantic query and retrieval is currently being implemented within the UMR Design Repository to facilitate designer-initiated Web-based design exploration. In the next section, further analysis of the camera family is discussed.

4.3. Step 3: Product family analysis and redesign

Once the information is stored in the appropriate format, a product family can be redesigned using either the *component-based approach* or the *product-based approach* described in Section 3.3.2. In this example, only seven components are used for ease of understanding; however, the methodology is scalable to larger problems involving more components and products. The components used here are listed in Figure 23. The products are listed in rows, and the components are listed in columns.

Once all of the concepts are defined, ToscanaJ is used again to automatically generate the concept lattice shown in Figure 24. Note that the nodes associated with objects are larger than the nodes only associated with concepts for ease of visualization. At the top of Figure 24 are the design artifacts that are common throughout the whole product family. In this case, six concepts are identified: *has Arm*, *has Arm retainer*, *has Back panel*, *has Identification label*, *has Shutter*, and *has Shutter cover*. In other words, these six concepts tell us that all the cameras have an arm, an arm retainer, a back panel, an identification label, a shutter, and a shutter cover.

To demonstrate the *product-based approach* to redesign, consider two cameras: the *MAX Flash* and the *Advantix Switchable*. The algorithm presented in Figure 13b first recommends making the *Shutter* common between the two products (by using *Shutter:vs1* in both cameras). In the current design, the two products both use a shutter, but two different instances exist. Implementing these recommendations will remove the concept *Shutter:vs3* and will move the *Advantix Switchable* under the node where *Shutter:vs1* is located.

Meanwhile, the *component-based approach* seeks to reuse critical component instances throughout a product family while giving preference to making variant components common over unique components variant. Consider the *Arm* as an example. We see that all the cameras have an arm (the concept *has Arm* is found in the top node). If we now go down into the graph, two concepts are found for the *Arm*: *Arm:va1* and *Arm:va2*. Below the node where *Arm:va1* is found, we identify six products: *MAX Outdoor*, *MAX Flash*, *MAX HQ*, *Black & White*, *Plus Digital*, and *MAX Water & Sport*, whereas there is only one product below the node *Arm:va2* (*Advantix Switchable*). The recommendation is to change *Arm:va2* to *Arm:va1*, if possible. Note that no other consideration except the algorithm presented in Figure 13a is used here, and other constraints, such as the cost of the

Instances	Arm 1	Arm retainer 1	Back Panel 1	Bottom Cover 1	Identification label 1	Shutter 1	Shutter cover 1
	Arm 2		Back Panel 2		Identification label 2	Shutter 2	Shutter cover 2
			Back Panel 3		Identification label 3	Shutter 3	Shutter cover 3
			Back Panel 4		Identification label 4		
			Back Panel 5		Identification label 5		
					Identification label 6		
					Identification label 7		

	Components						
Products	Arm	Arm Retainer	Back Panel	Bottom Cover	Identification label	Shutter	Shutter cover
MAX Outdoor	1	1	1		1	1	1
MAX Flash	1	1	2		2	1	1
MAX HQ	1	1	2		3	2	2
Plus Digital	1	1	2		4	2	2
Black & White	1	1	3		5	2	2
MAX Water & Sport	1	1	4		6	2	2
Advantix switchable	2		5	1	7	3	3

Fig. 23. The product family components analyzed in this study.

components, could affect the recommendation. Similarly, all the other components can be chosen, and the same algorithm can be applied.

Table 3 shows the results of using the five commonality indices mentioned in Section 3.3.1: DCI, TCCI, CI, PCI, and CMC. For the component-based approach, the focus is on making the arm common throughout the family. For the product-based approach, the shutter is made common between the two cameras concerned. The resulting changes to the commonality indices are noted in the table. Note that all increases are positive, and in many cases identical between the two approaches. This is because most of indices are simply component based, that is, making one additional component common, be it the shutter or the arm, has the same effect regardless of which component is changed; meanwhile, more comprehensive indices such as the CMC

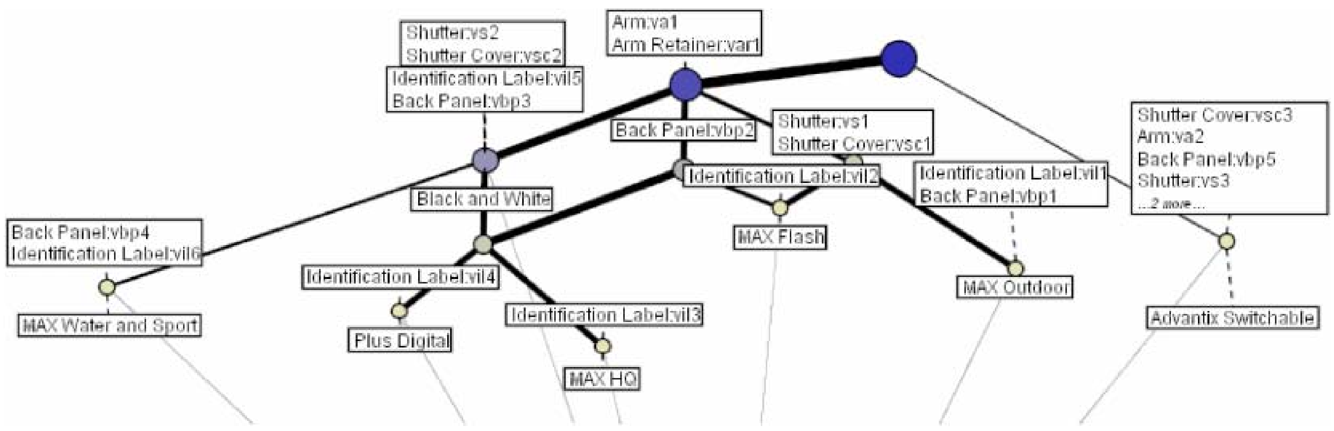


Fig. 24. The concept lattice for the one-time-use camera. [A color version of this figure can be viewed online at www.journals.cambridge.org]

Table 3. *Commonality within camera family before and after redesign recommendations*

	Before Redesign	Product-Based Redesign	Component-Based Redesign
CI	0.568	0.574 (+1.06%)	0.574 (+1.06%)
DCI	1.861	1.880 (+1.02%)	1.880 (+1.02%)
TCCI	0.465	0.4706 (+1.20%)	0.4706 (+1.20%)
PCI	54.000	54.500 (+0.91%)	54.500 (+0.91%)
CMC	0.500	0.512 (+2.37%)	0.539 (+7.73%)

also take the cost of components into account, rewarding commonality decisions based on the cost savings that result (Thevenot & Simpson, 2006*b*).

5. CONCLUSIONS AND FUTURE WORK

Representing product families in a knowledge base by preconceived common ontologies shows promise in promoting component sharing across multiple products in a family, while assisting search and exploration of linguistic and parametric design information over various phases of the product realization process. Unlike unstructured design information, the taxonomic generalization as well as partonomic aggregations captured as part of structured OWL ontologies provides the designer much greater control over the depth and breadth of the semantic graph that needs to be analyzed for product family design decision making. Product vector and function component mapping matrices along with the common ontologies are utilized for designer initiated information exploration, aggregation, and analysis. Use of FCA in the development of the NBOM and application of commonality indices provide a systematic way for redesigning existing product families to increase commonality within a product family. Recent developments with the UMR Design Repository (a PostgrsSQL database used to store the design artifact instances) are also presented. The use of OWL for metadata representation facilitates data access across proprietary software programs and computational platforms. An example involving a family of seven one-time-use cameras is presented to demonstrate implementation of the three steps of the proposed knowledge management framework.

Future work involves incorporating more low-level design features and additional information (e.g., process planning and assembly) into the knowledge management framework. We also plan to exploit the semantic descriptions within the OWL ontologies more, as it is backed by DL (Baader et al., 2003), which makes it easier for computers to interpret. Finally, as the product platform design ontologies grow, we will also explore inference as a tool for automatic design information interpretation as well as the appropriate level of granularity when analyzing large-scale complex systems.

ACKNOWLEDGMENTS

This work was funded by the National Science Foundation under Grants IIS-0325402, IIS-0325321, IIS-0325415, and DMI-0133923. Any opinions, findings, and conclusions or recommendations presented in this paper are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

Baader, F., Calvanese, D., McGuinness, D., Nardi, D., & Patel-Schneider, P. (2003). *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge: Cambridge University Press.

Becker, P. (2004). Numerical analysis in conceptual systems with ToscanaJ. *Concept Lattices: Second Int. Conf. Formal Concept Analysis (ICFCA)*, pp. 96–103, Sydney, Australia.

Bohm, M.R., Stone, R.B., Simpson, T.W., & Steva, E.D. (2006). Introduction of a data schema: the inner workings of a design repository. *ASME Int. Design Engineering Technical Conf. Computers and Information in Engineering Conf.*, Paper No. DETC2006/CIE-99518, Philadelphia, PA.

Bohm, M.R., Stone, R.B., & Szykman, S. (2005). Enhancing virtual product representations for advanced design repository systems. *Journal of Computer Information Science in Engineering 5(4)*, 360–372.

Boothroyd, G., & Dewhurst, P. (2002). *Product Design for Manufacture and Assembly*, 2nd ed., rev. New York: Marcel Dekker.

Bourke, R. (1999). Product information management: product lifecycle management in complex industries, MidRange ERP. Accessed at http://industrydirections.com/midrange/rb0499.htm

Burmeister, P. (2003). *Formal Concept Analysis With CONIMP: Introduction to the Basic Features*. Darmstadt, Germany: Darmstadt University of Technology, Department of Mathematics.

Clark, K.G., Ed. (2005). *SPARQL protocol for RDF*. World Wide Web Consortium. Accessed at http://www.w3.org/TR/2005/WD-rdf-sparql-protocol-20050527/

Collier, D.A. (1981). The measurement and operating benefits of component part commonality. *Decision Sciences 12(1)*, 85–96.

Daconta, M.C., Obrst, L.J., & Smith, K.T. (2003). *The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management*. Indianapolis, IN: Wiley.

Davies, J., Fensel, D., & van Harmelen, F. (2003). *Towards the Semantic Web: Ontology-Driven Knowledge Management*. West Sussex: Wiley.

Denny, M. (2004). *Ontology Tools Survey, Revisited*. O'Reilly. XML.com. Accessed at http://www.xml.com/pub/a/2004/07/14/onto.html

Dixon, J.R., & Poli, C. (1999). *Engineering design & design for manufacturing: A structured approach*. Conway, MA: Field Stone Publishers.

Douglas, K., & Douglas, S. (2003). *PostgreSQL*, 1st ed. Indianapolis, IN: Sams.

Duquenne, V., Chabert, C., Cherfouh, A., Delabar, J.-M., Doyen, A.-L., & Pickering, D. (2001). Structuration of phenotypes/genotypes through Galois lattices and implications. *Int. Workshop on Concept Lattice-Based Theory, Methods and Tools for Knowledge Discovery in Databases*, Stanford University, Stanford, CA.

Erné, M., Koslowski, J., Melton, A., & Strecker, G.E. (1991). A primer on Galois connections. *Proc. Summer Conf. General Topology and Applications in Honor of Mary Ellen Rudin and Her Work*, pp. 103–125, Madison, WI.

Fernandez-Manjon, B., & Fernandez-Valmayor, A. (1998). Building educational tools based on formal concept analysis. *Education and Information Technologies 3(3)*, 187–201.

Fikes, R., Hayes, P., & Horrocks, I. (2003). *OWL-QL—A Language for Deductive Query Answering on the Semantic Web*. Palo Alto, CA: Stanford University, Knowledge Systems Laboratory.

Ganter, B., & Wille, R. (1999). *Formal Concept Analysis: Mathematical Foundations*. Heidelberg: Springer–Verlag.

Gennari, J.H., Tu, S.W., Rothenfluh, T.E., & Musen, M.A. (1994). Mapping domains to methods in support of reuse. *International Journal of Human–Computer Studies 41(3)*, 399–424.

Godin, R., & Mili, H. (1993). Building and maintaining analysis-level class hierarchies using Galois lattices. *Proc. Eighth Annual Conf. Object-Oriented Programming Systems, Languages, and Applications*, pp. 394–410, Washington, DC.

Godin, R., Missaoui, R., & April, A. (1993). Experimental comparison of navigation in a galois lattice with conventional information retrieval methods. *International Journal of Man–Machine Studies 38(5)*, 747–767.

Gratzer, G.A. (1998). *General Lattice Theory*. Boston: Birkhäuser.

Groh, B., Strahringer, S., & Wille, R. (1998). Toscana-systems based on thesauri, conceptual structures: theory, tools and applications. *6th Int. Conf. Conceptual Structures (ICCS'98)*, pp. 127–138, Montpellier, France.

Hatvany, J., Newman, W.M., & Sabin, M.A. (1993). World survey of computer-aided design. *Computer Aided Design 25(12)*, 776–798.

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

Iwasaki, Y., & Chandrasekaran, B. (1992). Design verification through function and behavior-oriented representations: bridging the gap between function and behavior. *Second Int. Conf. Artificial Intelligence in Design*, Pittsburgh, PA.

Kirschman, C., & Fadel, G.M. (1998). Classifying functions for mechanical design. *ASME Journal of Mechanical Design 120(3)*, 475–482.

Koivunen, M.-R., & Miller, E. (2001). *W3C Semantic Web activity*. World Wide Web Consortium. Accessed at http://www.w3.org/2001/12/semweb-fin/w3csw

Kota, S., Sethuraman, K., & Miller, R. (2000). A metric for evaluating design commonality in product families. *ASME Journal of Mechanical Design 122(4)*, 403–410.

Kurtoglu, T., Campbell, M.I., Bryant, C.R., Stone, R.B., & McAdams, D.A. (2005). Deriving a component basis for computational functional synthesis. *Int. Conf. Engineering Design (ICED05)*, Melbourne, Australia.

Kuznetsov, S.O. (2001). Machine learning on the basis of formal concept analysis. *Automation and Remote Control 62(10)*, 1543–1564.

Liao, S.-H. (2003). Knowledge management technologies and applications—literature review from 1995 to 2002. *Expert Systems with Applications 25(2)*, 155–164.

Lindig, C. (1995). Concept-based component retrieval. *Int. Joint Conf. Artificial Intelligence: Formal Approaches to the Reuse of Plans, Proofs, and Programs (IJCAI-95)*, Montreal, Canada.

Martin, M.V., & Ishii, K. (1997). Design for variety: development of complexity indices and design charts. *1997 ASME Design Engineering Technical Conf. Design for Manufacturability*, Paper No. DETC97/DFM-4359, Sacramento, CA.

McBride, B. (2002). Jena: a semantic web toolkit. In *IEEE Internet Computing*, pp. 55–59.

McGuinness, D.L., & van Harmelen, F. (2004). *OWL web ontology language overview. Recommendation*. World Wide Web Consortium. Accessed at http://www.w3.org/TR/2004/REC-owl-features-20040210/

Miller, L., Seaborne, A., & Reggiori, A. (2002). *Three Implementations of SQUISHQL. A Simple RDF Query Language*. Bristol: Hewlett–Packard, Information Infrastructure Laboratory.

Murdock, J.W., Szykman, S., & Sriram, R.D. (1997). An information modeling framework to support design databases and repositories. *ASME Design Engineering Technical Conf.*, Paper No. DETC97/DFM-4373, Sacramento, CA.

Nanda, J., Simpson, T.W., Shooter, S.B., & Stone, R.B. (2005*a*). A unified information model for product family design management. *ASME Int. Design Engineering Technical Conf. Computers and Information in Engineering Conf.*, Long Beach, CA.

Nanda, J., Thevenot, H., & Simpson, T.W. (2005*b*). Product family representation and redesign: increasing commonality using formal concept analysis. *ASME Int. Design Engineering Technical Conf. Computers and Information in Engineering Conf.*, Paper No. DETC2005/DAC84818, Long Beach, CA.

Nanda, J., Thevenot, H.J., & Simpson, T.W. (2005*c*). Product family design knowledge representation, integration, and reuse. *IEEE Int. Conf. Information Reuse and Integration*, pp. 32–37, Las Vegas, NV.

Nanda, J., Simpson, T.W., Kumara, S.R.T., & Shooter, S.B. (2006). Product family ontology development using formal concept analysis and web ontology language. *ASME Journal of Computing and Information Science in Engineering 6(1)*, 103–113.

Nourine, L., & Raynaud, O. (1999). Fast algorithm for building lattices. *Information Processing Letters 71(5–6)*, 199–204.

Noy, N.F., Sintek, M., Decker, S., Crubézy, M., Fergerson, R.W., & Musen, M.A. (2001). Creating semantic web contents with Protégé-2000. *IEEE Intelligent Systems 16(2)*, 60–71.

Otto, K.N., & Wood, K.L. (2001). *Product Design: Techniques in Reverse Engineering and New Product Development*. Upper Saddle River, NJ: Prentice–Hall.

Priss, U. (2003). Linguistic applications of formal concept analysis. *Proc. First Int. Conf. Formal Concept Analysis (ICFCA'03)*, Darmstadt, Germany.

Saaksvuori, A., & Immonen, A. (2003). *Product Lifecycle Management*. Heidelberg: Springer–Verlag.

Shooter, S.B., Keirouz, W.T., Szykman, S., & Fenves, S.J. (2000). A model for the flow of design information in product development. *Journal of Engineering with Computers 16(3–4)*, 178–194.

Shooter, S.B., Simpson, T.W., Kumara, S.R.T., Stone, R.B., and Terpenny, J.P. (2005). Toward a multi-agent information infrastructure for product family planning and mass customization. *International Journal of Mass Customization 1(1)*, 134–155.

Siff, M., & Reps, T. (1997). Identifying modules via concept analysis. *IEEE Int. Conf. Software Maintenance*, pp. 170–179, Bari, Italy.

Simpson, T.W. (2004). Product platform design and customization: status and promise. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 18(1)*, 3–20.

Szykman, S., Racz, J., Bochenek, C., & Sriram, R.D. (2000). Web-based system for design artifact modeling. *Design Studies 21(2)*, 145–165.

Thevenot, H.J., & Simpson, T.W. (2006*a*). Commonality indices for product family design: a detailed comparison. *Journal of Engineering Design 17(2)*, 99–119.

Thevenot, H.J., & Simpson, T.W. (2006*b*). A comprehensive metric for evaluating commonality in a product family. *ASME Int. Design Engineering Technical Conf. Computers and Information in Engineering Conf.*, Paper No. DETC2006-DAC99268, Philadelphia, PA.

Tilley, T. (2004). Tool support for FCA, concept lattices. *Second Int. Conf. Formal Concept Analysis (ICFCA)*, pp. 104–111, Sydney, Australia.

Ulrich, K.T., & Eppinger, S.D. (2004). *Product Design and Development*. New York: McGraw–Hill/Irwin.

Valtchev, P., Missaoui, R., & Lebrun, P. (2002). A partition-based approach towards constructing Galois (concept) lattices. *Discrete Mathematics 256(3)*, 801–829.

van der Vegte, W.F., Kitamura, Y., Mizoguchi, R., & Horváth, I. (2002). Ontology-based modeling of product functionality and use—part 2: considering use and unintended behavior. *Proc. EdiPROD Conf.*, pp. 115–124.

Vogt, F., & Wille, R. (1995). Toscana—a graphical tool for analyzing and exploring data. *Proc. DIMACS Int. Workshop on Graph Drawing*, pp. 226–233, Princeton, NJ.

Wacker, J.G., & Trelevan, M. (1986). Component part standardization: an analysis of commonality sources and indices. *Journal of Operations Management 6(2)*, 219–244.

Jyotirmaya Nanda is a Research Scientist at Intelligent Automation, Inc., Rockville, MD. He obtained his his MS degree in industrial engineering from Penn State University in 2002, his BE in mechanical engineering from Visvesvaraya National Institute of Technology, Nagpur, India, in 1998, and his doctorate in industrial engineering with a minor in high performance computing at Penn State University with Dr. Timothy W. Simpson in 2006. Dr. Nanda's research interests are in mass customization, knowledge management using ontologies, autonomic agent systems, and design optimization.

Henri J. Thevenot is a Fixed-Term Instructor and a Post-doctoral Research Associate working in product family design at Penn State University. He received his MS degree in industrial engineering from both Ecole Centrale de Lyon (France) and Penn State University in 2004 and his PhD degree in industrial engineering from Penn State University in 2006. Dr. Thevenot is currently developing tools and methods for product family design and redesign in the context of globalization and innovation. He is also interested in developing tools and methods to help designers in the early stage of the product design process.

Timothy W. Simpson is a Professor of mechanical and industrial engineering and engineering design and the Director of the Product Realization Minor at Penn State University. He received his BS (1994) in mechanical engineering from Cornell University and his MS (1995) and PhD (1998) degrees in mechanical engineering from Georgia Tech. His research and teaching interests include product family and product platform design, product dissection and concurrent engineering, and visualization methods. Dr. Simpson is a member of the ASME Design Automation Executive Committee and the AIAA Multidisciplinary Design Optimization Technical Committee.

Robert B. Stone is an Associate Professor in the Interdisciplinary Engineering Department and the Director of the Student Design and Experiential Learning Center at the University of Missouri–Rolla. He joined the faculty in 1998 after completing his PhD in mechanical engineering from the University of Texas at Austin. Prior to academia, he worked as a Space Shuttle Flight Controller at NASA-Johnson Space Center. He assisted in creating the design-focused interdisciplinary engineering degree program. Dr. Stone's research interests include design theories and methodologies, specifically product architectures, functional representations, and automated conceptual design techniques. He has authored chapters on product architecture in design texts.

Matt Bohm is currently a PhD student. He is performing research with the Interdisciplinary Engineering Department and is a student in the Systems Engineering Department. He joined the Interdisciplinary Engineering Department as an undergraduate Researcher in December 2001 while working on a bachelors degree in mechanical engineering. After finishing his bachelors degree he continued performing research in the area of conceptual design and design information storage while working toward a masters degree in mechanical engineering.

Steven B. Shooter is an Associate Professor of mechanical engineering at Bucknell University where he has taught since 1995. He is a registered professional engineer in the state of Pennsylvania and has been the principal investigator on numerous projects with industry involving new product development and the design of production infrastructure. Dr. Shooter has been a Researcher at NIST in the Design Engineering Technologies Group, a Visiting Professor at the Swiss Federal Institute of Technology in Lausanne (EPFL), and a Process Engineer for Sony Music Corporation. His research interests involve information management for design and the design of mechatronic systems and products. Integral to this research is the exploration of approaches for the capture, storage, and retrieval of product development information.