International Journal of Production Research,
Vol. 44, No. 14, 15 July 2006, 2813–2837

Taylor & Francis
Taylor & Francis Group

DAKA: design activity knowledge acquisition through data-mining

Y. JIN*† and Y. ISHINO‡

†Department of Aerospace and Mechanical Engineering, University of Southern California, 3650 McClintock Avenue, OHE-430, Los Angeles, CA 90089-1453, USA
‡Graduate School of Science, Center for Quantum Life Sciences, Hiroshima University, 1-3-1 Kagamiyama, Higashi-Hiroshima, Hiroshima 739-8530, Japan

(*Revision received February 2006*)

Engineering knowledge is an important asset of industrial companies. The present research focuses on design activity knowledge and it attempts to develop effective ways to capture this operational knowledge from the design events monitored during the design process. A design activity is defined as a sequence of meaningful design operations carried out by designers to advance the design from its current state to the new state. The paper proposes a design activity knowledge acquisition (DAKA) framework that extracts designers' design activity knowledge from the computer-aided design (CAD) operation event data obtained through commercial CAD systems. DAKA is composed of a product model roadmap for capturing the trajectory of designers' design moves and a function-based design operation-mining algorithm for extracting meaningful design operations from CAD event databases. DAKA has been evaluated through case studies using real CAD operation event data as well as computer-generated synthetic data. In this paper, the details of the DAKA framework are described and a case example involving automotive door design is presented to demonstrate the effectiveness of the proposed approach.

Keywords: Knowledge acquisition; Design activity; Design operations; Data-mining; Product model

1. Introduction

Engineering design is knowledge-intensive. Designers must be experienced enough to understand the design problem, have sufficient domain knowledge to find solutions and possess expertise to know what design decisions to make and which design operations to take. In the current engineering practice where computer-aided design (CAD) tools are widely used, although extensive knowledge is applied during the design process, neither the knowledge applied nor the ways to apply the knowledge is left in a sharable form after the design is completed. The only result is the CAD drawings that describe only the static features of the designed product.

Not being able to capture design activity knowledge — i.e. what design operations were carried out, which part of the design requires more effort and careful thinking, etc. — during design leads to difficulties for design management, design

*Corresponding author. Email: yjin@usc.edu

International Journal of Production Research
ISSN 0020–7543 print/ISSN 1366–588X online © 2006 Taylor & Francis
http://www.tandf.co.uk/journals
DOI: 10.1080/00207540600654533

training and design efficiency. In the old days when drawings were produced by using papers and pencils rather than computers, a design manager could walk to a drawing table and tell whether there were problems, and which part of the design had problems, by identifying the places of the drawing that had repeating erasing traces. Working on a large drawing table, as many automotive designers did in the past, made it easier for the designers to collaborate and make sure their designs work together properly at the end. Furthermore, new designers could learn about design process by observing designers' design activities of drawing, erasing and talking.

As design problems become more complex, design can hardly be accomplished without using CAD tools. While applying CAD tools increases the quality and efficiency of design for complex design problems, the individual-oriented and result-driven nature of CAD tools has made CAD-based design a 'hidden process' that cannot be observed. Efforts have been made to record CAD operation events as the history of design process. However, the vast amount of CAD event data alone cannot provide any understandable and useful design process knowledge. Computer tools can be provided to allow designers to record their design knowledge explicitly, but this approach is not desirable because: it takes designers' time away from design, it interrupts designers' normal thinking process, and it is often impossible for designers to articulate their tacit knowledge explicitly. There is a need for technologies that can acquire design knowledge without direct involvement from the designers.

Design knowledge ranges from design product knowledge and design rationale knowledge to design activity knowledge and design process (or workflow) knowledge. While researchers in engineering and artificial intelligence have focused their knowledge acquisition research on documented domain knowledge (Bradshaw *et al*. 1997), research on capturing design activity knowledge received little attention, partly because design activity knowledge is often domain-specific and it is difficult to develop a generally useful approach.

The research on design rationale has explored various directions (Moran and Carroll 1996) including argumentation-based design rationale such as the Issue-Based Information System (IBIS) (Kunz and Rittel 1970), action-based design rationale (Lakin *et al*. 1989), and model-based design rationale such as the Active Design Document (ADD) system (Garcia and Howard 1992). Recently, many pursued the integration of the record of design rationale with the history of the evolution of the design artefact (Banares-Alcantara and King 1997, Shipman and McCall 1997, Liang *et al*. 1999, Shah *et al*. 2000). In the field of design rationale, the research focus is on capturing *why* a specific design feature was designed the way it is, rather than *how* it was designed. Moreover, most extent design rationale capture systems require designers to record reasons during the design process.

Some researchers attempted to capture design process knowledge as part of design rationale. Ganeshan *et al*. (1994) proposed a framework to model design as selections from predefined transformation rules. When a rule is selected, the choice is recorded along with rationale associated with that rule. In their approach, designers' activities are constrained and they are translated into the predefined rules beforehand. Myers *et al*. (1999) proposed a framework to capture design rationale from a general CAD database. They developed an experimental system, the Rationale Construction Framework (RCF), which automatically acquires rationale

information from the detailed design process. In RCF, they regarded design history as a conglomerate of detailed design rationales and focused on capturing many partly isolated design rationales in detail. In RCF, simple pattern matching is executed to detect design procedure using general predefined rules called design metaphors and qualitative reasoning is used to capture design rationale. Focusing on capturing know-how knowledge and design procedure features, rather than know-why knowledge, Ishino and Jin (2001, 2002) proposed a Grammar and Extended Dynamic Programming Approach (GEDP). The core idea of GEDP was to model a design process as a series of meaningful clusters of design events, called design operations. In this approach, designers' activities were constrained and translated into design operations using the predefined classification rules called grammar rules and EDP rules. All methods mentioned above were useful to reach their goals, respectively. However, they all needed predefined rules used for classification of recorded events. Therefore, these methods are limited by high cost of creating predefined rules.

In addition, some of the industrial Case Based Reasoning (CBR) systems can also be considered as capturing design process knowledge. Several recent publications have addressed specific issues of applying CBR to design (Maher and Pu 1997). Goel *et al*. (1997) directly addressed the representation of causal behaviour through the use of structure–behaviour–function (SBF) models. SBF models are useful to diagnose products. However, they focused on states of product models rather than design activities.

In the field of data mining, various techniques and algorithms have been studied to capture knowledge possessing high interestingness. The number of patterns generated in a process is usually very large and only a few of the patterns are likely to be of any interest to the domain expert analysing the data. To increase the utility, relevance, and usefulness of the discovered patterns, techniques are required to reduce the number of patterns that need to be considered. These techniques are referred to as interestingness measures in a data-mining domain (Hilderman and Hamilton 1999). To date, many techniques have been published such as Piatetsky-Shapiro's (1991) rule-interest function, Smyth and Goodman's (1991) *J*-Measure, Agrawal and Srikant's (1994) itemset measures, and Gray and Orlowska's (1998) interestingness. There have also been many application studies of the techniques in many fields such as marketing (Agrawal and Srikant 1994) and medical domains (Matheus and Piatetsky-Shapiro 1996, Ohsaki *et al*. 2004). For example, Piatetsky-Shapiro (1991) introduced his rule-interest function that was used to quantify the correlation between two attributes arbitrarily chosen, and the obtained knowledge is expressed as classification rules. And later Piatetsky-Shapiro and Matheus proposed Key Findings Reporter (KEFIR), which can perform an automatic drill-down through data and discover key findings in a database, and they tested it using healthcare data (Matheus and Piatetsky-Shapiro 1996). Hilderman and Hamilton's (1999) survey paper summarized major studies on the interestingness measure. Almost all such techniques focused on the correlation or deviation among attributes or components, and the acquired knowledge was expressed as classification rules or association rules. These techniques did not deal with the time series sequential patterns of the components.

Data mining has been applied at various stages of product development process, from marketing (Shaw *et al*. 2001) to design and manufacturing

(Kusiak 2000, Braha 2001). Berry and Linoff (1997) presented many examples and applications of data mining in marketing, sales, and customer support. Shaw *et al.* (2001) proposed a systematic methodology that uses data mining and knowledge management techniques to manage the marketing knowledge and support marketing decisions. Létourneau *et al.* (1999) developed a data-mining-based approach to support aircraft maintenance by predicting the problems of a variety of aircraft components based on collected aircraft operation data. Kusiak and his colleagues have led the research on data-mining technologies for developing and improving manufacturing processes (Kusiak 2000a, 2000b, 2002, 2005, Agard and Kusiak 2004a, b, Da Cunha *et al.* 2005). Agard and Kusiak (2004a) developed a data-mining-based technology to support design of product families that addresses the issues of maintaining adequate product diversity and reducing the time and cost of production. In addition, they also devised a model and an algorithm for the selection of subassemblies based on the analysis of prior orders received from the customers (Agard and Kusiak 2004b). Using the information extracted from production history, Da Cunha *et al.* (2005) developed a data-mining approach to determine the sequence of assemblies that minimize the risk of producing faulty products. To support effective configurations for cellular manufacturing systems, Chen (2003) developed a data-mining technique that can find association rules among machines from the process database so that the occurrence of some machines in a machine cell will cause the occurrence of other machines in the same cell. Neaga and Harding (2005) took a knowledge-discovery and data-mining approach to enterprise modelling and integration and suggested utilizing the existing reference architectures, models and integrations for developing a common knowledge enterprise model as novel combination of the previous ones.

The goal of the present research is to develop a generally applicable technology that can acquire design activity knowledge without distracting designers from their normal design processes. The paper takes a data-mining approach and attempts to extract design activity knowledge from automatically recorded vast amount of CAD event data.

The rest of the paper is organized as follows. Section 2 describes the DAKA framework and a general knowledge application model that provide a basis for event based design activity knowledge acquisition. Sections 3 and 4 describe DAKA's two core components: product model roadmap and knowledge acquisition algorithm, respectively. Section 5 presents a case example; and concluding remarks are drawn in section 6.

2. Design activity knowledge acquisition

To avoid requiring direct involvement of designers, our design activity knowledge acquisition (DAKA) framework is composed of five major steps, as shown in figure 1:

- Monitor designers' design actions.
- Store the monitored event data of each designer into corresponding task clusters, such as door design or instrument panel design.

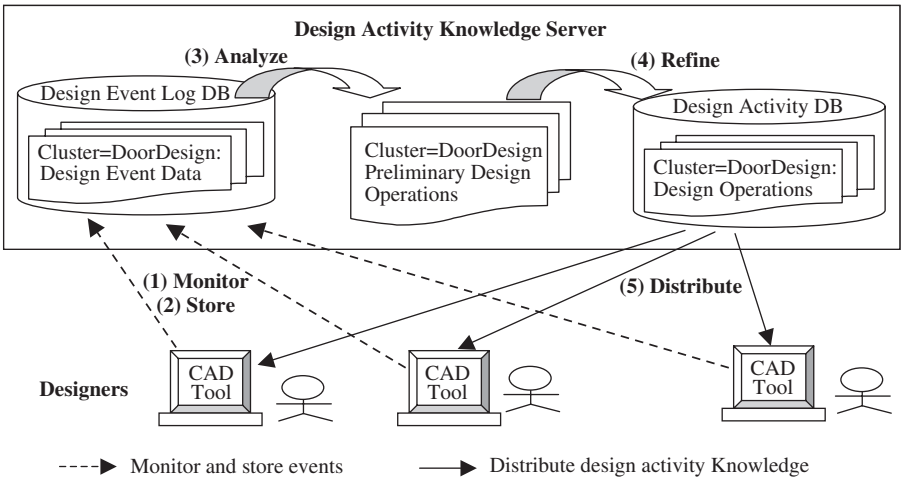


Figure 1. Design activity knowledge acquisition (DAKA) framework.

- Analyse data of each task cluster to mine design activities for the corresponding design task.
- Refine the obtained design activities by checking their meanings.
- Distribute the design activity knowledge for reuse.

These five steps are continually reiterated, while the data and design activity knowledge are updated.

In the *monitoring* step, a wrapper module, which translates designers' CAD operations into numerical predefined codes, enables one to monitor designers' design events, such as adding an object, changing the value of a parameter, and deleting an attribute or an object. The wrapper plugged into the CAD systems reports pairs of designer's design action and its corresponding resulting state of the design object in real time. In the *storing* step, the data transmitted by the wrapper are stored in the task clusters, characterized by design task IDs, of design event log database. In the *analysis* step, data of the same task cluster in the design event log database are analysed to extract preliminary design operations. In the *refining* step, the obtained preliminary design activities knowledge is examined by human expert designers. If needed the experts can modify or delete any of them. After the refinement, finally the resulting design activity knowledge is registered in a design activity knowledge database, which is called Design Activity DB in figure 1. Lastly, in the *distribution* step, the obtained design activity knowledge can be delivered and shown at designers' requests. As can be seen from figure 1, monitoring and storing are carried out at real time during design, while analysing and refining are processed in a batch mode after the complete event data become available.

To realize the DAKA framework described above, we need to model what are design activities, design operations, and design events. We also need knowledge acquisition algorithms that operate based on the model. In the following we first introduce the concepts of knowledge application unit and product model roadmap, and then present our function-based design operation-mining algorithm.

2.1 *Design knowledge application unit*

In this research, we consider design knowledge in four different categories: design rationale knowledge (i.e. reasons for making a specific design move), design activity knowledge (i.e. steps or operations taken to make design moves), design product knowledge (i.e. current and historical design product information), and design process knowledge (i.e. the overall workflow for creating a design). In the following section we introduce a general design knowledge application model to illustrate the definitions of these four types of design knowledge.

Let ***Kr*** denote the set of total design rationale involved, ***Pd*** the set of possible design product states (i.e. states of product model), and ***Av*** the set of design activities, i.e. meaningful design operations. We introduce a concept called knowledge application unit (KAU) to represent specific design moves.

Definition 1 (Knowledge application unit): a knowledge application unit denoted by $u_i \in \mathbf{U}$ is defined as:

$$u_i := \text{apply}\{Kr_i, Av_i, Pd_i\} => Pd_i \rightarrow Pd_{i+1},$$

where

$$Kri \in \boldsymbol{Kr},\ Av_i \in \boldsymbol{Av},\ Pd_i \in \boldsymbol{Pd},\ Pd_{i+1} \in \boldsymbol{Pd}.$$

By definition, a knowledge application unit, u_i, is a unit of action that involves application of rationale knowledge Kr_i, specification of design activity Av_i, and, as a result, the design move from product development state Pd_i to state Pd_{i+1}, as shown in figure 2.

Given a set of KAU $\boldsymbol{U}$, let $\boldsymbol{R}$ denote the set of relations between the KAUs in $\boldsymbol{U}$, i.e. $\forall r_k \in \mathbf{R}$, $\exists u_i \in \mathbf{U}$, $\exists u_j \in \mathbf{U}$ $(i \neq j) => r_k \rightarrow (u_i, u_j)$.

Definition 2 (Design process): a design process Pc is defined as:

$$Pc := \{\mathbf{U}, \mathbf{R}\} = \{u_1, u_2, \ldots, u_M, r_1, r_2, \ldots, r_N\},$$

where

$$u_i \in \mathbf{U}(0 < i \leq M), \text{ and } r_j \in \mathbf{R}(0 < j = N).$$

The above definition of KAU has important implications. First, it separates design rationale knowledge from the design activity knowledge. Every design move or progress is made possible by a design activity and a design activity is carried out under the guidance of specific rationale or reasons. Although it is usually difficult to capture design rationale of a designer without the direct involvement of the designer, it is possible to observe what the designer does and further capture or infer what are

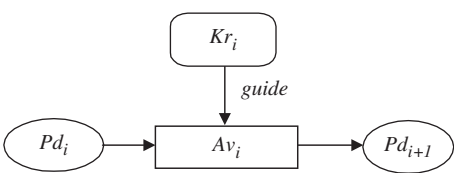


Figure 2. Knowledge application unit (KAU).

the intended operations. We argue that capturing this type of design activity knowledge is important for understanding the design behaviour of experienced designers and for reasoning about their design rationales. Second, the KAU definition indicates that in order to capture design activity knowledge and/or design rationale knowledge one must keep tracking the changes of the design product states. Matching design activities, product states and design rationale in terms of KAU provides a convenient way to organize design information and knowledge. In our research, the goal is to capture design activity knowledge based on the recorded CAD events and product state changes. The following section elaborates the concept of KAU to include design operations and events.

2.2 *Design operations and events*

In order to acquire design activity knowledge, we must first define what is design activity knowledge. Generally speaking, design activity knowledge is related to 'how to move the design product from its current state to the next state'. In our research, we view a design activity as composed of a sequence of *design operations* that reflect meaningful design actions such as 'decrease the weight of the object' and 'increase the strength of the robot arm'. If we can capture the design operations carried out by the experienced designers when they make a design move from one product state to another, then we will have the activity knowledge of that specific KAU. If we can capture all relevant KAUs of the overall design process of a relatively routine design, then we can say that we have good knowledge of how to do that design. In order to capture the change of product states and the operations carried out for making the design moves, we introduce the following definitions.

Definition 3 (Design event): a design event, denoted by e_i, associated with a knowledge application unit u_i is an observable happening that contributes to, but does not directly cause, the product state transition from Pd_i to Pd_{i+1}.

Design events are the most primitive design actions generated by designers while operating a CAD system. Design events can be recognized by observing changes of design variables and/or accesses to documents or databases. They usually do not reflect designers' intention. For example, 'Change length A from 15 to 30' or 'See document B' are design events. Many CAD tools provide mechanisms to automatically record design events.

Given the definitions of design activity and design event, we can define operations as follows:

Definition 4 (Design operation): a design operation, denoted by op_i, is defined as a *sequential pattern* of design events $\{e_1, e_2, \ldots\}$ that reflect a meaningful design action:

$$op_i := \{e_1, e_2, \ldots\},$$

where

$$\text{for } \forall\ op_i,\ \exists\ Av_i, op_i, \in Av_i.$$

Operations are meaningful in the sense that they are carried out to create or edit an *attribute* of the product being designed. For example, 'Decrease the weight of the object' and 'Increase the strength of the robot arm' are both design operations.

A design activity that moves design from one product state to another is composed of a sequence of design operations that are meaningful with respect to the intended design goals.

Based on the above definitions, the problem of capturing design activity knowledge can be decomposed into two sub-problems. First, we must be able to identify product states and their transitions so that we can identify what are the design activities that cause the transitions. Second, we must provide mechanisms to extract sequential patterns of design events so that the design operations within each design activity can be identified. In the following section, we introduce a product model roadmap to represent product states and their transitions during design.

3. Product model roadmap

During design, the design product states change over time. To capture design activity knowledge, we must first capture the changes of the product states. Since a product state at a given time can be represented by the product model at that time, we introduce a 'product model roadmap' to capture 'where' in the design space did a designer 'walk' into and how did he/she 'arrive' at the final 'destination'. Figure 3 illustrates such a product model roadmap.

A product model is composed of a set of objects of which each has various attributes representing physical and conceptual features of the product being designed. In DAKA, the product model roadmap is defined as a time-series tree of product models in which a newly developed product model often becomes a child of one of the existing leaf product models. A child product model can be derived from its parent by creating or modifying one or more major features. Sometimes, a new product model can become a new root. Figure 3 illustrates an example of the product model roadmap, in which the designer created a new product model Pd_3 by adding (or modifying) some features. After deciding Pd_3 would not lead to desired result, the designer moved back to Pd_2 and started creating (or modifying) some other features and moved the design to Pd_4.

Creating a product model roadmap from monitored CAD event data involves identifying formations of individual product models and then determining their parent–child relations. One simple way to do this is to use file version management system. For parametric design problems, however, a more sophisticated method is needed for generating a product model roadmap since designers tend to change

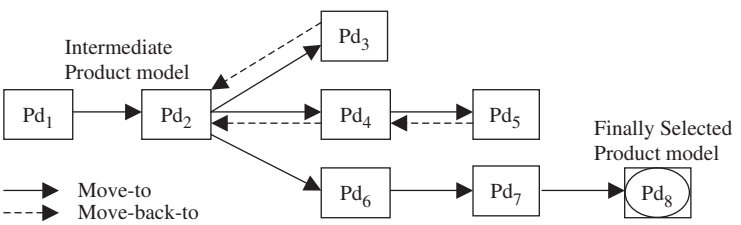


Figure 3. Design activity roadmap.

product models without saving current CAD data. We developed the following two-step approach:

Step 1: Identify formations of individual product models.

First, we introduce the concept of product model core to represent the current state of a product model being developed. A product model core is an essential part of its corresponding product model and is defined by a set of a few core parameters of the product model. Adding or updating the parameters of the product model core implies significant design moves.

Next, the formation, or birth, of a product model is judged as follows:

- An initial product model is generated from an empty or incomplete product structure — i.e. a product model with parameters having null values — by assigning non-null values to all the design parameters.
- A derived product model is created when a design change that causes changes in the product model core is made on an initial or derived product model.

Based on these definitions, we can determine when a product model is formed by monitoring the changes of the core parameters of the product model. For relatively routine design problems such as automotive design, designers can identify what are core parameters for given design products.

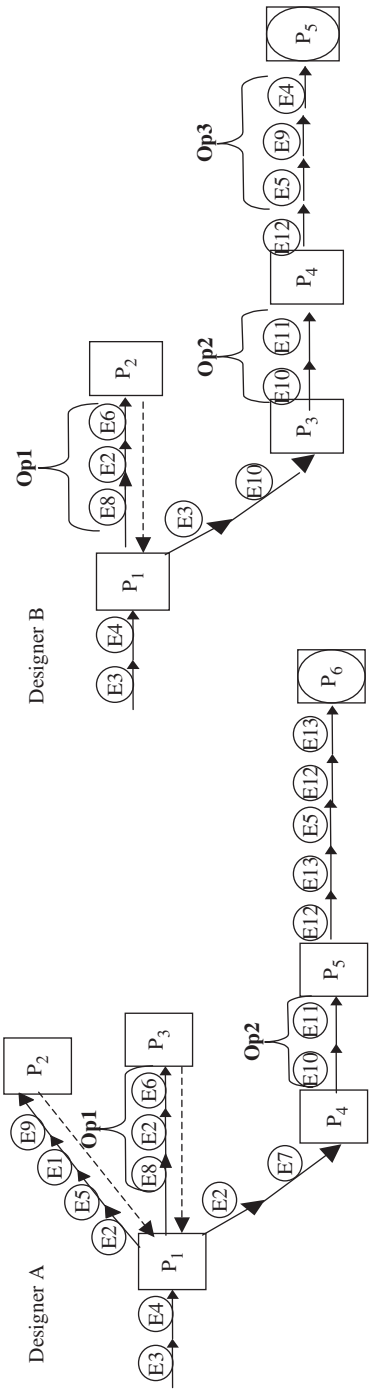
Step 2: Determine the parent–child relationship between product models.

To evaluate the relations between product models, we introduce the concept of virtual distance between product models. The distance between two product models is represented by the summation of a set of binary values resulting from the comparison of their core parameters. If the values of the same parameter of the two product models are different, then the binary value is 1; otherwise it is 0. When a newly derived product model is formed, distances between the product model and all other existing product models are calculated. After that, the product model that has the closest distance is selected as the parent of the newly derived product model. In the product model tree structure, a child product model is located and linked directly under its parent. If the closest distance is bigger than the predefined threshold, the product model is then considered to be independent of all other product models and has no parent.

Our goal is to capture design operations as design activity knowledge based on the data of monitored design events and the information of product model roadmap. We identify what were the important design operations and where they were applied in the product model roadmap. Figure 4 illustrates the monitored design events and the inferred design operations on the corresponding product model roadmaps of two skilful designers who were working on the similar design tasks. From figure 4, it can be seen that $Op1$, $Op2$ and $Op3$ are the important operations because they were always performed before the formation of a new derived product model.

4. Function-based design operation mining

Generally, there are two ways to extract design operations from monitored design events. The first is rule based, that is to define a set of rules that can be used to map predefined sequential patterns of events into known design operations. Our previous



P_i: Product Model with chronicle ID = i; E_n: Design Event of type *n*; Op_m: Design Operation of type *m*.

Figure 4. Design events, operations and product model roadmap.

work on GEDP to design knowledge acquisition (Ishino and Jin 2002) is an example of this approach. Although rule based approach is relatively simple, it has two limitations. One is that the predefined rules may not be complete to cover all interesting operations. The second limitation is that the rules are often design task-specific and it is difficult to reuse them for other design tasks. As a result, creating specific rules for specific design tasks can be very costly.

The other general approach is function based, i.e. to define a few functions, instead of many rules, and use the functions to determine whether a specific design event sequential pattern is interesting enough to be considered as a design operation. This approach is more general and data mining oriented since there is no need for predefined design event patterns and rules, and previously unknown and potentially useful sequential patterns can be discovered. As in the data mining domain, the term interestingness is used here as a concept to represent how seriously interesting and meaningful the detected patterns are. One of the most important aspects of interestingness is frequency, i.e. how many times a pattern appears in the design process. In design processes, however, it is not always true that the more frequently a pattern appears, the more meaningful it is. We need a new measure to calculate interestingness of event sequential patterns.

4.1 *Interesting sequential pattern*

To introduce a method that can be applied to design activity knowledge acquisition for a wide range of design tasks, we formulate the problem of design activity capture as follows. First, we introduce the following definitions:

$E = \{e_1, e_2, \ldots, e_n\}$ time series of all monitored design events, where n is the total number of events,

$Pd = \{Pd_1, Pd_2, \ldots, Pd_m\}$ time series of all identified product models, where m is the total number of product models,

$M_k = \{e_{i+1}, e_{i+2}, \ldots, e_{i+h}\}$ event sequence monitored for developing Pd_k after finishing the formations of Pd_{k-1}, where i and h are arbitrarily chosen to let $i+1$ and $i+h$, respectively, indicate the initial event number and the last event number in the product model Pd_k; $\boldsymbol{M}_k \subseteq E$. For any sub-event sequence X, if $X \subseteq \boldsymbol{M}_k$, then we say that $\boldsymbol{M}_k$ contains X.

Given a set of monitored design events E, the problem of design operation mining can be formulated as two problems: (1) discovering interesting sub-sequences X_{op1}, $X_{op2}, \ldots$, as design operations, and (2) locating positions of $X_{op1}, X_{op2}, \ldots$, on the product model roadmap. Since the positions of event sequences on the product model roadmap is known once the roadmap is built, problem (2) is solved. To solve problem (1), we need to address two questions: what are interesting sub-sequences and how can they be acquired. We focus on the first question first.

We consider the interestingness of a sequence of events X in terms of the information the events carry about the operational intent of the designer. We assume that there are two types of interestingness in a sequence X. One is how informative the sequence is in itself, represented by the number of its component literals. The other is how informative the sequence is in the design process as a whole,

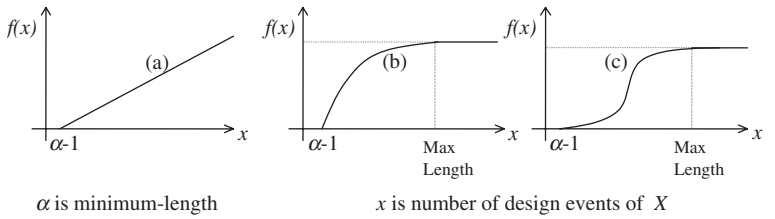


Figure 5. Examples of intrinsic interestingness functions.

represented as frequency of occurrence. We call the former intrinsic interestingness and the latter extrinsic interestingness.

The intrinsic interestingness of a sub-sequence X can be defined by the number of design events involved in X. We further assume that (1) the interestingness of any design event is no less than zero, and (2) the more events involved in a sub-sequence the more interesting the sub-sequence. We introduce an intrinsic interestingness function $f(x)$, where x represents the number of design events of X. Based on the above two assumptions, $f(x)$ should be a monotonic increasing function and always satisfy the conditions in formulas (1) and (2):

$$0 \leq {}^{\forall}x_1 < {}^{\forall}x_2 \Rightarrow f(x_1) \leq f(x_2) \tag{1}$$

$$^{\exists}\alpha > 0, \quad f(\alpha) > 0 \wedge f(\alpha - 1) = 0 \tag{2}$$

In formula (2), the integer α is the minimum length that represents the minimum number of design events that X must contain for having non-zero interestingness.

In fact, a designer can determine any function satisfying the above conditions as an intrinsic interestingness function of a given design task. Typical examples of such functions are schematically depicted in figure 5. Figure 5(a) shows a linear intrinsic interestingness function in which the informative interestingness increases in proportion with the number of design events. In practice, however, it is conceivable that the value tends to reach a ceiling and not increase any more after the number of events goes over a limit. We call this limit maximum length. Graphs (b) and (c) in figure 5 are other two different functions with ceilings.

Next, we define the extrinsic interestingness. Our interactions with engineering designers indicated that important design operations appear more than once in design processes. However, regarding the frequency of appearance of operations, there were two different opinions. Some designers felt that more frequent appearances mean it is more significant. Others voiced that the rarely appeared operations were the most important ones. After summarizing the opinions of the designers, we adopt the following assumption: a valuable sequential pattern always appears more than once in design processes.

Based on this assumption, we introduce extrinsic interestingness function, $g(y)$, where y indicates how many times a sequence X appears in the design process. The extrinsic interestingness function is expressed by equation (3), where $Freq(X)$ is a function of an event sequence X and indicates its number of appearances in the design process. The extrinsic interestingness function is not always a monotonic function, but needs to satisfy the condition in formula (4), which signifies the above assumption. In formula (4), an integer β represents the largest number of appearance

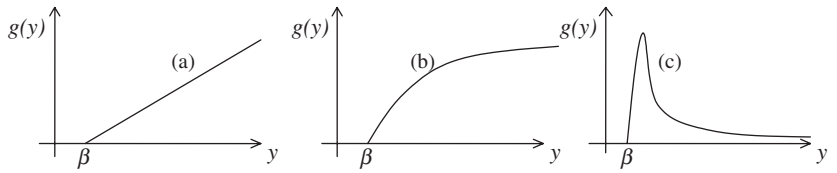


Figure 6. Examples of extrinsic interestingness functions.

when the extrinsic interestingness function gets zero-value. The integer β is called a frequency threshold:

$$g(y) = g(Freq(X)) \tag{3}$$

$$^{\exists}\beta \geq 1, \quad g(\beta + 1) > 0 \wedge g(\beta) = 0 \wedge g(\beta - 1) = 0 \wedge \ldots \wedge g(1) = 0 \tag{4}$$

A designer can determine a function satisfying the condition in formula (4) as an extrinsic interestingness function of a certain design task based on his/her experience. Typical extrinsic interesting functions are schematically depicted in figure 6. Graph (a) in figure 6 indicates the case that the rare sequences have great value; graphs (b) and (c) in figure 6 indicate the cases that more frequent means more interesting.

Finally, we integrate intrinsic and extrinsic interestingness into a comprehensive whole, called interestingness, expressed by $I(X)$, which indicates how seriously interesting and meaningful the detected patterns are for design experts. Since the interestingness should be zero-value when either the intrinsic interestingness or the extrinsic interestingness is zero-value, we define the interestingness as multiplication of the two, as shown in equation (5), where x represents the number of literals of event sequence X, and y represents number of appearance of X. Given equation (5), a design operation can be viewed as 'a sequence of events that has a high value of interestingness':

$$I(X) = f(x) \times g(Freq(X)) = f(x) \times g(y) \tag{5}$$

4.2 *Design operation mining method*

After defining what interesting patterns are, our next question is how can they be acquired? To avoid requiring predefined templates, we take a generation-and-test approach. The following are the steps of our method:

Step 1: Treat the chronicle sequence of monitored events as an applied field for data mining.

Step 2: Pick up a j-tuple event sequence from the applied field and treat it as a j-tuple template.

Step 3: Apply the template through the whole applied field to match the similar patterns and identify the j-tuple event sequences with higher-than-threshold interesting values as design operations.

Step 4 Alter j and go to Step 2.

```
1   Start:
2         Let j = minimum_length − 1
3     Iteration:
4       Let j = j + 1
5       If j > maximum_length Then Go to End_Iteration
6         Else
7         Create a new j-template from the Applied Field                ;; 1) Create Template
8           If no more j-template, Then go to Iteration
9           Else
10            Calculate Intrinsic interestingness of the j-template using Intrinsic Interestingness Function
11        Get Frequency of the j-template through Pattern Matching      ;; 2) Execute Pattern Matching
12        If Frequency > frequency_threshold
13            Then
14              Calculate Extrinsic interestingness of the j-template using Extrinsic Intrst. Function
15              Calculate Interestingness of the j-template
16              Add the j-template to the OperationList
17        If Stop Condition is satisfied Then Go to End_Iteration       ;; 3) Check and Stop Search
18        Else Go to Iteration
19    End_Iteration
20        Determine operation sequence                                  ;; 4) Determine operation sequence
21  End
```

Figure 7. Algorithm of function-based design operation mining.

The variable j is an integer between the minimum length and the maximum length of the successive event sequence. It can be seen that any j-tuple event sequence can be picked up as a template, but only the matched sequences with high enough interestingness will be selected as design operations. The pseudo-code of the algorithm is illustrated in figure 7.

To create an effective algorithm of pattern matching for design operation identification, we introduce a practically important assumption: a design operation often emerges with noise. To deal with noise in the design event data, we devised a dynamic programming based method to perform pattern matching for measuring $Freq(X)$. For example, if the sequence [w, o, r, d] is an original sequence where letters are literals, then [w, o, a, r, d] and [w, r, o, d] can be recognized as the same. The details of our dynamic programming based pattern matching can be found in our previous work (Ishino and Jin 2002).

There are three stop conditions (line 17 of figure 7): (1) when the number of literals in templates (i.e. j of j-tuple templates) reaches the maximum length, if it exists and the extrinsic interestingness function is monotonic; (2) if all j-tuple templates get their frequency equal to or less than the frequency threshold; and (3) if j of j-tuple template reaches the maximum literal length of the event set. These conditions are implemented in DAKA as *a priori* heuristics (Agrawal and Srikant 1994).

As shown in figure 7, there are four major functions in our algorithm, i.e. create a template, execute pattern matching, check and stop the search, and determine the design operation sequences. To illustrate these functions, let us consider the data in table 1 as a design process that consists of four product model event sets. Five literals, or types of events, described by a, b, c, d, and e are involved in this case. The maximum literal length in this design process is seven. In this example, for simplicity, we do not use dynamic programming in pattern matching.

Table 1. Example data of a design process.

Product model	Design events
Pd_1	M_1 = [b, c, a, c, d, a]
Pd_2	M_2 = [a, d, c, e]
Pd_3	M_3 = [b, a, d, c, e, d, c]
Pd_4	M_4 = [e, c, a, c, a]

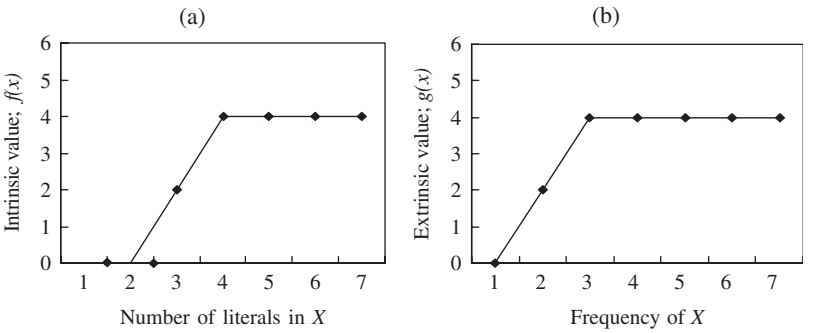


Figure 8. Examples of intrinsic value function and extrinsic value function.

Table 2. Examples of a three-tuple template.

Template	Frequency	Place	Intrinsic value	Extrinsic value	Interestingness
[a, d, c]	2	M_2(1,3), M_3(2,4)	2.0	2.0	4.0
[c, a, c]	2	M_1(2,4), M_4(2,4)	2.0	2.0	4.0
[d, c, e]	2	M_2(2,4), M_3(3,5)	2.0	2.0	4.0

First, the intrinsic value function $f(x)$ and the extrinsic value function $g(x)$ are determined as shown in figure 8, in which the lower length limit is 3, the upper length limit is 4, and the frequency threshold is 1.

Next, while j-tuple templates generated, the frequency of each template is investigated using pattern matching. In this example, the three-tuple templates include [b, c, a], [c, a, c], etc. They are generated and kept sorted in their lexicographic order. All three-tuple templates have the intrinsic value of 2.0 according to the intrinsic value function (figure 8(a)). The frequency of each three-tuple template is identified using pattern matching, and it turns out that three templates, [a, d, c], [c, a, c], and [d, c, e], exceed the frequency threshold because they all appear twice in the design process. Their extrinsic values are all 2.0 based on the extrinsic value function (figure 8(b)). Therefore, these three templates are stored in memory together with their interestingness and places of appearance in the design process, as shown in table 2. After that, the stop condition is checked; if an upper length limit exists and j in j-tuple templates reaches it, the iteration loop stops.

Table 3. Example of a four-tuple template.

Template	Frequency	Place	Intrinsic value	Extrinsic value	Interestingness
[a, d, c, e]	2	$M_2(1,4)$, $M_3(2,5)$	4.0	2.0	8.0

When j is 3, this stop condition is not satisfied. The same procedure is performed on four-tuple templates, resulting in only one 4 template stored in memory, as shown in table 3. When j is 4, the stop condition is satisfied and the iteration loop stops.

Finally, comes a finalizing process to determine operations. A template that has the highest value of interestingness is first chosen as the most possible operation sequence. At the same time, its places of appearance corresponding to where the template appears in the design process are recorded as a consumed area. In this example, the template [a, d, c, e] is selected as the first operation sequence, and its places of appearance, one from the first literal to the fourth in M_2 and the other from the second literal to the fifth in M_3, are recorded as the consumed area. Then, other templates that have the second highest value of interestingness are put in a candidate set. In the example, all three templates in table 2 are put in the candidate set. Then, a candidate template whose locations of appearance do not overlap with the consumed area is chosen as the second operation sequence, and the places of the second operation sequence are added to the consumed area. In the example, the sequence [c, a, c] is selected as the second operation sequence, and its places, from the second literal to the fourth in M_1 and from the second literal to the fourth in M_4, are added to the consumed area. This process is repeated until the candidate templates run out. In the example, no more candidates exist, and finally two sequences, [a, d, c, e] and [c, a, c], are selected as operation sequences.

5. Case example

To test the proposed approach for design activity knowledge acquisition, we chose the 'car front door design' task presented by Araki (2000). In this design task a designer has to design all of the geometric positions and sizes of the front door, the lock and striker part, and the seal part, based on the information of the cross-section of the car body depicted in figure 9. This design was built as a two-dimensional parametric design using a commercial CAD system. A wrapper module was developed and plugged into the CAD to monitor design events. The geometric model of this design is also shown in figure 9. There are 28 parameters in total.

We prepared three design cases: a middle-sized car, a compact car and a luxury car, and had a skilful designer design the cases in this order. There were common constraints and requirements for all cases, as well as specific ones for individual cases.

Constraints (symbols are defined in figure 9):

$\overline{ab}$ *is on* $\overline{ij}$, $b = j$, $\overline{fg}$ *is on* $\overline{km}$, $m = g$, $|mq| = 4$, where the notation of $|mq|$ means the length of segment mq.

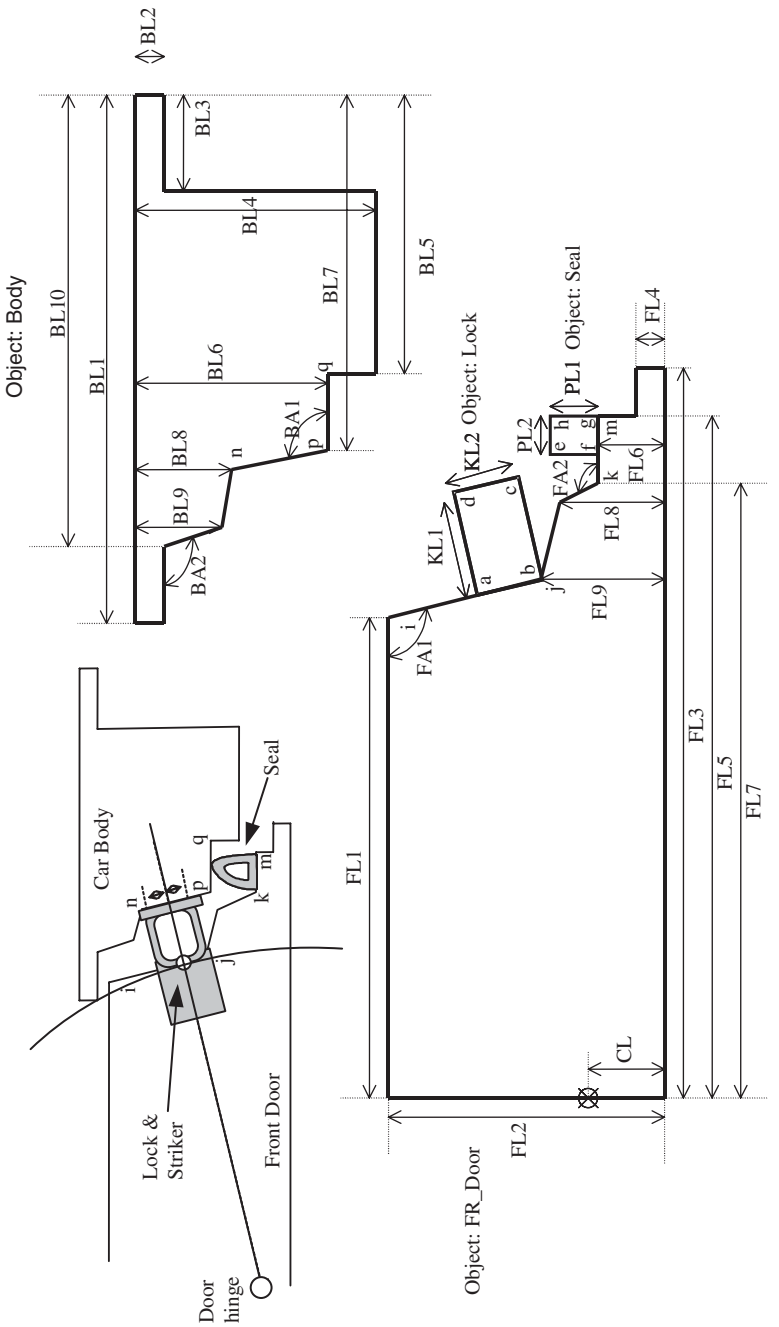


Figure 9. Geometric model of front door design.

Table 4. Case-specific requirements.

	Case A	Case B	Case C
FL5 + BL5	160	156	166
CL	12	9	13
PL1	8	6	8
PL2	5	4	5
BL1	75	70	79
BL2	7	5	9
BL3	12	11	12
BL4	47	45	50
BL5	30	28	30
BL6	39	35	39
BL7	44	40	44
BL10	69	64	72

Common requirements:

$$\overline{cd} \text{ is on } \overline{pn},\ \overline{eh} \text{ is on } \overline{pq}, \qquad 9 \le \mathrm{KL1} \le 16,\ 8 \le \mathrm{KL2} \le 14,$$
$$\mathrm{FL9} = \mathrm{FL1} \cdot \cos\theta \sin\theta + \mathrm{FL2} \cdot \sin^2\theta + \mathrm{CL} \cdot \cos^2\theta - \mathrm{KL2} \cdot \cos\theta, \qquad \text{where } \theta = \mathrm{FA1} - 90.$$

This equation in common requirements indicates the following: the radius of the circular arc that has one endpoint at the centre of the front door hinge and the other endpoint on the segment $\overline{ij}$ of the front door should be orthogonal with the segment $\overline{ij}$. The centre of the lock and striker part should be on the intersection of the radius and the segment $\overline{ij}$.

Case-specific requirements:

Case specific requirements are shown in table 4.

5.1 *Product model and monitored design events*

This case study did not distinguish between the 'product model' and 'product model core' because it was not initially clear which parameters were more important than the others. Therefore, all 28 parameters indicated in figure 9 were used to identify product model roadmaps. It was assumed that when the same parameters were manipulated the second time it is the start of a new product model. Furthermore, only the parameter change events were monitored. Therefore, there are a total of 28 possible types of events involved in this case study.

A designer with sufficient task knowledge executed all three design cases A, B, and C. In design case A, 74 design events were yielded and 18 product models were identified by analysing the states of the product model core. Similarly, 39 design events and nine product models were detected in design case B, and 39 design events and six product models were obtained in design case C. Table 5 shows a partial list of the design events monitored during case B design. The product model roadmaps on each design case are shown in figure 10.

Table 5. Partial list of the captured design events.

Event ID	Parameter	FL1	FL2	FL3	FL4	FL5	FL6	FL7	FL8	FL9	FA1	…
…	…	…	…	…	…	…	…	…	…	…	…	…
11	FL1	**94**	51	145	8	130	17	116	22	24	110	…
12	FL2	94	**49**	145	8	130	17	116	22	24	110	…
13	BA1	94	49	145	8	130	17	116	22	24	110	…
14	KL1	94	49	145	8	130	17	116	22	24	110	…
15	FA1	94	49	145	8	130	17	116	22	24	**105**	…
16	FL2	94	**49.4**	145	8	130	17	116	22	24	105	…
17	KL2	94	49.4	145	8	130	17	116	22	24	105	…
18	KL1	94	49.4	145	8	130	17	116	22	24	105	…
19	FL3	94	49.4	**140**	8	130	17	116	22	24	105	…
20	FL5	94	49.4	140	8	**125**	17	116	22	24	105	…
21	FL5	94	49.4	140	8	**128**	17	116	22	24	105	…
22	BA1	94	49.4	140	8	128	17	116	22	24	105	…
23	BA1	94	49.4	140	8	128	17	116	22	24	105	…
24	FA1	94	49.4	140	8	128	17	116	22	24	**107**	…
25	FL1	**81**	49.4	140	8	128	17	116	22	24	107	…
26	BA1	81	49.4	140	8	128	17	116	22	24	107	…
27	FA1	81	49.4	140	8	128	17	116	22	24	**104**	…
…	…	…	…	…	…	…	…	…	…	…	…	…

Note: highlighted cells indicate the 'parameter change' events in the context of the other parameters.

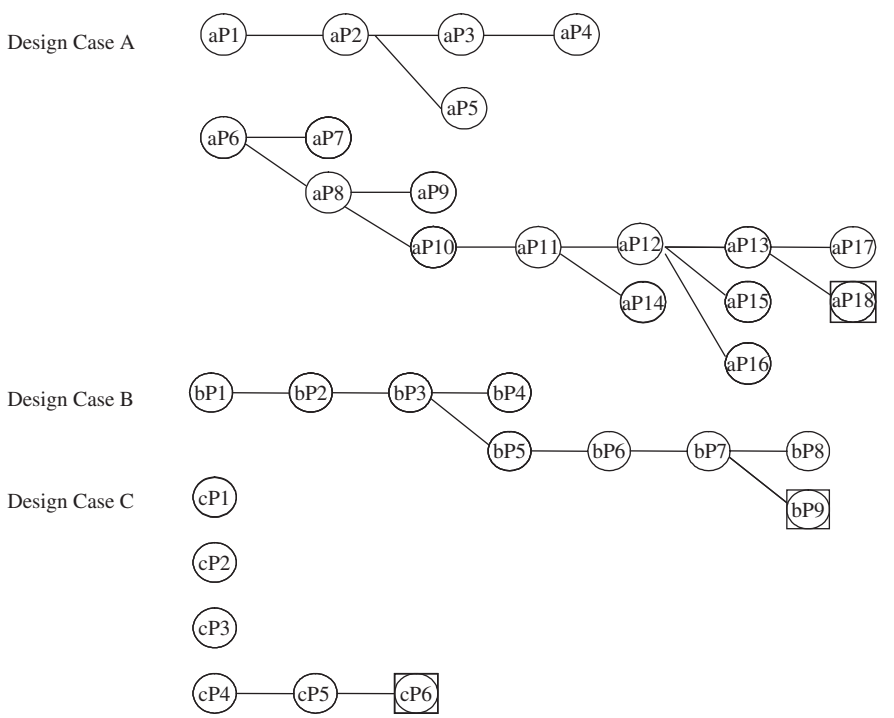


Figure 10. Product model roadmap (front door design).

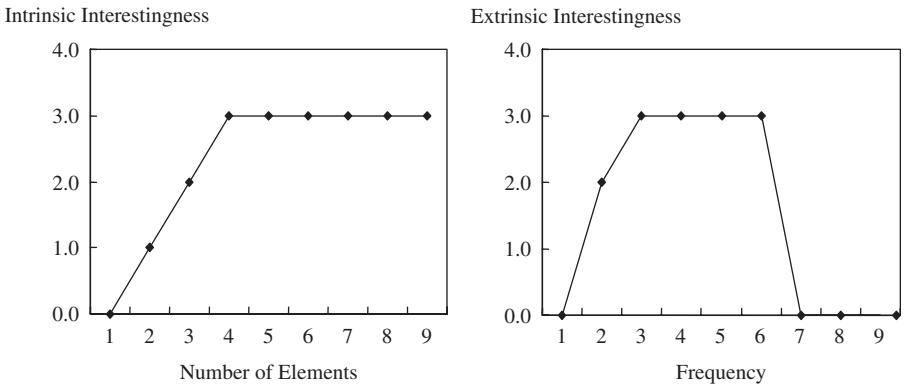


Figure 11. Intrinsic and extrinsic interestingness functions on front door design.

Table 6. Design operations acquired by DAKA in front door design.

ID	Sequence as operations	Frequency	Intrinsic value	Extrinsic value	Interestingness
Op1	FL1-FL2-FL9-KL2	5	3.0	3.0	9.0
Op2	FL2-FL9-BL9-BL8	3	3.0	3.0	9.0
Op3	FL2-FL9-FL1	3	2.0	3.0	6.0
Op4	BA1-FA1-FL1	2	2.0	2.0	4.0
Op5	FL4-FL6-BA1	2	2.0	2.0	4.0
Op6	FL6-FL8	2	1.0	2.0	2.0

Note: the parameter name in the operation sequence indicates the 'parameter value modification' event on that parameter, e.g. 'FL1' means 'modifying FL1's value'.

5.2 *Design operation mining*

The intrinsic interestingness function and the extrinsic interestingness function we set are shown in figure 11. These were determined based on hearing from the designer who had accomplished these three design cases.

The design operation mining was applied to all three design cases together as if one huge design process had included all 33 product models. The operation identification process is the same as described in section 4.2 except that the dynamic programming algorithm (Ishino and Jin 2002) was used for pattern matching. Six sequential patterns were discovered as valuable design operations, as shown in table 6.

6. Results

The acquired design operations were located on the product model roadmap of each design case, as shown in figure 12.

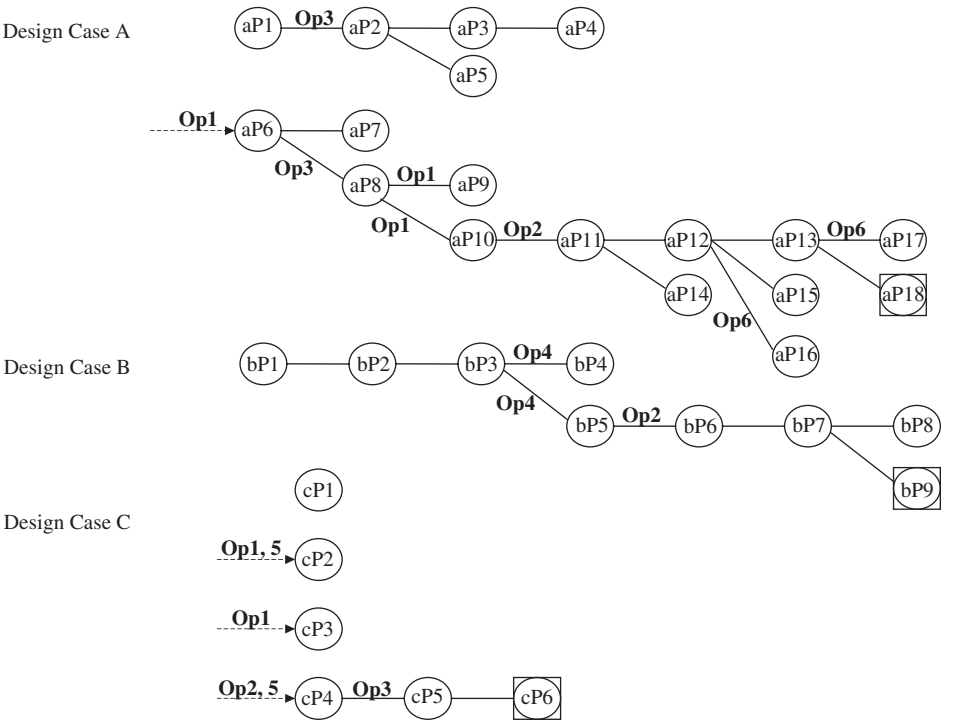


Figure 12. Emergence of the acquired design operations.

In order to examine how the acquired sequences of design events were qualified to be practical design operations, table 6 and figure 12 were presented to the designer who performed the designs. The designer evaluated the sequence ID 1, 2, 4 and 6 as important design operations and provided the following comments:

- *Operation Op*1: this seemed to be executed to meet the common requirement expressed by the following equation:

$$\mathrm{FL9} = \mathrm{FL1} \cdot \cos\theta \sin\theta + \mathrm{FL2} \cdot \sin^2\theta + \mathrm{CL} \cdot \cos^2\theta - \mathrm{KL2} \cdot \cos\theta,$$
$$\text{where } \theta = \mathrm{FA1} - 90$$

  This equation included FL1, FL2, FL9, CL, KL2, and FA1. The length of CL was designated as a case-specific requirement, so that the rest of the five parameters should be determined. Since FA1 was an angle and its change of even 1° affected other parameters much greater than the other parameters did, the point here was that FA1 should be examined and determined first and then FL1, FL2, FL9, and KL2 should be adjusted to meet the requirement. Op1 represented the natural order of changing parameters, FL1, FL2, FL9, and KL2, from the large parameter to the small parameter. Since the requirement was the most important and difficult requirement, Op1 tended to reappear in the early stage of design process.

- *Operation Op*2: this reflects the adjustment of the space between the front door and the body for placing the lock and striker. After the requirement mentioned above was satisfied, there often was not enough space left for the lock and striker. The series of design events, changing the parameters FL2, FL9, BL9, and BL8, was performed to solve this problem by changing the vertical length of the front door and the body.
- *Operation Op*4: this was executed to adjust the horizontal space between the front door and the body by changing angles, FA1 and BA1. This was a radical change, since the change of these angles affected the requirement mentioned above. Op4 might indicate that when it is necessary to change either one of the BA1 and FA1 in the middle of design, the other one should also be changed. Since the change of these angles affected the horizontal length of the space between the front door and the body, the length of FL1 should be changed successively.
- *Operation Op*6: this was conducted to adjust the vertical space for the seal. This was a small change for the design and was executed in the late stage of the design.

It is interesting to note that the design operations acquired by DAKA can be used by designers to recall the significant design operations they performed in their design processes. This experiment showed that DAKA can approximate significant design operations in practice.

7. Conclusions

Engineering knowledge is a key asset for industrial enterprises. This paper focused on design activity knowledge and proposed a DAKA framework that acquires design activity knowledge through mining-monitored CAD events. DAKA requires neither designer involvement at the design stage nor predefined and task-specific rules for the classification of design events. The key components of DAKA include a product model roadmap that represents the trajectory that designers walked through in their design process and an interestingness function-based design operation-mining algorithm that identifies the most interesting design event sequential patterns as design operations. The design activity knowledge acquired by DAKA reflects the trajectory of designers' approach to reach a final design product model and identifies key design operations for specific design tasks.

The case study results shown in table 6 and figure 12 indicate the features of design activity knowledge and the capabilities of the DAKA approach. The design activity knowledge is usually tacit and can hardly be explicitly articulated. Although the designer who performed the designs of the case study had reasons for performing certain design operations, such as Op1, Op4 and Op6, it would still be hard for the designer to articulate explicitly such knowledge before these operations were explicitly represented in terms of event sequences and presented in front of the designer. This approach of extracting operations from design events for understanding and improvement is similar to the business process intelligence approach in which process execution logs are explored to identify and improve the process execution quality (Grigori *et al.* 2004). The fact that DAKA does not require

predefined rules or templates for identifying design operations is a big advantage. Since design activity knowledge is tacit and not likely to be prespecifiable, applying interestingness functions for mining design operations made it possible for DAKA to acquire design operations. Although how to specify interestingness functions is still a question, the present authors believe the experience can be built by testing various types of functions for given design tasks. The case example described in section 5 required very little computation time for DAKA to generate the results shown in table 6. To verify the scalability of the proposed approach, a test case was created using computer-generated synthetic data in which about 5000 design parameters and tens of thousands of design events were involved. The computation for this case took 3.77 hours on a Pentium IV, 2-GHz, computer. The authors' current research investigates how different interestingness functions affect design activity knowledge-capturing by applying DAKA to more engineering case examples.

Acknowledgements

This research was supported, in part, by a NSF CAREER Award under Grant No. DMI-9734006. Additional support was provided by industrial sponsors. The authors are grateful to the NSF and industrial sponsors for their support.

References

Agard, B. and Kusiak, A., A data-mining based methodology for the design of product families. *Int. J. Prod. Res.*, 2004a, **42**, 2955–2969.

Agard, B. and Kusiak, A., Data mining for subassembly selection. *ASME Trans. J. Manuf. Sci. Eng.*, 2004b, **126**, 627–631.

Agrawal, R. and Srikant, R., Fast algorithms for mining association rules, in *Proceedings of the 1994 International Conference on Very Large Data Bases (VLDB'94)*, 1994, pp. 487–499.

Araki, H., A study of the collaborative environment for product development process innovation. Doctoral thesis, University of Tokyo, 2000.

Banares-Alcantara, R. and King, J.M.P., Design support systems for process engineering. iii – Design rationale as a requirement for effective support. *Comput. Chem. Eng.*, 1997, **21**, 263–276.

Bradshaw, J.M., Carpenter, R., Cranfill, R., Jeffers, R., Poblete, L., Robinson, T., Sun, A., Gawdiak, Y., Bichindaritz, I. and Sullivan, K., Roles for agent technology in knowledge management: Examples from applications in aerospace and medicine, in *Proceedings of the AAAI Spring Symposium on Artificial Intelligence in Knowledge Management (AIKM'97)*, 1997.

Braha, D., *Data Mining Des. Manuf. Meth. Appl.*, 2001 (Kluwer: Norwell, MA).

Chen, M., Configuration of cellular manufacturing systems using association rule induction. *Int. J. Prod. Res.*, 2003, **41**, 381–395.

Da Cunha, C., Agard, B. and Kusiak, A., Improving manufacturing quality by re-sequencing assembly operations: a data-mining approach, in *Proceedings of the 18th International Conference on Production Research — ICPR 18*, University of Salerno, Fisciamo, Italy, August 2005, pp. 1–6.

Ganeshan, R., Garrett, J. and Finger, S., A framework for representing design intent. *Design Stud.*, 1994, **15**, 59–84.

Garcia, A.C.B. and Howard, H.C., Acquiring design knowledge through design decision justification. *Artif. Intell. Eng. Des. Anal. Manuf.*, 1992, **6**, 59–71.

Goel, A., Bhatta, S. and Stroulia, E., KRITIK: an early case-based design system, In *Issues and applications of case-based reasoning in design*, edited by M. Maher and P. Pu, pp. 87–132 (Brlbaum: Hillsdale, NJ).

Gray, B. and Orlowska, M.E., Ccaiia: clustering categorical attributes into interesting association rules, In *Proceedings of the Second Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'98)*, edited by X. Wu, R. Kotagiri, and K. Korb, pp. 132–143, 1998 (Melbourne).

Grigori, D., Casati, F., Castellanos, M., Dayal, U., Mehmet Sayal, M. and Shan, M., Business process intelligence. *Comput. Ind.*, 2004, **53**, 321–343.

Hilderman, R. and Hamilton, H., *Knowledge Discovery and Interestingness Measures: A Survey*. Technical Report No. CS 99-04, 1999 (Department of Computer Science, University of Regina).

Ishino, Y. and Jin, Y., Acquiring engineering knowledge from design processes. *Artif. Intell. Eng. Des. Anal. Manuf.*, 2002, **16**, 73–91.

Ishino, Y. and Jin, Y., Data mining for knowledge acquisition in engineering design, In *Data mining for design and manufacturing: methods and applications*, edited by D. Braha, pp. 145–160 (Kluwer: Amsterdam).

Kunz, W. and Rittel, W., *Issues as Elements of Information Systems*. Working Paper No. 131, 1970 (Center for Planning and Development Research, University of California, Berkeley, CA).

Kusiak, A., A data-mining approach for generation of control signatures. *ASME Trans. J. Manuf. Sci. Eng.*, 2002, **124**, 923–926.

Kusiak, A., Decomposition in data mining: an industrial case study. *IEEE Trans. Elec. Packag. Manuf.*, 2000a, **23**, 345–353.

Kusiak, A., Evolutionary computing and data mining, in *Proceedings of the SPIE Conference on Intelligent Systems and Advanced Manufacturing*, edited by B. Gopalakrishnan and A. Gunasekaran. SPIE 4192, Boston, MA, USA, November 2000b, pp. 1–10.

Kusiak, A., Selection of invariant objects with a data-mining approach. *IEEE Trans. Elec. Packag. Manuf.*, 2005, **28**, 187–196.

Lakin, F., Wambaugh, J., Leifer, L., Cannon, D. and Steward, C., The electronic design notebook: performing medium and processing medium. *Vis. Comput. Int. J. Comput. Graph.*, 1989, **5**, 214–226.

Létourneau, S., Famili, F. and Matwin, S., Data mining to predict aircraft component replacement. *IEEE Intell. Sys.*, 1999, **14**, 59–66.

Liang, J., Shah, J.J., D'Souza, R., Urban, S.D., Ayyaswamy, K., Harter, E. and Bluhm, T., Synthesis of consolidated data schema for engineering analysis from multiple STEP application protocols. *Comput.-Aid. Des.*, 1999, **31**, 429–447.

Maher, M.L. and Pu, P., *Issues in the Application of Case-based Reasoning Design*, 1997 (Lawrence Erlbaum: Mahwah, NJ).

Matheus, C.J. and Piatetsky-Shapiro, G., Selecting and reporting what is interesting: the Kefir application to healthcare data, In *Advances in Knowledge Discovery and Data Mining*, edited by U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, pp. 495–515, 1996 (AAAI Press/MIT Press: Menlo Park, CA).

Moran, T.P. and Carroll, J.M., *Design Rationale: Concepts, Techniques, and Use*, 1996 (Lawrence Erlbaum: Mahwah, NJ).

Myers, K.L., Zumel, N.B. and Garcia, P., Automated capture of rationale for the detailed design process, in *Proceedings of the 11th Conference on Innovative Applications of Artificial Intelligence (IAAI'99)*, 1999.

Neaga, E.I. and Harding, J.A., An enterprise modeling and integration framework based on knowledge discovery and data mining. *Int. J. Prod. Res.*, 2005, **43**, 1089–1108.

Ohsaki, M., Sato, Y., Kitaguchi, S., Yokoi, H. and Yamaguchi, T., Comparison between objective interestingness measures and real human interest in medical data mining, in *Proceedings of the 17th Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE2004, LNAI3029)*, 2004, pp. 1072–1081.

Piatetsky-Shapiro, G., Discovery, analysis and presentation of strong rules, in *Knowledge Discovery in Databases*, pp. 229–248, 1991 (AAAI/MIT Press: Menlo Park, CA).

Shah, J.J., Rangaswamy, S., Qureshi, S. and Urban, S.D., Design history system: data models and prototype implementation, In *Knowledge Intensive Computer Aided Design*, edited by S. Finger, T. Tomiyama, and M. Mantyla, pp. 91–114, 2000 (Kluwer: Boston, MA).

Shaw, M.J., Subramaniam, C., Tan, G.W. and Welge, M.E., Knowledge management and data mining for marketing. *Dec. Supp. Sys.*, 2001, **31**, 127–137.

Shipman III, F.M. and McCall, R.J., Integrating different perspectives on design rationale: supporting the emergence of design rationale from design communication. *Artif. Intell. Eng. Des. Anal. Manuf.*, 1997, **11**, 141–154.

Smyth, P. and Goodman, R.M., Rule induction using information theory, *Knowledge Discovery in Databases*, pp. 159–176, 1991 (AAAI/MIT Press: Menlo Park, CA).