Available online at www.sciencedirect.com
SCIENCE DIRECT
Advanced Engineering Informatics 20 (2006) 89–107
ADVANCED ENGINEERING INFORMATICS
www.elsevier.com/locate/aei

An information value based approach to design procedure capture

Yoko Ishino[a,b,*], Yan Jin[a]

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

Received 19 August 2004; accepted 19 April 2005

Abstract

Capturing engineering knowledge and managing it effectively is important for enterprises to stay competitive in today's global market. In our research, we focus on specific engineering knowledge, called *design procedures*, and attempt to develop effective ways to capture this operational knowledge from the design events monitored during design process. We proposed a novel method called Information Value based Mining for Sequential Pattern, or VMSP for short. VMSP does not require any predefined design-task-specific rules for classification of design events. The basic idea of VMSP is to treat any list of monitored design events as a sequential pattern and search for most frequent and informative sequential patterns. VMSP automatically generates candidate templates of sequential patterns, and then identify valuable patterns as design operations based on the *interestingness* we set. The *interestingness* is a synthesized index of the two evaluation criteria: One is *intrinsic value* representing how many design events constitute the template; and the other is *extrinsic value* indicating how often the sequence appears in the design process. We have evaluated VMSP through two case studies using real CAD operation data recorded through gear design and automotive door design processes, respectively. A case study using synthetic data was also carried out to test the scalability of the proposed algorithm.

Keywords: Knowledge acquisition; Design procedure; Design process; Data mining; Interestingness; Sequential pattern

1. Introduction

Knowledge management has recently attracted much attention in various industries, including engineering design and manufacturing [4,15]. Industrial design projects have grown larger, and designing modern products such as cars and aircrafts requires vast amount of information and knowledge. Since designing different versions of products with similar mechanisms is common in practice, capturing design knowledge and sharing it among designers is indispensable to maintain product quality and corporate competence. It is especially useful to know *how* skillful designers carried out designs in their ways. Analyzing know-*how* knowledge can provide designers with valuable insights to effectively handle design tasks without diverting too much of their design efforts.

Though the value of know-*how* knowledge has long been recognized, capturing designers' know-*how* without disrupting their normal design processes is still a challenge. Designers usually acquire practical know-*how* knowledge from their design processes. Because design processes are often ill structured, ad hoc and may vary greatly depending on design contexts, designers often have to acquire their know-*how* knowledge through an implicit learning process. It is usually difficult for a designer to express it fully and accurately.

In this paper we look into a specific type of know-*how* knowledge, called *design procedure* knowledge. Design procedure knowledge explains how a designer approaches a design problem and achieves a final design: what operations were carried out, how the operations were organized, what iterations happened, and which alternatives were explored. Our objective is to *automatically* capture design procedure knowledge using commercial object-oriented CAD systems.

* Corresponding author. Address: Department of Aerospace and Mechanical Engineering, University of Southern California, 3650 McClintock Ave. OHE-430, Los Angeles, CA 90089-1453, USA. Tel.: +81 82 424 5735; fax: +81 82 424 5736.
E-mail address: y8ishino@hiroshima-u.ac.jp (Y. Ishino).

1474-0346/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.aei.2005.04.002

The captured design procedure knowledge will help us to understand how experienced designers carry out their designs and guide other designers to do better design. Moreover, the knowledge can be used for training of novice designers so that they can quickly learn how to execute prominent design.

In our previous research, we developed a Grammar and Extended Dynamic Programming Approach (GEDP) for design knowledge acquisition [12,13]. The core idea of GEDP is to model a design process as a series of meaningful clusters of design events, called design operations. In this approach, designers' activities are captured as design events and translated into design operations using a set of predefined classification rules, called grammar rules and Extended Dynamic Programming (EDP) rules. Although GEDP is a useful method to extract design procedure knowledge from operation event data, it has limitations. First, GEDP requires predefined rules, which may be incomplete for covering all interesting operations. Second, the rules are 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.

To overcome the limitations of GEDP, we propose a novel method called Value-based Mining for Sequential Pattern (VMSP). VMSP employs the same design process model as GEDP does, but it has the advantage of not requiring any predefined classification rules. The basic idea of VMSP is to treat any list of design events as a sequential pattern and search for most *meaningful* sequential patterns. VMSP automatically generates templates as candidates of design operations. The candidate templates are then evaluated, and only the meaningful ones are recognized as a design operation. The meaningfulness is defined by a user-defined *interestingness*, discussed in Section 3.1. The interestingness is a term generally used in data mining domain for representing how much valuable a generated pattern is. The principle of VMSP is so simple that it has the potential to be applied to a wide range of design tasks. We have verified VMSP through two case studies using practical CAD data captured through real design processes. We also tested scalability of the algorithm using a large set of synthetic design process data.

This paper is organized as follows. Section 2 introduces the framework for design procedure capture and explains several models developed in our previous research and used as basis for this research. In Section 3 we introduce the Value-based Mining for Sequential Pattern (VMSP) and describe its theoretical basis and algorithms. Section 4 shows two case studies in which the proposed method is applied; one is a 2D design of a double-reduction gear system and the other is a 2D design of the front door of a car. A case study using synthetic design process data is also discussed in this section. Section 5 discusses related work and Section 6 draws concluding remarks.

2. A framework for design procedure capture

Engineering design requires intensive and continuous thinking. It is usually not desirable to interrupt designers' normal design process. In our research, in order to avoid disrupting designers' design process, we first capture design operation event data from the CAD system that designers use, and then mine or infer the design procedures based on the captured CAD operation events.

Our framework contains five phases as depicted in Fig. 1: (1) monitor designers' design actions, (2) store the monitored event data of each designer into corresponding task clusters, such as *door design* or *instrument panel design*, (3) analyze data of each task cluster to mine design procedures for the corresponding design task, (4) refine the obtained design procedures by checking their meanings, and (5) distribute the design procedures as design knowledge for reuse. These five phases are continually reiterated, while the data and procedure knowledge are updated.

In the monitoring phase, a wrapper module, which translates designers' CAD operations into numerical predefined codes, enables us to monitor designers' design actions, 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 phase, 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 phase, data of the same task cluster in the design event log database are analyzed to extract preliminary design procedures. This phase is the main part of our framework and its detailed mechanism is described in Section 3. In the refining phase, the obtained preliminary design procedures are examined by human expert designers. If needed the experts can modify or delete any of them. After the refinement, finally the remaining design procedures are registered in a procedure knowledge database, which is called Design Procedure DB in Fig. 1. As an option, the procedure can be annotated with its procedure features and reasons. Lastly, in the distribution phase, obtained procedure knowledge can be delivered and shown at designers' requests. As can be seen from Fig. 1, monitoring and storing are carried out at real time during design, while analyzing and refining are processed in a batch mode after the complete event data become available.

2.1. Design process model

As mentioned above, the core of capturing design procedures is in the analysis phase. Since inferring design procedures from primitive event data is a bottom-up

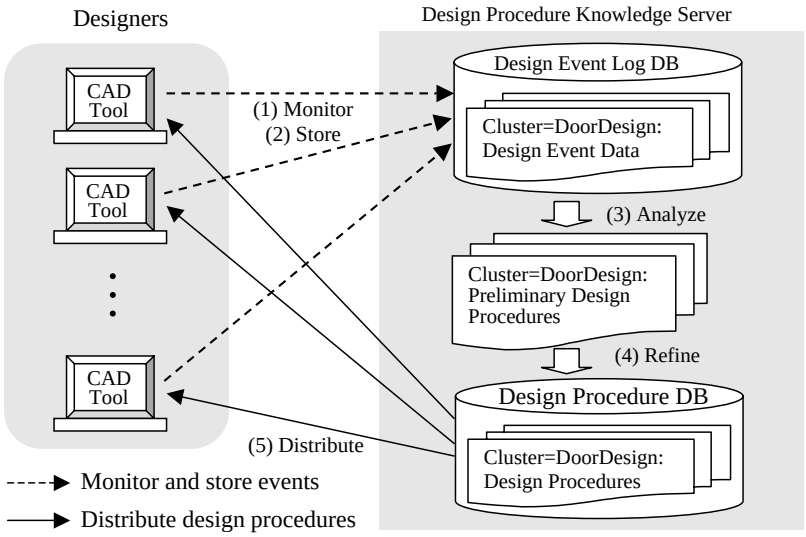
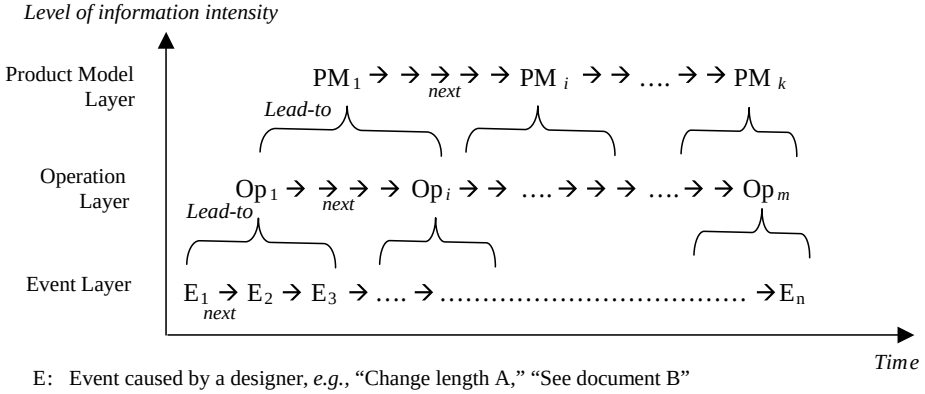


Fig. 1. A design procedure capture system framework.

approach, we need a general yet powerful model to represent design processes and an effective reasoning method for inference. We developed a *three-layer design process model* to represent general design processes [13], as schematically illustrated in Fig. 2.

The *three-layer design process model* represents generic design processes based on three layers of information, namely, Event Layer, Operation Layer, and Product Model Layer. The *Event Layer* captures primitive design events that are generated by designers while operating a CAD system. Design events are *observable* changes of design variables and accesses to documents. They do not reflect intentions of designers. For example, 'Change length A from 15 to 30' or 'See document B,' illustrated as 'E' in Fig. 2, is an event that occurred at the Event Layer. The *Operation Layer* represents higher-level design operations that reflect meaningful design actions. Elements at the Operation Layer are design operations that can be generated by reasoning based on multiple design events found at the Event Layer. 'Decrease the weight of the object' and 'Increase the strength of the arm' illustrated as 'Op' in Fig. 2 are examples of design operations. The elements in the *Product Model Layer* are design alternatives. Each product model corresponds to a specific intermediate design alternative illustrated as 'PM' in Fig. 2. The final design is the product model that is finally chosen as the result of design.

Based on our model of design processes, the goal of designers can be considered as to create a final product model. Before reaching the final product model, designers explore the solution space by creating and evaluating alternative product models. In each process of creating



E: Event caused by a designer, *e.g.*, "Change length A," "See document B"
Op: Meaningful Operation, *e.g.* "Decrease weight," "Increase strength of the arm"
PM: Product Model (design prototype) representing design alternatives

In real design processes, a product model (e.g., PM_1) is created by a sequence of operations (e.g., Op_1, ... Op_i); and an operation (e.g., Op_1) is performed through a sequence of observable events (e.g., [E_1, E_2, E_3]). In procedure capturing, a sequence of events can be used to infer meaningful operations and a sequence of meaningful operations can be linked to the formation of a product model.

Fig. 2. Three-layer design process model.

a product model, designers intentionally plan and perform various design operations ('Op's). Although what the performed design operations are and how they appear in a design process cannot be observed directly, we can use the observable events ('E's) as basis to reason about them.

We believe that the knowledge of 'how to proceed with design' is reflected in the patterns of 'how design operations appear in the design process'. Knowing how and where the design operations emerge in the design process can lead us to knowing what the key operations are and how a design was carried out. Our knowledge capture process has two steps. First we identify the structure of design process as the development history of product models. That is, we identify how many product models are created in the design process and how their parent–child relationships are formed. Secondly, we discover where key design operations emerge in the structured design process. The basis for the above two-step process is the monitored CAD operation data and their corresponding consequential states of the design objects (i.e. elements of the product model) in the CAD system.

2.2. Hierarchical structure of product models

As part of the design process model described above, we introduce a hierarchical structure of product models that represents the development trajectory of product models during the process of design. A product model is composed of a set of objects with various attributes representing physical and conceptual feature of the product being designed. The hierarchical structure is made by relationships among product models. The relationships are determined by comparing the features of product models and acquiring the similarity among them. The hierarchical structure is a time-series tree of product models in which a newly developed product model becomes a child of one of the existing leaf product models. A child product model can be derived from its parent by major feature modifications or additions. Of course, a newly developed product model does not always become a child of an existing product model. It is possible a new product model becomes a new root. We will show how to make the structure later. Analyzing the parent–child relationship between product models provides an overall map or path of how product models were created, explored, discarded, and finally adopted. Fig. 3 illustrates an example of the product model hierarchy. In this example,

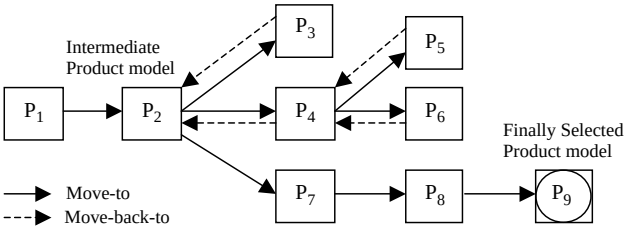


Fig. 3. Hierarchical structure of product models.

the designer created a new product model P_3 by modifying or creating one or more key features. After realizing P_3 will not lead to desired result, the designer moved back to P_2 and started modifying or creating some other features which led to P_4.

Creating a hierarchical structure from monitored CAD operation event data involves differentiating formations of individual product models and then determining their parent–child relationships. One simple way to do this is to use file version management system. When designers save their files into different versions, each version can be viewed as a product model and the version tree can be viewed as the product model structure.

For parametric design problems, however, one needs more sophisticated procedures to generate a hierarchical structure of product models since designers tend to change product models without *saving* current CAD data. We developed the following approach.

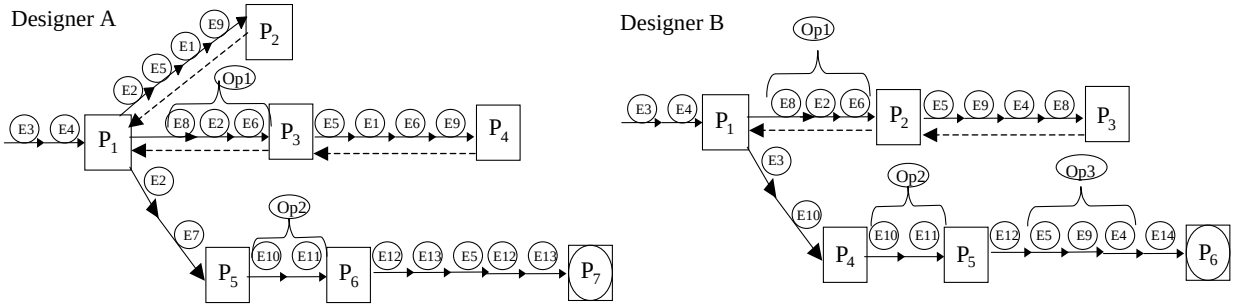
2.2.1. Differentiate formations of individual product models

First, the concept of *product model core* is introduced to represent the current status 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 key parameters of the product model. Updating the parameter values of the product model core implies significant progress in product model development.

Next, the formation, or birth, of a product model is judged as follows: (i) 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, and (ii) 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, one can determine *when a product model is formed* by monitoring the changes of the parameters of the product model core.

2.2.2. Determine the parent–child relationship between product models

To evaluate the relationship 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 new 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



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

This figure illustrates two designers' design processes represented in our design process model. Product models are constructed based on the object-oriented CAD data; design events are monitored through CAD tools, and design operations are inferred based on our proposed algorithms described in the following section

Fig. 4. Capturing design procedures.

threshold, the product model is then considered to be independent of all other product models and has no parent.

2.3. Design operations

Our goal is to capture design operations as procedure knowledge using the data of monitored design events and the information of product models. We identify what were the important design operations and where were they conducted in the design process. For example, Fig. 4 illustrates the monitored event data and the inferred operations of two skillful designers who were working on the similar design tasks. From Fig. 4 it can be seen that Op1, Op2 and Op3 are the important operations that were always performed before the formation of a new derived product model.

3. Value-based mining for sequential pattern (VMSP)

The previously unknown, potentially useful, interesting, and understandable sequential patterns should be successfully discovered in a design event log database. This objective is very similar to the one of data mining. In the data mining domain the term *interestingness* is used 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 many cases, however, it is not always true that the more frequently a pattern appears, the more meaningful it is. The design process may contain a lot of undistinguished repeating steps accompanied by early decisions in the design process. We propose a novel measure to calculate interestingness of sequential patterns generated. Next, we introduce a novel method called Value-based Mining for Sequential Pattern (VMSP) that includes the proposed interestingness measure. VMSP should be able to overcome the limitations of GEDP described in Section 1. VMSP is also based on the same *three-layer design process model* described above. However, VMSP does not require predefined rules.

3.1. Valuable sequential pattern

To introduce a method that can be applied to procedure acquisition for a wide range of design tasks, we formulate the problem of procedure capture as follows.

Definitions.

$E=\{e_1,e_2,\ldots,e_n\}$	time series of all monitored *design events*; where n is *total number* of events.
$P=\{P_1,P_2,\ldots,P_m\}$	time series of all identified product models, where m is *total number* of product models.
$M_k=\{e_{i+1},e_{i+2},\ldots,e_{i+h}\}$	the *event sequence* monitored for developing P_k after finishing the formations of P_{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 P_k; $M_k \subseteq E$.

For any *sub-event sequence* X, if $X \subseteq M_k$, then we say that M_k *contains* X.

Given a set of monitored design events E, the problem of mining design procedures can be formulated as discovering *valuable* sub-sequences $X_{op1},X_{op2},\ldots$ as design operations, and locating positions of $X_{op1},X_{op2},\ldots$ in the product model structure. The problem can be decomposed into two sub problems: what are the *valuable sub-sequences*, and how can they be acquired. We first focus on the former one here.

The value of a sub-event sequence X should be estimated from observations of E, P and M_k. We consider the value 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 values in a sequence X. One is the inherent value of the sequence, that is, how informative the sequence is in itself, represented by the number of its component literals. The other type is how informative the sequence is in the design process as a whole, represented as how often it appears. We call the former *intrinsic value* and the latter *extrinsic value*.

Firstly, we define the intrinsic value. Since we attempt to develop a general way of capturing design procedures that can be applied to various design tasks, we pay attention to the generic and objective information that sequential patterns emit. Therefore, the number of design events involved in a sub-sequence X becomes the only information we can access for formulating the intrinsic value. Here, we introduce an axiom related to the value of a single design event.

Axiom. the value of any design event is no less than zero.

The value of each design event is vague and can fluctuate through a design context. Each design event is part of a design operation. While redundant design events may have zero value, any identifiable design event should contribute to the overall value of the design operation. The axiom therefore should be valid. It indicates that a sequence with more elements provides more information. From the axiom, the more design events a sub-sequence X contains, the higher value (or at least the same value) the X should be able to generate. We introduce an *intrinsic value function* $f(x)$ in which x represents the number of design events that constitute sequence X. The function $f(x)$ should be a monotonic increasing function and always satisfy the conditions in logical 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 represents the lowest number of design events that constitute the sequence X when the intrinsic value function has value bigger than zero. We call α a *lower length limit*.

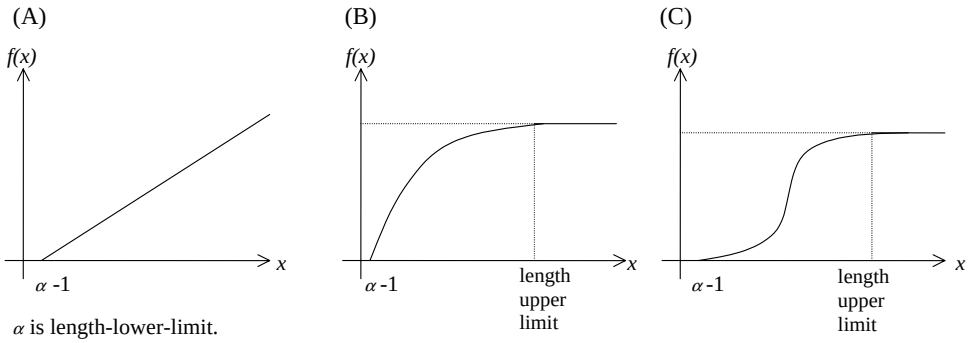
In fact a designer can determine any function satisfying the above conditions as an intrinsic value function of a given design task. We assume that the designer who did the design has empirical knowledge about how the value of a sub-sequence of design events relates to the number of the events constituting the sub-sequence. Typical examples of intrinsic value function are schematically depicted in Fig. 5. Fig. 5(A) shows a linear intrinsic value function in which the informative value 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 *upper length limit*. Graphs B and C in Fig. 5 are two different functions with ceilings.

Next, we define the extrinsic value. Our interactions with industrial 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 appearance means 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 No. 1.

Assumption No. 1. A valuable sequential pattern always appears more than once in design processes.

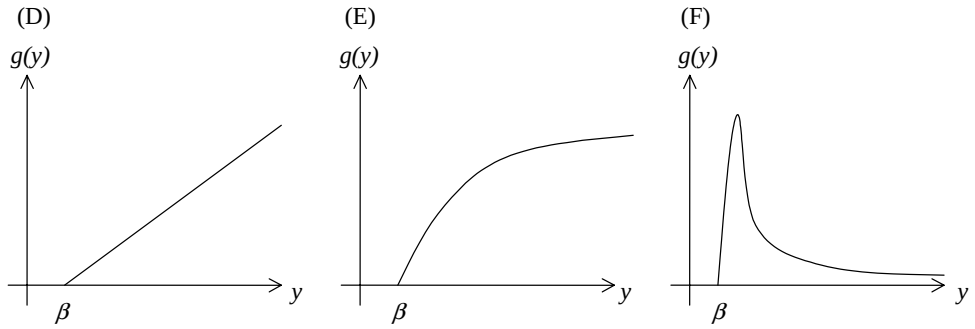
Based on this assumption, we introduce *extrinsic value function*, $g(y)$, where y indicates how many times a sequence X appears in the design process. The extrinsic value function is expressed by Eq. (3), where $Freq(X)$ is a function of an event sequence X and indicates its number of appearance in the design process. The extrinsic value function is not always a monotonic function, but needs to satisfy the condition in formula (4), which signifies Assumption No. 1. In formula (4) an integer β represents the largest number of appearance when the extrinsic value function gets zero-value. The integer β is called a *frequency threshold*.

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



α is length-lower-limit.
x is the number of design events that constitute X.

Fig. 5. Examples of intrinsic value functions.



y is the frequency of X in the design process.

β is a frequency threshold in the design process.

Fig. 6. Examples of extrinsic value functions.

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

As in the case of intrinsic value function, a designer can determine a function satisfying the condition in formula (4) as an extrinsic value function of a certain design task based on his/her experience. Typical extrinsic value functions are schematically depicted in Fig. 6. The graph F in Fig. 6 indicates the case that the rare sequences have great value, and graphs D and E in Fig. 6 indicate the cases that more frequent means more valuable. The word 'extrinsic value' is similar to the notion of 'support' in data mining. However, the word 'support' is usually used for finding an association rule. Strictly speaking, our resultant sequences differ from association rules, because association rules do not consider the order of components. So, we use the word, 'extrinsic value' instead of 'support'.

Finally, we integrate intrinsic and extrinsic values into a comprehensive measure of information value, called *interestingness*, expressed by $I(X)$. The concept of it is very similar to the term interestingness used in data mining domain. The interestingness indicates how seriously interesting and meaningful the detected patterns are for design experts. Since the interestingness should be zero-value when either the intrinsic value or the extrinsic value is zero-value, we define the interestingness as multiplication of intrinsic value and extrinsic value, as shown in Eq. (5). In Eq. (5), x represents the number of literals of event sequence X, and y represents number of appearance of X. Given the interestingness in Eq. (5), 'a valuable sequence (or design operation)' can be viewed as 'a sequence that has a high value of interestingness.'

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

As mentioned above, the intrinsic value function and the extrinsic value function should be determined by reflecting designers' experience. The typical forms of functions illustrated in Figs. 5 and 6 represent some practically useful valuations. The designers can choose from them depending on the scale and contents of their design tasks. The applications of these value functions are not restricted to the same design tasks. They are applicable to other similar design tasks. When deciding the intrinsic and extrinsic value functions, one should pay attention to the balance between them. While the proposed method does not pose any constraint on the relationship between intrinsic and extrinsic value functions, we think both intrinsic and the extrinsic values are important. Eq. (6) shows the condition we used to try balancing the values. In Eq. (6), k represents the *upper length limit* if exits, or *maximum literal length* (i.e. the largest possible number of events constituting an event sequence). On the other hand, u represents the ceiling value of frequency of an event sequence if exits. Otherwise u equals the number of product models assuming the sequence appears once for each product model generation.

$$\int_1^k f(x)\mathrm{d}x = \int_1^u g(x)\mathrm{d}x \tag{6}$$

3.2. VMSP method

After defining what valuable patterns are, our next question is how can they be acquired? We propose a new method called Value-based Mining for Sequential Pattern (VMSP) to capture valuable sequential patterns as design procedure knowledge based on the interestingness. In order not to require 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 *template*.

Step 3 Apply the template through the whole applied field to match the similar patterns and identify the valuable j-tuple event sequences as design operations.

```
Start:
    Let j = length_lower_limit – 1
Iteration:
    Let j = j + 1
    If j > length_upper_limit Then Go to End_Iteration
    Else
      Create a new j-template from the Applied Field
      If no more j-template, Then go to Iteration
      Else
          Calculate Intrinsic Value of the j-template based on Intrinsic Value Function
          Get Frequency of the j-template through Pattern Matching
          If Frequency > frequency_threshold
          Then
            Calculate Extrinsic Value of the j-template based on Extrinsic Value Function
            Calculate Interestingness of the j-template
            Add the j-template to the OperationList
          If Stop Condition is satisfied Then Go to End_Iteration
          Else Go to Iteration
End_Iteration
    Determine operation sequence
End
```

Fig. 7. Algorithm of VMSP.

Step 4 Alter j and go to Step 2.

The variable j is an integer between the minimum length and the maximum length of the successive event sequence. It can be seen that in VMSP 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.

To create an effective algorithm of pattern matching for design operation identification, we introduce two practically important assumptions:

Assumption No. 2. A design operation often emerges with noise.

Assumption No. 3. A design operation appears literally at least once in the design process.

Assumption No. 2 arises because design is a human activity. It is usually hard for designers to perform completely 'clean' operations. Assumption No. 3 indicates that all design operations should be observable from the whole applied field of events. We do not need to worry about anything beyond this field. Our VMSP algorithm uses the two assumptions as described below.

3.2.1. VMSP algorithm

The VMSP algorithm is developed based on the definitions of intrinsic and extrinsic values and the three assumptions described above. The pseudo-code of the algorithm is illustrated in Fig. 7. As shown in Fig. 7, the core mechanisms of VMSP include: (1) create templates, (2) execute pattern matching, (3) check and stop the search, and (4) determine design operation sequences.

To further illustrate the above four 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 utilized in this case. The maximum literal length in this design process is seven. In this example, in order to simplify the understanding of the overall process of VMSP we do not use dynamic programming in pattern matching.

First of all, the intrinsic value function $f(x)$ and the extrinsic value function $g(x)$ are determined as shown in Fig. 8.

According to the functions, the lower length limit is 3, the upper length limit is 4, and the frequency threshold is 1.

Next, while j-templates—i.e. the set of all j-tuple templates $(1 \leq j \leq 5)$—are generated, the frequency of each template is investigated using pattern matching. In this example, *3-templates* include: [b, c, a], [c, a, c], etc. They are generated and kept sorted in their lexicographic order. All *3-templates* have the intrinsic value of 2.0 according to the intrinsic value function. The frequency of each 3-template is tested using pattern matching, and it turns out that only three templates, [a, d, c], [c, a, c], and [d, c, e], exceed the frequency threshold, since they all

Table 1
Example data of design process

Product model	Design events
P1	M_1 = [b, c, a, c, d, a]
P2	M_2 = [a, d, c, e]
P3	M_3 = [b, a, d, c, e, d, c]
P4	M_4 = [e, c, a, c, a]

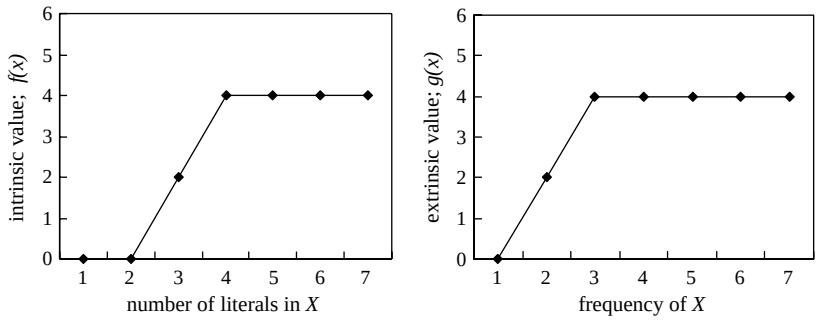


Fig. 8. Examples of intrinsic value function and extrinsic value function.

appear twice in the design process. Their extrinsic values are all 2.0 based on the extrinsic value function. Therefore, these three templates are stored in memory 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-templates reaches it, the iteration loop is stopped. When j is 3, this stop condition is not satisfied. The same procedure is performed on 4-templates, and consequently only one 4-template is stored in memory, as shown in Table 3. When j is 4, the stop condition is satisfied and the iteration loop stops.

Next comes a finalizing process, the operation determination process. 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 places 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.

Table 2
Examples of 3-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

Table 3
An Example of 4-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

3.2.2. Implementation of VMSP

For effective and efficient implementation of VMSP the following two issues must be addressed. First, how to tolerate the noise in design event sequences while searching valuable patterns, and second, how to ensure a successful discovery of the most valuable design operations without excessive calculation cost. The pattern matching using dynamic programming was employed in VMSP for the former objective. A heuristic algorithm called *Apriori* heuristic was used in VMSP for the latter one.

3.3. Pattern matching in VMSP

We introduce pattern matching using dynamic programming in order to measure $Freq(X)$, that is, how many times a template X appears in design processes. A pattern matching method in VMSP should tolerate noise, according to Assumption No. 2. Simple pattern matching such as a word search cannot satisfy this requirement. The pattern matching using dynamic programming is noise-tolerable. For example, if the sequence [w, o, r, d] is an original sequence where letters are literals, [w, o, a, r, d] and [w, r, o, d] can be recognized as the same.

Originally, dynamic programming was an approach to solving sequential decision problems, developed by Richard

Bellman in 1957. The simplest dynamic programming context involves an n-step decision-making problem, where the states reached after n steps are considered terminal states and have known utilities. The main concept of dynamic programming is that choosing the best state in each step leads to the optimum final state. Recently, dynamic programming has been applied to solving various pattern matching problems, e.g. speech recognition [26], image recognition [6], and bioinformatics [16].

Dynamic programming enables us to detect not only the same sequence as the template but also approximately similar ones. It allows us to deal with noisy design event data, such as missing, additional or substituted events. However, our pattern matching in VMSP allows only insertion (addition) and substitution (interchange) of literals. This means that VMSP recognizes substrings of a sequence as different ones from an original sequence. The reason is as follows: our interestingness of a template is determined by multiplying the intrinsic value and the extrinsic value. It is tacitly based on the assumption that all sequences counted as reappearance of templates have the same information content. If we allow a deletion of literals of a template while pattern matching, the assumption is broken. In practice, to recognize only the insertion and substitution of literals, we adjust the weights used in dynamic programming.

3.4. Apriori Heuristic in VMSP

VMSP can capture sequences as design operations that have the highest value of interestingness. Moreover, VMSP saves calculation cost based on the *Apriori* heuristic. The Apriori heuristic was first proposed in mining association rules between items in a large database of sales transactions [1]; *the frequency of any super-pattern of a given pattern is always equal or less than the frequency of the given pattern*. Although the Apriori heuristic enables methods of mining association rules to reduce the search space, it works in VMSP to determine when the search for templates should stop.

There are three stop conditions: (1) In the case that the upper length limit exists and the extrinsic value function is a monotonic function, when the number of literals in templates (i.e. j of j-template) reaches the upper length limit, VMSP stops the generation-and-test process and does not make $(j+1)$-templates. We call this situation 'the search stops.' (2) If all j-templates get their frequency equal to or less than the frequency threshold while searching, the search stops. Or (3) if j of j-template reaches the maximum literal length, the search stops. The third condition is a full search of all the possible templates, independent of the Apriori heuristic. However, the first and second conditions are based on the Apriori heuristic.

The first condition can be explained as follows: Let k_{upper} be the upper length limit. The intrinsic value of all k_{upper} templates equals $f(k_{\text{upper}})$. If k_{upper} exists, then formula (7) stands true.

$$f(k_{\text{upper}}) = f(k_{\text{upper}}+1) = f(k_{\text{upper}}+2) = \cdots = f(k_{\max}) \tag{7}$$

where $k_{\max}$ is the maximum literal length

On the other hand, formula (8) stands true based on the Apriori heuristic.

$$\text{max of Freq}(T_{k_{\text{upper}}}) \geq \text{max of Freq}(T_{k_{\text{upper}}+1}) \geq \cdots \geq \text{max of Freq}(T_{k_{\max}}), \text{ where } T_{\text{j}} \text{ represents } j\text{-template.} \tag{8}$$

Moreover, if the extrinsic value function is a monotonic function, then formula (9) stands true.

$$g(\text{max of Freq}(k_{\text{upper}}) - \text{template})\ g(\text{max of Freq}(T_{k_{\text{upper}}})) \geq g(\text{max of Freq}(T_{k_{\text{upper}}+1})) \geq \cdots \geq g(\text{max of Freq}(T_{k_{\max}})) \tag{9}$$

In this case, formula (10) stands true supported by formulas (7) and (9).

$$\text{max of } I(T_{k_{\text{upper}}}) \geq \text{max of } I(T_{k_{\text{upper}}+1}) \geq \cdots \geq \text{max of } I(T_{k_{\max}}) \tag{10}$$

According to formula (10), the search after k_{upper}-templates is not needed to obtain the template that has the highest interestingness.

Next, the second condition can be proved as follows: Let k_{t} be the length, where all k_{t}-templates get their frequency equal to or less than the frequency threshold. From formula (4) and the Apriori heuristic, formulas (11)–(13) stand true.

$$\text{max of Freq}(T_{k_{\text{t}}}) \leq \beta,\ \text{max of Freq}(T_{k_{\text{t}}+1}))) \leq \beta, \cdots \tag{11}$$

, where β represents the frequency threshold.

$$g(\text{max of Freq}(T_{k_{\text{t}}})) = g(\text{max of Freq}(T_{k_{\text{t}}+1})) = \cdots = 0 \tag{12}$$

$$I(T_{k_{\text{t}}}) = I(T_{k_{\text{t}}+1}) = \cdots = 0 \tag{13}$$

According to formula (13), the search after k_{t}-templates is not needed to obtain the template that has the highest interestingness.

4. Case study

We have tested the VMSP method on two design problems: one is double-reduction gear system design and the other is car front door design. Since we had already acquired design operations using GEDP method for the gear design problem [13], the objective of the gear design experiment was to compare the captured operations of VMSP with the ones from GEDP. The car door design problem is a more practical one. The objective of the experiment with this problem is to qualitatively evaluate the effectiveness of VMSP.

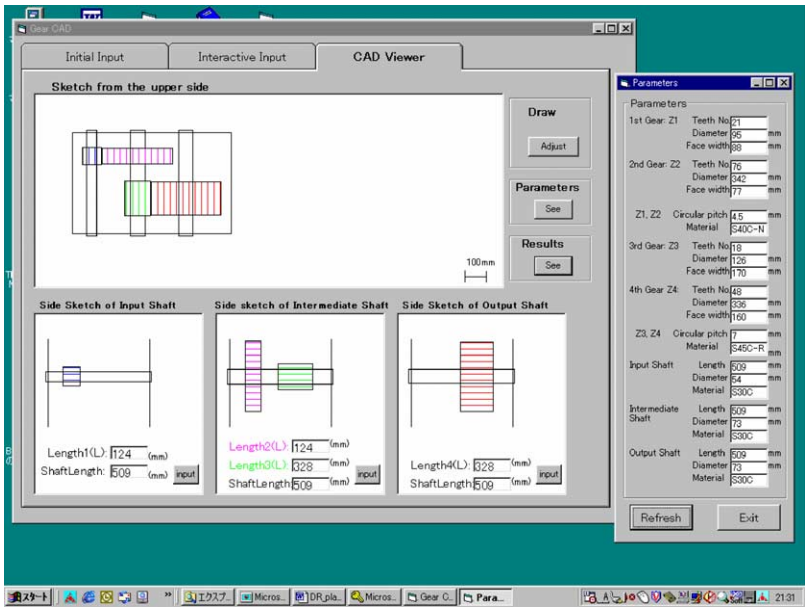


Fig. 9. Graphical user interface of GearCAD.

4.1. VMSP for gear system design

4.1.1. Design task

The double-reduction gear system is composed of four gears, three shafts, bearings, and a case. Since the power of the revolution makes the torque and the bending moment, gears and shafts are designed to stand up to the force. We developed a 2D CAD system called 'Gear-CAD.' Gear-CAD is equipped with needed domain knowledge for supporting design of double-reduction gear systems. Besides, it records the designer's operational events and the status of the objects throughout the design process. Fig. 9 shows an example of the Gear-CAD screens.

The constraints and requirements for the gear design problem were:

Constraints:

(1) Spur gears that have teeth with a 20-degree pressure angle are utilized in this system.
(2) The input power and speed of rotation are 10.0 kW and 500 rpm, respectively.

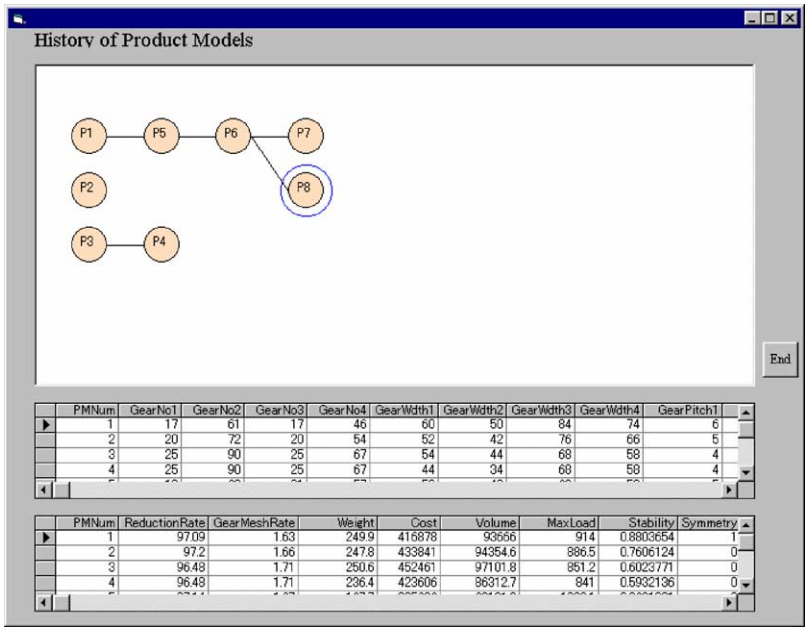


Fig. 10. Hierarchical structure of product design (gear design).

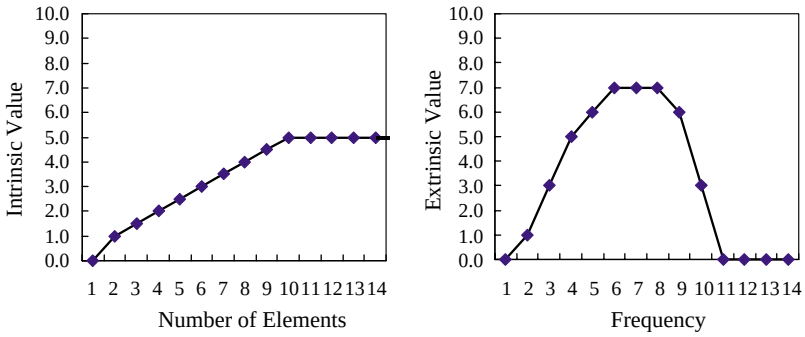


Fig. 11. Intrinsic and extrinsic value functions on gear design.

Requirements:

(1) All design components are determined in detail, i.e. size and position.
(2) Required reduction ratio is 10:1.
(3) Lighter, smaller, and cheaper is better (because the gear is to be used in space).

4.1.2. Monitored design events

In our gear design experiment, a user with enough knowledge of gear design designed the double-reduction gear system using Gear-CAD. Gear-CAD stored 472 events in a list during the design process. The list consists of the event-ID and the associated action, e.g. 'Event-ID 5; See document No. 2' and 'Event-ID 150; Input the number of gear teeth of pinions.'

And then, eight product models were obtained in this case. The parent–child relationships among product models are shown in Fig. 10, where the final product model chosen was product model No. 8.

4.1.3. VMSP application to design event log

We set the intrinsic value function and the extrinsic value function as shown in Fig. 11. The shapes of the two value functions were determined by the subjective intuition of the designer who executed this design task.

VMSP was applied to the data where 472 design events had been divided into eight product models, and 11 design operations were discovered as design operations. The result is shown in Table 4.

4.1.4. Comparison of GEDP and VMSP results

We previously indicated that GEDP successfully acquired beneficial design procedure knowledge using a case study of a double-reduction gear system [13]. Although GEDP is a useful method to obtain design procedure knowledge, it needs predefined classification rules, which is a burden for users. In contrast, VMSP has the advantage of not requiring any predefined classification rules. The objectives of GEDP and VMSP are the same; to acquire beneficial design procedure knowledge. To ensure that VMSP, without predefined rules,

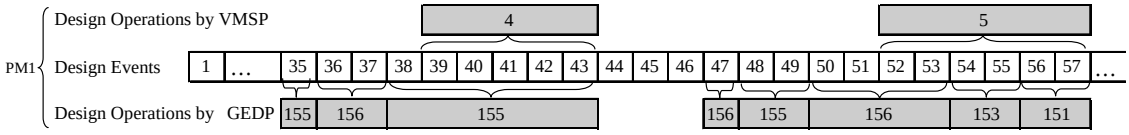


Fig. 12. Example of design operations from different algorithms.

Table 4
Obtained design operations by VMSP on gear design

ID	Sequence as operations	Frequency	Intrinsic value	Extrinsic value	Interestingness
1	[E21,E30,E31,E130,E129,E125,E126,E41,E50,E51]	4	5.0	5.0	25.0
2	[E67,E69,E82,E133,E83,E84,E134,E85,E135,E86]	4	5.0	5.0	25.0
3	[E132,E131,E127,E128,E60,E91]	4	3.0	5.0	15.0
4	[E93,E94,E96,E95,E97]	5	2.5	6.0	15.0
5	[E94,E96,E98,E91,E92]	4	2.5	5.0	12.5
6	[E97,E65,E65,E67,E67,E67,E66,E69]	3	4.0	3.0	12.0
7	[E87,E137,E89,E138,E88,E136]	3	3.0	3.0	9.0
8	[E123, E124]	6	1.0	7.0	7.0
9	[E93,E94,E96,E98,E67,E67]	2	3.0	1.0	3.0
10	[E82,E133,E83,E85,E135,E86]	2	3.0	1.0	3.0
11	[E87,E137,E88,E136,E89,E138]	2	3.0	1.0	3.0

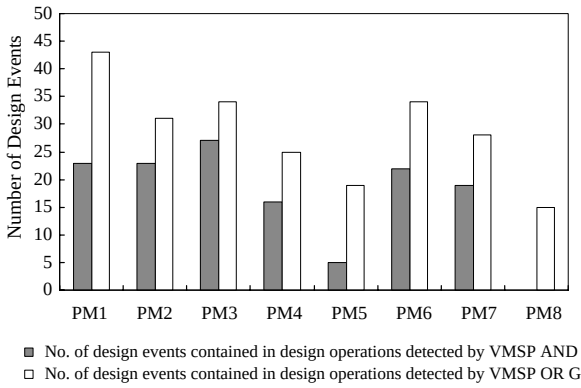


Fig. 13. Comparison of design operations from two different algorithms.

can obtain design procedure knowledge to the same level as GEDP did, we applied both methods for a double-reduction gear system design and compared the results.

The designer who had conducted design of the double-reduction gear system knew that one of the most important operations of this design task was related to the change of the geometric position and size of the gears and shafts. We picked up the sequences that were supposed to be connected with such changes from a set of operation sequences obtained by VMSP. The selected sequences were VMSP ID 2, 4, 5, 6, 9, and 10 in Table 4. On the other hand, we had analyzed the same data by GEDP. The design operations ID 14, 151, 152, 153, 154, and 155 in GEDP were regarded as ones connected to the change mentioned above. We compared the design operations, respectively, obtained by the two algorithms. Fig. 12 gives us a partial illustration of the outcome. For example, VMSP captured a design operation from design event ID 52–57, though GEDP captured three different design operations from the corresponding sequence: the first from design event ID 50–53, the second from design event ID 54–55, and the last from design event ID 56–57.

We calculated some index numbers to compare two algorithms: (1) the total number of obtained design operations related to the change of the geometric position and size of the gears and shafts, (2) the average number of design events composing the design operations, (3) the number of design events contained in the design operations detected both VMSP *and* GEDP in each product model, and (4) the number of design events contained in the design operations detected VMSP *or* GEDP in each product model. In the case of VMSP the values of index-1 and index-2 were, respectively, 20 and 7.1, though GEDP yielded 75 and 3.0 for those values. This means that GEDP tends to capture design operations in further detail in comparison with VMSP. This occurred because VMSP can acquire only those design procedures that repeatedly appear in design process. However, as far as the design procedure knowledge is concerned, the contents of design operations captured by GEDP and VMSP were very similar, as shown in Fig. 13 where the values of index-3 and index-4 are indicated. The high ratio of index-3 in index-4 means the design operations obtained by the two algorithms are overlapped to each other to a great extent. Fig. 13 shows that the similarity is conspicuously observed in PM1, PM2, PM3, PM4, PM6 and PM7. The design operations obtained by VMSP were significantly included in design operations by GEDP. We can see that the sequences obtained by VMSP are not exactly the same as the ones by GEDP, but they correlate greatly well with each other. This means that the sequential patterns discovered by VMSP without predefined GEDP rules can be very valuable in practice. Manually refining

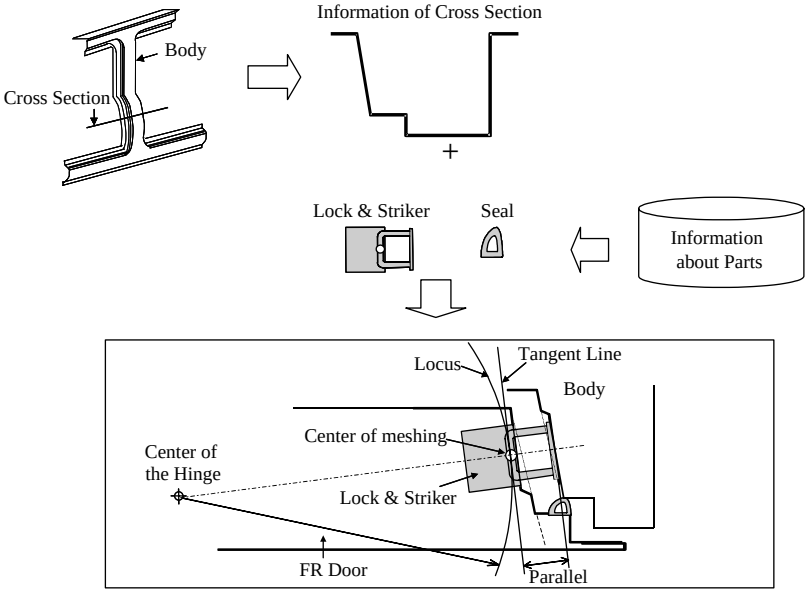


Fig. 14. Design of front door of car.

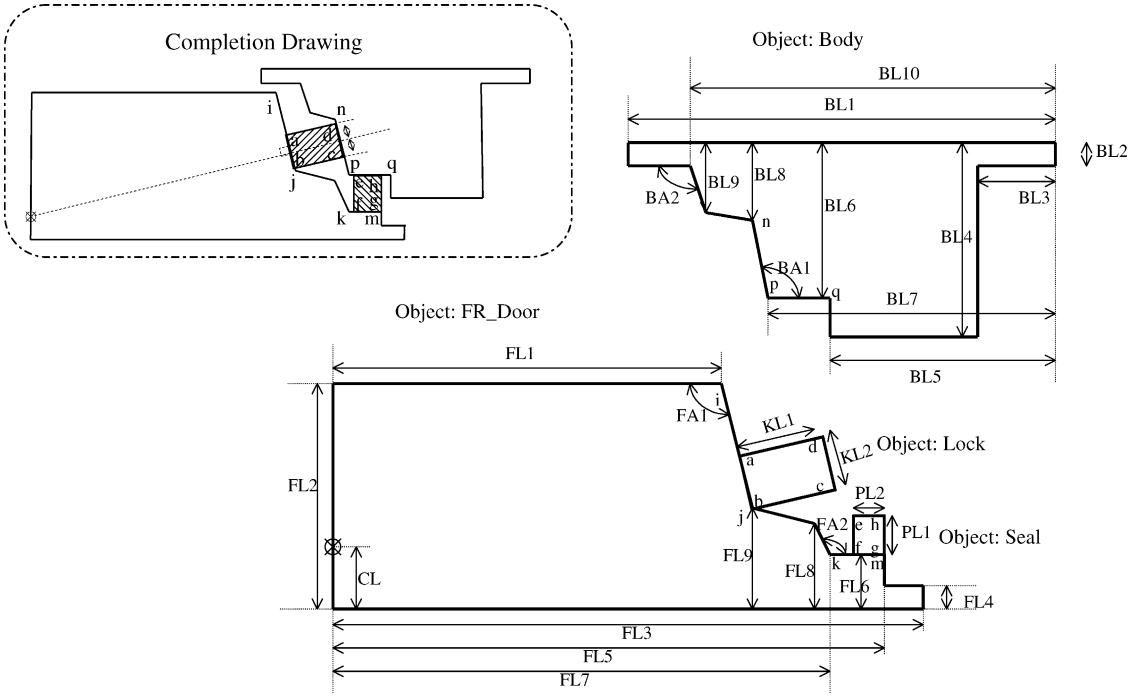


Fig. 15. Geometric model of front door design.

the sequences makes them more reliable in terms of the meanings of the sequences.

4.2. VMSP for front door design of an automobile

4.2.1. Design task

In practice, designing a car is decomposed into many sub-activities [2]. From the automotive design process documented by Araki [2], we picked up one of the sub-activities, designing the front door, and applied design data presented in [2] in our case study. In this design task a designer has to design all of the geometric positions and size 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 Fig. 14. This design was built as a 2D parametric design using a commercial CAD system. The wrapper module was developed and plugged into the CAD to monitor design events. The geometric model of this design is shown in Fig. 15. There are 28 parameters in total.

Since we prepared three design cases, (A) a middle-sized car, (B) a compact car, and (C) a luxury car, a skillful designer dealt with all design 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 Fig. 15)

Table 5
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

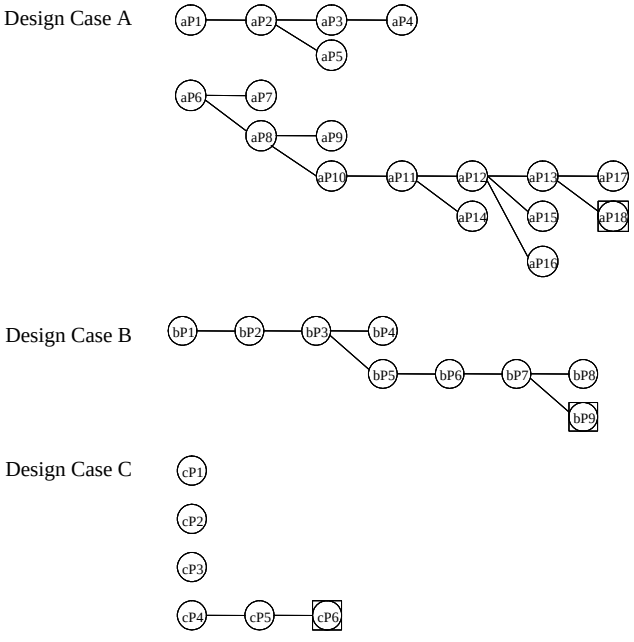


Fig. 16. Hierarchical structure of product models (front door design).

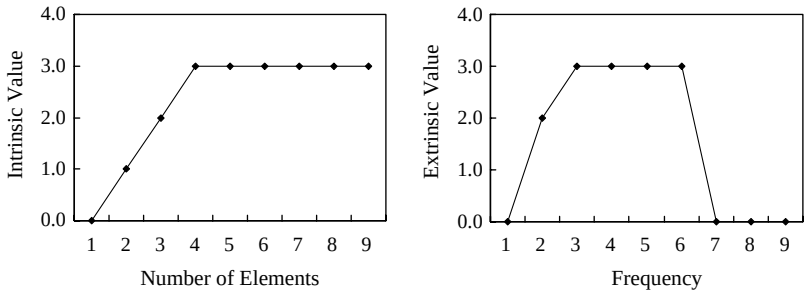


Fig. 17. Intrinsic and extrinsic value functions on front door design.

$\overline{ab}$ 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.
Common requirements:

$\overline{cd}$ is on $\overline{pn}$, $\overline{eh}$ is on $\overline{pq}$,

$9 \leq KL1 \leq 16$, $8 \leq KL2 \leq 14$,

$$FL9 = FL1 \cdot \cos\theta \sin\theta + FL2 \cdot \sin^2\theta + CL \cdot \cos^2\theta - KL2 \cdot \cos\theta$$

, where $\theta = FA1\text{-}90°$

This equation in common requirements indicates the following: The radius of the circular arc that has one endpoint at the center 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 center of the lock and striker part should be on the intersection of the radius and the segment $\overline{ij}$.
Case-specific requirements:
Shown in Table 5.

4.2.2. Monitored design events

A designer who had sufficient knowledge to accomplish this design executed all three design cases A, B, and C in this order. In design case A, 74 design events were yielded and 18 product models were detected by analyzing the status 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. The hierarchical structures of product models on each design case are shown in Fig. 16.

Table 6
Obtained design operations by VMSP on front door design

ID	Sequence as Operations	Frequency	Intrinsic Value	Extrinsic Value	Interestingness
1	FL1-FL2-FL9-KL2	5	3.0	3.0	9.0
2	FL2-FL9-BL9-BL8	3	3.0	3.0	9.0
3	FL2-FL9-FL1	3	2.0	3.0	6.0
4	BA1-FA1-FL1	2	2.0	2.0	4.0
5	FL4-FL6-BA1	2	2.0	2.0	4.0
6	FL6-FL8	2	1.0	2.0	2.0

4.2.3. VMSP application to design event log

The intrinsic value function and the extrinsic value function we set are shown in Fig. 17. These were determined based on the hearing from the designer who had accomplished these three design cases.

VMSP applied to all three design cases together as if one huge design process had included all 33 product models. Six sequential patterns were discovered as valuable design operations, as shown in Table 6.

4.2.4. Results of VMSP

The obtained design operations were located on the design process of each design case, as shown in Fig. 18.

In order to examine how obtained sequences of design events were qualified to be practical design operations, we interviewed the designer who had accomplished these

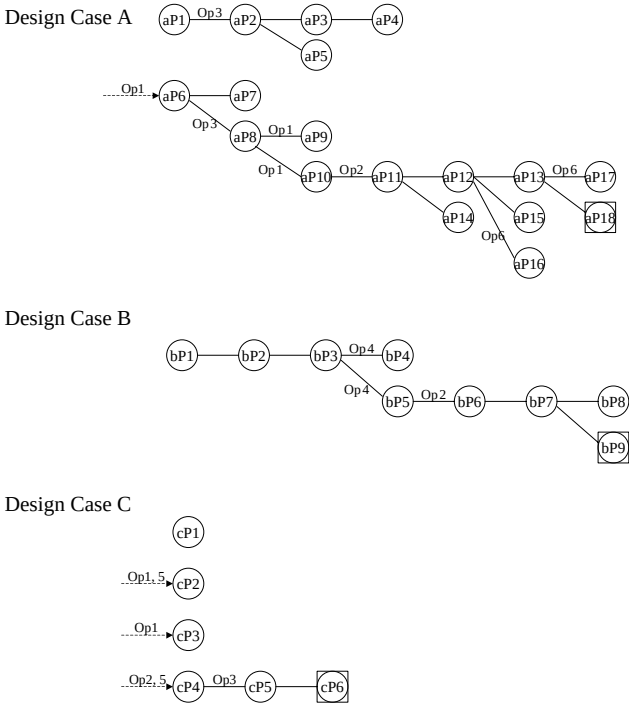


Fig. 18. Emergence of obtained design operations in design process.

design cases by showing her Table 6 and Fig. 18. She evaluated the sequence ID 1, 2, 4, and 6 as important operations. Following is a summary of her comments:

- The sequence No. 1 seemed to be executed to meet the common requirement expressed by the following equation.

$$FL9 = FL1 \cdot \cos\theta \sin\theta + FL2 \cdot \sin^2\theta + CL \cdot \cos^2\theta - KL2 \cdot \cos\theta,$$

, where $\theta = FA1\text{-}90$

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 five parameters should be determined. Since FA1 was an angle and the change of its one-degree affected other parameters much greater than other parameters did, the point was that first FA1 should be examined and determined and then other FL1, FL2, FL9, and KL2 be adjusted to meet the requirement. The sequence No. 1 represented the natural order of changing parameters, FL1, FL2, FL9, and KL2, from large parameter to small parameter. Since the requirement was the most important and difficult requirement, the sequence No. 1 tended to reappear in early stage of design process.

- The sequence No. 2 seemed to be executed to adjust the space between the front door and the body in order to put the lock and striker part. After the requirement mentioned above was satisfied, sometimes came the problem there was not enough space left for the lock and striker part. The series of design events, changing the parameters FL2, FL9, BL9, and BL8, behaved to solve it by changing vertical length of the front door and the body.
- The sequence No. 4 seemed to be 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. The sequence No. 4 might indicate the following things: When the necessity to change the parameter, either BA1 or FA1, occurred in the middle of design, both of them should be also changed. Since the change of these angles often affected the horizontal length of the space between the front door and the body, successively the length of FL1 should be changed.
- The sequence No. 6 seemed to be conducted to adjust the vertical space for the seal part. This was a small change for the design as a whole, and executed in the late stage of the design.

According to the designer's comment, the sequences obtained by VMSP can approximate significant design operations in practice. Although the sequential patterns discovered by VMSP cannot always become meaningful design operations, they can be used by designers to recall the significant design operations they did in their design processes. This experiment showed that VMSP has the potential to produce valuable design procedures even in practical design cases using a commercial CAD system. It is important to note that VMSP captures design operations without predefined design-task-specific rules. Instead, it uses intrinsic and extrinsic values functions which are more design task independent and can be shared across given design domains.

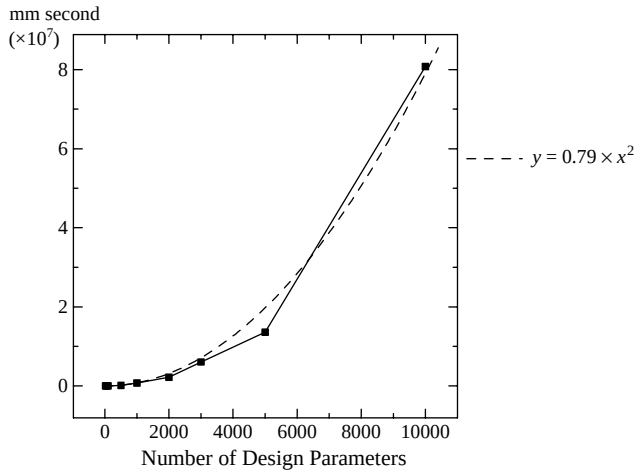


Fig. 19. Required time for acquiring target patterns according to data size.

4.3. VMSP for synthetic data

To evaluate the scalability of VMSP, we applied it to synthetic event data sets generated by a computer program. The parameters used for generating data include: (1) the total design event numbers, (2) the total number of product models in design process, (3) the maximum number of design events in a product model, (4) the minimum number of design events in a product model, (5) the maximum threshold of a target sequence length of design events as a design operation, (6) the number of a target model pattern in design process. We made 27 data sets by changing values of parameter No. 1 to No. 5. The pattern such as '108, 300, 412, 800, 707, 228, 589' was set as a target model of a design operation to be captured. Then the target model was randomly inserted using parameter No. 6. We changed parameter No. 1 from 20 to 10000 and parameter No. 2 from 10 to 500, so that the size of test data set was changed from 390 byte to 153 kb. The required time for calculation was shown in the Fig. 19. The required time for acquiring design operations tends to increase not linearly but 2-order exponentially, corresponding to the number of existing design parameters. However, generally speaking, the number of design parameters in a normal engineering design task in industry is 3000–5000 at the most. In our experiment, the case with 5000 design parameters required 13589137.0 mm s (=3.77 h). It is reasonable in time scale. As a result, VMSP showed the ability to be used in actual engineering design.

5. Related work

Capturing design procedure knowledge of experienced designers is important for effective engineering knowledge management. While many researchers in engineering and artificial intelligence have focused their knowledge capturing research on documented domain knowledge [5], research on capturing *know-how* knowledge such as design procedures received little attention. One reason for this is that *know-hows* and *procedures* are much domain-specific and it is difficult to deal with the capturing problem in a general way. Despite the difficulty, some researchers pointed out its importance and attempted to tackle the problem; e.g. Knowledge Infrastructure for Collaborative and Agent-based Design (KICAD) was proposed to use agent-based systems to capture and manage procedure knowledge in collaborative design [14,15]. However, the general way to capture the design procedure knowledge remains to be a challenging research topic.

On the other hand, the research on *design rationale* has been done from different point of views [22]. There are three major models: argumentation-based design rationale such as the Issue-Based Information System (IBIS) [17], action-based design rationale [18], and model-based design rationale such as the Active Design Document (ADD) system [8]. Recently, as the computer systems supporting all aspects of the design process evolved, many have pursued the integration of the record of design rationale with the history of the evolution of the design artifact [19,27]. Furthermore, a number of recent design rationale systems including PHIDIAS [28] and KBDS-IBIS [3] were proposed. In the field of design rationale, the research focus is on capturing *why* a specific design feature was designed the way it is. The process of *how* it was designed is not an issue. Moreover, few researchers cared about not to interrupt designers' normal thinking process. Many require designers to record reasons during the design process.

Several researches in the action- and model-based design area worked on both *know-how* and *know-why* knowledge capturing. Ganeshan et al. [7] proposed the framework to capture *how* and *why*, in which the core idea was 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. proposed a framework to capture design rationale from a general CAD database [23]. 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 proposed a Grammar and Extended Dynamic Programming Approach (GEDP) [12,13]. 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. So, these methods are limited by high cost of creating predefined rules. Our VMSP approach, however, does not need any predefined rules. It is simple and applicable to a wide range of design tasks.

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 [20]. Goel et al. directly addressed the representation of causal behavior through the use of *structure-behavior-function* (SBF) models [9]. SBF models are useful to diagnose products. However, they focused on states of product models rather than design history.

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 analyzing 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 data mining domain [11]. To date, many techniques have been published such as Piatetsky–Shapiro's rule-interest function [25], Smyth and Goodman's *J*-Measure [29], Agrawal and Srikant's itemset measures [1], and Gray and Orlowska's interestingness [10]. There have also been many application studies of the techniques in many fields such as marketing [1] and medical domains [21,24]. For example, Piatetsky–Shapiro introduced his rule-interest function which was used to quantify the correlation between two attributes arbitrarily chosen, and the obtained knowledge is expressed as classification rules [25]. 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 health-care data [21]. The survey paper [11] 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 treat the time series sequential patterns of the components. On the other hand, our proposed measure and

implemented method focused on the sequential patterns frequently appeared and considered to be valuable. The sequential patterns are not combinations of attributes or components, but permutations of them. We introduced a novel function to express the interestingness of sequential patterns considering sequential variations occurred with components' insertions and substitutions.

6. Concluding remarks

Since engineering design is highly knowledge intensive, capturing, managing, and utilizing knowledge can yield significant benefits for both designers and enterprises. In this paper we focus on capturing *design procedure* knowledge through monitored operational events from CAD systems. We proposed a novel method called Value-based Mining for Sequential Pattern (VMSP). VMSP does not require any predefined and task-specific rules for classification of design events. The core concept of VMSP is to regard a list of design events as a sequential pattern and search for frequent and informative patterns as design operations. VMSP automatically generates templates of sequential patterns, and then identify valuable sequential patterns as design operations based on the templates and the *interestingness* defined by non-task-specific *intrinsic* and *extrinsic* value functions. The captured design procedure knowledge shows how a designer approaches and achieves a final product model, and what are the key design operations for a specific design task. We have evaluated VMSP through two case studies in which practical CAD data monitored through design processes were used.

The advantage of our approach is that a user can utilize VMSP without having to make predefined rules about the design task concerned. We achieved this advantage by introducing a meta-level concept called value function. Comparing with predefined pattern matching rules, value functions are more general and usually applicable to a wide range of design tasks. Our algorithm allows this important saving of rule-definition cost without substantial increase of computation cost. The Apriori heuristic and our fetch-generation mechanisms described in Section 3.2 limit the maximum number of templates proportional to the number of product models and events.

One issue of our approach is that some of the sequential patterns discovered by VMSP may not be literally meaningful design operations. Manually refining the sequences obtained by VMSP is needed to make them more meaningful. Moreover, the tuning up of the intrinsic and extrinsic value functions is sometimes needed to acquire more reasonable sequences. Despite the limitations, we have demonstrated that VMSP can produce meaningful primitives of design operations that can help designers to recall their practical design procedures and make this know-how knowledge explicit. Our current research addresses these limitations and explores issues related to the scalability of this method.

Acknowledgements

This research was supported in part by a NSF CAREER Award under grant DMI-9734006. Additional support was provided by industrial sponsors. The authors are grateful to NSF and industrial sponsors for their support.

References

[1] Agrawal R, Srikant R. Fast algorithms for mining association rules. Proceedings of the 1994 international conference very large data bases (VLDB'94) 1994 p. 487–99.
[2] Araki, Hiromi (2000). A Study of the collaborative environment for product development process innovation. Doctoral Thesis, University of Tokyo, Nov. 2000.
[3] Banares-Alcantara R, King JMP. Design support systems for process engineering iii—design rationale as a requirement for effective support. Comput Chem Eng 1997;21(3):263–76.
[4] Bernardo Y, Rothe A. Knowledge management in engineering: supporting analysis and design processes in innovative industries. In: Cunningham P, Cunningham M, Fatelnig P, editors. Building the knowledge economy, issues, applications, case studies. Amsterdam: IOS Press; 2003. p. 931–8.
[5] Bradshaw JM, Carpenter R, Cranfill R, Jeffers R, Poblete L, Robinson T, et al. Roles for agent technology in knowledge management: examples from applications in aerospace and medicine. Proceedings of the AAAI spring symposium on artificial intelligence in knowledge management (AIKM'97) 1997.
[6] Chikada T, Yoshimura M. An off-line signature verification method based on a hidden Markov model using column images as features. Proceedings of the 9th Biennial conference of the international graphonomics society (IGS'99) 1999 p. 79–82.
[7] Ganeshan R, Garrett J, Finger S. A framework for representing design intent. Design Stud 1994;15(1):59–84.
[8] Garcia ACB, Howard HC. Acquiring design knowledge through design decision justification. Artificial Intelligence Eng Des Anal Manuf 1992;6(1):59–71.
[9] Goel A, Bhatta S, Stroulia E. KRITIK: an early case-based design system. Issues and applications of case-based reasoning in design 1997 p. 87–132.
[10] Gray B, Orlowska ME. Ccaiia: clustering categorical attributes into interesting association rules. In: Wu X, Kotagiri R, Korb K, editors. Proceedings of the second Pacific-Asia conference on knowledge discovery and data mining (PAKDD'98), Melbourne, Australia, 1998. p. 132–43.
[11] Hilderman R, Hamilton H. (1999). Knowledge discovery and interestingness measures: a survey. Technical Report CS 99-04, Department of Computer Science, University of Regina.
[12] Ishino Y, Jin Y. Data mining for knowledge acquisition in engineering design. Data mining for design and manufacturing: methods and applications. New York: Kluwer; 2001.
[13] Ishino Y, Jin Y. Acquiring engineering knowledge from design processes. Artif Intelligence Eng Des Anal Manuf 2002;16:73–91.
[14] Jin Y, Zhou W. Agent-based knowledge management for collaborative engineering. Proceedings of the design engineering technical conferences (DETC'99) in ASME 1999.
[15] Jin Y, Zhao L, Raghunath A. ActivePROCESS: a process-driven and agent-based approach to collaborative engineering. Proceedings of the design engineering technical conferences (DETC'99) in ASME 1999.
[16] Krogh A, Brown M, Mian IS, Sjölander K, Haussler D. Hidden Markov models in computational biology: application to protein modeling. J Mol Biol 1994;235:1501–31.

[17] Kunz W, Rittel W. Issues as elements of information systems. Working paper 131, center for planning and development research. Berkeley: University of California; 1970.

[18] Lakin F, Wambaugh J, Leifer L, Cannon D, Steward C. The electronic design notebook: performing medium and processing medium. Visual Comp: Int J Comp Graphics 1989;5:214–26.

[19] Liang J, Shah JJ, D'Souza R, Urban SD, Ayyaswamy K, Harter E, Bluhm T. Synthesis of consolidated data schema for engineering analysis from multiple STEP application protocols. Comput-Aided Des 1999;31(7):429–47.

[20] Maher ML, Pu P, editors. Issues and applications of case-based reasoning in design. London: Lawrence Erlbaum; 1997.

[21] Matheus CJ, Piatetsky-Shapiro G. Selecting and reporting what is interesting: the kefir application to healthcare data. In: Fayyad UM, Piatetsky-Shapiro G, Smyth P, Uthurusamy R, editors. Uthurusamy, advances in knowledge discovery and data mining. Menlo Park, CA: AAAI Press/MIT Press; 1996. p. 495–515.

[22] Moran TP, Carroll JM. Design rationale: concepts, techniques, and use. Mahwah, New Jersey: Lawrence Erlbaum Associates; 1996.

[23] Myers KL, Zumel NB, Garcia P. Automated capture of rationale for the detailed design process. Proceedings of the 11th conference on innovative applications of artificial intelligence (IAAI'99) 1999.

[24] Ohsaki M, Sato Y, Kitaguchi S, Yokoi H, Yamaguchi T. Comparison between objective interestingness measures and real human interest in medical data mining. Proceedings of the 17th industrial and engineering applications of artificial intelligence and expert systems, (IEA/AIE2004, LNAI3029) 2004 p. 1072–81.

[25] Piatetsky-Shapiro G. Discovery, analysis and presentation of strong rules. Knowledge discovery in databases.: AAAI/MIT Press; 1991 p. 229–48.

[26] Sakoe H, Chiba S. In: Waibel A, Lee K, editors. Dynamic programming algorithm optimization for spoken word recognition, readings in speech recognition. San Mateo, CA: Morgan Kaufmann; 1990. p. 159–65.

[27] Shah JJ, Rangaswamy S, Qureshi S, Urban SD. Design history system: data models and prototype implementation. In: Finger S, Tomiyama T, Mantyla M, editors. Knowledge intensive computer aided design. New York: Kluwer; 2000. p. 91–114.

[28] Shipman III FM, McCall RJ. Integrating different perspectives on design rationale: supporting the emergence of design rationale from design communication. Artif Intelligence Eng Design, Anal Manuf 1997;11(2):141–54.

[29] Smyth P, Goodman RM. Rule induction using information theory. Knowledge Discovery in Databases.: AAAI/MIT Press; 1991 p. 159–76.