Expert Systems with Applications 29 (2005) 913–925

Expert Systems with Applications

www.elsevier.com/locate/eswa

Applying case-based reasoning for product configuration in mass customization environments

Hwai-En Tseng[a,*], Chien-Chen Chang[b], Shu-Hsuan Chang[c]

[a]*Institute of Production System Engineering and Management, National Chin-Yi Institute of Technology, 35, Lane 215, Section 1, Chung-Shan Road, Taiping City, Taichung County, 411 Taiwan, ROC*
[b]*Department of Industrial Design, Huafan University, No. 1, Huafan Road, Shihtin Hsiang, Taipei Hsien, Taiwan, ROC*
[c]*Department of Industrial Education, National Changhua University of Education, 1 Jin-De Road, Changhua 500, Taiwan, ROC*

Abstract

Product variation and customization is a trend in current market-oriented manufacturing environment. Companies produce products in order to satisfy customer's needs. In the customization environment, the R&D sector in an enterprise should be able to offer differentiation in product selection after they take the order. Such product differentiation should meet the requirement of cost and manufacturing procedure. In the light of this, how to generate an accurate bill of material (BOM) that meets the customer's needs and gets ready for the production is an important issue in the intensely competitive market.

The purpose of this study is to reduce effectively the time and cost of design under the premise to manufacture an accurate new product. In this study, the Case-Based Reasoning (CBR) algorithm was used to construct the new BOM. Retrieving previous cases that resemble the current problem can save a lot of time in figuring out the problem and offer a correct direction for designers. When solving a new problem, CBR technique can quickly help generate a right BOM that fits the present situation.

Keywords: Mass-customization; Product configuration; Case-based reasoning; Bill of material; Feature tree

1. Introduction

The manufacturing trend of producing a smaller number but wider variety of products forces enterprises to adopt differentiation strategy to offer customers more choices of products. Such kind of variation strategy often makes the interwoven constraint relationship of products even more complicated, which is one of the characteristics of in a customization manufacturing environment (Jiao, Ma, & Tseng, 2003; Salvado & Forza, 2004). Fohn, Liau, Greef, Young, and O'Grady (1995) once used computers as a case study and demonstrated that approximately 30–85% of product information was wrong and that this kind of mistake would causes in engineering design and substantial burden to an enterprise. Therefore, how to bring the complexity and accuracy of product configuration into control has become one of the important challenges enterprises have to face nowadays. In dealing with product configuration, it is easy to lose control of product configuration due to the incomplete communication or cognition conflict if an enterprise totally depends on the knowledge or experience of the professional personnel. This will increase the difficulty of design alternation and the pressure of cost.

Different approaches have been adopted to solve the product configuration problem. For example, the generic bill of material (GBOM) concept had been used to solve the problem of product configuration management (Hegge & Wortmann, 1991; Jiao, Tseng, Ma, & Jhou, 2000; Olsen & Saetre, 1997) and the object-oriented concept had been used to replace traditional database viewpoint (Kobler & Norrie, 1997). Constraint Satisfactory Problem (CSP) Algorithm offers another way to solve the product configuration problem (Ryu, 1999). Jiao et al. (2003) claimed that it is necessary to build a Product Family Structure (PFA), which could adjust the new product variation and satisfy

* Corresponding author. Tel.: +886 4 2392 4505x6001; fax: +886 4 2393 0062.
E-mail address: hwai_en@seed.net.tw (H.-E. Tseng).

0957-4174/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2005.06.026

the customer's needs. Simpson (2001) attempted to establish a product variety tradeoff evaluation method, which applied goal programming and statistical analysis techniques to optimization of product family. Du, Jiao, and Tseng (2002) dealt with product variation and flexibility by a graph method similar to a programming syntax with the viewpoint of product family design. In general, current researchers in the field mostly focus on the issues about creating information system environment and solving optimization-based problem for product family design.

As a mater of fact, the maintenance of accurate product configurations starts right after an order is placed. After the confirmation of customers, an initial product configuration can be quickly generated. With the data transmitted to R&D sector, it is sometimes necessary to redesign and reorganize these data so as to generate accurate BOM that will guarantee the smooth production procedure. If previous successful cases can be fully applied to the design alternative derived from customization, the error rate of BOM will be lowered, thus enhancing the commonality of parts of products and reducing the total cost of an enterprise. Different from traditional views of customization, the case-based database is built to solve product configuration problem. case-based reasoning (CBR) can help solve the problem through the retrieval of similar previous cases (Kolodner, 1993). In terms of product configuration, this approach has the following the advantages:

(1) It reuses the previous successful reasoning case to solve a new problem an enterprise is encountered with.
(2) Through previous successful cases, the same mistakes can be avoided and alternatives can be generated to improve the quality of problem solving.
(3) It is easy to collect previous failed or successful cases, which reduces the bottleneck of knowledge retrieval.
(4) CBR can prevent the loss of an enterprise know how when experienced technicians leave a company.

In this paper, integration of graph-based BOM tree and CBR is explored for the mass customization environment. Basic ideas regarding CBR are reviewed in Section 2. In Section 3, the proposed CBR algorithms are discussed with a ballpoint pen as an illustrated example. In Section 4, a CNC lather is used as an example to verify the mythology mentioned in this study. Finally, conclusions are made and future work is suggested in Section 5.

2. Basic concepts of case-based reasoning

There are two fundamental concepts for CBR. One is that similar problems will have similar solutions. The other is that the same problems will often occur. More importantly, CBR simulates the human problem-processing model and can have the self-learning function by constant accumulation of past experience. When the user enters a new problem in CBR, CBR will search for the data that have the highest similarity with the existing cases and adjust the previous cases to suit the new problem. General CBR algorithms are composed of the following steps (Kim & Han, 2001; Kolodner, 1993):

Step 1: Index assignment. Classify cases in the database through different features that serve as indexes.
Step 2: Case retrieval in the database. For a new problem, enter the index values for its features and compare cases to look for the one that has the highest similarity.
Step 3: Old case adaptation. Adjust the retrieved cases to fit the solution to the current status.
Step 4: New case evaluation. Evaluate the adjusted case to ensure its feasibility.
Step 5: Case storage. Store the newly adapted case in the database to achieve the self-learning function.

Generally, CBR deals with the experience previously set and turns it into a dependent one in the database for further retrieval. For a related case, the user only needs to key in the known indexes and CBR will look for a case that has the highest similarity in the database to serve for problem solution. Then, through partly adapting of the content of the retrieved case, it is possible to solve the new problem from the old experience. At last, saving the case of new problem solution in the database will reach the purpose of knowledge regeneration for future reuse.

In the past, CBR had been successfully applied to the solution to many problems. For example, on-line services to help desk application (Göker & Roth-Berghofer, 1999), scheduling and process planning (Chang, Dong, Liu, & Lu, 2000; Schmidt, 1998), hydraulic machine design (Vong, Leung, & Wang, 2002), architecture design (Heylighen & Neuckermans, 2001), customer relationship management (Choy, Lee, & Lo, 2003), fault diagnosis (Liao, Zang, & Mount, 2000; Yang, Han, & Kim, 2004), design and implementation of knowledge management (Lau, Wong, Hui, & Pun, 2003; Wang & Hsu, 2004), prediction of information systems outsourcing success (Hsh, Chiu, & Hsh, 2004), customer and market plan (Changchien & Lin, 2005; Chiu, 2002). In summary of the review, the evolution of CBR methods depends on the integration of domain problem and application specific.

Traditionally, BOM deals with a database through tables (Cunningham, Higgins, & Browne, 1996; Olsen & Saetre, 1997; Vang & Wortmann, 1992). Such a structure cannot handle the product configuration in a customization environment. In a BOM hierarchical structure, situations vary. Sometimes, only node values will be changed while sometimes a part of or even the whole structure will be changed. To solve this kind of problem, a tree hierarchy method and CBR technique are incorporated for product configuration.

3. Proposed CBR algorithms for product configuration

3.1. Expression of product configuration

For different types of products, product configuration can be described according to their features, which can be form, function, appearance, quality feature, stock mode, and technology specification. Currently, there is no consistent way in describing features. Instead, features of products and companies should be taken into consideration.

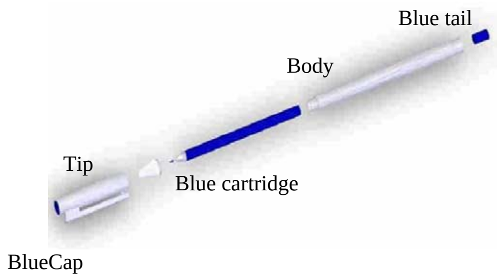
In this study, traditional BOM's tree structure and the expression of features are combined to describe characteristics of product configuration, which is defined as a product feature tree. In such a tree structure, the relationship of parent–child levels can represent the relationships between parts and subassembly. Take the PILOT pen (see Fig. 1(a)) for instance. Fig. 1(b) stands for its product feature tree, in which the contoured circles and gray circles symbolize the nodes and features, respectively. Here, the so-called features describe the characteristics of the parent nodes. For example, in Fig. 1(b), there are three features for the pen cap: the color is blue; the diameter of the ink cartridge is 0.4 mm; and the shape is circular. In the diagram, the straight line represents the line that connects nodes, indicating the hierarchical family relationship. The child points are dependent on the changing of parent nodes. To make sure that this property can be handled, the following rules should be obeyed:

Rule 1: The connected nodes should not form a loop.
Rule 2: Features should be used to describe nodes so as to guarantee that there is one and only one node in the parent level.
Rule 3: When nodes stand for parts, the nodes in the child level represent the subparts or features of the node.
Rule 4: When nodes stand for features, there will be no nodes in the child level; the features describe the characteristics of the node in the parent level.
Rule 5: The standard of setting up the features and nodes of a product configuration should be established by the personnel in charge of the planning of product configuration.

In terms of data storage, the menu was used in this study because of the following advantages:

1. *Visualized user-friendly interface:* Types of data storage are identical to those in the user interface. Therefore, no specific training is needed for database maintenance.

(a)



(b)

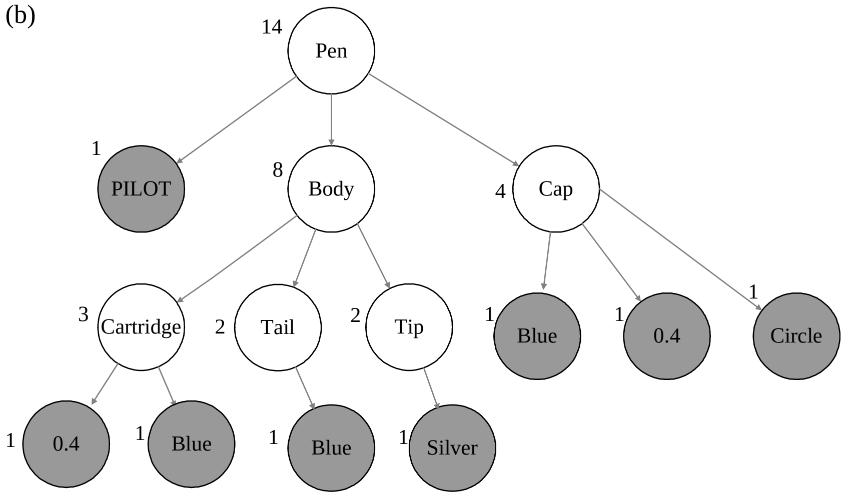


Fig. 1. Pen: (a) diagram of a blue pen (b) feature tree (Left number represents weight for each node). (For interpretation of the reference to colour in this legend, the reader is referred to the web version of this article.)

2. *Hierarchical tree representation:* It features the parent–child level way of representation, the concept of parts and subparts.

In the interface design, contoured circles and gray circles represent nodes and features, respectively. Black lines represent the ones that connect nodes. The diagram is depicts the coordination of nodes, where the root node serves as the initial point (0,0) and other nodes are deployed in an ascending order. If a parent node is set to be (X,Y), and its child node is (X_1,Y_1), then $X_1=X+1$; $Y_1=Y+(1+N)$; N is the number of total nodes and cannot appear twice. The data storage of Fig. 1 is shown in Fig. 2.

3.2. Retrieval and comparison of cases

For the retrieval and comparison of cases, we need to check first whether the cases are equal in terms of each attribute; f_j^I represents the value of the jth attribute of input Case (I), and f_{ij}^R is the value of the jth attribute of the ith case in the database (R). In this study, it is hypothesized that $S(f_j^I, f_{ij}^R) \in [0,1]$. When $f_j^I \equiv f_{ij}^R$, the output equals to 1 whereas when $f_j^I \neq f_{ij}^R$, the output equals 0. Finally, z equals

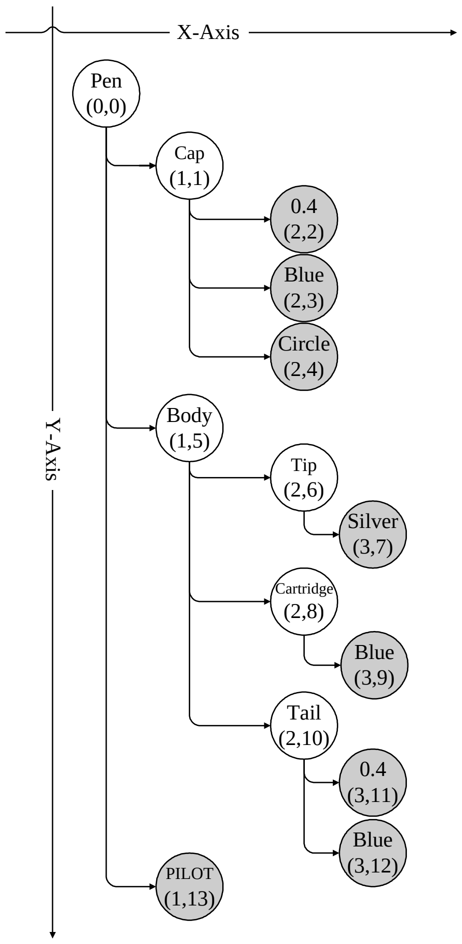


Fig. 2. Storage format of feature tree.

to the sum of the number of nodes in the child level node of the jth attribute and the node itself, namely, 1.

Second, we need to calculate the similarity of each node. $\mathrm{Sim}(f_j^I, f_{ij}^R)$ denotes the similarity of the jth attribute of input Case (I) and the jth attribute of the ith Case in the database (R). To begin with, we have to check whether the node is equal to its child level nodes and calculate the total number of nodes from $S(f_j^I, f_{ij}^R)$. The similarity of the node can be obtained from Formula (1):

$$\mathrm{Sim}(f_j^I, f_{ij}^R) = \frac{\sum_{k=j}^{l=j+z-1} S(f_k^I, f_{ik}^R)}{z} \tag{1}$$

Finally, we need to calculate the similarity between nodes. This can be done by dividing the summation of the product of the weight W_j of the jth attribute and the similarity output of Formula (1) by the sum of weights, $\sum_{j=1}^{n} W_j$, denoted as Formula (2):

$$\mathrm{Similary}(f^I, f_i^R) = \frac{\sum_{j=1}^{n} [W_j \mathrm{Sim}(f_j^I, f_{ij}^R)]}{\sum_{j=1}^{n} W_j} \tag{2}$$

In this study, it is hypothesized that nodes on the higher level are closer to the final product and that a node is composed of the nodes on the child level (parts or subassembly) and features. Therefore, when a node is different, at least one node on its child level will be changed accordingly. Therefore, the relative importance of the node on the parent level is bigger than that of the nodes on the child level. The summation of the number of nodes itself and that of the nodes below decides the weight of the node. This is shown in Fig. 1(b), in which the number by the node indicates its weight.

It can be seen from Fig. 1(b) that the nodes below the node Pen are, from the top to the bottom and from the left and the right, PILOT, body, cap, ink cartridge, tail, tip, blue, 0.4, rounded tip, 0.4, blue, blue, silver, 13 child nodes in total. With the node itself 1, added, the weight of the node equals 14. The node PILOT does not have any child node on the lower levels. So, it has a weight of 1, the node PILOT itself. The node Body has seven child nodes, Ink Cartridge, Tail, Tip, 0.4, Blue, Blue, and Silver. As a result, its weight is 8. The node Cap has three child nodes, Blue, 0.4, Rounded Tip, and the value of its weight is 4. In the same way, the weights of the nodes in the third level are 3 for node Ink Cartridge, 2 for node Tail, 2 to node Tip, 1 for node Blue, 1 for node ‘0.4’, and 1 for node Rounded Tip. Accordingly, the weights for the nodes on the fourth level will be 1 for node ‘0.4’, 1 for node Blue, 1 for node Blue, and 1 for node Silver.

The algorithms for case comparison can be listed as follows (see Fig. 3).

Step 1: Input nodes
Enter the node to be compared and the node in the database.

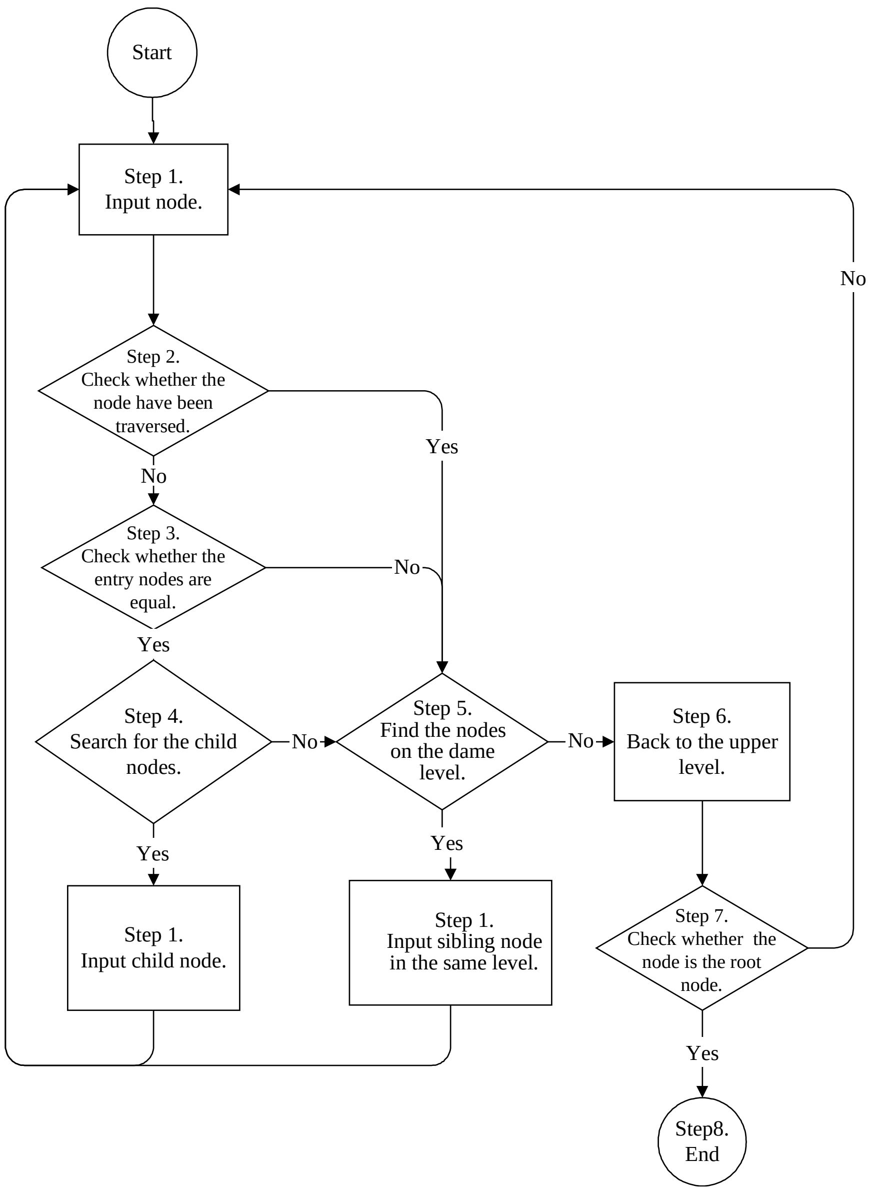

Start
Step 1.
Input node.
No
Step 2.
Check whether the node have been traversed.
Yes
No
Step 3.
Check whether the entry nodes are equal.
No
Yes
Step 4.
Search for the child nodes.
No
Step 5.
Find the nodes on the dame level.
No
Step 6.
Back to the upper level.
Yes
Yes
Step 1.
Input child node.
Step 1.
Input sibling node in the same level.
Step 7.
Check whether the node is the root node.
Yes
Step8.
End

Fig. 3. Algorithms for case comparison.

Step 2: Check whether the entry nodes have been traversed.
  Yes: Jump to Step 5.
  No: Process Step 3.

Step 3: Check whether the entry nodes are equal.
  Yes: $\mathrm{Sim}(f_j^I, f_{ij}^R) = 1$, process Step 4.
  No: $\mathrm{Sim}(f_j^I, f_{ij}^R) = 0$, jump to Step 5.

Step 4: Search for the child nodes.
  Check whether or not the nodes on the child level have not been traversed.
  Yes: Jump to Step 1.
  No: Process Step 5.

Step 5: Find the nodes on the same level.
  Check if there are nodes on the same level that have not been traversed.
  Yes: Jump to Step 1.
  No: Process Step 6.

Step 6: Back to the upper level.

Step 7: Check whether the node is a root node.
  Yes: Process Step 7.
  No: Jump to Step 1.

Step 8: End. The comparison of two hierarchical trees comes to the end.

Take the PILOT pens in Figs. 1(a) and 4 as an example for the comparison algorithms, the similarity and the traverse order. There are differences in their product configuration. As can be seen in Fig. 5(a) and (b), contoured circles and gray circles stand for nodes and features, respectively. Gray lines represent the ones that connect nodes. Black lines indicate the traverse direction and the numbers show the traverse order.

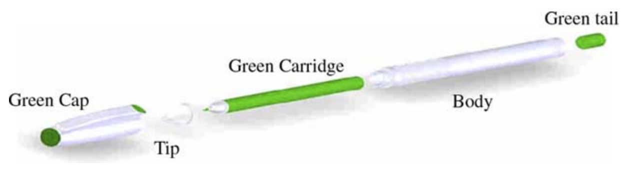


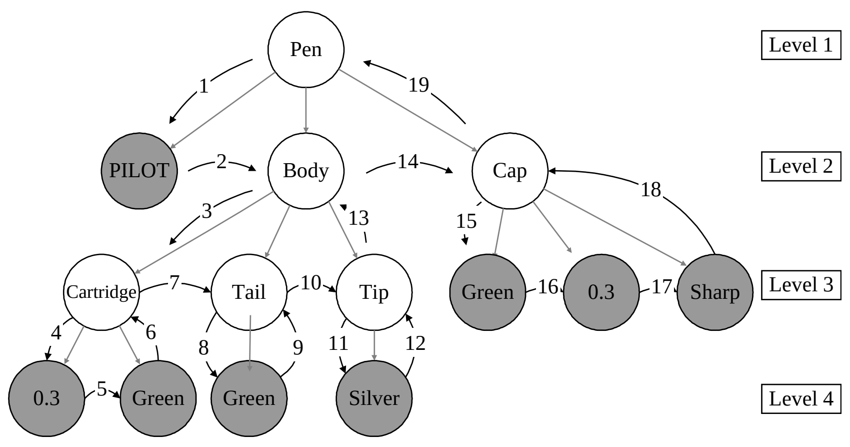
Fig. 4. Green 0.3: PILOT pen. (For interpretation of the reference to colour in this legend, the reader is referred to the web version of this article.)

Step 1: Input the node (pen) in Fig. 5(a) and the node (pen) in Fig. 5(b)
Step 2: Check whether the entry nodes have been traversed. Process Step 3.
Step 3: Check whether the entry nodes are equal. Enter Yes, Sim(f_j^I,f_{ij}^R) = 1. Process Step 4.
Step 4: Find nodes on the child level. The child nodes of the node (pen) have not been traversed. Therefore, jump to Step 1.

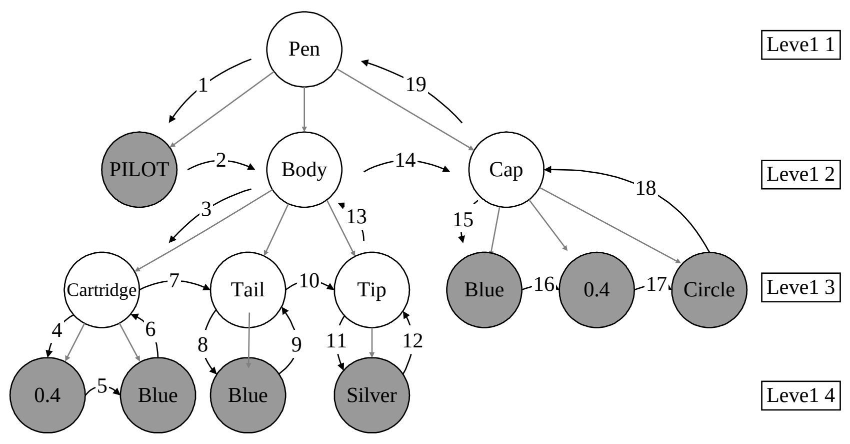
Step 1: Input nodes. Enter the node(PILOT) in Fig. 5(a) and the node(PILOT) in Fig. 5(b).
Step 2: Check whether these two nodes have been traversed. Process Step 3.
Step 3: Check whether these entry nodes are equal. Enter Yes, Sim(f_j^I,f_{ij}^R) = 1. Process Step 4.
Step 4: Look for the child nodes. There are no child levels for the node (PILOT) in Fig. 5(a) and (b). Process Step 5.
Step 5: Find the nodes on the same level. The node (body) has not been traversed. Therefore, jump to Step 1.

Step 1: Input nodes. Enter the node(body) in Fig. 5(a) and the node(body) in Fig. 5(b).
Step 2: Check whether the entry nodes have been traversed. Process Step 3.
Step 3: Check whether these nodes are equal. Enter Yes, Sim(f_j^I,f_{ij}^R) = 1. Process Step 4.
Step 4: Look for the child nodes. The child levels of the node(body) in Fig. 5(a) and (b) have not been traversed. Jump to Step 1.

Step 1: Input nodes. Enter node (ink cartridge) in Fig. 5(a) and node(ink cartridge) in Fig. 5(b).
Step 2: Check whether the entry nodes have traversed. Process Step 3.



(a) BOM diagram of PILOT pen (green).



(b) BOM diagram of PILOT pen (blue).

Fig. 5. BOM diagram (number on arc represents traverse order).

Step 3: Check whether the entry nodes are equal. Enter Yes, $\text{Sim}(f_j^I, f_{ij}^R) = 1$. Process Step 4.
Step 4: Look for the child nodes. The child levels of the node(ink cartridge) in Fig. 5(a) and (b) have not been traversed. Jump to Step 1.

Step 1: Input nodes. Enter node(0.3) in Fig. 5(a) and node(0.4) in Fig. 5(b).
Step 2: Check whether the entry nodes have been traversed. Process Step 3.
Step 3: Check whether the nodes are equal. Enter No, $\text{Sim}(f_j^I, f_{ij}^R) = 0$. Process Step 5.
Step 5: Find the nodes on the same level. The node(green) in Fig. 5(a) and the node(blue) in Fig. 5(b) have not been traversed. Jump to Step 1.

Repeat the same steps until the comparison of the two tree structures is done. From calculation, the similarity can be generated:

- n:$n = 14$
- W_j:$j = \{1,2,\ldots,14\}$ = {pen, PILOT, body, ink cartridge, 0.3, green, tail, green, tip, silver, cap, green, 0.3, sharp tip}

$$W_1 = 14, \quad W_2 = 1, \quad W_3 = 8, \quad W_4 = 3,$$
$$W_5 = 1, \quad W_6 = 1, \quad W_7 = 2, \quad W_8 = 1,$$
$$W_9 = 2, \quad W_{10} = 1, \quad W_{11} = 4, \quad W_{12} = 1,$$
$$W_{13} = 1, \quad W_{14} = 1$$

$$\sum_{j=1}^{n} W_j = 41$$

- f_j^I:$j = \{1,2,\ldots,14\}$ = {pen, PILOT, body, ink cartridge, 0. 3, green, tail, green, tip, silver, cap, green, 0.3, sharp tip}
- f_{ij}^R:$j = \{1,2,\ldots,14\}$ = {pen, PILOT, body, ink cartridge, 0. 4, blue, tail, blue, tip, silver, cap, blue, 0.4, rounded tip}
- $\text{Sim}(f_j^I, f_{ij}^R)$:$j = \{1,2,\ldots,14\} = \{1, 1, 1, 1/3, 0, 0, 1/2, 0, 1, 1, 1/4, 0, 0, 0\}$

$$\text{Similary}(f^I, f_i^R) = \frac{\sum_{j=1}^{n} W_j \times \text{Sim}(f_j^I, f_{ij}^R)}{\sum_{j=1}^{n} W_j}$$

$$= \frac{14\times1+1\times1+8\times1+3\times\frac{1}{3}+1\times0+1\times0+2\times\frac{1}{2}+1\times0+2\times1+1\times1++4\times\frac{1}{4}+1\times0+1\times0+1\times0}{41}$$

$$= 70.73\%$$

In the case of PILOT pen, there are 14 features, so $n = 14$. Since the weight of each feature equals the total numbers of the node itself and the nodes below, and we can get $W_j = 41$. f_j^I means the input features and f_{ij}^R denotes the ith item of information in the database. $\text{Sim}(f_j^I, f_{ij}^R)$ indicates the similarity weight of each feature between f_j^I and f_{ij}^R; 1 for equal and 0 for not equal. At last, through the formula of $\text{Similary}(f^I, f_i^R)$, we can get a similarity value of 70.73% in the PILOT pen case.

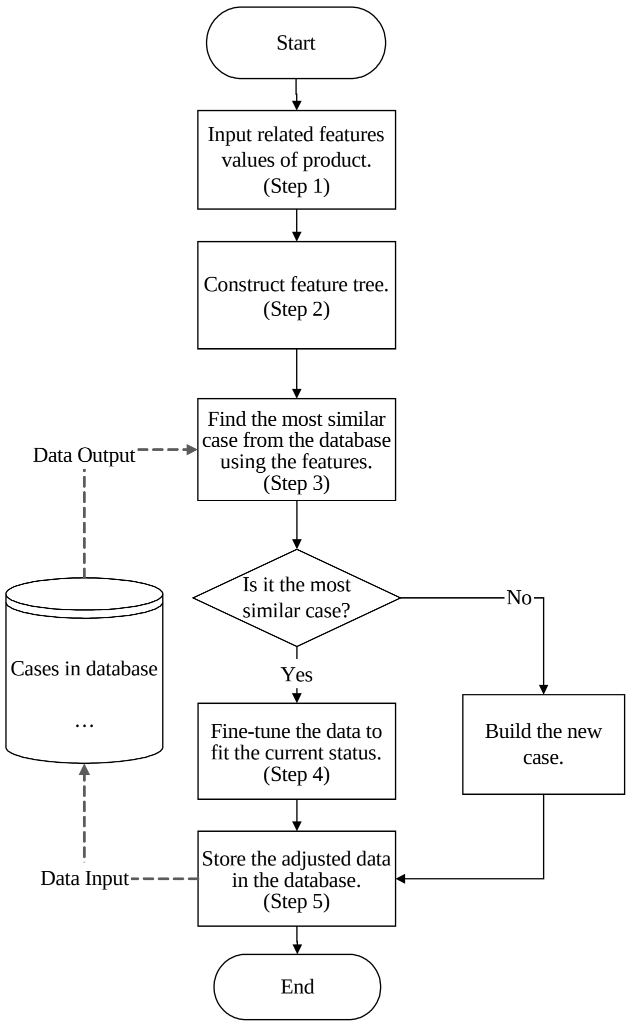

Start
Input related features values of product. (Step 1)
Construct feature tree. (Step 2)
Find the most similar case from the database using the features. (Step 3)
Data Output
Is it the most similar case?
No
Yes
Cases in database
...
Fine-tune the data to fit the current status. (Step 4)
Build the new case.
Data Input
Store the adjusted data in the database. (Step 5)
End

Fig. 6. CBR algorithms for product configuration.

3.3. Algorithms of case-based reasoning

Fig. 6 shows the algorithms of CBR for the comparison of product configuration. It is made up of the following steps:

Step 0: Start.
Step 1: Input related feature values according to the standardized format.
Step 2: Construct the product feature tree in accordance with the data types defined in Fig. 2.

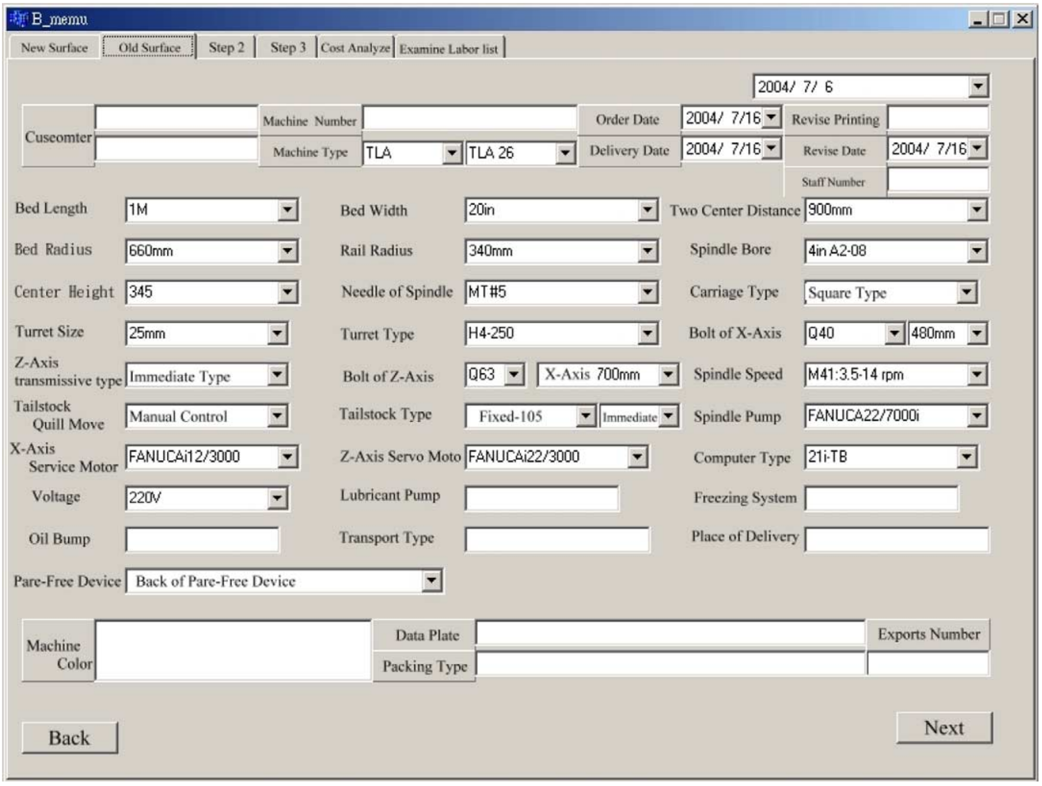
Table 1
Features of CNC machine at Taichung, Taiwan

Feature	Option	Feature	Option
Model one	3	Tailstock	
Model two	9	Tailstock quill diameter	1
Bed radius	6	Tailstock thimble	1
Rail radius	6	Tailstock body movement	1
Center height	6	Tailstock quill movement	2
Cross-slide length	2	Tailstock quill	1
Carriage width	2	Feeding	
Center distance	15	*X*-axis speedy feeding	2
CNC controller	4	*X*-axis feeding rate	1
Carriage width	3	*X*-axis	2
Rail type	1	*X*-axis servo motor	1
Bed length	15	*Z*-axis speedy feeding	6
Headstock		*Z*-axis feeding rate	2
Spindle bore	10	*Z*-axis transmission type	3
Main spindle tip	10	*Z*-axis servo motor	4
Main spindle rotational speed	10	Standard accessories	
Main spindle thimble	7	Lubrication pump	1
Main spindle horse-power	5	Cooling system	1
Turret		Oil bump	1
Turret type	9	Eliminates truncating device	3
Turret size	2	Enclosure guarding	2

Step 3: According to the product feature tree, search the most similar case from the database and check whether it is the case the user needs following Formulae (1) and (2).
  If a previous case has been found, process Step 4.
  If no previous cases have been found, check whether it is necessary to adjust the product feature tree.
    If the product feature tree needs to be adjusted, jump to Step 3.
    If the current case is set to be a new case, jump to Step 5.
Step 4: Fine-tune the data of the previous case to fit the current status.
Step 5: Store the adjusted data in the database.
Step 6: End.

In addition to the CBR algorithms for product configuration knowledge database, the following managerial steps are essential to building the type of product configuration, and to verify the retrieval and comparison model.

(1) Identify the system goal: Meetings can be held to set up customization as the system goal and application direction.



2004/ 7/ 6
Cuseomter:
Machine Number:
Machine Type: TLA / TLA 26
Order Date: 2004/ 7/16
Delivery Date: 2004/ 7/16
Revise Printing:
Revise Date: 2004/ 7/16
Staff Number:
Bed Length: 1M
Bed Width: 20in
Two Center Distance: 900mm
Bed Radius: 660mm
Rail Radius: 340mm
Spindle Bore: 4in A2-08
Center Height: 345
Needle of Spindle: MT#5
Carriage Type: Square Type
Turret Size: 25mm
Turret Type: H4-250
Bolt of X-Axis: Q40 480mm
Z-Axis transmissive type: Immediate Type
Bolt of Z-Axis: Q63 / X-Axis 700mm
Spindle Speed: M41:3.5-14 rpm
Tailstock Quill Move: Manual Control
Tailstock Type: Fixed-105 / Immediate
Spindle Pump: FANUCA22/7000i
X-Axis Service Motor: FANUCAi12/3000
Z-Axis Servo Moto: FANUCAi22/3000
Computer Type: 21i-TB
Voltage: 220V
Lubricant Pump:
Freezing System:
Oil Bump:
Transport Type:
Place of Delivery:
Pare-Free Device: Back of Pare-Free Device
Machine Color:
Data Plate:
Packing Type:
Exports Number:

Fig. 7. Input interface of features.

(2) The expression and storage of product configuration: This should be designed by domain experts and built in a modular way, which will enhance the flexibility and reliability of retrieval and comparison processes. Furthermore, adding the knowledge to the database will improve the experience of the system.
(3) The retrieval and comparison of cases: Applying CBR algorithms to the case of retrieval and comparison will provide suggestions for decision references.
(4) The structure and flow chart of the system: With the construction of the database and the design of similarity algorithms, CBR inference mechanism can serve as the kernel for the system.
(5) The construction and verification of the system: The C++ programming language will be used to design the system and applied in practical cases.

4. Practical example

A local CNC machine manufacturer (http://www.llcnclathe.com) at Taichung, Taiwan was used as a case study. The company intended to build up their product configuration system for the purpose of product specification management. The sectors involved in the project are the business division responsible for order processing and the R&D division for design projects. It also offered them a good opportunity to reevaluate the company's operation system. The computer system was built on the basis of the algorithms shown in Fig. 6 (see Section 3).

Through many meetings, 37 features of a CNC lather were decided. As shown in Table 1, there are three features corresponding to MODEL_ONE in that there are three different manufacturing procedures for MODEL_ONE.

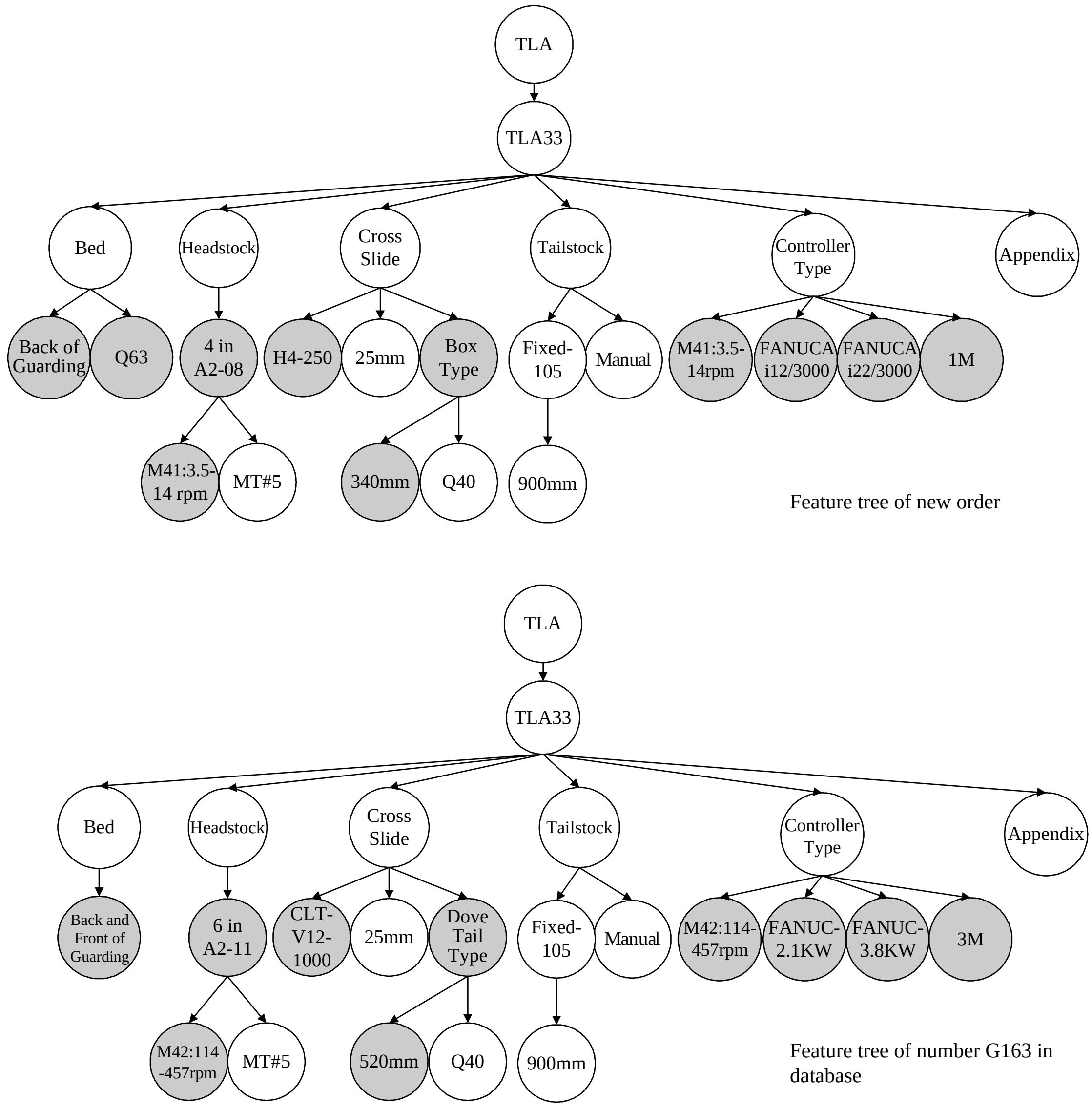


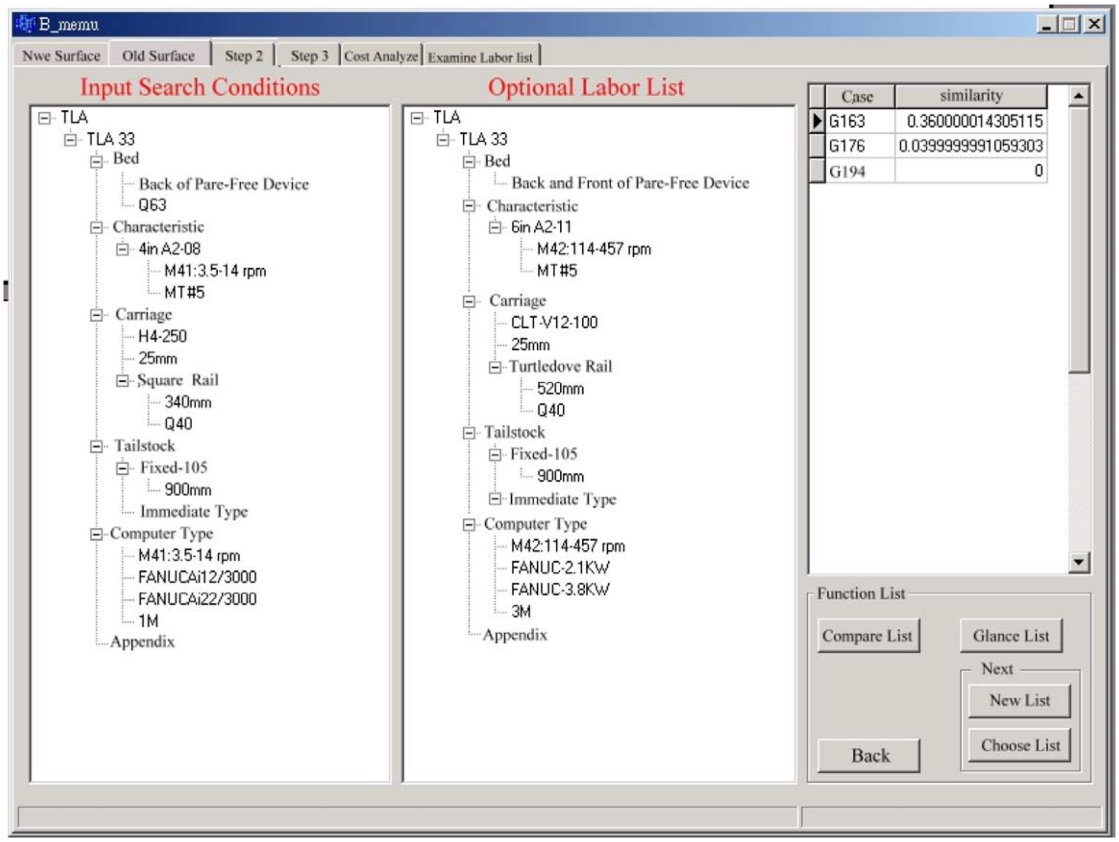
Fig. 8. Product feature trees of the new order and old order G163.

For these 37 features, there will be 2.74×10^{17} types of product configuration, which is too huge for engineers to handle. This is one of the major reasons why the variation in some features would cause trouble to the management sector. Fig. 7 shows the input interface the authors built for the business sector. The business personnel turned the customer's needs into new job orders. With the algorithms mentioned in Section 3, a newly confirmed order would be changed into a product feature tree as seen in Fig. 8(a), from which the R&D personnel would get a tree structure as the details shown on the left side of Fig. 9.

The right side of Fig. 9 shows the cases in the database and the similarity data from the comparison procedure. In the center of Fig. 9, the data of a previous case with the highest similarity are prompted. Comparison shows that the similarities between two job orders and the new one are 0.36 and 0.039 for job order G163 and job order G176, respectively. Furthermore, using the algorithms in Fig. 6, we can obtain that G163 is the most similar previous case. The fact that job order G163's product feature tree (Fig. 8(b)) is pretty close to that of the new order (Fig. 8(a)) demonstrated that G163 can be used as reference for further design and development. The management sector can also use G163 for further cost estimation.

In accordance with the algorithms in Fig. 6, previous case G163 was adjusted to suit the present condition. In Fig. 9, the tree structure of job order G163 is shown on the left side while the right side describes the features of the nodes in the tree structure. After adjustment of the features, the data are updated and stored as a new case in the database for later retrieval. The output demonstrated that when the BOM tree structure is properly designed, engineers might find that seldom do some parts or subassembly need to be changed right in the early design stage. This will reduce the probability of design adaptation. Furthermore, from previous experience of design, such as engineering or assembly drawings, some design might not be necessary and could be deleted. As a result, when the similarity between the new problem and the previous case is higher, the degree of differentiation between these two cases is lower, and vice versa. Fig. 10 shows the comparison procedure.

According to the requirement of the company, the cost of the CNC lather is divided into three parts: (1) material cost, (2) machining expenses, and (3) deduction expenses as shown in Fig. 11. This will make it possible for engineers to conduct related cost analysis of the parts during design stages. For example, when it is necessary to change the tailstock and bed, the cost analysis of the system demonstrates that the cost for the bed variation is much bigger than that of the tailstock. Hence, the bed has a higher priority in design. If it happens that some design variation should be done, the flexible cost in this part can be reduced, thus reaching a substantial cost cut in a company.



Input Search Conditions

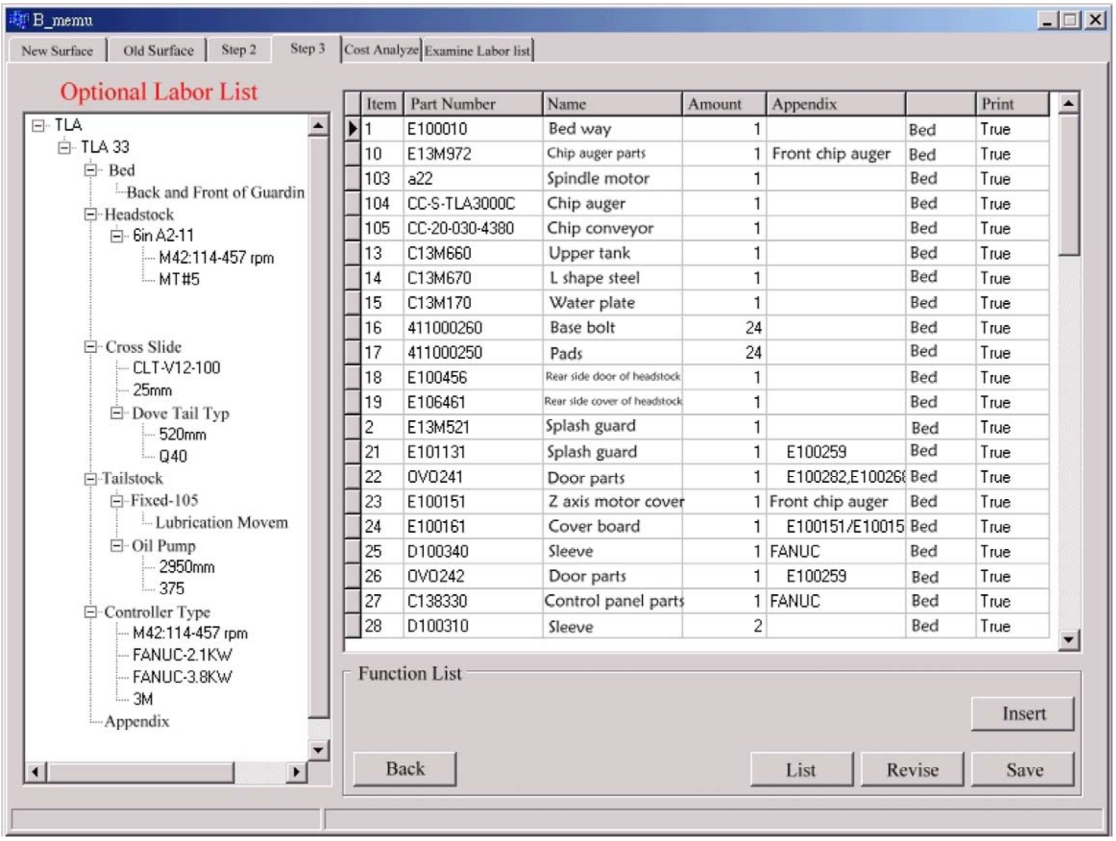
```
TLA
  TLA 33
    Bed
      Back of Pare-Free Device
      Q63
    Characteristic
      4in A2-08
        M41:3.5-14 rpm
        MT#5
    Carriage
      H4-250
      25mm
      Square Rail
        340mm
        Q40
    Tailstock
      Fixed-105
        900mm
      Immediate Type
    Computer Type
      M41:3.5-14 rpm
      FANUCAi12/3000
      FANUCAi22/3000
      1M
    Appendix
```

Optional Labor List

```
TLA
  TLA 33
    Bed
      Back and Front of Pare-Free Device
    Characteristic
      6in A2-11
        M42:114-457 rpm
        MT#5
    Carriage
      CLT-V12-100
      25mm
      Turtledove Rail
        520mm
        Q40
    Tailstock
      Fixed-105
        900mm
      Immediate Type
    Computer Type
      M42:114-457 rpm
      FANUC-2.1KW
      FANUC-3.8KW
      3M
    Appendix
```

Case	similarity
G163	0.360000014305115
G176	0.0399999991059303
G194	0

Fig. 9. Analysis and evaluation for product feature trees.

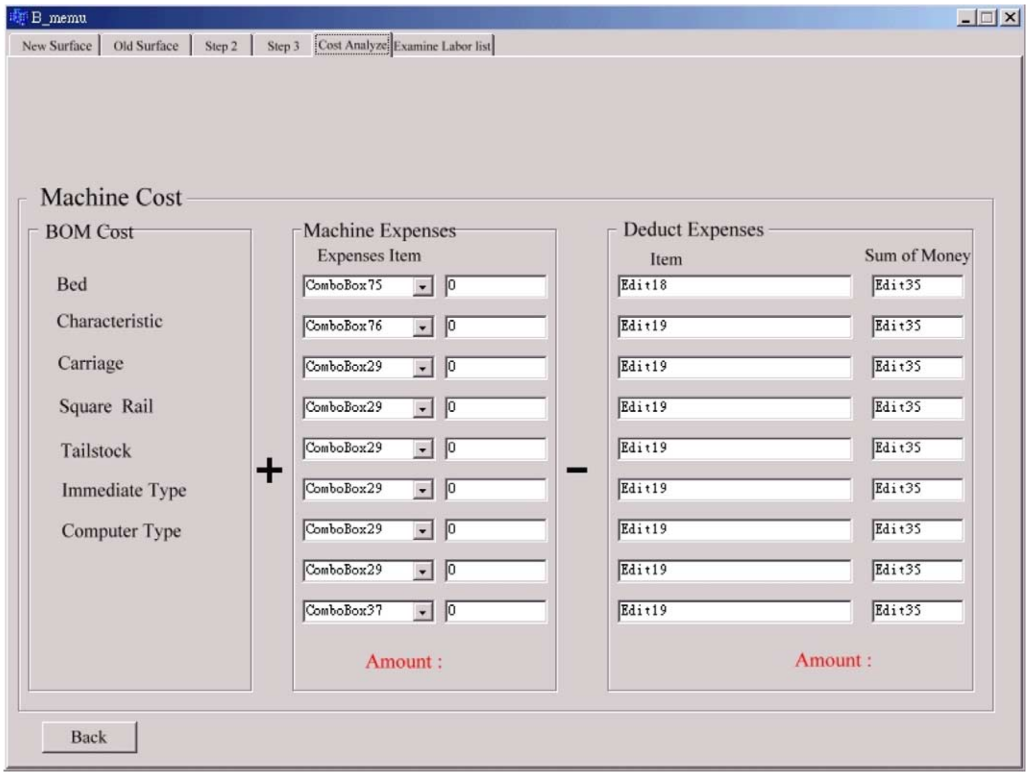


Optional Labor List

- TLA
  - TLA 33
    - Bed
      - Back and Front of Guardin
    - Headstock
      - 6in A2-11
        - M42:114-457 rpm
        - MT#5
    - Cross Slide
      - CLT-V12-100
      - 25mm
      - Dove Tail Typ
        - 520mm
        - Q40
    - Tailstock
      - Fixed-105
        - Lubrication Movem
      - Oil Pump
        - 2950mm
        - 375
    - Controller Type
      - M42:114-457 rpm
      - FANUC-2.1KW
      - FANUC-3.8KW
      - 3M
    - Appendix

Item	Part Number	Name	Amount	Appendix		Print
1	E100010	Bed way	1		Bed	True
10	E13M972	Chip auger parts	1	Front chip auger	Bed	True
103	a22	Spindle motor	1		Bed	True
104	CC-S-TLA3000C	Chip auger	1		Bed	True
105	CC-20-030-4380	Chip conveyor	1		Bed	True
13	C13M660	Upper tank	1		Bed	True
14	C13M670	L shape steel	1		Bed	True
15	C13M170	Water plate	1		Bed	True
16	411000260	Base bolt	24		Bed	True
17	411000250	Pads	24		Bed	True
18	E100456	Rear side door of headstock	1		Bed	True
19	E106461	Rear side cover of headstock	1		Bed	True
2	E13M521	Splash guard	1		Bed	True
21	E101131	Splash guard	1	E100259	Bed	True
22	OVO241	Door parts	1	E100282,E10026	Bed	True
23	E100151	Z axis motor cover	1	Front chip auger	Bed	True
24	E100161	Cover board	1	E100151/E10015	Bed	True
25	D100340	Sleeve	1	FANUC	Bed	True
26	OVO242	Door parts	1	E100259	Bed	True
27	C138330	Control panel parts	1	FANUC	Bed	True
28	D100310	Sleeve	2		Bed	True

Fig. 10. Comparison procedure of BOM tree.



Machine Cost

BOM Cost: Bed, Characteristic, Carriage, Square Rail, Tailstock, Immediate Type, Computer Type

Machine Expenses: Expenses Item		Deduct Expenses: Item	Sum of Money
ComboBox75	0	Edit18	Edit35
ComboBox76	0	Edit19	Edit35
ComboBox29	0	Edit19	Edit35
ComboBox29	0	Edit19	Edit35
ComboBox29	0	Edit19	Edit35
ComboBox29	0	Edit19	Edit35
ComboBox29	0	Edit19	Edit35
ComboBox29	0	Edit19	Edit35
ComboBox37	0	Edit19	Edit35
Amount :		Amount :	

Fig. 11. Cost analysis of product case.

Insofar as the practical case shows, the following notes merit greater:

1. The CBR algorithms can be applied under the condition that the product feature tree is built up according to a certain specification, from which an effective database is formed.
2. The previous experience can be sufficiently recorded and transmitted, which will dramatically reduce the error rate in dealing with the product specification.
3. The result of this study clarifies that there might exist some differences between the current job order and the most similar previous job order. Therefore, designers can focus on the differentiation parts of job order, hence saving time and energy. For example, the differences between the current job order and previous job order G163 lie in the tailstock and the headstock (Fig. 9). Therefore, designers need only to adjust the related parts.
4. Because the design variation is done from the most similar previous case, CBR system can effectively reduce the frequency of design adaptation. When it is necessary to perform some change in design, it can lower the cost for design differentiation. For instance, when there are two other parts to be changed, we can analyze their cost and choose to adjust only the one of lower cost.

5. Conclusions

In this study, CBR technique was applied to the planning of BOM. Previous cases that are similar to the current problem were retrieved and adjusted in accordance with the present constraints. The results demonstrated that such an approach could reduce the time needed to generate BOM and offer R&D staff members an accurate direction to follow. In this way, designers can share with each other their knowledge and experience. More importantly, it can avoid the loss of experience and the same mistakes in product design. In addition, because the newly generated BOM comes from the adjustment of a previous case, the situation in which parts are of raw universal usage will be prevented, thus solving the diversity problem in product configuration.

Some of the concrete contributions the authors make in this study are listed below.

1. Through the product configuration specified by CBR algorithms, the R&D knowledge and experience in a manufacturing company has been effectively taken down and systematically standardized.
2. In the traditional graph-based system, the rules derived from the constraint relationships of features will cause difficulty in the maintenance routine, especially, for complicated products. As far as this is concerned, CBR can effectively solve the problem.
3. The authors employed Borland C++ Builder 6.0 to write an program, which can simulate the learning ability of the R&D personnel. Furthermore, it can efficiently help generate an accurate BOM.

In the future, emphasis can be placed upon the application of cluster analysis in the database so as to enhance the retrieval speed and lower the burden of the database. In the comparison of cases, the viewpoint of Fuzzy comparison can be added to reinforce the inference algorithms.

Acknowledgements

The authors would like to thank the CNC manufacturing company in Taichung, Taiwan for their support in the test and evaluation of the program. This paper is part of a project for the industry and personnel promotion program (NSC 91-2622-E-212-018-CC3). The financial assistance from National Science Council, Taiwan, ROC is also gratefully acknowledged.

References

Chang, H. C., Dong, L., Liu, X. F., & Lu, W. F. (2000). Indexing and retrieval in machining process planning using case-based reasoning. *Artificial Intelligent in Engineering*, *14*(1), 1–13.

Changchien, S. W., & Lin, M. C. (2005). Design and implementation of a case-based reasoning system for market plans. *Expert Systems with Applications*, *28*, 43–53.

Chiu, C. (2002). A case-based customer classification approach for direct marketing. *Expert Systems with Applications*, *22*, 163–168.

Choy, K. L., Lee, W. B., & Lo, V. (2003). Design of an intelligent supplier relationship management system: A hybrid based neural network approach. *Expert Systems with Applications*, *24*, 225–237.

Cunningham, M., Higgins, P., & Browne, J. (1996). A decision support tool for planning bills-of-material. *Production Planning and Control*, *7*(3), 312–328.

Du, X., Jiao, J., & Tseng, M. M. (2002). Product family modeling and design support: An approach based on graph rewriting system. *Artificial Intelligent for Engineering Design, Analysis and Manufacturing*, *16*(2), 103–120.

Fohn, S. M., Liau, J. S., Greef, A. R., Young, R. E., & O'Grady, P. J. (1995). Configuring computer systems through constraint-based modeling and interactive constraint satisfaction. *Computer in Industry*, *27*, 3–21.

Göker, M. H., & Roth-Berghofer, T. (1999). The development utilization of the case-based help-desk support system HOMER. *Engineering Applications of Artificial Intelligence*, *12*, 665–680.

Hegge, H. M. H., & Wortmann, J. C. (1991). Generic bill-of-material: A new product model. *International Journal of Production Economics*, *23*(1–3), 117–128.

Heylighen, A., & Neuckermans, H. (2001). A case base of case-based design tools for architecture. *Computer-Aided Design*, *33*, 1111–1122.

Hsh, C. I., Chiu, C., & Hsh, P. L. (2004). Prediction information systems outsourcing success using a hierarchical design of case-based reasoning. *Expert Systems with Applications*, *26*, 435–441.

Jiao, J., Ma, Q., & Tseng, M. M. (2003). Towards high value-added products and services: Mass customization and beyond. *Technovation* , 809–821.

Jiao, J., Tseng, M. M., Ma, Q., & Zou, Y. (2000). Generic bill-of-materials-and-operations for high-variety product management. *Concurrent Engineering: Research and Applications*, *8*(4), 297–319.

Kim, K. S., & Han, I. (2001). The cluster-indexing method for case-based reasoning using self-organizing maps and learning vector quantization for bond rating cases. *Expert Systems with Applications*, *21*, 147–156.

Kobler, A., & Norrie, M. C. (1997). A Product information system based on an object-oriented internet database System. *Proceedings of the Sixth IEEE Workshops on Enabling Technologies Infrastructure for Collaborative Enterprises (WET-ICE'97), June*, *97*, 43–48.

Kolodner, J. L. (1993). *Case-based reasoning*. Pali Alto: Morgan Kaufman.

Lau, H. C. W., Wong, C. W. Y., Hui, I. K., & Pun, K. F. (2003). Design and implementation of an integrated knowledge system. *Knowledge-Based Systems*, *16*, 69–76.

Liao, T. W., Zang, Z. M., & Mount, C. R. (2000). A case-based reasoning system for identifying failure mechanism. *Engineering Applications of Artificial Intelligence*, *13*(2), 199–213.

Olsen, K. A., & Saetre, A. P. (1997). Managing product variability by virtual products. *International Journal of Production Research*, *35*(8), 2093–2107.

Ryu, Y. U. (1999). A hierarchical constraint satisfaction approach to product selection for electronic shopping support. *IEEE Transactions on Systems, Man and Cybernetics-Part A: Systems and Humans*, *29*(6), 525–532.

Salvador, F., & Forza, C. (2004). Configuring products to address the customization-reponsiveness squeeze: A survey of managing issues and opportunities. *International Journal of Production Economics*, *91*, 273–291.

Schmidt, G. (1998). Case-based reasoning for production scheduling. *International Journal of Production Economics*, *56–57*, 537–546.

Simpson, T. W., Seepersad, C. C., & Mistree, F. (2001). Balancing commonality and performance within the concurrent design of multiple products in a product family. *Concurrent Engineering: Research and Applications (CERA)*, *9*(3), 177–190.

Vang, E. A. V., & Wortmann, J. C. (1992). New developments in generative BOM processing systems. *Production Planning and Control*, *3*(3), 327–335.

Vong, C. M., Leung, T. P., & Wang, P. K. (2002). Case-based reasoning and adaptation in hydraulic production machine design. *Engineering Applications of Artificial Intelligence*, *12*, 567–585.

Wang, S. L., & Hsu, S. H. (2004). A web-based CBR knowledge management system for PC troubleshooting. *International Journal of Advanced Manufacturing Technology*, *23*, 532–540.

Yang, B. S., Han, T. H., & Kim, Y. S. (2004). Integrated of art-Kohonen neural network and case-based reasoning for intelligent fault diagnosis. *Expert Systems with Applications*, *26*(3), 387–395.