6

# Modeling of Product Life-cycle Knowledge and Data for an Intelligent Concurrent Design System

Deyi Xue
*Department of Mechanical and Manufacturing Engineering, The University of Calgary, Calgary, Alberta, Canada T2N 1N4*

**Key words**: Product life-cycle, Concurrent design, Artificial intelligence, Optimization, Object-oriented modeling

**Abstract**: This paper discusses the issues in modeling product life-cycle knowledge and data for an intelligent concurrent design system. Requirements for intelligent concurrent design are first outlined. Mathematical formulation for intelligent concurrent design is then introduced. Development and implementation of an intelligent concurrent design system and its applications are subsequently presented based upon the theoretical research results. A case study example is given at last to demonstrate the effectiveness of the introduced methods. This research focuses on the following four issues: (1) representation of various product life-cycle aspects, (2) automated generation of product life-cycle aspects, (3) maintenance of relations among product life-cycle aspects, and (4) identification of the optimal design considering relevant product life-cycle aspects.

## 1. INTRODUCTION

### 1.1 Product life-cycle and concurrent design

A product changes from its birth to its death through a sequence of life-cycle phases, including design, production process planning, manufacturing, inspection, distribution, operation/maintenance, and disposal/recycle. With the advances in computer technologies, many of the product development activities have been automated by introducing computer-based systems, such as Computer-Aided Design (CAD), Computer-Aided Process Planning

S. Finger et al. (eds.), *Knowledge Intensive Computer Aided Design*
© Springer Science+Business Media New York 2000

(CAPP), Computer-Aided Manufacturing (CAM), and so on (Chang, Wysk, and Wang, 1991; Singh, 1996). Many intelligent systems have also been developed for further improving the computer-based product development systems (Kusiak, 1992; Dong, 1994). Despite of the progress, information in most of these automated and intelligent systems still flows only in one direction – from design to other downstream product development phases. Since many re-designs have to be conducted to achieve good evaluation in these downstream product life-cycle aspects, the sequential product development process requires long product development lead time.

Concurrent design is an approach to incorporate considerations in different downstream product development life-cycle aspects, including manufacturing, assembly, maintenance, disposal/recycle, etc., into design phase, thus generating the design with the best overall life-cycle performance (Kusiak, 1993; Prasad, 1996). Because evaluations to these downstream product development aspects are carried out simultaneously during the design phase, concurrent design approach can reduce the number of costly re-designs and shorten product development lead time. Many industrial companies have employed this approach for improving the competitiveness of their products (Kusiak, 1993; Prasad, 1996).

The research on computer-aided concurrent design was initiated by the early work on *feature recognition* (Choi, Barash, and Anderson, 1984; Henderson, 1984). Feature recognition is an approach to automatically extract the geometric features, such as holes and slots, to be produced by certain manufacturing operations from the CAD database for planning production process and evaluating the design from manufacturing point of view. Due to the difficulty in feature extraction, another approach, called *feature-based design* that defines features as a library and uses the features to model a design, has been employed by many researchers (Pratt, 1984; Luby, Dixon, and Simmons, 1986; Shah and Rogers, 1988). Detailed review on feature recognition and feature-based design can be found in Reference (Shah and Mantyla, 1995).

To improve design with good evaluation in downstream product development life-cycle aspects, research on *design-for-X* was started by incorporating considerations in these life-cycle aspects into the early design phase. Typical downstream life-cycle aspects considered in design-for-X methods include manufacturing (Bralla, 1986; Dong, 1993), assembly (Boothroyd and Dewhurst, 1983), maintenance (Makino, Barkan, and Pfaff, 1989; Gershenson and Ishii, 1993), disposal/recycle (Zhang *et al.,* 1997), etc. Many advanced computational techniques, including constraint-network (Young, Greef, and O'Grady, 1992), optimization (Dowlatshahi, 1992), etc., have been employed for automating concurrent design processes.

## 1.2 Knowledge intensive engineering

As the result of market competition and technology advances, more products are manufactured nowadays to satisfy the increasing demand from customers. The current mass production paradigm brings tremendous advantages to us such as to improve the quality of living. Despite of these advantages, problems arose due to the mass production. These problems, called modern evils by Yoshikawa (Yoshikawa, 1993), are caused by natural constraints (e.g., materials, energy, and environmental capacity), social constraints (e.g., market and transportation), and human constraints (e.g., acceptance of high technology) (Tomiyama, 1997).

To solve these problems in the current mass production paradigm, a new manufacturing paradigm, namely Post Mass Production Paradigm (PMPP), was proposed (Tomiyama, 1997). PMPP aims at reducing the volume of production and consumption to an adequate and manageable size considering the limitations of natural, social, and human resources, while improving our living quality. In PMPP, the economical growth is decoupled from the resource/energy consumption and waste creation.

To reach the goal of PMPP, the past evaluation of living quality in terms of quantitative sufficiency must be replaced by the future evaluation in terms of qualitative satisfaction. The economic development therefore relies on the creation of high value products using intellectual resources rather than natural resources. Knowledge plays a crucial role in developing the value-added products.

Since the development of a product involves a sequence of life-cycle phases, the concept of *knowledge-intensive engineering*, which aims at organizing the knowledge at different life-cycle stages in a flexible manner to generate more added-value to products, was proposed (Tomiyama, 1994).

This research focuses on improving the current data-centered CAD systems into the next generation CAD systems – the knowledge intensive CAD systems. In knowledge intensive CAD systems, different product life-cycle knowledge is described in different modules. The relevant life-cycle knowledge is used when required to evaluate the design from a certain product life-cycle aspect. Since both the knowledge and data play important roles in the knowledge intensive CAD systems, modeling of product life-cycle knowledge and data is addressed in this paper.

## 1.3 Previous research on intelligent concurrent design

During the past years, the author has devoted the efforts on the development of an intelligent concurrent design system that supports the activities in all product development life-cycle aspects (Xue and Dong, 1993; Dong, Hu, and Xue, 1994; Xue and Dong, 1994; Xue, Rousseau, and Dong, 1996; Xue, 1997; Xue and Dong, 1997).

In this research, first modeling of product life-cycle aspects was studied (Xue and Dong, 1993). The three major life-cycle aspects, design, geometry, and manufacturing, are modeled by aspect primitives called features, including design features (mechanisms and components such as a gear and a shaft), geometry features (geometric primitives such as a box and a cylinder), and manufacturing features (geometric elements to be produced such as a hole and a slot). Representation of features follows the scheme of a product modeling language – Integrated Data Description Language (IDDL) (Tomiyama and ten Hagen, 1987; Xue *et al.,* 1992), which was originally developed at University of Tokyo. A system combining knowledge-based reasoning and optimization was developed for generating aspect models automatically and identifying the optimal design (Xue and Dong, 1994). Due to the large size of feature library, a design-function based design feature coding system and a manufacturing-function based manufacturing feature coding system were introduced for organizing feature library and for automatically generating design candidates and planning production process (Xue and Dong, 1997). Since production cost is a key measure for evaluating design from manufacturing point of view, the cost models considering different production processes and tolerance requirements were then developed (Dong, Hu, and Xue, 1994). An optimization model was developed for achieving the design with the best tradeoff between functional performance and production cost (Xue, Rousseau, and Dong, 1996). A number of global optimization models were also introduced to identify the optimal design (Xue, 1997).

The goal of the research presented in this paper is to further develop the intelligent concurrent design system and its applications with focus on modeling product life-cycle knowledge and data. The requirements for intelligent concurrent design are discussed first. The mathematical formulation for intelligent concurrent design is then introduced. The development and implementation of the intelligent concurrent design system and its applications are subsequently presented based upon the theoretical research results. Effectiveness of the introduced methods is illustrated using a case study example at the end of this paper.

## 2. REQUIREMENTS FOR INTELLIGENT CONCURRENT DESIGN

The requirements for intelligent concurrent design were achieved based upon an extensive study on the activities in concurrent design. These requirements are summarized as follows:

1. *An efficient method for modeling product life-cycle aspects should be introduced.*
   The different product life-cycle aspects, including design, manufacturing, assembly, and so on, should be described as aspect models. Each aspect model is composed of aspect descriptions for representing and evaluating the product from that life-cycle perspective. To improve the modeling efficiency, aspect models should be constructed using aspect building primitives. A building primitive is represented by a group of relevant descriptions for a particular purpose in the product development process. For instance, descriptions of a gear can be grouped as a building primitive for modeling a design candidate, and a hole can be described by a collection of geometric elements to be produced by a certain manufacturing operation. These building primitives also serve as the elements for evaluating the product aspect models. Descriptions in aspect models, including data and their relations, are classified into two categories: qualitative descriptions and quantitative descriptions. Different aspect models are associated by their relations.
2. *An automated product life-cycle aspect model generation mechanism should be achieved.*
   This mechanism aims at further improving the efficiency of constructing product life-cycle aspect models and their relations. Since the aspect models and their relations are built based upon relevant knowledge in product development process, a knowledge-based system is required to generate these aspect models and their relations automatically. In addition, because the development of a product undergoes a sequence of processes, including design candidate generation, design geometry modeling, manufacturing operation identification, and so on, this evolutionary nature of product realization process should also be represented.
3. *An integrated environment to maintain the relations among aspect models should be developed.*
   The aspect models are used for representing the same product from different life-cycle perspectives. Aspect models are associated by their relations. Because a concurrent design is carried out by considering these different life-cycle aspects simultaneously, a mechanism to maintain the relations among aspect models in an integrated environment is required.

The relations include qualitative relations and quantitative relations. Any change in one aspect model should be propagated to other aspect models automatically using these relations.

4. *An optimal design model considering all relevant product life-cycle aspects should be identified.*

   Because the same design requirement can be reached by alternative design candidates and different design parameter values, a mechanism to identify the optimal design alternative and its parameter values considering relevant life-cycle aspects should be developed. The product life-cycle aspect performance measures, including manufacturability, assemblability, serviceability, disposalability/recyclability, etc., should be employed for evaluating the product from the different product life-cycle aspects. Optimization approach can be used to achieve this objective.

The mathematical models and the intelligent concurrent design system were developed based upon the requirements summarized above. Details of these mathematical models and system implementation are presented in the following sections.

## 3. MATHEMATICAL FORMULATION FOR INTELLIGENT CONCURRENT DESIGN

A number of mathematical models have been developed for modeling intelligent concurrent design. The intelligent concurrent design system and its applications were implemented based on these mathematical models.

### 3.1 A feature-based product life-cycle aspect representation model

Models of different product life-cycle aspects are called aspect models, including design aspect model, $\boldsymbol{M}^{(D)}$, manufacturing aspect model, $\boldsymbol{M}^{(M)}$, and so on. These aspect models are built using aspect primitives, namely aspect features (Xue and Dong, 1993). An aspect feature, $F_i^{(P)}$ $(i=1,2,...,n_F^{(P)})$, is represented by a group of relevant descriptions in the aspect model for a particular product development purpose. The superscript $(P)$ of $F_i^{(P)}$ denotes the life-cycle aspect of the feature, such as design aspect, $(D)$, manufacturing aspect, $(M)$, and so on. For instance, a motor is a design feature for creating rotational motion, and a pocket is a manufacturing feature to be produced by a milling machining operation. A feature usually consists of quantitative descriptions called attributes. An attribute, $A_{ij}^{(P)}$, of the feature, $F_i^{(P)}$, is described by

$$A_{ij}^{(P)} = A_{ij}^{(P)}(F_i^{(P)}),\ \ i = 1,2,\cdots,n_F^{(P)}; j = 1,2,\cdots,n_{A_i}^{(P)} \tag{1}$$

For instance, rotational speed and output power are two attributes of a motor.

Features and attributes are associated by their qualitative relations, $R_{Fl}^{(P)}$, and quantitative relations, $R_{Am}^{(P)}$, respectively. These relations are defined as

$$R_{F_l}^{(P)} = R_{F_l}^{(P)}(F_1^{(P)}, F_2^{(P)}, \cdots, F_{n_F^{(P)}}^{(P)}),\ l = 1,2,\cdots,n_{R_F}^{(P)} \tag{2}$$

$$R_{A_m}^{(P)} = R_{A_m}^{(P)}(A_1^{(P)}, A_2^{(P)}, \cdots, A_{n_A^{(P)}}^{(P)}),\ m = 1,2,\cdots,n_{R_A}^{(P)} \tag{3}$$

For example, the connection relation between a gear and its shaft is a qualitative relation, while the relation between the speeds of the gear and the shaft is a quantitative one.

An aspect model, $\boldsymbol{M}^{(P)}$, consists of aspect features, $\boldsymbol{F}^{(P)}$, attributes of these features, $\boldsymbol{A}^{(P)}$, qualitative relations among features, $\boldsymbol{R}_F^{(P)}$, and quantitative relations among attributes, $\boldsymbol{R}_A^{(P)}$, as defined by

$$\boldsymbol{M}^{(P)} = \{\boldsymbol{F}^{(P)}, \boldsymbol{A}^{(P)}, \boldsymbol{R}_F^{(P)}, \boldsymbol{R}_A^{(P)}\} \tag{4}$$

where,

$$\boldsymbol{F}^{(P)} = \{F_1^{(P)}, F_2^{(P)}, \cdots, F_{n_F^{(P)}}^{(P)}\} \tag{5}$$

$$\boldsymbol{A}^{(P)} = \{A_1^{(P)}, A_2^{(P)}, \cdots, A_{n_A^{(P)}}^{(P)}\} \tag{6}$$

$$\boldsymbol{R}_F^{(P)} = \{R_{F_1}^{(P)}, R_{F_2}^{(P)}, \cdots, R_{F_{n_{R_F}^{(P)}}}^{(P)}\} \tag{7}$$

$$\boldsymbol{R}_A^{(P)} = \{R_{A_1}^{(P)}, R_{A_2}^{(P)}, \cdots, R_{A_{n_{R_A}^{(P)}}}^{(P)}\} \tag{8}$$

A product, $\boldsymbol{P}$, is defined by all its life-cycle aspect models, $\boldsymbol{M}$, and their relations, $\boldsymbol{R}_F$ and $\boldsymbol{R}_A$, using

$$\boldsymbol{P} = \{\boldsymbol{M}, \boldsymbol{R}_F, \boldsymbol{R}_A\} \tag{9}$$

where, $\boldsymbol{M}$ is a collection of aspect models described by

$$\boldsymbol{M} = \{\boldsymbol{M}^{(D)}, \boldsymbol{M}^{(M)}, \cdots\} \tag{10}$$

## 3.2 A product realization process model

The product aspect models are built gradually from design to other downstream life-cycle aspects. Usually design candidates are first created based upon design requirements. Product geometry is then achieved to model design details. Manufacturing descriptions are subsequently obtained from the product geometry. In this research, a product realization process model was introduced for representing this progressive nature of product development activities. This model is an extension of the General Design Theory (GDT), in which a design is considered as a process of mapping from function space to attribute space (Yoshikawa, 1981; Tomiyama and Yoshikawa, 1987).

In the product realization process model, new product descriptions, $M'$, at a certain product development stage are derived from the product descriptions, $M$, at an earlier product development stage using relevant knowledge, $K$, as described by

$$M \cap K \Rightarrow M' \tag{11}$$

In this equation, $\cap$ and $\Rightarrow$ are logical symbols representing AND relation and description derivation relation respectively. The relations among the derived data are of two types: AND relations and OR relations. For instance, the two machining operation descriptions derived by

$$\textit{internal thread} \cap K_1 \Rightarrow \textit{drilling} \cap \textit{threading} \tag{12}$$

have an AND relation, while the two design candidate descriptions generated by the following two equations

$$\textit{rotational motion} \cap K_2 \Rightarrow \textit{electrical motor} \tag{13}$$

$$\textit{rotation motion} \cap K_3 \Rightarrow \textit{gasoline engine} \tag{14}$$

have an OR relation.

## 3.3 A data relation maintenance model

Since different product life-cycle aspect models are used for representing different aspects of the same product, any change in one aspect model should be propagated to other aspect models to keep the consistency of the product database. In this research, the consistency of aspect models is maintained using the relations among these aspect models including qualitative relations among features and quantitative relations among attributes.

In the process of product development, since a piece of product description is usually derived from other descriptions using relevant knowledge, change of an earlier created description should have influence on the derived descriptions. For instance, the two machining operation descriptions in Eq. (12) are derived from the internal-thread manufacturing feature. If the internal-thread manufacturing feature is removed from the database, the two derived machining operation descriptions should also be deleted. This dependency relation is described by

$$d_j \leftarrow d_1, d_2, \cdots, d_n \tag{15}$$

where, $d_i$ could be a feature, an attribute, a qualitative relation among features, or a quantitative relation among attributes.

A quantitative relation among attributes can be further described by

$$A_j = A_j(A_1, A_2, \cdots, A_n) \tag{16}$$

where, $A_j$ is calculated using $A_1$, $A_2$, ..., $A_n$ as input attributes.

## 3.4 An optimal concurrent design model

Since design requirements can be satisfied by alternative design candidates, each of these candidates is further described by attributes with different values, in this research an optimal concurrent design model was introduced to identify the optimal design alternative and its attribute values considering relevant life-cycle aspects. The optimization is conducted at two different levels: alternative optimization level and attribute optimization level.

A feasible design alternative is modeled by a number of features and their attributes. Since the qualitative descriptions of features remain the same for a design alternative, a feasible alternative, $\boldsymbol{P}_i$, can therefore be described by a collection of attributes:

$$\boldsymbol{P}_i = \{A_{i1}, A_{i2}, \cdots, A_{in_i}\} \tag{17}$$

The optimal attribute values regarding one design alternative are obtained using constrained optimization approach:

$$\underset{w.r.t.\ A_{i1}, A_{i2}, \cdots A_{in_i}}{Min} \quad f_i(A_{i1}, A_{i2}, \cdots, A_{in_i})$$

*subject to:*

$$h_{ij}(A_{i1}, A_{i2}, \cdots, A_{in_i}) = 0, \ j = 1, 2, \cdots, k_i$$

$$g_{ij}(A_{i1}, A_{i2}, \cdots, A_{in_i}) \leq 0, \ j = k_i + 1, k_i + 2, \cdots, m_i \tag{18}$$

The objective function, $f_i(A_{i1}, A_{i2}, \ldots, A_{in_i})$, is an evaluation measure of the design from a certain product life-cycle perspective, such as manufacturing.

The optimal objective function evaluation measure is described as $f_i(\boldsymbol{P}_i^*)$. The optimal alternative is identified from all possible alternatives using

$$\underset{w.r.t.\ \boldsymbol{P}_i^*}{Min} f_i(\boldsymbol{P}_i^*) \tag{19}$$

where $\boldsymbol{P}_i^*$ is iterated among the feasible alternatives with the optimal attribute values.

## 4. DEVELOPMENT OF AN INTELLIGENT CONCURRENT DESIGN SYSTEM AND ITS APPLICATIONS

Based upon the mathematical formulation introduced in the previous section, an intelligent concurrent design system was developed for automating concurrent design activities. In this system, product life-cycle aspects are described by aspect models, as shown in Figure 1. The mathematical models introduced in the previous section were implemented as follows: (1) The product life-cycle aspects are modeled using a feature representation language. (2) The product realization process model is automated through knowledge-based inference. (3) The data relations among aspect models are maintained through data relation networks. (4) The optimal concurrent design is identified using a multi-level optimization approach. The system was implemented using Smalltalk, an object oriented programming language (Goldberg and Robson, 1983).

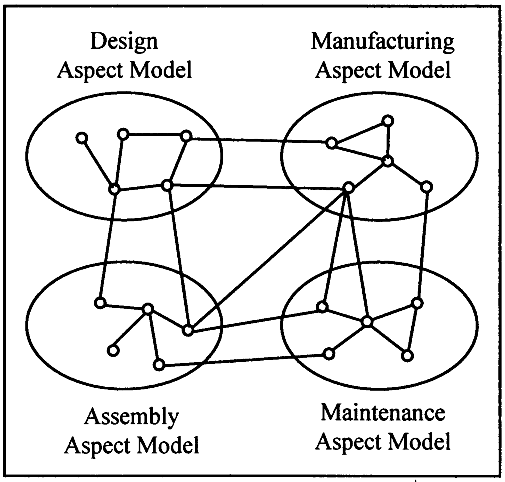


*Figure 1.* Aspect models of a product

Many industrial applications were also developed using the intelligent concurrent design system. One of these applications is used for designing building products for a local manufacturing company – Gienow Building Products Ltd. Many examples used in this paper were implemented for this application.

## 4.1 Modeling product life-cycle aspects using a feature representation language

In the feature-based product life-cycle aspect representation model introduced in Section 3.1, features are primitives for modeling product life-cycle aspects including design, manufacturing, and so on. In the intelligent concurrent design system, a feature representation language was introduced for modeling these aspect features.

In the feature representation language, features are described at two different levels, class level and instance level, corresponding to generic feature libraries and specific data for modeling particular products, respectively. Class features are used as templates for creating instance features. Object-oriented programming approach is employed for implementing class features and instance features.

Class features are used for modeling product libraries. All class features are organized in a hierarchical data structure. A new class feature is defined as a sub-class of an existing class feature. All the descriptions in a super-class feature are inherited by its sub-class features automatically. The top level class feature is a built-in class feature called Feature. Examples of class feature definitions used for the building product design application are shown in Figure 2. A class feature has 4 types of major components: (1) element-features, (2) attributes, (3) qualitative relations among features, and (4) quantitative relations among attributes.

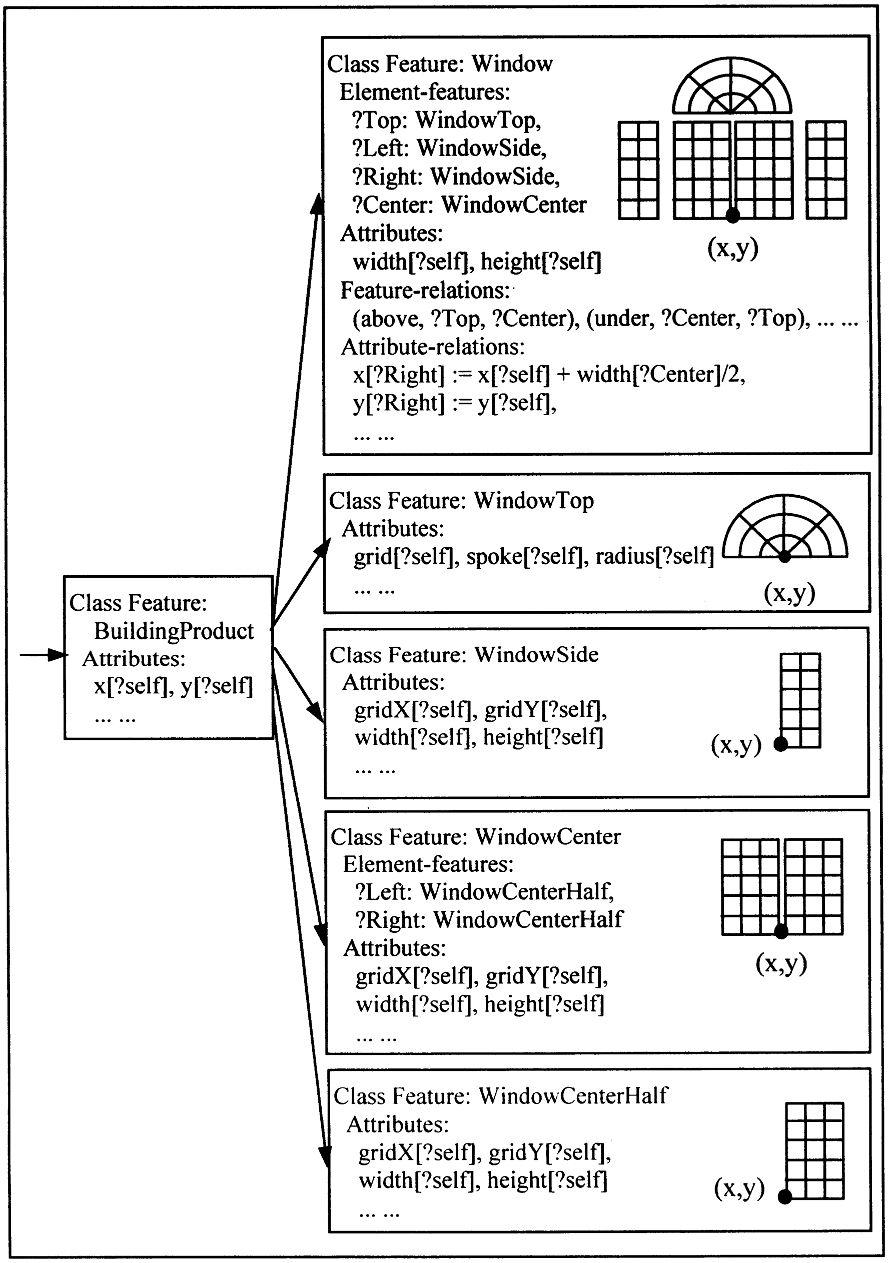

Class Feature: Window
Element-features:
?Top: WindowTop,
?Left: WindowSide,
?Right: WindowSide,
?Center: WindowCenter
Attributes:
width[?self], height[?self]
(x,y)
Feature-relations:
(above, ?Top, ?Center), (under, ?Center, ?Top), ... ...
Attribute-relations:
x[?Right] := x[?self] + width[?Center]/2,
y[?Right] := y[?self],
... ...
Class Feature: WindowTop
Attributes:
grid[?self], spoke[?self], radius[?self]
... ...
(x,y)
Class Feature: BuildingProduct
Attributes:
x[?self], y[?self]
... ...
Class Feature: WindowSide
Attributes:
gridX[?self], gridY[?self],
width[?self], height[?self]
... ...
(x,y)
Class Feature: WindowCenter
Element-features:
?Left: WindowCenterHalf,
?Right: WindowCenterHalf
Attributes:
gridX[?self], gridY[?self],
width[?self], height[?self]
... ...
(x,y)
Class Feature: WindowCenterHalf
Attributes:
gridX[?self], gridY[?self],
width[?self], height[?self]
... ...
(x,y)

*Figure 2.* Class feature definitions

An element feature is a component that composes the feature being defined. For instance, the Window class feature, shown in Figure 2, is composed of four element features. An element feature is defined by a variable and its class feature type. A variable is described by a string starting with "?". In the class feature Window, the four element features are associated with four variables: ?Top, ?Left, ?Right, and ?Center. When an instance feature is generated using this class feature as the template, the element features should also be created according to their class feature types defined in the class feature. The variables that are associated with element features can be used in other parts of the class feature definition. For instance, the four variables in class feature Window are used for defining feature relations and attribute relations of this class feature. The feature itself is associate with a built-in variable ?self. When a class feature is used for generating an instance feature, all the variables, representing element features, should be replaced by the actual element instance feature names in the created instance feature.

An attribute is a piece of quantitative description of a feature. An attribute is described by an attribute name and an attribute value. In a class feature definition, an attribute can be associated with a default value. Attributes are used in the form of attribute[feature] in other parts of the feature definition. For instance, width[?Center] in Figure 2 represents the attribute width of the feature ?Center.

A qualitative relation among features is described by a predicate, in the form of $(x_1, x_2, ..., x_n)$, where $x_1, x_2, ..., x_n$ are terms of this predicate represented by strings (e.g., above, under), integers (e.g., 25, −14), floats (e.g., 2.5, 1.2e−25), variables (e.g., ?Left, ?Center), and attributes (e.g., width[?self], height[?self]). A predicate without variable terms is called a fact.

A quantitative relation among attributes is defined by a function with a number of input attributes and one output attribute, as shown in Figure 2. Syntax of functions follows the syntax of Smalltalk. Element feature variables and attributes are also allowed in function definitions.

Instance features are generated from class features and used for modeling actual products. When a class feature is selected as the template for generating an instance feature, the element features defined in this class feature should also be created as instance features. All the descriptions defined in the class feature and its super-class features are inherited by the generated instance feature automatically. Figure 3 shows the instance features that are generated from the class features given in Figure 2 for representing a window product. In an instance feature, the variables in its class feature definition, representing element features, are replaced by the names of the actually created element instance features. Descriptions of instance features can be modified, added, and deleted.

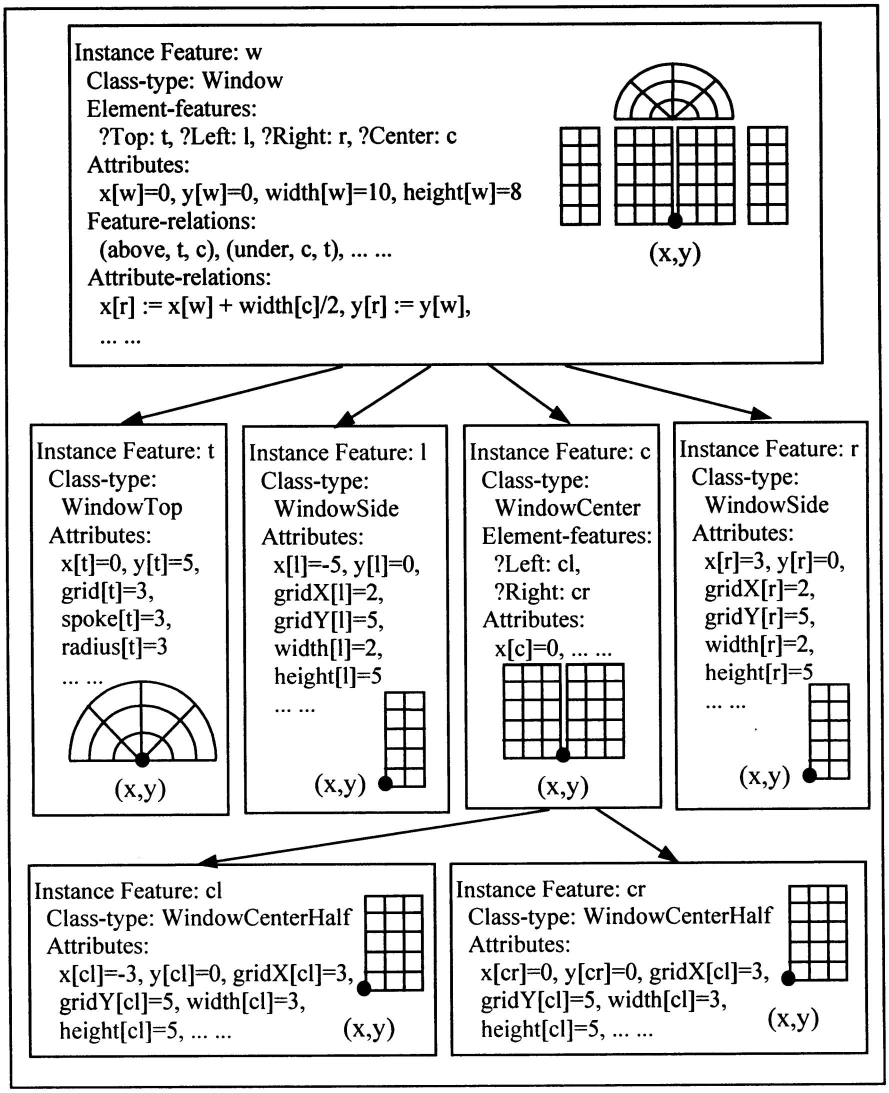

Instance Feature: w
Class-type: Window
Element-features:
?Top: t, ?Left: l, ?Right: r, ?Center: c
Attributes:
x[w]=0, y[w]=0, width[w]=10, height[w]=8
Feature-relations:
(above, t, c), (under, c, t), ... ...
Attribute-relations:
x[r] := x[w] + width[c]/2, y[r] := y[w],
... ...
(x,y)
Instance Feature: t
Class-type: WindowTop
Attributes:
x[t]=0, y[t]=5, grid[t]=3, spoke[t]=3, radius[t]=3
... ...
(x,y)
Instance Feature: l
Class-type: WindowSide
Attributes:
x[l]=-5, y[l]=0, gridX[l]=2, gridY[l]=5, width[l]=2, height[l]=5
... ...
(x,y)
Instance Feature: c
Class-type: WindowCenter
Element-features:
?Left: cl, ?Right: cr
Attributes:
x[c]=0, ... ...
(x,y)
Instance Feature: r
Class-type: WindowSide
Attributes:
x[r]=3, y[r]=0, gridX[r]=2, gridY[r]=5, width[r]=2, height[r]=5
... ...
(x,y)
Instance Feature: cl
Class-type: WindowCenterHalf
Attributes:
x[cl]=-3, y[cl]=0, gridX[cl]=3, gridY[cl]=5, width[cl]=3, height[cl]=5, ... ...
(x,y)
Instance Feature: cr
Class-type: WindowCenterHalf
Attributes:
x[cr]=0, y[cr]=0, gridX[cl]=3, gridY[cl]=5, width[cl]=3, height[cl]=5, ... ...
(x,y)

*Figure 3.* Instance feature definitions

## 4.2 Automating product realization process through knowledge-based inference

In Section 3.2, product realization process is modeled by mapping among different product development life-cycle aspects. In the intelligent concurrent design system, the mapping process is described by an AND/OR graph, as shown in Figure 4. In this graph, product descriptions, including features, attributes, qualitative relations among features, and quantitative relations among attributes, are described by nodes. The AND and OR relations among sub-nodes are achieved using the knowledge as described in Eqs. (12), (13) and (14).

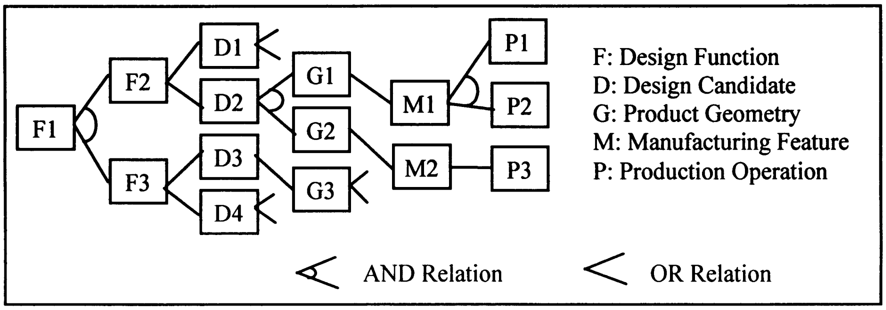


*Figure 4.* An AND/OR graph of product realization process

To automate the product realization process, a knowledge base system has been developed for generating the AND/OR graph through knowledge-based reasoning. In this system, product development knowledge is described by rules. Rules are grouped as rule-bases. The idea for organizing rules in groups follows the concept introduced for the Intelligent Integrated Interactive CAD (IIICAD) system (Tomiyama and ten Hagen, 1987; *Xue et al.*, 1992).

A rule-base is defined by a rule-base name and a collection of rules, as shown in Figure 5. Each rule description consists of a rule name and the rule itself in the form of IF-THEN data structure representing a piece of cause-result knowledge. Both the IF part and the THEN part of a rule are described by a number of patterns linked with logical-and (&). A pattern is described by a predicate in the form of $(x_1, x_2, ..., x_n)$, where $x_1, x_2, ..., x_n$ are terms described by strings, numbers, variables, and attributes. The condition part and result part of a rule are used for matching, creating, deleting, and modifying the data in product models, including features, attributes, qualitative relations among features, and quantitative relations among attributes, through knowledge-based inference.

To improve the efficiency of reasoning, only partial database and partial knowledge base are considered during product development process. Since an instance feature is composed of element features, attributes, qualitative relations among features, and quantitative relations among attributes, an instance feature is such a database unit selected for inference. The partial knowledge base used in inference is a collection of selected rule-bases.

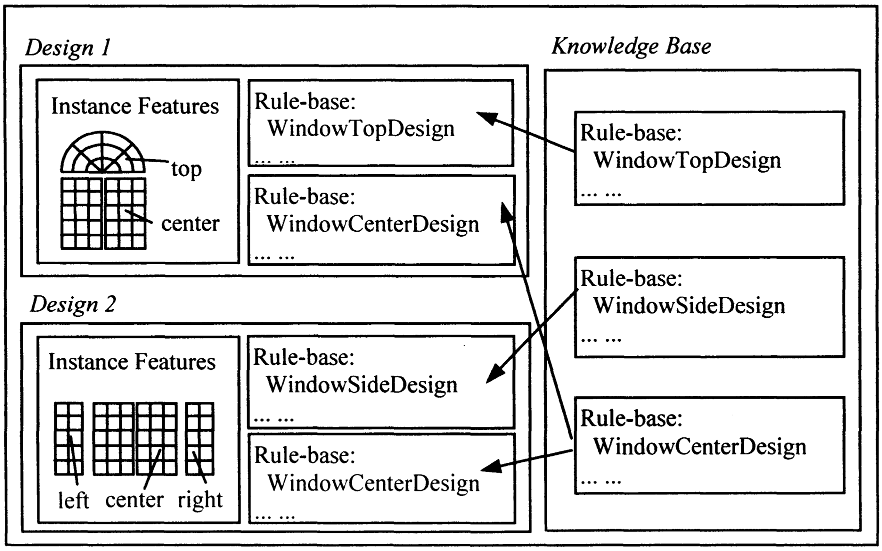
Therefore, each instance feature is associated with a number of rule-bases. This idea is illustrated in Figure 6.

```
Rule-base: WindowDesign

Rule: changeLeftGridY
IF (left, ?L) & (center, ?C) & (<>, gridY[?L], gridY[?C]) &
   (message, 'gridY values of left and center are not equal!') &
   (confirm, 'Do you want to change gridY of left?')
THEN (=, gridY[?L], gridY[?C])

Rule: changeCenterGridY
IF (left, ?L) & (center, ?C) & (<>, gridY[?L], gridY[?C]) &
   (message, 'gridY values of left and center are not equal!') &
   (confirm, 'Do you want to change gridY of center?')
THEN (=, gridY[?C], gridY[?L])
... ...
```

*Figure 5.* A rule-base definition

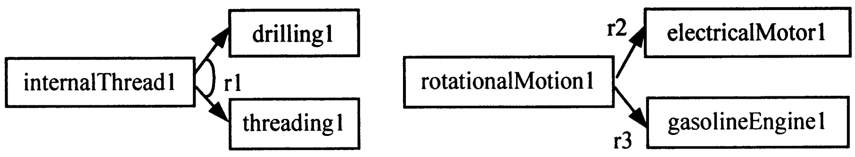


Design 1
Knowledge Base
Instance Features
top
center
Rule-base: WindowTopDesign ... ...
Rule-base: WindowCenterDesign ... ...
Design 2
Instance Features
left
center
right
Rule-base: WindowSideDesign ... ...
Rule-base: WindowCenterDesign ... ...
Rule-base: WindowTopDesign ... ...
Rule-base: WindowSideDesign ... ...
Rule-base: WindowCenterDesign ... ...

*Figure 6.* Selection of partial knowledge base and database in product design

During the process of knowledge-based product modeling, first an instance feature is selected as the partial database to be considered in inference. A number of rule-bases are then selected as the partial knowledge for this instance feature. All the rules in these rule-bases should be registered in this instance feature. The inference is conducted first by matching the condition parts of all the registered rules with the instance feature database. If multiple rules are matched, the best rule is then selected and the result part of this rule is executed. In this research, the first matched rule is considered as the best rule to be fired.

The AND/OR graph of product realization process is also generated by rule-based reasoning. When condition part of a rule matches with the database, the data created by executing the result part of this rule should have an AND relation. If several rules match with the same data, the data generated by executing these rules have an OR relation. Figure 7 shows the AND/OR graphs generated through rule-based reasoning.



(a) An AND relation (b) An OR relation

```
Rule: r1
   IF (featureType, ?X, InternalThread)
   THEN (assertFeature, ?Y1, Drilling) & (assertFeature, ?Y2, Threading).
Rule: r2
   IF (featureType, ?X, RotationalMotion)
   THEN (assertFeature, ?Y, ElectricalMotor).
Rule: r3
   IF (featureType, ?X, RotationalMotion)
   THEN (assertFeature, ?Y, GasolineEngine).
```

(c) Rules used for generating AND/OR relations

*Figure 7.* Generation of AND/OR graphs through rule-based reasoning

## 4.3 Maintaining data relations among aspect models through data relation networks

In Section 3.3, a data relation maintenance model was introduced for keeping the consistency of the product database using qualitative dependency relations among features and quantitative relations among attributes. In the intelligent concurrent design system, these two types of relations among product data are described by two associated networks, *feature relation network* and *attribute relation network*. The consistency of the product life-cycle aspect models is maintained using these two data relation networks. Any change in one life-cycle aspect is propagated automatically to other aspects through the data relation networks.

The feature relation network is described by an AND/OR tree of the product realization process. This network is composed of instance features and their dependency relations, which are generated either manually or through knowledge-based system. Other feature descriptions, including element features, attributes, qualitative relations among element features, and quantitative relations among attributes, are associated with these instance features. A feature relation network, representing two design alternatives shown in Figure 8 (a), is given in Figure 8 (b).

Product realization process alternatives, such as design alternatives or production process alternatives, can be achieved from the AND/OR tree of a feature relation network. An alternative is described by a collection of instance features. The process to identify product realization process alternatives is formulated in the following steps:

1. From the AND/OR tree, select the instance feature that needs alternative subsequent product realization processes.
2. If a selected instance feature has descendant instance features with an AND relation, all these descendant instance features should be selected.
3. If a selected instance feature has descendant instance features with an OR relation, only one of these descendant instance feature should be selected.

   Steps 2 and 3 are conducted continuously until no selection is required.

In the example shown in Figure 8, two alternatives, described by instance features of (1) doorWindow1, door1, window1, win2Frame1, dl1, dc1, dr1, s1, wl1, and wr1, and (2) doorWindow1, door1, window1, win3Frame1, dl1, dc1, dr1, s1, wl2, wc2, and wr2, are obtained from the AND/OR tree.

For each product realization process alternative, the quantitative relations among attributes then form another network, called attribute relation network. An attribute relation network is composed of two types of nodes: attribute nodes and function nodes. A function node is linked with a number of input attribute nodes and one output attribute node. The attribute relation network for the first design alternative is shown in Figure 8 (c).

## 4.4 Identifying the optimal concurrent design using a multi-level optimization approach

In Section 3.4, an optimal concurrent design model was introduced to identify the optimal design considering relevant product life-cycle aspects. In the intelligent concurrent design system, this optimal concurrent design model was implemented using a multi-level optimization approach, based upon the optimization algorithms introduced in (Xue, 1997).

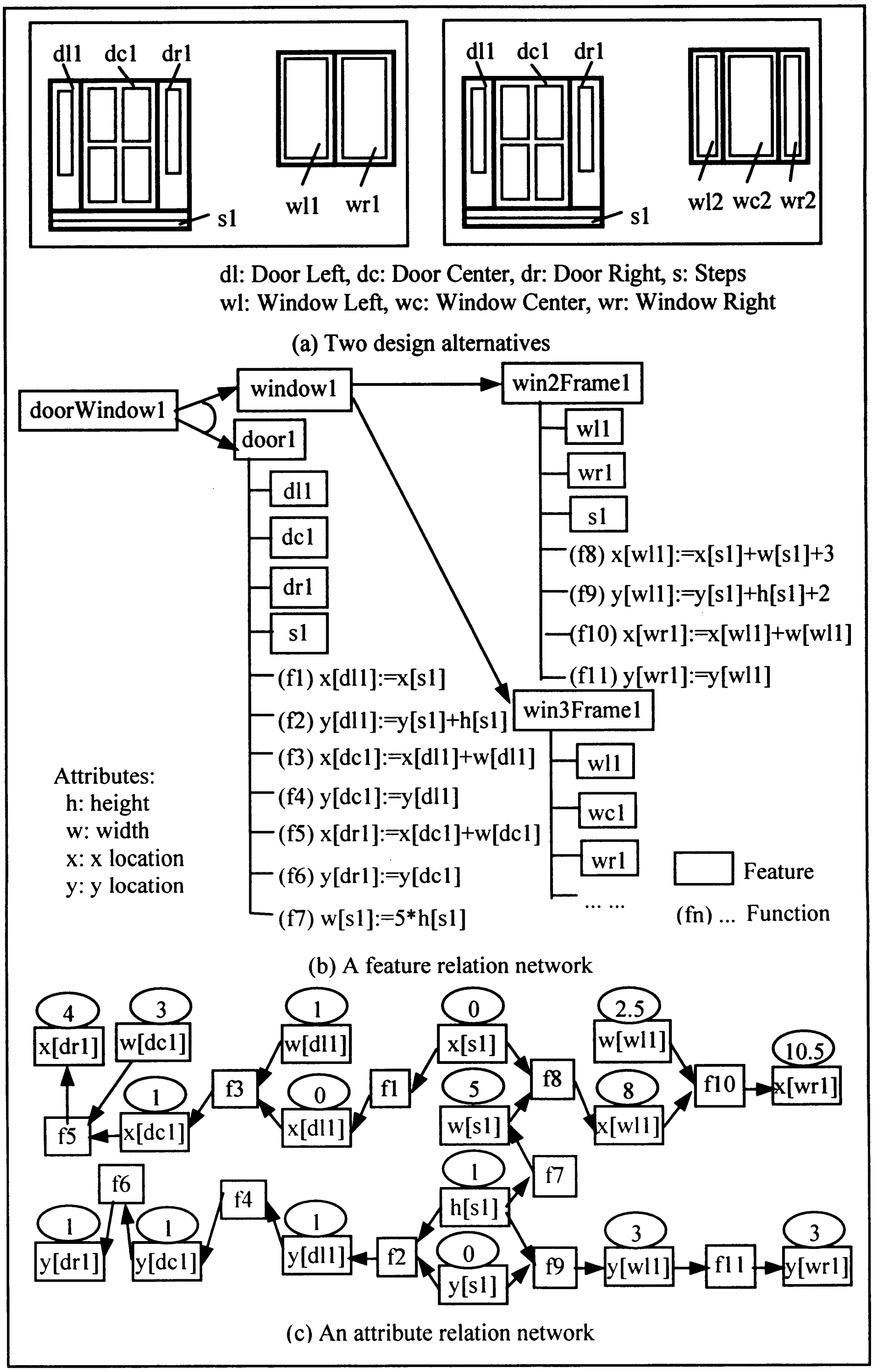


*Figure 8.* Maintenance of data relations

Since product realization process can be described by many alternatives, and each of these alternatives can be further described by attributes, the optimization is conducted at two different levels: attribute optimization level and alternative optimization level. First feasible design alternatives, which are described by instance features and relevant descriptions, are obtained from the AND/OR graph using the method introduced in Section 4.3. For each alternative, attribute optimization is carried out to identify the optimal attribute values using constrained optimization search. The optimal alternative is achieved from all the feasible alternatives at alternative optimization level. Many advanced optimization methods, including genetic algorithm and simulated annealing, were employed for improving the search efficiency and quality (Xue, 1997).

The objective function selected for optimization is based upon the concurrent design requirements. In the previous research for improving manufacturability, three types of objective functions have been introduced (Xue, Rousseau, and Dong, 1996). They are: (1) production cost function, (2) production time function, and, (3) combined cost and time function. Suppose production cost and time regarding *i*-th alternative are represented by $C_i(A_{i1}, A_{i2}, \ldots, A_{in_i})$ and $T_i(A_{i1}, A_{i2}, \ldots, A_{in_i})$ respectively, the objective function $f_i(A_{i1}, A_{i2}, \ldots, A_{in_i})$ in Eq. (18) can be represented by one of the three functions shown in Table 1. The $\alpha$ and $\beta$ in this table are weighting factors between 0 and 1 for representing the importance of production cost and time in manufacturability evaluation.

*Table 1.* Objective functions considering manufacturability

| Manufacturability Considerations | Objective Functions |
|---|---|
| Production cost only | $C_i(A_{i1}, A_{i2}, \ldots, A_{in_i})$ |
| Production time only | $T_i(A_{i1}, A_{i2}, \ldots, A_{in_i})$ |
| Both production cost and production time | $\alpha_i\, C_i(A_{i1}, A_{i2}, \ldots, A_{in_i}) + \beta_i\, T_i(A_{i1}, A_{i2}, \ldots, A_{in_i})$ |

## 5. IMPLEMENTATION OF THE INTELLIGENT CONCURRENT DESIGN SYSTEM

The intelligent concurrent design system was implemented using Smalltalk, an object oriented programming language (Goldberg and Robson, 1983), based upon the methods introduced in Section 4. Figure 9 shows the architecture of this implemented system. Users of this system are classified into two types: knowledge modeling users and product modeling users. The knowledge modeling users use two interface windows, *Class Feature Browser* and *Rule-Base Browser,* to model knowledge libraries including class features and rule-bases. The product modeling users use *Instance*

***Feature Browser*** to model product database by generating instance features. Instance features are created from class features either manually or through rule-based inference. The consistency of the database is maintained by the data relation maintenance module using qualitative relations among features and quantitative relations among attributes. The optimal design and its attribute values considering relevant life-cycle aspects are identified through optimization using the optimal design identification module. A snapshot of the implemented system is shown in Figure 10.

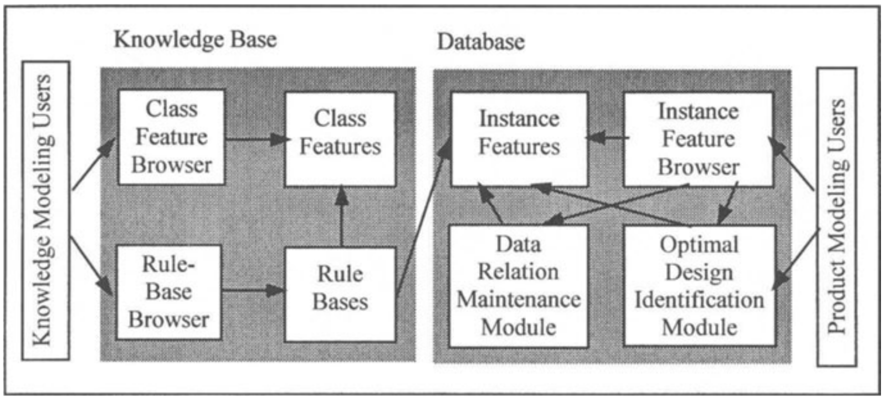
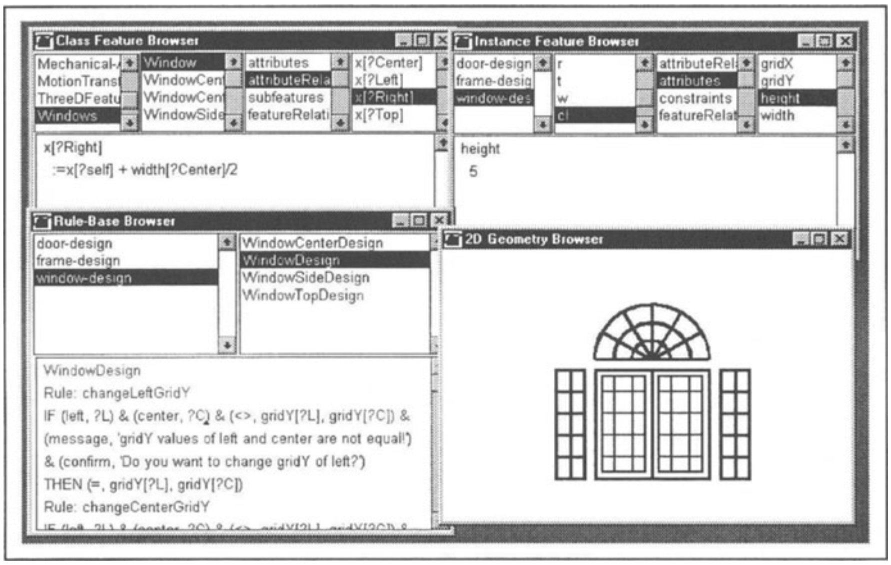


*Figure 9.* Architecture of the intelligent concurrent design system



```
x[?Right]
  :=x[?self] + width[?Center]/2
```

```
height
 5
```

```
WindowDesign
Rule: changeLeftGridY
IF (left, ?L) & (center, ?C) & (<>, gridY[?L], gridY[?C]) &
(message, 'gridY values of left and center are not equal!')
& (confirm, 'Do you want to change gridY of left?')
THEN (=, gridY[?L], gridY[?C])
Rule: changeCenterGridY
```

*Figure 10.* A snapshot of the implemented system

An industrial application for designing building products has also been developed for a local industrial company – Gienow Building Products Ltd., using the implemented system.

## 6. A CASE STUDY EXAMPLE

In this section, a case study example is given to show how a concurrent design is conducted to achieve the optimal design alternative and its attribute values using the introduced method.

The problem is to design a window frame with 2 glass panels. The required dimensions for this window are 150 cm by 100 cm. Two design candidates, *c1* and *c2* as shown in Figure 11 (a) and (b), are created from the design requirements. The width of the frame material is 5 cm. Through strength analysis, the length *x* must be long enough to support the whole window frame based upon the following two constraints: (1) *x* must be at least 20 cm, and (2) *x* must be greater than 0.035*L*, where *L* is the total frame length in the 150 cm by 100 cm window area.

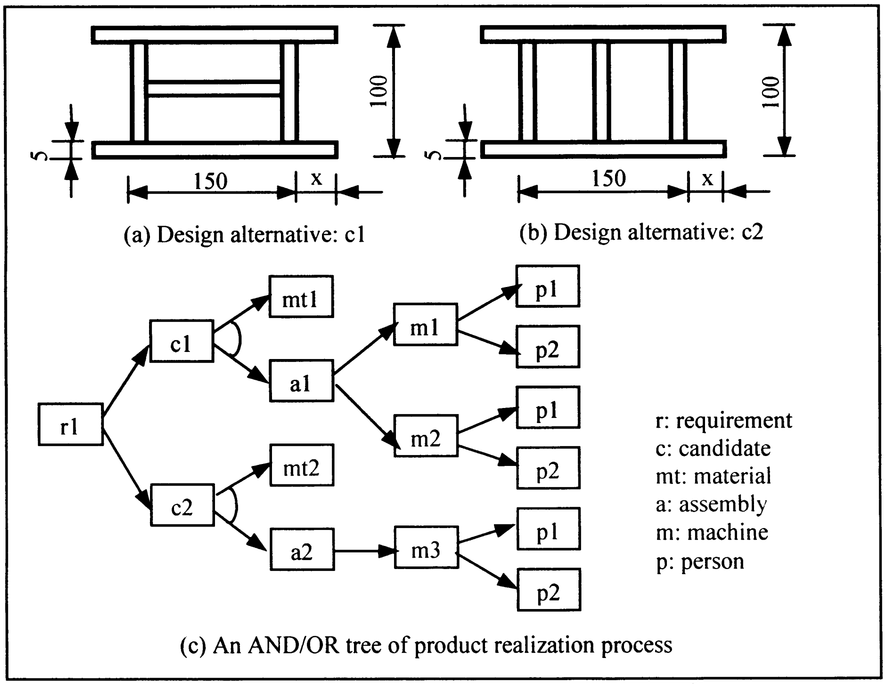


*Figure 11.* A case study example

Each of these design candidates is produced by processes of material ordering and assembly. The assembly process for candidate *c1* can be conducted using machine *m1* or *m2*. Each of these machines can be operated by person *p1* or *p2*. The assembly process for candidate *c2* can be conducted by machine *m3*, which is also operated by *p1* or *p2*. The AND/OR graph of the product realization process, which is generated through knowledge-based reasoning, is illustrated in Figure 11 (c). The nodes in the AND/OR graph are described by instance features, which are generated from class features. It requires 6 minutes for machine *m1*, or 3 minutes for machine *m2* to produce the candidate *c1*. It requires 6 minutes for machine *m3* to produce candidate *c2*. The unit machine costs and labor costs are given in Table 2. The unit cost of material is ***$0.05/cm***.

*Table 2.* Unit costs for different machines and persons

| Machines and Persons | Unit Costs ($/hour) |
|---|---|
| m1 | 10 |
| m2 | 30 |
| m3 | 15 |
| p1 | 25 |
| p2 | 30 |

From the AND/OR tree, six manufacturing alternatives are generated, as shown in Table 3. For each alternative, the length attribute *x* of the frame is selected as the variable at the attribute optimization level. The total production cost is selected as the objective function to be minimized for this problem. Using the cost models introduced in Table 2, the total cost is a function of the selected variable attribute for each alternative.

*Table 3.* Feasible alternatives and their optimal attribute values

| Alternatives | Optimal Attribute Values (cm) | Costs ($) |
|---|---|---|
| 1. r1, c1, mt1, a1, m1, p1 | $x[c1]^*=21.7$ | 38.84 |
| 2. r1, c1, mt1, a1, m1, p2 | $x[c1]^*=21.7$ | 39.34 |
| 3. r1, c1, mt1, a1, m2, p1 | $x[c1]^*=21.7$ | 38.09 |
| 4. r1, c1, mt1, a1, m2, p2 | $x[c1]^*=21.7$ | 38.34 |
| **5. r1, c2, mt2, a2, m3, p1** | $\mathbf{x[c2]^*=20.0}$ | **36.5** |
| 6. r1, c2, mt2, a2, m3, p2 | $x[c2]^*=20.0$ | 37.0 |

The optimal attribute value regarding one alternative is identified using constrained optimization. For instance, attribute optimization for the first alternative is formulated as:

$$\underset{w.r.t.\ x[c1]}{Min}\ cost[mt1] + cost[m1] + cost[p1]$$

$$= 0.05(620 + x[c1] * 4) + 6 \times 10/60 + 6 \times 25/60$$

*subject to:*

$$x[c1] \geq 20$$

$$x[c1] \geq 0.035 \times 620$$

The optimal attribute value is achieved as:

$$x^*[c1] = 21.7\ (cm)$$

and the total cost is calculated as *$38.84*. In the same way, the minimum costs for other alternatives are calculated as shown in Table 3. The optimal alternative is then identified from all the feasible alternatives.

## 7. CONCLUSIONS

In this research, modeling of knowledge and data for an intelligent concurrent design system is discussed.

The different product life-cycle aspects, including design, manufacturing, etc., are modeled by aspect primitives called features. A feature is a collection of qualitative and quantitative descriptions and their relations. Features are described at two levels, class level and instance level, representing product modeling libraries and actual product data, respectively. Instance features are generated using class features as their templates. The product realization process is modeled by an AND/OR tree. Generation of product models is carried out either manually or through knowledge-based inference. Rules are used for representing the product modeling knowledge and are organized in rule-bases. The qualitative and quantitative relations among product life-cycle aspects are maintained by two associated networks, the feature relation network and the attribute relation network. Any change can be propagated to other parts through these two associated networks. The optimal design alternative and its attribute values are identified by a multi-level optimization approach.

The introduced method has greatly improved the product modeling efficiency. This approach also provides a platform for developing the next generation CAD systems with intelligent concurrent design capabilities.

## ACKNOWLEDGEMENTS

I would like to give my thanks to my graduate student, S. Yadav, at University of Calgary for his work on improving and implementing the intelligent concurrent design system, and Prof. D. H. Norrie at University of Calgary for his collaboration on developing the industrial application. I would also like to thank Prof. H. Yoshikawa and Prof. T. Tomiyama at University of Tokyo, and Prof. Z. Dong at University of Victoria for their support in developing the product description language and the early version of the intelligent concurrent design system. Financial support from the grant of Natural Sciences and Engineering Research Council (NSERC) of Canada and Gienow Building Products Ltd. is also acknowledged.

## REFERENCES

Boothroyd, G. and Dewhurst, P. (1983) *Design for Assembly: A Designer's Handbook*, Boothroyd Dewhurst Inc., Wakerfield, RI.

Bralla, J. G. (ed.) (1986) *Handbook of Product Design for Manufacturing*, McGraw-Hill.

Chang, T. C., Wysk, R. A., and Wang, H. P. (1991) *Computer-Aided Manufacturing*, Prentice Hall.

Choi, K., Barash, M., and Anderson, D. (1984) "Automatic Recognition of Machined Surfaces from a 3-D Solid Model," *Computer-Aided Design*, Vol. 16, No. 2, pp. 81-86.

Dong, Z. (1993) "Design for Automated Manufacturing," *Concurrent Engineering: Automation, Tools, and Techniques*, Kusiak, A. (ed.), John Wiley & Sons, pp. 207-234.

Dong, Z. (ed.) (1994) *Artificial Intelligence in Optimal Design and Manufacturing*, Prentice Hall.

Dong, Z., Hu, W., and Xue, D. (1994) "New Production Cost-tolerance Models for Tolerance Synthesis," *Journal of Engineering for Industry, Transaction of ASME*, Vol. 116, pp 199-206.

Dowlatshahi, S. (1992) "Product Design in a Concurrent Engineering Environment: An Optimization Approach," *International Journal of Production Research*, Vol. 30, No. 8, pp. 1803-1818.

Gershenson, J. and Ishii, K. (1993) "Life-cycle Serviceability Design," *Concurrent Engineering: Automation, Tools, and Techniques*, Kusiak, A. (ed.), John Wiley & Sons, pp. 363-384.

Goldberg, A. and Robson, D. (1983) *Smalltalk-80: The Language and Its Implementation*, Addison-Wesley.

Henderson, M. R. (1984) "Extraction of Feature Information from Three Dimensional CAD Data," Ph.D. Dissertation, Purdue University.

Kusiak, A. (ed.) (1992) *Intelligent Design and Manufacturing*, John Wiley & Sons.

Kusiak, A. (ed.) (1993) *Concurrent Engineering: Automation, Tools, and Techniques*, John Wiley & Sons.

Luby, S. C., Dixon, J. R., and Simmons, M. K. (1986) "Creating and Using a Feature Databases," *Computers in Mechanical Engineering*, Vol. 5, No. 3, pp. 25-33.

Makino, A., Barkan, P., and Pfaff, R. (1989) "Design for Serviceability," *Proceedings of the 1989 ASME Winter Annual Meeting*, San Francisco, CA.

Prasad, B. (1996) *Concurrent Engineering Fundamentals: Volume I*, Prentice Hall.

Pratt, M. J. (1984) "Solid Modeling and the Interface between Design and Manufacturing," *IEEE Computer Graphics and Application Magazine*, pp. 52-59.

Shah, J. J. and Mantyla, M. (1995) *Parametric and Feature-Based CAD/CAM*, John Wiley & Sons.

Shah, J. J. and Rogers, M. T. (1988) "Functional Requirements and Conceptual Design of the Feature-based Modeling System," *Computer-Aided Engineering Journal*, Vol. 5, No. 1, pp. 9-15.

Singh, N. (1996) *Systems Approach to Computer-Integrated Design and Manufacturing*, John Wiley & Sons.

Tomiyama, T. (1994) "From General Design Theory to Knowledge-Intensive Engineering," *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing*, Vol. 8, No. 4, pp. 319-333.

Tomiyama, T. (1997) "A Manufacturing Paradigm towards the 21st Century," *Integrated Computer Aided Engineering*, Vol. 4, pp. 159-178.

Tomiyama, T. and ten Hagen, P. J. W. (1987) "The Concept of Intelligent Integrated Interactive CAD Systems," CWI Report No. CS-R8717, Centre for Mathematics and Computer Science, Amsterdam, The Netherlands.

Tomiyama, T. and Yoshikawa, H. (1987) "Extended General Design Theory," *Design Theory for CAD, Proceedings of the IFIP Working Group 5.2 Working Conference 1985 (Tokyo)*, Yoshikawa, H. and Warman, E. A. (eds.), North-Holland, Amsterdam, pp. 95-130.

Xue, D. (1997) "A Multilevel Optimization Approach Considering Product Realization Process Alternatives and Parameters for Improving Manufacturability," *Journal of Manufacturing Systems*, Vol. 16, No. 5, pp. 337-351.

Xue, D. and Dong, Z. (1993) "Feature Modeling Incorporating Tolerance and Production Process for Concurrent Design," *Concurrent Engineering: Research and Applications*, Vol. 1, pp. 107-116.

Xue, D. and Dong, Z. (1994) "Developing a Quantitative Intelligent System for Implementing Concurrent Engineering Design," *Journal of Intelligent Manufacturing*, Vol. 5, pp. 251-267.

Xue, D. and Dong, Z. (1997) "Coding and Clustering of Design and Manufacturing Features for Concurrent Design," *Computers in Industry*, Vol. 34, pp. 139-153.

Xue, D., Rousseau, J. H., and Dong, Z. (1996) "Joint Optimization of Performance and Costs in Integrated Concurrent Design: Tolerance Synthesis Part," *Engineering Design and Automation*, Vol. 2, No. 1, pp. 73-89.

Xue, D., Takeda, H., Kiriyama, T., Tomiyama, T., and Yoshikawa, H. (1992) "An Intelligent Integrated Interactive CAD – A Preliminary Report," *Intelligent Computer Aided Design*, Waldron M. B., Brown, D., and Yoshikawa, H. (eds.), North-Holland, Amsterdam, pp. 163-192.

Yoshikawa, H. (1981) "General Design Theory and CAD Systems," *Man-machine Communication in CAD/CAM*, Sata, T. and Warman, E. (eds.), North-Holland, Amsterdam, pp. 35-58.

Yoshikawa, H. (1993) *Techno-Globe*, Sangyo Chosa-kai, Tokyo.

Young, R. E., Greef, A., and O'Grady, P. (1992) "An Artificial Intelligence-based Constraint Network System for Concurrent Engineering," *International Journal of Production Research*, Vol. 30, No. 7, pp. 1715-1735.

Zhang, H. C., Kuo, T. C., Lu, H., and Huang, S. H. (1997) "Environmentally Conscious Design and Manufacturing: A State-of-the-Art Survey," *Journal of Manufacturing Systems*, Vol. 16, No. 5, pp. 352-371.