Design Process Modeling for the Future CAD Systems

D. Xue*, H. Yang, and Y. L. Tu

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

E-mails: xue@enme.ucalgary.ca, yhg@enme.ucalgary.ca, paultu@enme.ucalgary.ca

Abstract

To develop the future CAD systems that support functions of design process modeling, an evolutionary design database model is introduced in this research to describe design requirements and design results developed at different design stages from conceptual design to detailed design. In this model, the evolutionary design database is represented by a sequence of worlds corresponding to the design descriptions at different design stages. In each world, only the differences with its ancestor world are recorded. When the design descriptions in one world are changed, these changes are then propagated to its descendant worlds automatically. Case study is conducted to show the effectiveness of this evolutionary design database model.

1. Introduction

Although many computer aided design (CAD) systems have been developed and employed for engineering design and manufacturing, these CAD systems are primarily used for modeling the final geometric design result developed at the detailed design stage. New CAD systems need to be developed to support various design activities in different design stages from conceptual design to detailed design. The issues to be addressed for developing the future CAD systems are primarily summarized into the following two categories: modeling of design objects and modeling of design processes.

In our previous research, a design database representation scheme was introduced for modeling design objects developed at different design stages by integrating geometric descriptions and non-geometric descriptions into the same environment [1]. This scheme was developed based on the research results achieved by the Design Repository Project at the

National Institute of Standards and Technology (NIST) [2] and the Design Database Modeling Project at the University of Calgary [3-5]. The research given in this paper focuses on the design process modeling aspect by introducing an evolutionary design database model to record the process from conceptual design to detailed design.

Many researches on design process modeling have been conducted in the past 3 decades. In Yoshikawa's General Design Theory, a design process is defined as a mapping from functional space to attribute space [6]. To model the mapping process, Tomiyama and Yoshikawa introduced a meta-model mechanism to describe the design descriptions at different design stages using a sequence of meta-models [7]. Kiriya et al. employed the qualitative physics (QP) theory to describe the meta-models and to integrate the design descriptions, usually created at different design stages, into the same environment using meta-models [8]. Takeda et al. developed a computable design process model to manipulate the evolution of design database [9].

Despite the progress, the following two issues need to be addressed for design process modeling.

1. When design history needs to be recorded in the conventional CAD systems, a number of files are then created by saving the design database at different design stages. Changes of design descriptions in one file are not propagated to the subsequent databases saved in other files. The evolutionary process of design database from conceptual design to detailed design is not well studied.
2. Although multiple results of a product family, usually called different configurations, can be modeled in some CAD systems (e.g., SolidWorks and Pro/ENGINEER) by assigning dimensions with different sets of values or by activating/freezing different features and parts in one CAD database,

* Corresponding author.

these different results only describe the design alternatives developed at the same design stage. Modeling of the design alternatives created at different design stages has never been discussed.

This research aims at solving these problems by introducing an evolutionary design database model. In this model, the evolutionary design database is described by a collection of worlds. Each world models the design descriptions created at a certain design stage. The database representation scheme introduced in our previous research [1] is employed in this work to model the design descriptions developed at different design stages.

2. An Evolutionary Design Database Model

The evolutionary design database is modeled by a collection of *worlds* as shown in Fig. 1.

Figure 1. An evolutionary design database modeled by worlds

$$W_1 \rightarrow W_2 \rightarrow W_3 \rightarrow \dots \rightarrow W_n \quad (1)$$

Each world, W_i , models design descriptions, including design requirement and design result, developed at a certain design stage. In Fig. 1, a design is evolved from the world W_1 with the design requirement to the world W_{10} with the final design result.

In each world, design descriptions, including design requirement and design result, are modeled by *objects*.

$$W_i = \{O_{i,1}, O_{i,2}, \dots, O_{i,m_i}\}, \quad i = 1, 2, \dots, n \quad (2)$$

such as

$$\{\text{Gear-Pair-1}, \text{Gear-1}, \text{Gear-2}, \text{Shaft-1}, \text{Shaft-2}\}$$

An object is composed of *elements*.

$$O_{i,j} = \{O_{i,j}.E_1, O_{i,j}.E_2, \dots, O_{i,j}.E_{p_{i,j}}\}, \\ i = 1, 2, \dots, n; j = 1, 2, \dots, m_i \quad (3)$$

An element of the object is assigned with a value of either a *primitive* (e.g., 3.14) or an object in the form of

$$\langle \text{object} \rangle. \langle \text{element} \rangle = \langle \text{value} \rangle$$

such as

$$\text{Gear-1.tooth-number} = 25$$

$$\text{Gear-Pair-1.g1} = \text{Gear-1}$$

The database in one world is changed by the following five types of design description operations.

- *Object creation* to add a new object to the world.

$$o_{i,j}^{(c)} = O^{(c)} O_{i,j}, \quad i = 1, 2, \dots, n; j = 1, 2, \dots, m_i \quad (4)$$

- *Object removal* to remove an existing object in the world.

$$o_{i,j}^{(r)} = O^{(r)} O_{i,j}, \quad i = 1, 2, \dots, n; j = 1, 2, \dots, m_i \quad (5)$$

- *Element creation* to add an element to an existing object in the world.

$$e_{i,j,k}^{(c)} = E^{(c)} O_{i,j}.E_{i,j,k}, \quad i = 1, 2, \dots, n; \\ j = 1, 2, \dots, m_i; k = 1, 2, \dots, p_{i,j} \quad (6)$$

- *Element removal* to remove an existing element of an existing object in the world.

$$e_{i,j,k}^{(r)} = E^{(r)} O_{i,j}.E_{i,j,k}, \quad i = 1, 2, \dots, n; \\ j = 1, 2, \dots, m_i; k = 1, 2, \dots, p_{i,j} \quad (7)$$

- *Element modification* to modify an existing element of an existing object in the world.

$$e_{i,j,k}^{(m)} = E^{(m)} O_{i,j}.E_{i,j,k}, \quad i = 1, 2, \dots, n; \\ j = 1, 2, \dots, m_i; k = 1, 2, \dots, p_{i,j} \quad (8)$$

A world W_i is evolved to another world W_j through a sequence of design description operations.

$$W_i \xrightarrow{o_1^{(c)} o_2^{(c)} \dots o_1^{(r)} o_2^{(r)} \dots e_1^{(c)} e_2^{(c)} \dots e_1^{(r)} e_2^{(r)} \dots e_1^{(m)} e_2^{(m)} \dots} W_j \quad (9)$$

where W_i is called an *ancestor world* of W_j , while W_j is called a *descendant world* of world W_i .

When only object creation operations and element creation operations are conducted, the descendant world W_j is then a super set of world W_i , representing more design descriptions are added to the database.

$$W_i \subset W_j \quad (10)$$

The evolution of the worlds is carried out by the following five types of world operations.

- *World creation* to create a new world without any design descriptions.

$$\xrightarrow{W^{(c)}} W_i \quad (11)$$

The W_1 and W_5 in Fig. 1 are created in this way.

- *World evolution* to evolve an existing world to a new world.

$$W_i \xrightarrow{W^{(e)}W_i} W_i \rightarrow W_j \quad (12)$$

The W_2 in Fig. 1 is evolved from W_1 . Multiple worlds, representing different design alternatives, can be evolved from the same world. For example, both W_3 and W_4 in Fig. 1 are evolved from W_2 .

- *World removal* to remove an existing world.

$$W_i \xrightarrow{W^{(r)}W_i} \quad (13)$$

When a world is removed, its ancestor world is evolved to its descendant world directly. For example, when W_3 in Fig. 1 is removed, W_2 is then connected to W_6 directly.

- *World insertion* to insert a new world between two existing worlds.

$$W_i \rightarrow W_j \xrightarrow{W^{(i)}W_i, W_j} W_i \rightarrow W_k, W_k \rightarrow W_j \quad (14)$$

- *World merge* to combine several worlds into a new world.

$$W_i, W_j, \dots \xrightarrow{W^{(m)}W_i, W_j, \dots} W_i \rightarrow W_p, W_j \rightarrow W_p, \dots \quad (15)$$

In Fig. 1, the W_4 and W_5 are merged into W_8 .

3. A Design Database Representation Scheme

The design database representation scheme introduced in our previous research [1] is employed in this work to model design descriptions in one world. In this scheme, design descriptions are modeled at three different levels: *meta-class* level, *class* level, and *instance* level, as shown in Fig. 2.

The meta-classes describe the concepts used in the database. These concepts are classified into five major categories: *entities*, *properties*, *relationships*, *tasks*, and *specifications*. The behaviors of these concepts are also described in the meta-classes. Meta-classes cannot be modified by the design engineers. The classes describe the generic design libraries, such as gear, shaft, gear-pair, etc. The design engineers can define their own classes, usually using the existing classes as the super-classes. The instances describe the special design cases. The instances are created using the classes as the templates. In Fig. 2, the created mechanism is composed of 1 motor and 2 gear-pairs, which are generated using the motor class and the gear-pair class as the templates.

Figure 2. Three levels of design database descriptions

The relations among meta-classes are summarized in Table 1. A class can be defined using a meta-class as the template. An instance is created either as an *object* with composing elements (e.g., a gear-pair composed of two gears) using a class as the template or as a *primitive* (e.g., an attribute defined as `diameter[gear-1]=20`) without using a class as the template. The relations among meta-classes, classes, and instances are also given in Table 1.

Entities are physical units for modeling designs. Entities are classified into *artifacts* and *features*. An artifact, either a *component* or an *assembly*, represents a distinct entity separable from others and created in a design. A feature is a collection of descriptions in an artifact for a certain purpose. The different features of the same artifact represent the different views of the artifact in the design. For example, a hole can be viewed both as a design feature for ventilation function and as a manufacturing feature to be machined by a drilling process.

Properties are the inherited characteristics belonging to artifacts. Typical properties include *data*, *functions*, *behaviors*, *forms*, and *flows*. A piece of data, in the form of an *attribute*, a *dimension*, or a *tolerance*, represents the value of a certain aspect of an artifact. Functions are the teleological descriptions of artifacts. Functions are usually linked by *flows*, including *material flows*, *energy flows*, and *information flows*. Functions are classified into four categories according to their input flows and output flows: *atomic functions* without input and output flows, *source functions* with only output flows, *destination functions* with only input flows, and *transfer functions* with both input and output flows. Behaviors of artifacts are the ways to implement the functions of the artifacts. Behaviors are derived from the *forms* (physical characteristics including *geometry* and *materials*) of the artifacts and science/engineering principles. Behaviors are usually described by the responses to the stimulus of artifacts, such as a sequence of artifact description changes,

Table 1. Meta-classes, classes, and instances

Meta-Class	Class	Instance
Entity		
Artifact		
Component	Yes	Object
Assembly	Yes	Object
Feature		
Design-Feature	Yes	Object
Manufacturing-Feature	Yes	Object
Assembly-Feature	Yes	Object
Maintenance-Feature	Yes	Object
Recycle-Feature	Yes	Object
Property		
Data		
Attribute		Primitive
Dimension		Primitive
Tolerance		Primitive
Function		
Atomic-Function	Yes	Object
Source-Function	Yes	Object
Destination-Function	Yes	Object
Transfer-Function	Yes	Object
Behavior		
Continuous-Time-Behavior	Yes	Object
Discrete-Time-Behavior	Yes	Object
State-Transition-Behavior	Yes	Object
Form		
Geometry	Yes	Object
Material	Yes	Object
Flow		
Material-Flow	Yes	Object
Energy-Flow	Yes	Object
Information-Flow	Yes	Object
Relationship		
Declarative-Relationship		Primitive
Dependency-Relationship		
Attribute-Relationship		Primitive
Logic-Relationship		Primitive
Constraint		
Arithmetic-Constraint		Primitive
Temporal-Constraint		Primitive
Spatial-Constraint		Primitive
Task	Yes	Object
Specification	Yes	Object

usually with respect to the change of the time parameter. Behaviors are classified into three categories: *continuous-time behaviors*, *discrete-time behaviors*, and *state-transition behaviors*.

Relationships associate descriptions of artifacts. Relationships are classified into *declarative relationships*, *dependency relationships*, and *constraints*. Declarative relationships are defined by predicates such as (pair, gear-1, gear-2). A dependency relationship defines the relationship between one output and a number of inputs. When the input descriptions are changed, the output description should be updated automatically. The dependency relationships are classified into two categories:

attribute relationships and *logic relationships*. An attribute relationship defines the relationship between an output attribute and a number of input attributes. A logic relationship associates a number of artifact descriptions using an AND-OR tree. Constraints are relationships that should always be satisfied during the design process. Constraints are classified into three categories: *arithmetic constraints*, *temporal constraints*, and *spatial constraints*. An arithmetic constraint defines the relationship among attributes by expressions linked by comparison operators and logic operators. Temporal constraints are used to define the sequential relationships among tasks. Spatial constraints are those related to geometric elements, such as surfaces, edges, vertices, and so on.

A *task* defines an activity in a down-stream product development life-cycle aspect. Typical tasks include *manufacturing tasks*, *assembly tasks*, *maintenance tasks*, *recycle tasks*, etc. These tasks are associated with life-cycle aspect features. The sequential relationships among these tasks are defined by temporal constraints.

A *specification* defines the design requirements that must be satisfied. A specification is described by existing entities (including artifacts and features), relationships, and functions linked with logic-AND and logic-OR relationships. The constraints are defined inside functions. A design is completed when all the requirements in the specification are satisfied.

Figure 3. Design requirement and design result

Fig. 3 shows the design descriptions modeled in one world. The design requirement is described by an AND tree with 1 function and 2 flows. The design result is described by a tree with 1 assembly and 4 components.

4. Evolutionary Design Database Modeling

This section provides details of the evolutionary design database modeling method.

4.1. Evolution of Worlds

The evolutionary design database is modeled by a collection of worlds as shown in Fig. 1. Each world describes design requirement and design result developed at a certain design stage. A design is evolved from the world with original design requirement to the world with final design result. When a sequential relation between two worlds is defined, the world created at an earlier stage is called an *ancestor world*, while the world created at a later stage is called a *descendant world*. Among all these worlds, only one is selected as the *active world* for design database modeling.

Five types of world operations, as shown in Fig. 4, can be conducted to manipulate the evolution process of the worlds.

- *World creation* to create a new world without any design descriptions. This world is used either for specifying design requirements or for modeling partial design database to be merged with other worlds.
- *World evolution* to evolve an existing world to a new world. All the design descriptions in the ancestor world are considered as design descriptions in this descendant world. The user can add new design descriptions or remove existing design descriptions in the new world. Multiple worlds, representing different design alternatives, can be evolved from the same world.
- *World removal* to remove an existing world. When a world is removed, the design descriptions defined in that world are removed from its descendant worlds. When the removed world is connected with an ancestor world and a descendant world, the ancestor world is then connected to the descendant world directly after the world is removed.
- *World insertion* to insert a new world between two existing worlds. This mechanism allows the design engineer to make fine modifications to the design database between two major design stages.
- *World merge* to combine several worlds into a new world. The design engineer can first create several

Figure 4. Five types of world operations

worlds to model partial design results in these worlds, and then achieve the final design result by combining these partial design results.

4.2. Design Database Modeling in the Active World

When a descendant world created from an ancestor world is selected as the active world as shown in Fig. 5, the design descriptions in the ancestor world and its ancestor worlds are viewed as *virtual design descriptions* in the active world. The design engineer can create new design descriptions in the active world. The design engineer can also remove or modify design descriptions created in the ancestor worlds. The design descriptions defined in the current world are called *true design descriptions*. The true design descriptions in a world represent the differences between this world and its ancestor world. Both the virtual design descriptions and the true design descriptions are treated the same when creating the design requirement tree and the design result tree. Since the virtual design descriptions

Figure 5. Virtual and true design descriptions

are defined in the ancestor worlds, when these virtual design descriptions are changed in the ancestor worlds, these changes are propagated to the active world automatically.

Design database modeling in the active world is conducted by the following five types of design description operations.

- *Object creation* to add a new object to the world. An object is a special instance composed of elements. The classes to be used to create objects can be found in Table 1. An object is created in the form of

CreateObject <Class> <Instance>

such as

CreateObject Gear gear-1

- *Object removal* to remove an existing object created in an ancestor world. The object is removed in the form of

RemoveObject <Instance>

When an object is removed, all the relevant design descriptions should also be removed from the database. For example, when the object gear-1 is removed, the attributes of this gear and its relationship with another gear, (pair, gear-1, gear-2), should also be removed from the database.

- *Element creation* to add an element to an existing object created in an ancestor world. An element is added in the form of

CreateElement <Type> <Instance> <Element>

such as

CreateElement Attribute gear-1 d[gear-1]=40

- *Element removal* to remove an existing element of an existing object. Both the object and the element are created in ancestor worlds. An element is

removed in the form of

RemoveElement <Type> <Instance> <Element>

- *Element modification* to modify an existing element of an existing object. Both the object and the element are created in ancestor worlds. An element is modified in the form of

ModifyElement <Type> <Instance>
<New Element>

The design requirement tree and the design result tree in the active world are created using the design descriptions in both the active world and its ancestor worlds. These two trees in the active world are created through the following steps.

- (1) Obtain the design descriptions, including both the virtual design descriptions and the true design descriptions, from its ancestor worlds.
- (2) Change the obtained design descriptions using the design description operations conducted in the active world.
- (3) Create the tree of design requirement using the Specification object as the root node of this tree.
- (4) Create the tree of design result using the Product object as the root node of this tree.

The design attributes in the active world, World-4, shown in Fig. 6 are created in the following steps.

- (1) From the World-4, requests are sent to its ancestor worlds World-2 and World-3.
- (2) From the World-2, a request is sent to its ancestor world World-1.
- (3) In World-1, the design attributes are achieved as

a1[C1]=11
a2[C1]=12
a3[C1]=13
a1[C2]=21

Figure 6. Design database modeling by design description operations

These design attributes are returned to World-2.

(4) In World-2, the design attributes are achieved as

a1[C1]=11
a2[C1]=120
a3[C1]=13
a4[C1]=14

These design attributes are returned to World-4.

(5) In World-3, the design attributes are achieved as

a1[C3]=31
a2[C3]=32

These design attributes are returned to World-4.

(6) In World-4, the design attributes are achieved as

a1[C1]=11
a2[C1]=120
a4[C1]=14
a1[C3]=31
a2[C3]=32
a1[C4]=41
a2[C4]=42

Details of the formation and evolution of the design requirement tree and the design result tree will be discussed in Section 4.3 and Section 4.4, respectively.

4.3. Evolution of Design Requirements in Worlds

The design requirement in a world is modeled by a tree with design functions as the major nodes of this tree, as shown in Fig. 7. A function can be decomposed into sub-functions in the design process. The root node of the tree is modeled by a Specification object. When alternative sub-functions need to be created, multiple descendant worlds are then generated. In Fig. 7, since function F2 can be decomposed into either F4 AND F5 or F6 AND F7, two worlds, World-2 and World-3, are created for modeling these two alternative design requirements. The World-3 is further evolved by decomposing the functions F6 and F7 into sub-functions. The functions in the design requirement tree

Figure 7. Evolution of design requirements

are created in the active world and its ancestor worlds using the five types of design description operations introduced in Section 4.2. A function created in an ancestor world is called a *virtual function* described in the form of

<world>@<function>

such as

world-1@rotation-to-rotation-1

When the design requirements are defined by an AND-OR tree as shown in Fig. 8 (a), creation of multiple AND trees from this AND-OR tree is then conducted. These AND trees form the descendant worlds for modeling different design alternatives. The following rules are used for creating the alternative AND trees.

- (1) The root node, Specification object, has to be selected first.
- (2) When a node is selected and all of its sub-nodes are associated with an AND relationship, all these sub-nodes should be selected.
- (3) When a node is selected and all of its sub-nodes are associated with an OR relationship, only one of these sub-nodes should be selected.

(a). An AND-OR tree

(b). Created alternative AND trees

Figure 8. An AND-OR tree and its alternative AND trees

The AND trees created from the AND-OR tree shown in Fig. 8. (a) are given in Fig. 8 (b).

4.4. Evolution of Design Results in Worlds

The design result in a world is modeled by a tree with artifacts including components and assemblies, as shown in Fig. 9. The root node of the design result tree is a special assembly called **Product**. The artifacts in the design result tree are created in the active world and its ancestor worlds using the five types of design description operations introduced in Section 4.2. An artifact created in an ancestor world is called a *virtual artifact* described in the form of

<world>@<artifact>

such as

world-1@gear-pair-1

The elements of artifacts are also created in the active world and its ancestor worlds. An element defined in an ancestor world is described in the form of

<world>@<artifact>.<world>@<element>

such as

world-1@gear-1.world-2@diameter[gear-1]=40

During the design process, first the design result is created based on the design requirement either manually or using a knowledge based system. The behaviors of the design result are then achieved to check whether the design requirement, usually modeled by functions and constraints, can be satisfied. Multiple design candidates, modeled by different components and assemblies, can be created from the same design requirement. For each design candidate, the design attributes can be optimized based upon a selected objective function such as the efficiency of a power generator or the cost of this power generator. The design candidate with the best evaluation measure is then selected for further design improvement.

Figure 9. Evolution of design results

In the design result evolution process shown in Fig. 9, first an assembly with two components C1 and C2 is created in World-1. Two design alternatives, (1) an assembly A1 with components C3 and C4, and (2) an assembly A2 with components C5 and C6, are created in two descendant worlds World-2 and World-3. Since the cost of the product modeled in World-3 is lower than the cost of the product modeled in World-2, the World-3 is selected to create a descendant world World-4 for further design development.

5. System Implementation and a Case Study Example

The evolutionary design database modeling system has been developed as a Web-based system. Dynamic HTML, ASP, Microsoft Access, and VRML have been used to implement this system. A snapshot of the implemented system is given in Fig. 10.

A case study has been conducted to illustrate the effectiveness of the introduced evolutionary design database modeling approach. This case study was originally developed in our previous research on design database modeling [1] and modified in this research. The objective of the design is to create a mechanism to transform the rotation motion of a motor to the translation motion of a piston for a power fluid pump in a heating system. The evolutionary design database, modeled by multiple worlds, is given in Fig. 11.

The design requirements are defined as the design specification with the function R-L-1 (rotation-to-translation-1) in World-1. This function is to transform a rotational energy-flow created by the motor artifact into the translational energy-flow to drive the piston artifact. Design constraints are also defined in the specification. All the requirements and constraints defined in the specification have to be satisfied by the created design. In this case study, the following constraints are defined in the specification.

- (1) The rotational speed of the rotation motion created by the motor should be between 1000 rpm and 3000 rpm.
- (2) The ratio of the output rotational speed to the input rotational speed for the rotation-to-rotation transformation function should be between 0.05 and 1.
- (3) The maximum velocity of the piston should be less than 1.8 m/s.
- (4) The capacity of the pump, calculated by multiplying the cross section area of the piston (m^2), stroke of the piston (m), and number of cycles per unit time (cycles/min), should be between $0.32 m^3/min$ and $1.85 m^3/min$.

Figure 10. A snapshot of the implemented system

Figure 11. An evolutionary design database modeled by multiple worlds

This main function is decomposed into two sub-functions: the function R-R-1 (rotation-to-rotation-1) to transform the high-speed rotation motion to the low-speed rotation motion, and the function R-L-2 (rotation-to-translation-2) to transform the low-speed rotation motion to the translation motion of the piston.

Two design alternatives, a pulley-belt-drive mechanism and a gear-pair mechanism, are created in two different worlds, World-2 and World-3, to satisfy the design function R-R-1.

Since the gear-pair mechanism provides higher power output and takes less space compared with the

Table 2. Evaluation of the two design candidates

Design Parameter	Specification	Candidate 1	Candidate 2
Rotational speed of the motor (rpm)	1000-3000	1500	1500
Ratio of the rotational speeds for the gear-box	0.05-1	0.0625	0.0625
Maximum speed of the piston (m/s)	< 1.8	0.937	1.022
Capacity of the pump (m ³ /min)	0.32-1.85	0.33	0.33
Design Function Evaluation		Satisfied	Satisfied
Manufacturing Cost Evaluation		\$994.35	\$238.83
Design Decision		Not Selected	Selected

pulley-belt-drive mechanism, the World-3 is selected for further design evolution. Two design alternatives, a cam-follower mechanism and a crank-slider mechanism, are created separately in World-4 and World-5 to satisfy the R-L-2 function.

These two design candidates are evaluated through behavior-based simulation. The evaluation results for these two candidates are shown in Table 2. All the design constraints are satisfied by both design candidates.

Although all the requirements and constraints defined in the specification have been satisfied by each design candidate, the manufacturing costs for these two designs are significantly different as shown in Table 2. The design candidate with the crank-slider mechanism is selected due to its lower production cost.

6. Conclusions

An evolutionary design database model is introduced in this research to describe design requirements and design results developed at different design stages for developing future CAD systems that support functions of design process modeling. In this model, the evolutionary design database is described by a sequence of worlds. A design database representation scheme introduced in our previous research is employed in this work to model the database in these worlds. In each world, only the differences with its ancestor world are recorded. When the database descriptions in one world are changes, these changes are then propagated to its descendant worlds automatically.

Characteristics of this research are summarized as follows.

1. By modeling design descriptions, including design requirements and design results, developed at different design stages using a collection of worlds,

the evolutionary process of the design database can be recorded.

2. By building the evolutionary design database using our previously introduced design database representation scheme that integrates non-geometric design descriptions and geometric design descriptions, the design descriptions developed at different design stages, such as conceptual design and detailed design, can be modeled in this evolutionary design database.
3. By describing only the differences between a world and its ancestor world, modifications to the design descriptions developed at an early design stage can be propagated to the subsequent design stages automatically.

The effectiveness of the introduced evolutionary design database modeling approach has been demonstrated by a case study to obtain the optimal design through design evolution with 5 worlds.

References

- [1] D. Xue and H. Yang, "A Concurrent Engineering-Oriented Design Database Representation Model," *Computer-aided Design*, Vol. 36, No. 10, 2004, pp. 947-965.
- [2] S. J. Fenves, A Core Product Model for Representing Design Information, NISTIR 6736, National Institute of Standards and Technology, Gaithersburg, Maryland, 2001.
- [3] D. Xue, S. Yadav and D. H. Norrie, "Knowledge Base and Database Representation for Intelligent Concurrent Design," *Computer-Aided Design*, Vol. 31, No. 2, 1999, pp. 131-145.
- [4] F. Zhang and D. Xue, "Distributed Database and Knowledge Base Modeling for Concurrent Design," *Computer-Aided Design*, Vol. 34, No. 1, 2002, pp. 27-40.
- [5] D. Xue and Y. Xu, "Web-based Distributed System and Database Modeling for Concurrent Design," *Computer-Aided Design*, Vol. 35, No. 5, 2003, pp. 433-452.
- [6] H. Yoshikawa, "General Design Theory and CAD Systems," *Man-machine Communication in CAD/CAM*, T. Sata and E. Warman (eds.), North-Holland, Amsterdam, 1981, pp. 35-58.
- [7] T. Tomiyama and H. Yoshikawa, "Extended General Design Theory," *Design Theory for CAD, Proceedings of the IFIP Working Group 5.2 Working Conference 1985 (Tokyo)*, H. Yoshikawa and E. Warman (eds.), North-Holland, Amsterdam, 1987, pp. 95-130.
- [8] T. Kiriya, T. Tomiyama and H. Yoshikawa, "The Use of Qualitative Physics for Integrated Design Objects Modeling," *Proceedings of ASME Design Theory and Methodology Conference*, Miami, FL, 1991, pp. 53-60.
- [9] H. Takeda, P. Veerkamp, T. Tomiyama and H. Yoshikawa, "Modeling Design Process," *AI Magazine*, Vol. 11, No. 4, 1990, pp. 37-48.