Design Database Modeling for the Future CAD Systems

Deyi Xue* and Haoguang Yang
Department of Mechanical and Manufacturing Engineering, University of Calgary
E-mails: xue@enme.ucalgary.ca, yhg@enme.ucalgary.ca

Abstract
To develop the future CAD systems with concurrent engineering functions, a new CAD model, called Concurrent Engineering-oriented Design Database Representation Model (CE-DDRM), is introduced in this research for supporting various life-cycle aspects in concurrent design. In this model, concepts and behaviors of different design database modeling components, including entities, properties, relationships, tasks, and specifications, are defined at meta-class level. Design database is modeled at two different levels, class level and instance level, representing generic design libraries and special design cases, respectively.

1. Introduction
Although many CAD and other computer-based systems have been developed for engineering design and manufacturing, these systems are primarily used for modeling and manipulation of geometric information, such as solid modeling, CNC machining path generation, and so on. Modeling of non-geometric design information, such as design requirements, conceptual design candidates and other product development life-cycle considerations, is not well understood and studied for developing these computer-based systems. Therefore, the CAD systems that support the entire product development life-cycle, especially the processes at the early conceptual design stage, have to be introduced.

Among all the design database modeling approaches, the research on design functions, behaviors, and forms (also called structures) has attracted attention of many researchers since the 1980s [1,2]. At the National Institute of Standards and Technology (NIST), USA, a Design Repository Modeling Project was initiated to describe design knowledge using the Function-Behavior-Form database structure [3-6]. At the University of Calgary, a Design Database Modeling Project was conducted to support concurrent design using primitives called features [7-11].

The objective of this research is to introduce a new database representation model for concurrent engineering design by incorporating the Function-Behavior-Form based design modeling approach introduced at the NIST and the concurrent engineering based design modeling approach introduced at the University of Calgary.

2. Overview of the Concurrent Engineering-oriented Design Database Representation Model (CE-DDRM)

2.1. Requirements for Developing the CE-DDRM
Through extensive studies on commercial computer-based design tools, research prototype systems, and industrial design activities, the major requirements for developing the CE-DDRM are summarized as follows.

(1) The model should describe both geometric information and non-geometric information.
(2) The model should incorporate the results achieved in the research on design theory and methodology to develop tools for supporting design engineers.
(3) The model should be developed based upon the advances of recent computing technologies.

2.2. System Architecture
Architecture of the concurrent engineering-oriented design database modeling system is shown in Fig. 1. In this system, the design database is described at two different levels, class level and instance level, representing generic design libraries and special design cases, respectively. Various product development systems, such as structure analysis system, motion analysis system, etc., can be used to access the design databases.

Knowledge in the CE-DDRM is described at three different levels: meta-class level, class level, and
instance level, as shown in Fig. 2. The meta-class knowledge describes the concepts used in the CE-DDRM. These concepts are classified into five major categories: entities, properties, relationships, tasks, and specifications. The behaviors of these concepts are also described as the meta-class knowledge. Meta-class knowledge cannot be modified by the design engineers. The class design knowledge describes the generic design libraries, such as gear, shaft, gear-pair, etc. The design engineers can define their own class knowledge, usually using the existing classes as the super-classes. The instance design knowledge describes the special design cases. The instance knowledge is created using the libraries of the class knowledge 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.

3. Knowledge Representation in CE-DDRM

3.1. Entities

Entities are physical units for modeling designs. Entities are classified into artifacts and features.

An artifact represents a distinct entity separable from others and created in a design. Typical artifacts include components and assemblies. A component is a primitive for modeling a design. An assembly is composed of components and sub-assemblies for modeling a design. Fig. 4 shows an example of class artifact and instance artifact modeling.
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 as a design feature for ventilation function and a manufacturing feature to be machined by a drilling task. The features include design features, manufacturing features, assembly features, maintenance features, recycle features, etc. A design feature is a group of descriptions in an artifact for achieving a certain design function. A manufacturing feature represents the partial artifact geometry used for identifying production processes.

3.2. Properties

Properties are the inherited characteristics belonging to artifacts. Typical properties in the CE-DDRM include data, functions, behaviors, forms, and flows, as illustrated in Fig. 5.

3.2.1. Data

A piece of data represents the value of a certain aspect of an artifact. Typical data include attributes, dimensions, and tolerances. An attribute is described by a pair of attribute name and attribute value. A dimension describes the nominal value of a linear or an angular parameter used for modeling artifact geometry. A tolerance describes allowance of the geometric error produced in the manufacturing process.

3.2.2. Functions

Functions are the teleological descriptions of artifacts. For instance, the major function of a gear-pair is to transform a rotational motion to another rotational motion with a constant rotational speed ratio. Functions are usually used for modeling design intentions, design requirements, and so on.

During a design process, the design requirements are usually first described as design functions and their relationships. These functions are achieved through the process of identifying the forms of the artifacts whose behaviors satisfy these functions and their relationships.

Functions are usually linked by flows. The functions and their input flows and output flows for modeling the mechanism shown in Fig. 6 (a) are given...
in Fig. 6 (b). Flows are classified into material flows, energy flows, and information flows. The flows given in Fig. 6 (b) are energy 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.

3.2.3. Behaviors

Behaviors of artifacts are the ways to implement the functions of the artifacts. Behaviors are derived from the forms (physical characteristics) 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, usually with respect to the change of the time parameter. Behaviors are classified into different categories in the CE-DDRM, including continuous-time behaviors, discrete-time behaviors, and state-transition behaviors.

A continuous-time behavior describes the changes of the artifact descriptions when the time parameter value changes continuously. A parameter, such as an attribute or a dimension, can be defined as a numerical function of the continuous time parameter, in the form of

\[ f(X, t) \]

where, \( X \) is a vector of design parameters (e.g., attributes, dimensions, etc.), and \( t \) is the time parameter.

A discrete-time behavior describes the changes of the artifact descriptions when the time parameter value can only be selected from a set of discrete values (e.g., 1, 2, 3, ...). A parameter, such as an attribute or a dimension, can be defined as a numerical function of the discrete time parameter, in the form of

\[ f(X_{t-1}, t) \]

where, \( X_{t-1} \) is a vector of design parameters (e.g., attributes, dimensions, etc.) at the previous time index, and \( t \) is the current time index.

A state-transition behavior describes a number of states and conditions to change among these states. A state is represented by a node in a graph and a condition between two nodes is represented by an arc. Each state is described by a collection of descriptions in an artifact. Among all the states, only one is active. The descriptions in the active state are used to model the design. When an output arc of the active state is satisfied, the active state is then changed to this adjacent state. Fig. 7 shows a three-level-transition behavior to model a 2-power-level electrical fan. When the switch button of the electrical fan is pressed down, the state is then changed from the off-state to the first-level state, and then to the second-level state, and then to the off-state.

3.2.4. Forms

Forms represent the physical descriptions of the artifacts. The geometry descriptions and material
3.2.5. Flows

Flows are used to model the inputs and outputs of functions. Flows link the functions into a function network. Flows are classified into three categories: material flows, energy flows, and information flows. Typical flows are summarized in Table 1.

<table>
<thead>
<tr>
<th>Flow Type</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Energy Flow</td>
<td>Mechanical Flow (e.g., Force, Position), Thermal Flow (e.g., Entropy), etc.</td>
</tr>
<tr>
<td>Information Flow</td>
<td>Analog Flow, Digital Flow, etc.</td>
</tr>
</tbody>
</table>

3.3. Relationships

Relationships associate descriptions of artifacts. In the CE-DDRM, relationships are classified into declarative relationships, dependency relationships, and constraints.

3.3.1. Declarative Relationships

Declarative relationships define the relationships among entities, functions, behaviors, and flows. The declarative relationships are described by predicates. A predicate takes the form of 

\[(\text{relationship}, p_1, p_2, \ldots, p_n)\]

A predicate without variable terms is called a fact. Facts can be used in rule-based reasoning.

3.3.2. Dependency Relationships

A dependency relationship defines the relationship between one output and a number of inputs. The output is dependent on the inputs. When the input descriptions are changed, the output description should be updated.

Figure 8. A mechanism and VRML coordinate systems of its artifacts
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. Fig. 9 shows a gear-pair with three attribute relationships. The attribute relationships form a network called attribute dependency relationship network. An attribute dependency relationship network is composed of two types of nodes: attribute nodes and relationship nodes. Each attribute node is associated with an attribute value. Each relationship node is linked with one or several input attribute nodes and one output attribute node. When an attribute value is changed, the relationships that use this attribute as the input node are then activated to update this change to the output attribute nodes. This attribute propagation process is carried out continuously until no attribute value change is required.

Logic relationships associate a number of artifact descriptions, usually the facts (i.e., the predicates without variables), created using IF-THEN rules. 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 \((p, x_1, x_2, ..., x_n)\). Logic relationships are described by AND-OR trees. When a fact is created using a rule that matches with a number of facts, the derived fact is dependent on the existing facts, and the derived fact has an AND relationship with these existing facts, as shown in Fig. 10 (a). When a fact can be created using a number of rules each having one matched fact, the derived fact is dependent on these existing facts, and the derived fact has an OR relationship with these existing facts, as shown in Fig. 10 (b). When a fact can be created using a number of rules each having several matched facts, the derived fact is dependent on these existing facts, and the derived fact has an AND-OR tree relationship with these existing facts, as shown in Fig. 10 (c).

### Figure 10. Modeling of logic relationships

#### 3.3.3. Constraints

Constraints are relationships that should always be satisfied during the design process. In this research, constraints are classified into three major categories: arithmetic constraints, temporal constraints, and spatial constraints.

An arithmetic constraint defines the relationship among attributes by expressions linked by comparison operators, including \(<, \leq, >, \geq, =, !=\), and logic operators, including `AND` (\&\&), `OR` (\|\|), `EQUAL` (==), `NOT` (!), and `XOR` (!=), such as

\[
\begin{align*}
    r[gp1] &= n[g1] / n[g2] \\
    r[gp1] &> 2.0 \\
    (x[e1] > 2) &== true \\
    ((x[e1] > 2) &\& (x[e1] < 12)) &== true
\end{align*}
\]

Temporal constraints are those related with time parameters. In the CE-DDRM, the temporal constraints are used to define the sequential relationships among tasks. Table 2 summarizes some of the fundamental temporal logic operators used in the CE-DDRM.
Spatial constraints are those related to geometric elements, such as surfaces, edges, vertices, and so on. In an assembly, the spatial constraints are used to connect a number of components and sub-assemblies together. In the CE-DDRM, first these geometric elements should be defined or extracted. The spatial constraints among geometric elements are defined by built-in predicates. The spatial constraints are classified into basic spatial constraints and derived spatial constraints. Some of the basic spatial constraints are summarized in Table 3. Many CAD systems provide other types of spatial constraints that are derived from the basic spatial constraints for improving the constraint modeling efficiency. Table 4 gives examples of the derived spatial constraints.

### Table 3. Modeling of basic spatial constraints

<table>
<thead>
<tr>
<th>Spatial Constraint</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point-Point</td>
<td>(Point-Point, vertex-1[desk-1], vertex-4[desk-cover-1])</td>
</tr>
<tr>
<td>Point-Line</td>
<td>(Point-Line, center-point[ball-2], centerline[hole-1])</td>
</tr>
<tr>
<td>Point-Surface</td>
<td>(Point-Surface, center[disk-bottom-surface], top-surface[desk-1])</td>
</tr>
<tr>
<td>Line-Line</td>
<td>(Line-Line, centerline[bolt-1], centerline[nut-1])</td>
</tr>
<tr>
<td>Line-Surface</td>
<td>(Line-Surface, edge-1[hutch-2], top-surface[desk-1])</td>
</tr>
<tr>
<td>Surface-Surface</td>
<td>(Surface-Surface, top-surface[desk-leg-1], top-surface[desk-leg-2])</td>
</tr>
</tbody>
</table>

### Table 4. Modeling of derived spatial constraints

<table>
<thead>
<tr>
<th>Spatial Constraint</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parallel</td>
<td>(Parallel, centerline[hole-1], centerline[hole-2])</td>
</tr>
<tr>
<td></td>
<td>(Parallel, centerline[hole-1], surface-2[part-3])</td>
</tr>
<tr>
<td>Perpendicular</td>
<td>(Perpendicular, surface-1[part-1], surface-2[part-2])</td>
</tr>
<tr>
<td>Offset</td>
<td>(Offset, surface-1[part-3], surface-2[part-2], 20)</td>
</tr>
<tr>
<td>Alignment</td>
<td>(Alignment, surface-1[part-1], surface-1[part-2])</td>
</tr>
<tr>
<td>Contact</td>
<td>(Contact, surface-1[part-1], surface-1[part-3])</td>
</tr>
<tr>
<td>Coaxial</td>
<td>(Coaxial, hole-1, shaft-1)</td>
</tr>
</tbody>
</table>

### 3.4. Tasks

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. In the CE-DDRM, these tasks are defined in manufacturing features, assembly features, maintenance features, and recycle features, respectively. The sequential relationships among these tasks are defined by temporal constraints.

The task requirements preserved in artifacts are used for process planning/scheduling, usually considering manufacturing resources. The process planning/scheduling results are also used as down-stream product life-cycle evaluation measures for identifying the optimal design candidate.

The manufacturing tasks to produce each artifact are linked as a graph, as shown in Fig. 11, representing the sequential constraints among these tasks. A task in a higher level artifact is conducted only when all the tasks in the lower level artifacts have been completed. Each task specifies the resource requirements and time period to carry out this task. For instance, the shaft component shown in Fig. 11 can be produced by 3
tasks including turning, grooving, and milling. The turning task, C1, requires a lathe and a machinist for 10 minutes.

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

Design is the process to identify the artifacts that satisfy the design specification. The design candidates can be identified in two different ways: top-down design to map from the design requirements in the specification to the design artifacts, and bottom-up design to select the design candidates and evaluate them to see whether the design requirements in the specification can be satisfied.

4. System Implementation and a Case Study Example
The concurrent engineering-oriented design database modeling system has been developed as a Web-based system. A snapshot of the implemented system is given in Fig. 12. In this system, different product development databases are modeled at different locations, represented as Internet nodes. End users access these nodes through Web-browsers.

A case study example to design a power fluid pump for a heating system is given in this section to illustrate the effectiveness of the introduced concurrent engineering-oriented design database modeling approach. The objective of this design is to create a mechanism to transform the rotation motion of a motor to the translation motion of a piston. Since the rotational speed of the motor is usually high, this high-speed rotation motion needs to be transformed into a low-speed rotation motion first.

The design requirements are defined as the design specification using an AND/OR graph shown in Fig. 13. The main function, RotationToTranslation1, is to transform the energy-flow RotationMotion1 created by the artifact Motor1 into the energy-flow OscillatoryTranslationMotion1 to drive the artifact Piston1. The main function is decomposed into two sub-functions: the function RotationToRotation1 to transform the high-speed rotation motion RotationMotion1 to the low-speed rotation motion RotationMotion2, and the function RotationToTranslation2 to transform the RotationMotion2 to the OscillatoryTranslationMotion1.

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 (m²), stroke of the piston (m), and number of cycles per unit time (cycles/min), should be between 0.32 m³/min and 1.85 m³/min.

The top-down design approach is employed in this work to create the design candidates. A gear-box assembly that is composed of two gear-pairs, called Gearbox1, is created to satisfy the RotationToRotation1 function. A behavior, called GearboxBehavior1, is also created to model the behaviors of this gear-box, such as the relations among rotational speeds and the numbers of teeth of the gears in the gear-box.

For the RotationToTranslation2 function, two design candidates, a cam-follower mechanism called CamFollower1 and a crank-slider mechanism called CrankSlider1, are created separately to satisfy this function. The created two design candidates are shown in Fig. 14.

These two design candidates are evaluated through behavior-based simulation. The evaluation results for these two candidates are shown in Table 5. 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 5. The design candidate with the crank-slider mechanism is selected due to its low production cost.
5. Conclusions

The characteristics of the CE-DDRM are summarized as follows.
(1) This model can describe both the geometric information and the non-geometric information.
(2) The model incorporates many of the results achieved in the design theory and methodology research area. The model integrates the functions, behaviors, and forms of artifacts into the same computing environment.
(3) The model employs object-oriented modeling approach, distributed database modeling techniques, and Web technologies for designing the system architecture.

Acknowledgements

The authors would like to thank Dr. R. D. Sriram at the NIST for his encouragement, support, and discussion to the development of this concurrent engineering-oriented design database representation model.

References