G. Chen
Y.-S. Ma[1]
e-mail: mysma@ntu.edu.sg
G. Thimm
S.-H. Tang

School of Mechanical & Aerospace Engineering,
Nanyang Technological University,
50 Nanyang Avenue,
Singapore 639798, Singapore

Associations in a Unified Feature Modeling Scheme

Features allow one to associate human knowledge and product geometry. The authors proposed, in earlier publications, a unified feature modeling scheme with the aim to maintain the integrity and consistency of a product model. Different application feature models within and across different product life-cycle stages are integrated, and especially, nongeometric relations (besides geometric ones) are handled. In this paper, as an improvement to the previous work, two types of associations are introduced: sharing and dependency. In the context of conceptual and detail design stages, these associations are described and the implementation is discussed in detail. [DOI: 10.1115/1.2194910]

Keywords: unified feature modeling, association mechanism, information consistency, product life cycle

1 Introduction

Historically, computer-aided tools, such as computer-aided design (CAD), computer-aided process planning (CAPP), and computer-aided manufacturing (CAM) systems were developed to support individual product development life-cycle stages; they are design, process planning, and manufacturing, respectively. Since these stages are interrelated, concurrent engineering which requires system integration among these stages, is commonly applied. However, because of complex product models, high pressure from time-to-market requirements, and distributed production environments, the efficiency of concurrent engineering approach has not been fully achieved.

The major obstacles for true system integration are incompatible data structures and the lack of data associations across these stages. Features are considered as a means to overcome these obstacles by clustering and transferring data to a higher abstraction level. Publications reviewed in [1] propose to share data from the design stage to other stages by using design-by-feature or feature recognition approaches. However, due to the inherent rigidity of traditional feature definitions in design-by-feature approaches and the nongeneric nature of feature recognition algorithms, current commercial tools mainly use features to provide high-level geometric representations to facilitate certain product development tasks, such as the design of parametrized product geometry. The integration of the design stage with other stages is still limited only to geometry. Recent research showed that features can represent involved product information rather than only pure geometry by grouping data types together [2–6].

Feature-based integration of different product lifecycle stages attracted a lot of attention [7–12]. However, the respective results have limited impact on industrial practices due to the following reasons:

1. Most works focused only on the integration of the detail design with the process planning stage. However, a detail design represents only a single solution from the view of the required product functionality. It is common that the conceptual design is changed or enhanced along with the progress of different life-cycle stages. Consequently, the detail design must be modified iteratively. The detail design has also to be evolved, continuously, in order to address different specific design aspects. This industrial requirement stands in a stark contrast to the commercial systems, where models of life-cycle stages are disconnected. Therefore, the exploration of alternative design solutions is costly as the synchronization of the models relies on human intervention and a repetition of the same product development tasks. This has as a consequence that only one design is explored and that downstream application optimizations are confined in individual stages.
2. Most multiple-view feature modeling systems use only the product geometry to connect lifecycle stages. This is due to the fact that features mainly define geometry. However, in real applications, non-geometric associative relations broadly exist and should be managed systematically. For example, material specifications are derived from the product's functional requirements. At the same time, they have significant influences on the machining processes.
3. Relations among different stages, whether they are geometric or nongeometric, are usually not explicitly modelled and maintained during the product development processes. This makes later product modifications difficult and commonly impairs model integrity and consistency.

The proposed unified feature-based modeling scheme solves these problems by:

1. including the conceptual design, together with the functional requirements, into the multiple-view feature modeling process
2. supporting nongeometric associations across different stages
3. Assisting in the creation and maintenance of the following two aspects of associations:

(a) associations between the canonical form of a feature and the boundary representation
(b) associations between application features and the facts in the higher-level knowledge model

The first aspect is necessary to maintain the geometric consistencies among feature models, whereas the second is for knowledge-based reasoning processes, such as design intent validation. The framework of the unified feature modeling scheme proposed in [13] does just this, and this paper reports the continued effort. Two types of associations used in the unified feature

[1]Corresponding author.

Contributed by the Engineering Informatics (EIX) Committee of ASME for publication in the JOURNAL OF COMPUTING AND INFORMATION SCIENCE IN ENGINEERING. Manuscript received January 5, 2005; revised manuscript received February 14, 2006. Guest Editors: R. Sriram, S. Szykman, D. Durham.

modeling scheme, sharing and dependency, are introduced with the implementation described covering the integration of the conceptual and the detail design stages. This paper is organized as follows. Section 2 reviews the related research works. Section 3 describes the unified feature modeling scheme very briefly. In Sec. 4 and 5, the information flow between the conceptual and the detail design stages is analyzed in order to identify the required associations. The implementation of these association types is discussed in Sec. 6. Criteria for evaluating data validity, integrity, and consistency are introduced in Sec. 7. A case study is given in Sec. 8. Finally, discussions and conclusions are given.

2 Related Research

Some of the earlier feature-related research focused on geometric relations within a feature or among features [14,15]. To maintain feature validity, many researchers tried to embed these relations into feature definitions as constraints [3,16,17]. In addition, nongeometric relations between features were also recognized to be important for the validity of a feature model [3,18]. Multiple-view feature modeling was proposed to represent different product development stages using different but coherent application feature models. Connections between these models are established on the basis of common faces [10,19] or volumetric overlaps [9,11]. However, such connections are applied only to the product geometry. Hoffman and Joan-Arinyo [12] proposed a product master model to link different views. Each view deposits a part of their private models into a public repository representing the master model and associates their private data to the common and shared data. The master model is responsible for propagating modifications, while specific applications are responsible for modifying their data to achieve the interapplication consistency. No implementation details are mentioned. Ma et al. [5] revealed the importance of using and maintaining the associative nature of features during the design processes. An associative feature concept was proposed to model the associated geometric entities [4,5] via a generic feature class. It was highlighted that features should be self-contained and flexible. They should have built-in associative links, different representation forms, and self-validation methods. The consistency of relations has to be managed while the geometric representations pertaining to various life-cycle stages are evolved. One conclusion from the above-mentioned research is that feature validity can be represented by constraints on feature properties (intrafeature relations), relations between different features (interfeature relations) as well as relations between features and other entities (functions).

Recently, Roy and Bharadwaj explored relations among product function, part behavior, geometry, and specifications [20]. They claim that both spatial and energy transfer relations between part faces are important to describe product functions and to determine product specifications, such as surface finish. They emphasized the importance of including the conceptual design stage into an integrated computer-aided product development environment.

Bronsvoort and Noort [8] proposed a framework which includes conceptual design into a multiple-view feature modeling system. They discussed some nongeometric relations between the conceptual design view and other views [8]. However, they used only geometric relations to connect different views. Furthermore, knowledge-based reasoning, which is especially indispensable in the conceptual design and the manufacturing planning stages, was not mentioned.

Xue et al. [21] and Xue and Yang [22] used various product information entities, such as functions, geometries, and machining processes, in a design optimization process. They suggested the use of features to model different product life-cycle aspects but did not discuss the semantics of features and consistency control mechanisms.

Other researchers also contributed to the integration of conceptual design with other life-cycle stages. Welch and Dixon used behavior graphs to relate product function and the product embodiment configurations [23]. Umeda et al. used physical features to relate product functions and behaviors [24]. A physical feature is a combination of physical components and phenomenon. Qian and Gero modeled the mapping relations among the product functions, behaviors, and physical structures [25]. Ranta et al. [26] proposed a generic ontology map across stages, common-function-based associations among geometric entities, and a map of abstract functional requirements to geometric constraints. Brunetti and Golob described the data mapping from the conceptual design to the detail design stages [27]. Brunetti and Grimm used feature and shape ontologies to support the integration of a knowledge-based system with CAx systems [28]. The basic ontologies for six layers of a shape model (i.e., geometry, topology, parametrics, form features, application features, and assembly features) were described. Pratt and Srinivasan proposed a three-level architecture (generic, canonical, and embedded levels) to represent form features in an application-independent way [29]. A list of basic informational requirements for the application-independent form-feature representation was identified. The unified feature modeling scheme [13] extends the associative feature concept [4] by including nongeometric feature associations.

In general, features can be used not only as building blocks to develop the product information model within a single stage, but also as an associative mechanism to link:

1. high-level, knowledge models with low-level, geometric models

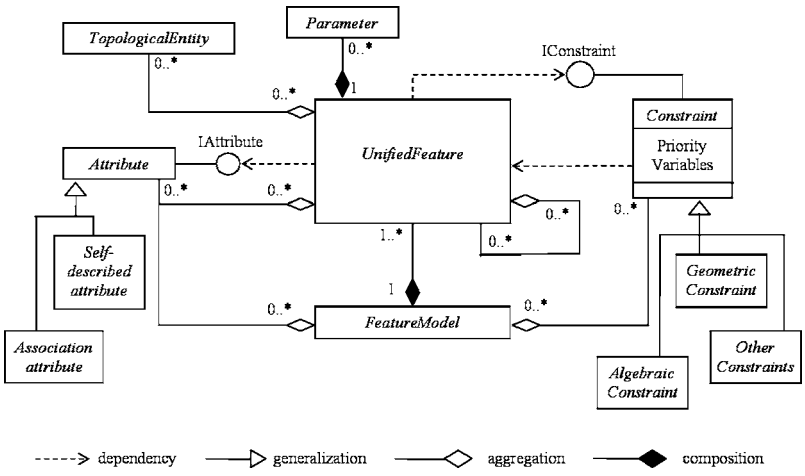

Fig. 1 Unified feature definition

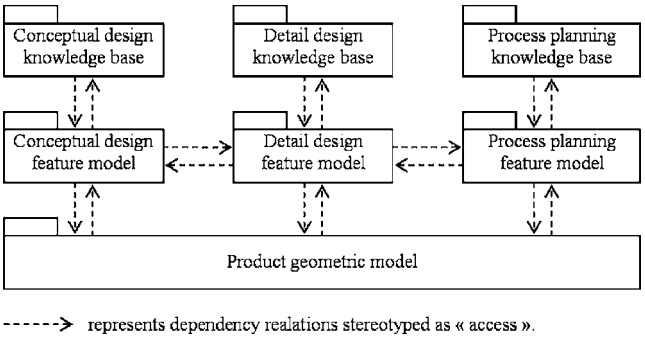

Fig. 2 Relations in the unified feature modeling scheme

2. application models of different stages

To achieve these two extensions, inherent relations in the feature-based modeling process must be identified and represented as associations. These associations are then used along with the product development processes. The main idea is to use geometric and nongeometric associations to propagate modifications as well as to maintain data validity, integrity, and consistency. These issues are discussed in the Secs. 3–6.

Note that the unified feature modeling approach is quite distinct from the product life-cycle modeling scheme using the unified modelling language (UML) [30,31], as proposed by some of the authors [32]. The main difference to the here-proposed approach is that the UML-based approach is a model-oriented, high-level approach, which, by design, ignores detail product information. However, unlike the unified feature modeling scheme, it is very suited for modeling the entire product life-cycle and business processes.

3 Unified Feature Modeling

The unified feature modeling scheme [13] was proposed as a new solution for information sharing and integration in a multiple-view feature modeling system. This scheme is a part of an encompassing research project, which tries to develop a Web-enabled, feature-based, multiapplication-oriented product development framework. This research project targets on providing an information layer above the geometric kernel to integrate different computer-aided application functions. The unified feature modeling scheme is characterized by definitions of unified features and several association mechanisms.

The unified feature definition provides a generic feature format and granularity for different product development stages. The unified feature definition differentiates itself from other feature definitions by enabling associations among feature entities as well as among three information layers, features, knowledge bases, and the geometry. In the object-oriented prototype implementation, features are implemented as objects. Common characteristics and behaviors of conceptual design features, detail design features, and process planning features are extracted as a unified feature's member variables and methods. Features of a specific stage are derived from the unified feature class and hence inherit its variables and methods. The generic feature definition is given using a UML diagram [30] (see Fig. 1).

For the reader's convenience, the UML symbols used in the figure are explained here. Rectangles represent classes, such as the *UnifiedFeature* class. Dashed and directed lines represent dependency relations. The lines are directed from the depending class to the class on which it depends. Solid and directed lines with triangular, open arrowheads represent generalization relationships, pointing to the more general class that defines basic properties. Solid and directed lines with open diamonds represent aggregation relationships, pointing from the parts to the whole, aggregated object. Composition (indicated by a filled diamond) is a variation of a simple aggregation relationship. It describes strong ownership and coincident lifetime between the parts and the whole. The ranges indicated at the start and the arrow points of an aggregation arrow show how many parts can or must be in a whole [30]. For example, a unified feature may include none or many other unified features. A circle attached to a class represents an interface (such as the *IAttribute*) realized by (undirected lines) the class. Other classes can use this interface, e.g., the *UnifiedFeature* class uses the *IAttribute* interface.

Four kinds of feature properties are specified for the unified feature class:

1. *Attributes* represent those feature properties that can be represented as a pair of *name* and *value(s)*. The *value(s)* can be of type integer, real, string, or their expressions. They usually do not directly describe a feature's geometry. Attributes are further divided into self-described and associated ones. *Self-described attributes* represent those properties of a feature that are independent of geometry, such as color, material, etc. *Associated attributes*, in the form of identifiers, refer to the entities associated to owning features, such as geometric entities, rules, facts, constraints, other features, etc. For example, functions and behaviors in the conceptual design stage or machines and operations in the process planning stage are represented as associated attributes.
2. *Parameters* represent feature properties that control a feature's geometry, e.g., the shape, size, position, and orientation.
3. *Constraints* specify intra- or interfeature associations.

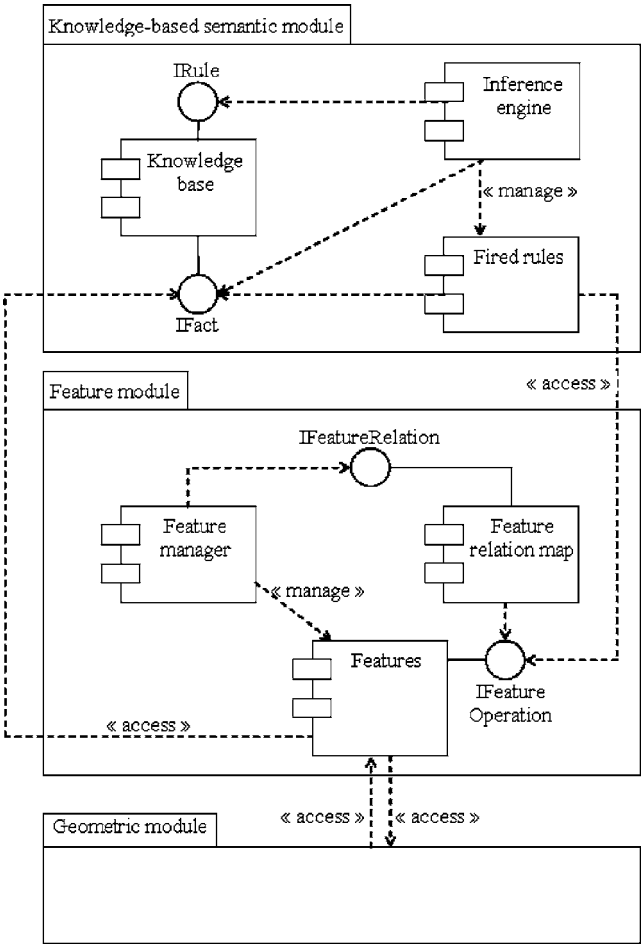

Fig. 3 Information entities and vertical associations in the unified feature modeling scheme

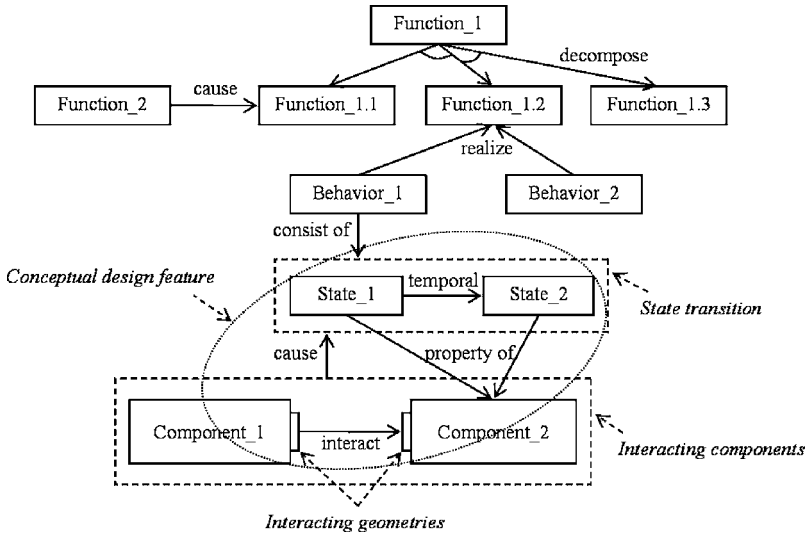

Fig. 4 Conceptual design feature

Constraints may be algebraic, geometric, or other types specified on its variables. Constraints have priorities.

4. *Geometric references* are a collection of pointers to topological entities in the product's geometric model. Since features are used to describe specific relations among geometric and nongeometric entities, a feature's geometry is not necessarily volumetric, connected, or twomanifold. Geometric references are not included in the higher-level knowledge model directly. They serve as the identifiers to the product's geometric model.

Associations in the unified feature modeling scheme can be roughly classified into horizontal and vertical (see Fig. 2). Horizontal ones represent feature associations via common topological entities. The associated features may belong to the same or different life-cycle stages. Vertical associations serve as the intermediate interfaces upward to the knowledge model and downward to the geometric model. A more detailed illustration is given in Fig. 3. The rectangles with tabs represent components or modules, which are modular and replaceable parts of a system. A component or a module conforms to and provides the realization of a set of interfaces [30]. The contents of Fig. 3 are further explained in Sec. 4.

4 Information Entities

A product information model consists of submodels that correspond to individual development stages. Each of them can be constructed with three layers: a knowledge-based semantic model, a geometric model, and a feature model (see Fig. 3):

1. The knowledge-based semantic model comprises abstract engineering knowledge, which is represented in the form of rules in this work, existing facts, and reasoning algorithms [33]. The inference engine accesses the knowledge base to find rules that are ready to fire. The fired rules add new facts to the knowledge base or invoke feature methods.
2. The geometric model handles geometric and topological entities of a product.
3. The geometry modeling systems are inherently restricted in representing nongeometric information. The feature model, as the intermediate layer between the above two layers, associates nongeometric information, in the form of attributes, parameters, constraints, etc., with the geometric model. This feature model also connects the declarative knowledge-based semantics and the procedural geometry modeling functions. It should be noted here that via features, application-specific nongeometric entities, such as functions and behaviors in the conceptual design stage or machines and operations in the process planning stage, are involved in the product information model. These entities support the knowledge-based reasoning process and execute the decisions through feature-based processes, such as feature generation, editing, object fusion, sharing, messaging, and deletion. Usually, they are not directly linked to the product geometry.

Exemplarily, the conceptual design and the detail design stages are analyzed hereafter to identify these entities. Major tasks in the conceptual design stage include function decomposition, functions to physical components mapping, and the determination of critical dimensions and specifications [34]. These tasks are often carried out with the assistance of a conceptual design knowledge base. As shown in Fig. 4, the functions of a mechanical product are generally realized through behaviors of single components or behavioral relations between components. If a state is defined as a set of properties (e.g., position, orientation) of the owning component, then the behavior of a component can be represented by these states and state transitions. Usually, the state of a component is changed (hence, displaying a specific behavior) due to the interactions with other components.

Relations between primitive design functions that could not be further decomposed, required behaviors, and respective states of interacting components, determine critical characteristics of the product, and hence, they are modeled as *conceptual design features*. Conceptual design features can be understood as guiding constraints in the detail design stage.

The conceptual design features are derived from the unified feature class (see Fig. 1):

1. Self-described attributes record data related to the behavior represented by the feature, such as the initial state, the end state, and other characteristics. The associated attributes could refer to related functions, behaviors, rules, or other conceptual design features.
2. Parameters are characteristics of interacting geometries, such as their shapes, sizes, positions, and orientations.
3. Constraints are specified on attributes, parameters, or between conceptual design features.
4. As the product's physical structure is not detailed in this stage, conceptual design features have to allow the representation of incomplete, inaccurate product specifica-

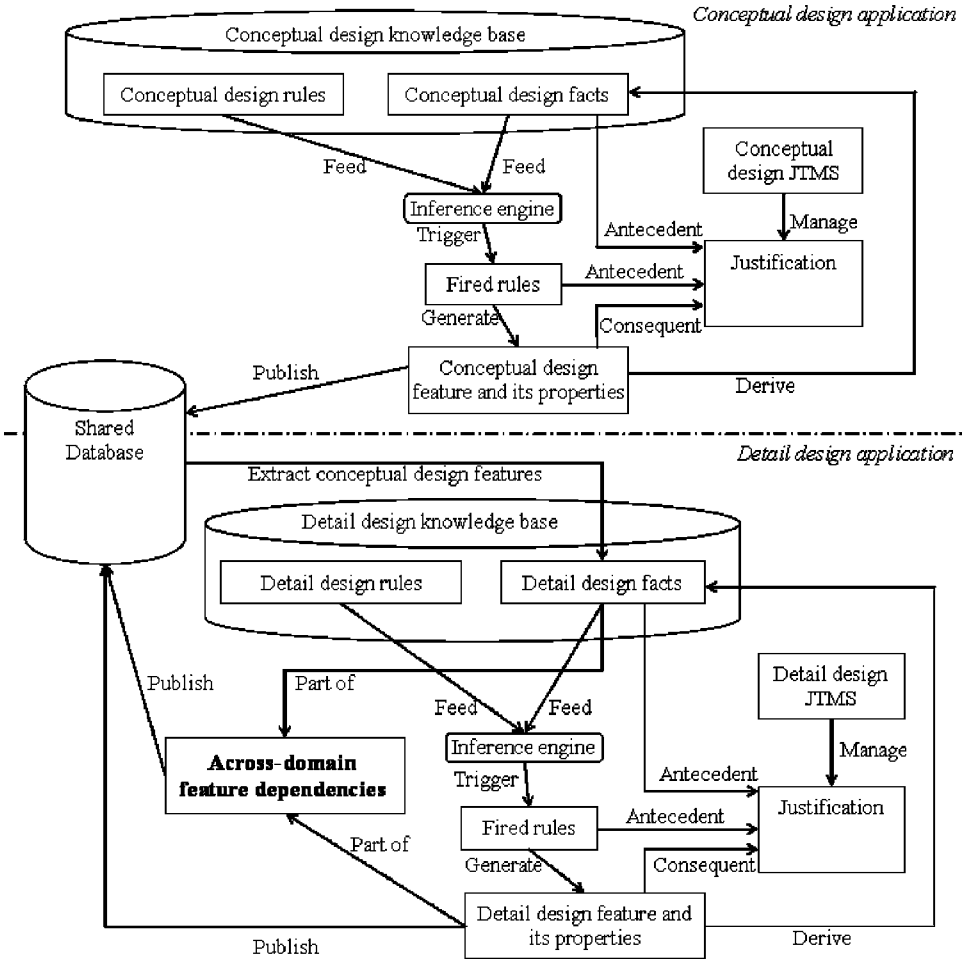

Fig. 5 Feature dependencies between the conceptual design and detail design

tions, such as the value range used in [35]. A conceptual feature could include parametrized but nonmanifold geometries. In such cases, default values, interpolations, or even a symbolic object may be used. In this way, the conceptual design feature model provides an abstract representation of the product.

It is possible to define a generic and abstract conceptual design feature class that is suitable for all mechanical products. For each specific type of product, it is necessary to develop a set of standard conceptual design subfeatures that reflect commonly used specific functions and behaviors of components at different levels of subassemblies, level by level, until the whole product has been covered.

In the detail design stage, the geometric shape, all dimensions, and specifications are determined. The detail design feature model is developed by using the conceptual design features as guiding constraints. The consistency of detail design features with the primitive design functions is ensured through their relations with the corresponding conceptual design features. The attributes, parameters, and constraints specified in the conceptual design feature model are usually mapped into the detail design stage. Geometries outlined by conceptual design features are materialized or refined by detail design features. Detail design features can also be created or modified due to DFX considerations, such as design for machining or design for assembly.

5 Associations

The validity of an information model is evaluated through associations among its composing entities. For example, a feature's validity depends on whether constraints specified on it are satisfied. To determine whether a constraint is satisfied or not, the entities and variables to which the constraint refers have to be evaluated and validated. Associations are also indispensable for propagating modifications. In this section, two new association types, namely, the sharing and dependency, are introduced. Their implementation is described in Sec. 5.1.

5.1 Sharing. Sharing association represents the relationship that different features refer to the same geometric or topological entities in the geometric model. If the shared geometric or topological entity is modified, then all the depending features must be notified and validated.

For example, explicitly specified product geometry in the conceptual design stage are shared via the conceptual design features and the corresponding detail design features. Sharing associations are also encountered frequently within a single stage, such as two detail design features sharing the same face either fully or partially. *Sharing* associations can be classified as a type for *common-geometry-based* feature associations.

More specifically, sharing associations are realized through assigning and associating feature's geometric references with the corresponding geometric or topological pointers. The feature model carries only the pointers to the geometric or topological entities, not the actual geometry. Hence, this association mechanism enables different features to be associated with the same geometric or topological entity.

5.2 Dependency. Because of the inherent sequential nature of the product development processes, new data entities are gener-

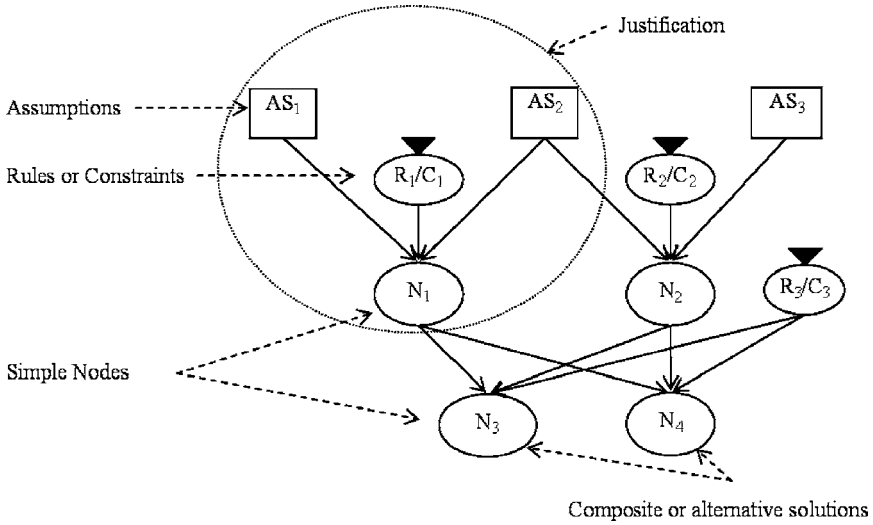

Fig. 6 JTMS-based dependency network

ated by procedures from other data entities belonging to the same application or different applications. For example, a detail design feature is generated to materialize a conceptual design feature. This kind of relation is represented with dependency associations. Dependency associations are used to describe geometric and nongeometric relations. Figure 5 shows the partial dependency associations in both conceptual and detail design stages.

5.2.1 Dependencies Between the Knowledge Base and the Feature Model. As shown in Fig. 5, if a feature (or its properties) is created by a knowledge-based reasoning process, then it is, by default, associated with the respective rules and facts using a dependency association [36]. Later feature modifications have to be validated against the rules that create the feature. It should be noted here that dependency associations also exist within the knowledge base: consequent facts of a fired rule depend on its antecedent facts.

The knowledge-based system establishes the first kind of nongeometric associations between features: *common-rule-based* feature association. For example, two conceptual design features can be used together as a pair to realize a specified function by firing a function-to-feature mapping rule. Another example is the order of two process planning features necessary to fulfill the machining requirement, which is also constrained with a rule.

In certain cases, the expert system may not find valid rules to fire. In these cases, the designer may follow the traditional design routine and create features and specify constraints without rule support. Designers may also define new rules. These rules are saved in the knowledge base for future use. In general, an expert system should be regarded as an assistant to the designers.

5.2.2 Single-Stage Intra- or Interfeature Dependencies. Intrafeature dependencies exist between constraints and the constrained attributes or parameters. For example, values of constrained feature properties depend on the constraints as well as values of other relevant properties. In addition, a value of a feature parameter may depend on values of properties of geometric entities used by the feature. The properties of features may also have dependency associations through constraint specifications, which are interfeature dependencies. Again, referring to Fig. 5, both intra- and interdependencies are recorded and managed by a justification-based truth maintenance system (JTMS) module in each life-cycle stage.

It should be noted that a geometric entity class includes its properties as well as its manipulation methods, which are based on the modeling functions of the geometric modeler. Dependency associations control the values of geometric properties, whereas a modification of an actual geometric object can only be carried out via geometric modeler methods.

5.2.3 Interfeature Dependencies Across Multiple Stages. Dependencies across different stages are maintained in a shared database as shown in Fig. 5. Usually, in the conceptual design stage, the geometry of a feature is not fully defined. The resulted entities could be, for instance, only surface shapes, abstract mechanism concepts, or parametrized volumes without assigning detailed properties. An abstract conceptual design feature has its concrete counterparts in the detail design feature model. Because a conceptual design feature represents a primitive design function that is usually realized through the interactions between a few components, it is very likely that an individual conceptual design feature is transformed into several features belonging to different components in the detail design stage. On the other hand, one detail design feature may also participate in the realizations of several conceptual design features. Such *feature object dependency* associations are the second kind of nongeometric associations between features.

Feature attributes, parameters, or constraints specified in the conceptual design feature model are transformed into attributes, parameters, or constraints for corresponding detail design features. For example, a parameter of a conceptual design feature may be transformed into a constraint between two detail design features of different components. A conceptual design constraint could be related to several constraints in the detail design feature model. Such *feature property dependency* associations are the third kind of nongeometric associations across features of different stages.

6 Implementation of Sharing and Dependency Associations

To prove the feasibility, a prototype system was developed using MICROSOFT VISUAL C++ 6.0, ACIS 7.0, and MYSQL 5.0.0. This section describes the implementation of the sharing and dependency associations, in detail.

6.1 Implementing Dependency Associations. As shown in Fig. 5, a JTMS is used to implement dependency associations in different life-cycle stages. Figure 6 shows the concept of JTMS network. According to the definitions in [37], two major types of entities are defined in the implemented JTMS node and justification. Nodes correspond to the entities in the inference engine:

rules, facts, functions, machines, constraints, features, feature properties, and so on. Three kinds of JTMS nodes are shown: premises (circles with a black triangle), assumptions (squares), and simple nodes (circles). Justifications (dashed circles) represent primitive reasoning processes, i.e., if the antecedent nodes are all believed, then also the consequent nodes.

The current prototype system is developed using an ACIS geometric kernel. For convenience, the dependency associations are defined as ACIS entities using an object-oriented approach because ACIS supports user-defined classes with saving and restoring mechanisms. The system automatically extracts data from objects of user-defined and its built-in classes.

The JTMS cooperates with the inference engine (a rule-based expert system and a constraint solver) to realize the problem-solving capability. Whenever a dependency association is generated, the corresponding JTMS nodes and justifications are automatically inserted into the JTMS dependency network via the association class generation method. Then each simple node records its direct related justifications, including antecedent and consequent ones (for later change propagation) and its current belief status.

Whenever a modification to the JTMS dependency network occurs, such as adding or retracting assumptions, the JTMS dependency network is searched for affected nodes as well as related justifications and both are recursively updated. The result is a new status of each affected JTMS node or a rejection of the modification on the basis of contradicting beliefs. As a result, an association network is established and updated along with the product modeling processes.

Using the knowledge-feature associations, engineering intents can be represented explicitly as rules in the product model instead of implicit and tedious constraints or attributes. The associations between the knowledge-based model entities and the feature model entities are implemented as dependency type.

An embedded forward-chaining rule-based expert system is used to incorporate engineering rules into the feature-based modeling system. The major classes of the expert system are illustrated in Fig. 7. The *rule* class comprises antecedent and consequent patterns. The prototype system supports users in defining rules according to their specific requirements, regardless of the purpose of each rule, either a function decomposition rule, a function feasibility checking rule, or a function-to-feature mapping rule. The rule-based reasoning mechanisms, including pattern matching, forward chaining, etc., are implemented generically instead of being confined in a particular domain.

In each application, the inference engine matches the input facts with applicable rules and fires them. The input facts can be, for example, primitive design functions in the conceptual design stage or the input conceptual design features in the detail design stage. Feature creation and editing commands are specified as consequent actions of a rule. When a rule fires, features, feature properties or interfeature constraints are created or modified as the rule's consequents. Dependency associations between features and a knowledge base are automatically established by the system. It should be noted that such feature entities can also be interactively created. In that case, they are transformed into new facts and inserted into the knowledge base by the corresponding entity creation methods. The facts are linked to the corresponding entities through a unique entity identifier in the fact objects.

6.2 Implementing Sharing Associations. In the current implementation, each application feature class has its geometry creation and manipulation functions. Feature geometry is created by an invocation of such functions and inserted into the product's geometric model. The created topological entities are then associated with the feature through the features' geometric references.

During the prototype development, we encountered the problem that although features are initially directly linked to the boundary representation. Later, Boolean operations on the boundary representation may destroy these links due to feature interactions. Hence, the prototype system uses, instead of a boundary representation model, an ACIS cellular model. The cellular topology is used to preserve these links (see Fig. 8). Feature geometric references actually refer to cellular entities, i.e., cells, cell faces, or cell edges [19]. These cellular entities further correspond to topological entities in the boundary representation.

Each cell carries attributes that record its properties, such as its owning features and the material status, depending on whether the cell contributes to the part solid or not. Cells never volumetrically overlap. Whenever feature interaction occurs, the overlapping portions are converted into new cells. Since they belong to both interacting features, they record the both features into their respective owner lists. The sharing associations between features are therefore established automatically via the cellular model. The feature cells are not discarded from the part geometric model as long as the feature exists. In a case in which a feature's geometry does not belong to the final product geometry, the corresponding cells are marked as void, but are not removed. By using this approach, the canonical definitions of features and their cells are

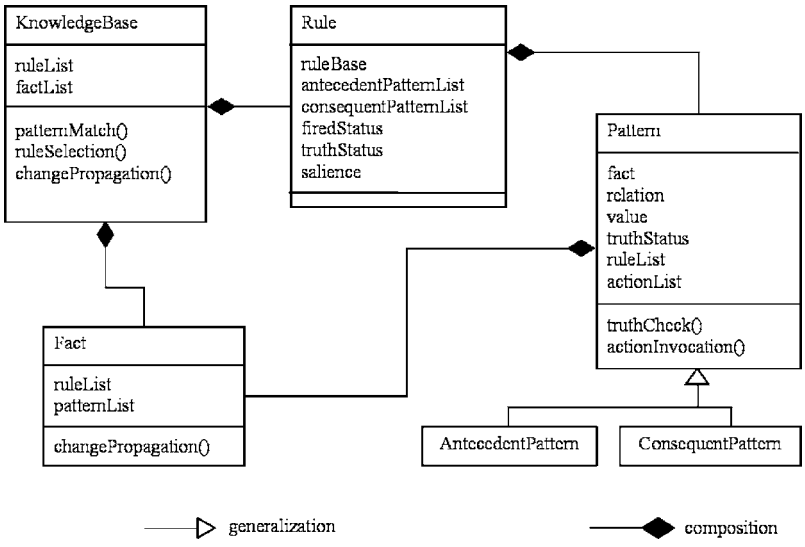

Fig. 7 Class diagram of the prototype rule-based expert system

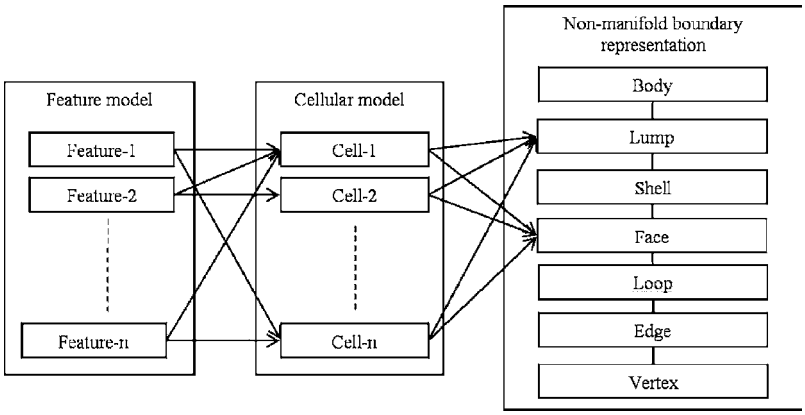

Fig. 8 Relations among feature model, cellular model, and the boundary representation

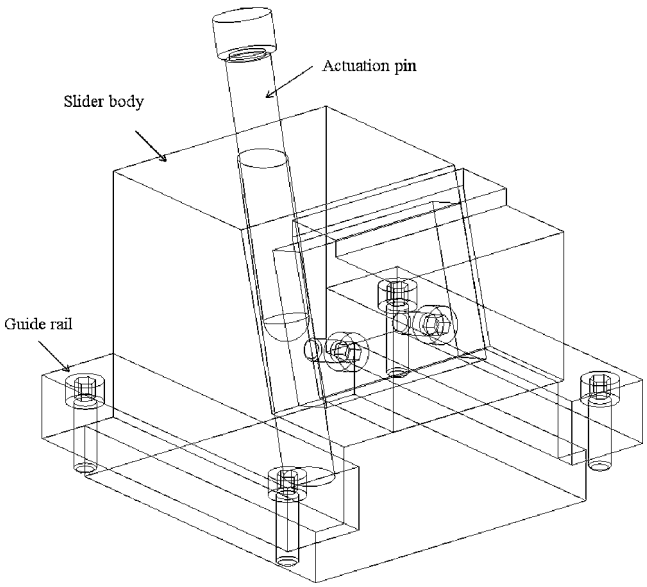

Fig. 9 Wire-frame picture of the slider mechanism

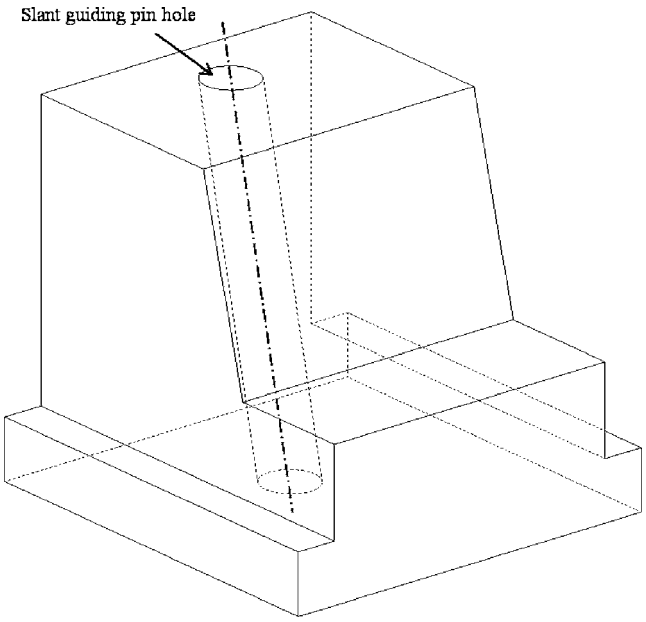

Fig. 10 Wire-frame picture of the slider body component

persistently linked to the boundary representation. Traditional feature interaction problems [17], especially the naming inconsistency, can hence be avoided.

If the geometry of a conceptual design feature has been explicitly specified, it is associated permanently with the corresponding detail design features by sharing cells, i.e., both features refer to the same cells. Such cells can even be nonsolid entities. When one of the owning features is modified, the corresponding cells' owner lists are searched to find other associated features for validation.

6.3 Interapplication Communication and Change Propagation. It is assumed in this work that each application has its own specific feature classes that other applications cannot access directly. On the other hand, these applications must communicate and exchange data with each other. The developed prototype system therefore uses a shared relational database manager, MYSQL [38], to address the communication problem. All applications publish their data to the shared database. Other applications can access and inquire the shared data through the database.

When an interapplication dependency association is established, it and the involved data are stored in this database. When an application modifies its private model, it must check the database for relevant interapplication associations. If such associations exist, a validity-checking process is triggered. The involved appli-

Conceptual Design Rule 1:

IF the function is "Create an external undercut region of a molding"

THEN decompose it into "Form the undercut region during the molding process"

and "Release the undercut prior to the ejection process"

Fig. 11 Conceptual design rule for function decomposition

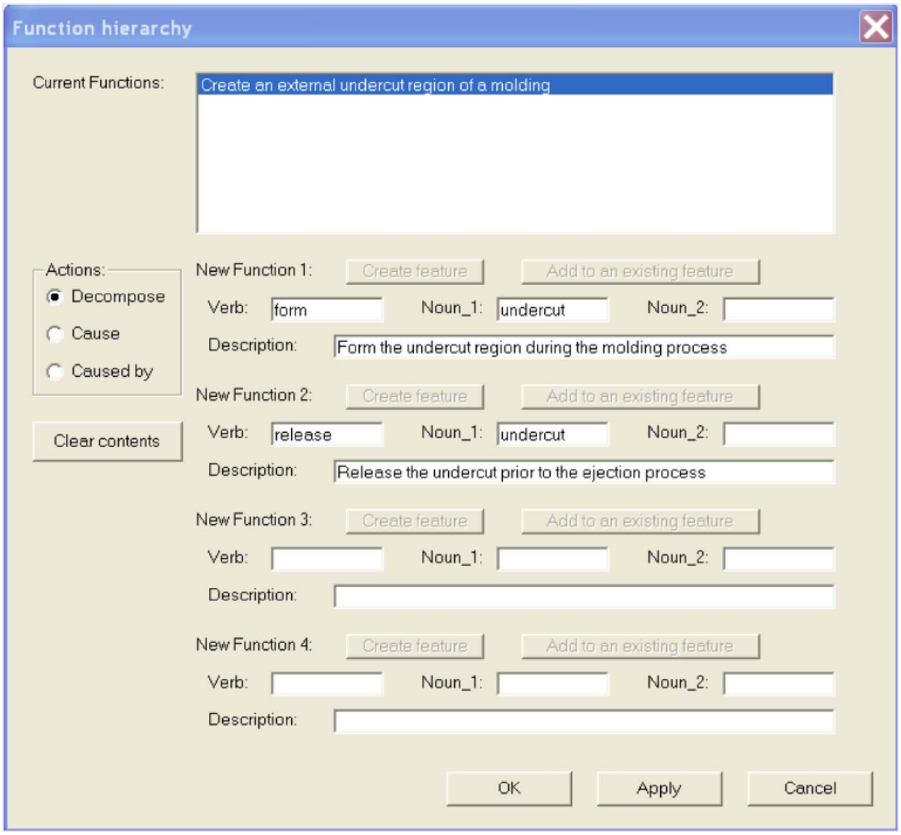

Fig. 12 Dialog box for establishing a function hierarchy, interactively

cations are responsible for maintaining the consistency, while the shared database is a medium for storing common data, interapplication associations, and change propagation.

For each modification made in an application, the application's JTMS dependency network is first searched for affected nodes. They are validated to ensure the modification is locally accepted within this application. In a second step, the association table in the shared database is searched for related features in other applications. The found features or feature properties are propagated to the corresponding applications for global validity check. These two processes are transparent to the designer, to whom only the invalid results are presented and efforts for maintaining the model consistency are reduced.

7 Criteria for Evaluating Information Validity

A general requirement for a valid product information model is its consistency within each partial model and among them. The detailed evaluation criteria are classified into three levels:

1. A feature is valid if
   - (a) the feature's geometric references point to valid topological entities
   - (b) the values of feature parameters are consistent with the product's geometric model
   - (c) all constraints on the feature are satisfied
   - (d) any feature property, if included in the JTMS dependency network, has a belief status, i.e., its supporting justification is labeled as believed
2. A model of an individual stage is valid if
   - (a) all features in the model are valid
   - (b) in its knowledge base, the antecedent conditions of all fired rules, which are used for generating attributes or parameters of, or constraints specified on, any existing feature, are satisfied
   - (c) dependency associations among the consequent facts and the respective feature attributes, parameters, or constraints, hold. For example, if a consequent fact is derived from, or used by some feature properties, the required feature properties must exist
   - (d) cellular entities referred to by the geometric references of any existing feature must exist and have a status (material or void, on the boundary or not on the boundary) that agrees with the feature sequence in its owning feature list
   - (e) all of the existing sharing and dependency associations hold
3. The conceptual and detail design are consistent if
   - (a) sharing associations between the conceptual design features and the corresponding detail design features hold
   - (b) each feature in the conceptual design is linked to features in the detail design via valid dependency associations
   - (c) each feature property or interfeature constraint in the conceptual design has its valid counterparts (might be one-to-many or many-to-one relations) in the detail design

8 Case Study

The conceptual and detail design models of a slider mechanism for a plastic injection mold are used as the example case for the unified feature modeling process. The interapplication associations in the mechanism are illustrated to show their implementa-

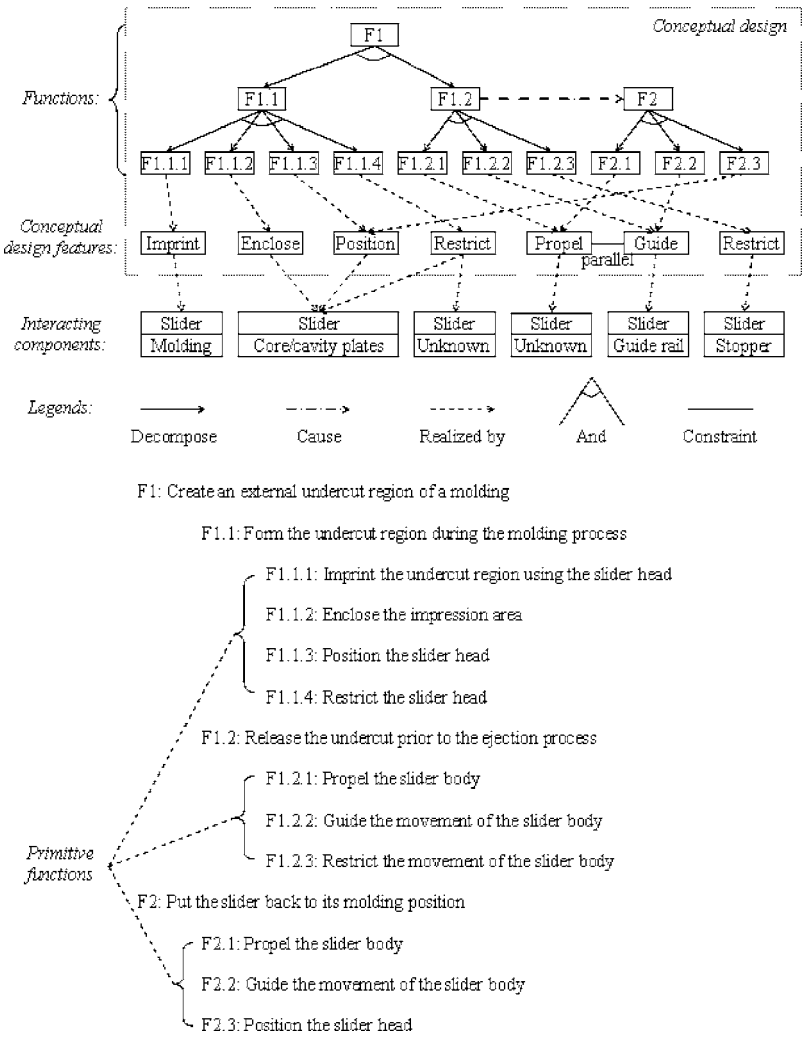

Fig. 13 Function hierarchy in the conceptual design

tion methods. Slider mechanisms are usually used to deal with the external undercuts of a plastic part in mold design [39].

The actuation device pushes the slider forward to its molding position before the melting plastic is injected into the closed mold cavity. The slider is kept in its molding position during the injection and cooling processes. The slider head forms the external undercut. Before the part is demoulded, the slider moves backward to release the undercut. Figure 9 shows a wire-frame picture of the slider mechanism assembly, whereas Fig. 10 illustrates the slider body, which is the object being designed.

8.1 Conceptual Design Stage. The conceptual design process begins with the function decomposition and ends with the mapping from the primitive design functions to the conceptual design features.

The main design function of a slider mechanism can be summerized as "*create an external undercut region of a molding*" [39]. Using the predefined (see Fig. 11) or interactively specified (see Fig. 12) function decomposition rules, this main design function is further decomposed until primitive design functions are obtained (Fig. 13).

A primitive design function is defined as a function that needs not further decomposition as it can be matched to a predefined conceptual design feature. In Fig. 13, ten primitive design functions are generated and the corresponding seven conceptual design features are created. The number of final conceptual design features is less than the number of primitive design functions due to function clustering. The users can also define new rules via the user interface (UI) as shown in Fig. 14. The user-specified function decomposition relations are also stored as rules for future use. Figure 15 shows one of the generated conceptual design features, PROPEL, in its symbolic abstract form. The behavior, represented by this feature, can be described as *a driving component pushes a driven component in a particular direction for a specific distance*. The conceptual design feature data are saved in the database. The system also permits the user to specify values or value ranges for feature properties (Fig. 16). Thus, a solution space, rather than a single solution point, is specified.

The design of a slider mechanism includes choosing the type of the actuation mechanism. According to the size of the undercut (especially its depth) as well as the required clamping and opening forces, the slider can be actuated by a heel cam, angular pins, or hydraulic units. The principle requirements for selecting a particular type of actuation are to facilitate mold opening and to provide sufficient clamping force.

In this case study, the depth of undercut is associated with one of the properties of a conceptual design feature (the PROPEL feature's face orientation property). This association is represented as an algebraic constraint

$$D = L \sin \phi$$

where D is the depth of the undercut; L is the working length of actuation device; and ϕ is the face orientation property of the

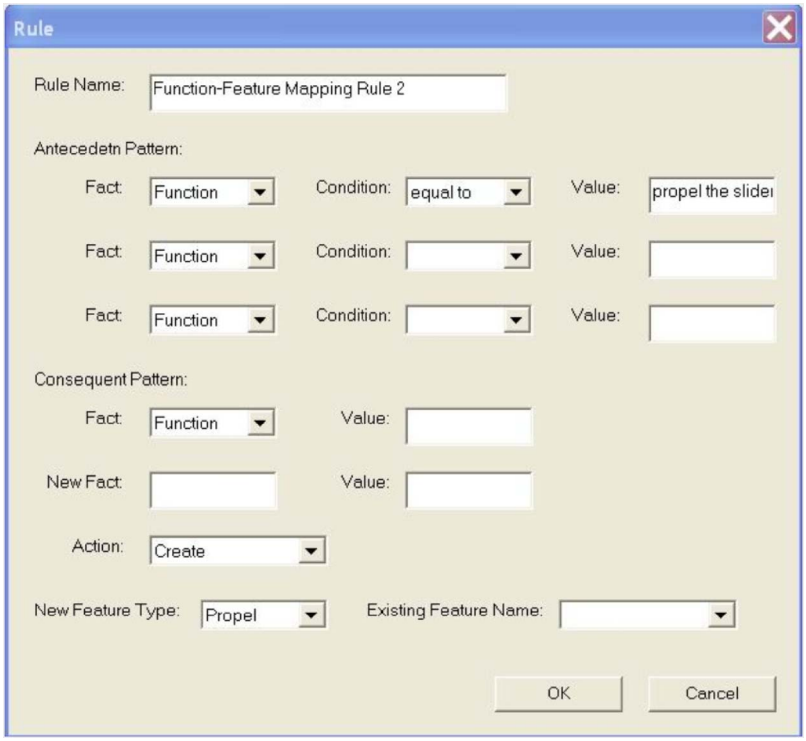

Fig. 14 User interface for defining a new rule

PROPEL feature.

In the conceptual design stage, the above design principle is roughly represented by the following rules:

1. If ϕ is <15 deg, then use a heel cam (to simplify the assembly).
2. If ϕ is between 15 deg and 25 deg, then use angular pins.
3. If ϕ is >25 deg, then use hydraulic units.

During a conceptual design session, the appropriate rule will fire and generate constraints specified on the PROPEL feature's face orientation property.

8.2 Detail Design Stage. In the detail design stage, the user loads and extracts information from the conceptual design. The user may choose (i) to develop the detail design on the basis of the existing conceptual design and thus create an integrated model with the built-in associations, or (ii) to create a separate detail design first and add the required associations afterward.

In both situations, the system supports the association of detail design features with conceptual design features. Figure 17 shows the dialog box used to establish an association between the face orientation property of the PROPEL feature of the conceptual design, and the orientation property of the *hole* feature of the detail

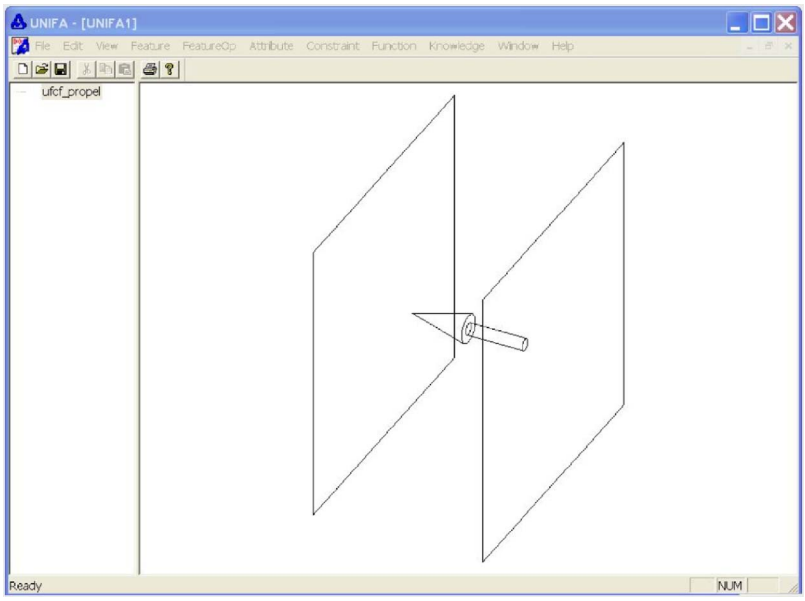

Fig. 15 PROPEL conceptual design feature

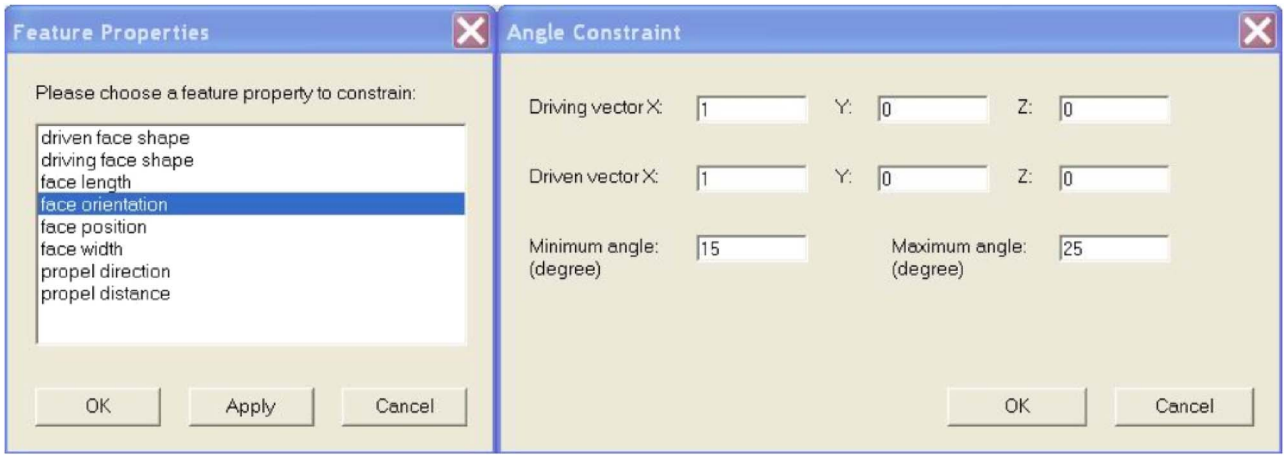

Fig. 16 Specifying the value ranges of the properties of the PROPEL feature

design. The hole feature is used to accommodate the actuation pin. These interapplication associations are stored in the database for later inquiry and change propagation. The prototype system can create the detail design of the slider mechanism as shown in Fig. 10.

8.3 Design Modification and Change Propagation. During the detail design stage, suppose the user modifies the hole feature on the "slider body" part by changing its orientation (Fig. 10). Then the feature-editing method searches the JTMS dependency network and the shared database automatically. Because the hole feature is generated (together with other detail design features) to realize the conceptual design feature PROPEL, it is identified as the associated feature. Furthermore, one of the properties of the PROPEL feature—face orientation—is found to be an associated entity of the property (hole orientation angle) of the modified hole feature. Then the values of the cylindrical face orientation are temporarily changed in the shared database. A message about the modification, which includes feature name, feature identifier, property name, and new value of the property, is sent to the conceptual design application. After receiving the modification, the conceptual design application checks the validity of the modification using the new values of the face orientation.

If no related constraints are violated or if a constraint is violated but can be resatisfied, a "valid" message is sent to the detail design application. Otherwise, a message about the modification being rejected as well as the violated rules or constraints is sent. Similarly, changes to the conceptual design may require further changes in the detail design. For example, if the face orientation of the PROPEL feature becomes <15 deg or >25 deg, major changes in the detail design is needed, i.e., the type of the actuation device has to be changed. This is not purely a geometric modification since clamping forces and the smoothness of the movement of the actuation device need to be considered together by using rule-based reasoning when choosing a suitable type of the actuation. Only after all intra- and interapplication constraints are checked and satisfied, the initial feature modification is globally accepted.

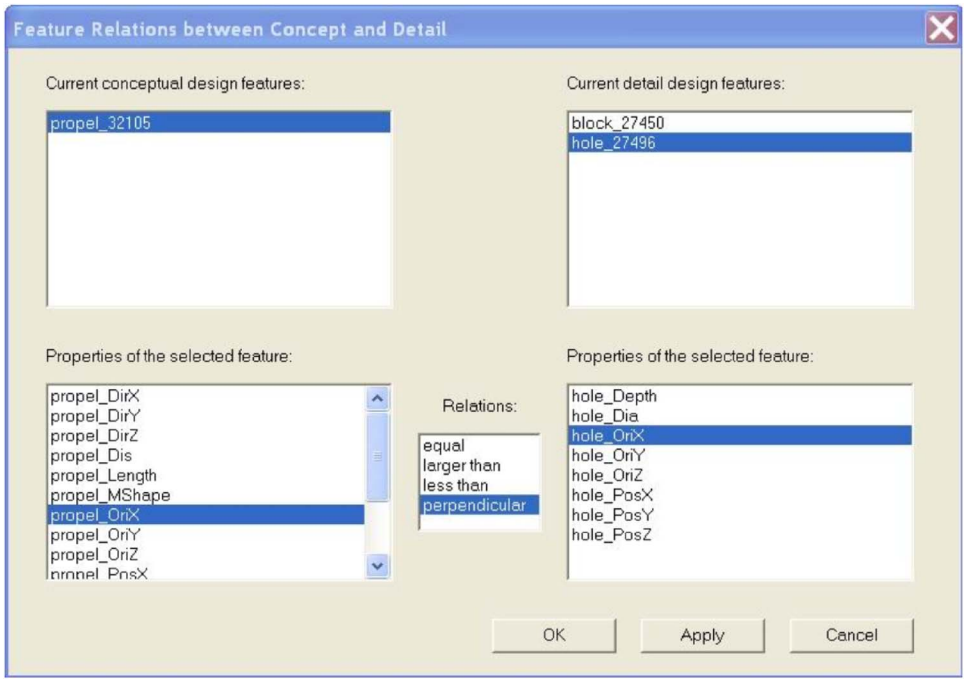

Fig. 17 Dialog box for feature association

9 Discussions and Conclusions

First, product development processes, such as, for example, product conceptualization, detailed design, and process planning, can be regarded as a set of decision-making processes, for instance, choosing a specific structure (shape, dimension, etc.), specifying material properties, or determining machining sequence. These decisions are all results of reasoning processes based on built-in system knowledge or human experience. Traditional computer-aided systems are either data oriented or knowledge oriented. Without coherently integrating the data- and knowledge-oriented approaches, engineering intents must be embeded as rigid data-oriented entities as well as geometric or algebraic constraints. They are implicit and inflexible. Explicitly incorporating knowledge rules into a feature modeling system enables designers to specify their intents more effectively.

Second, well-defined features, which include geometric references, attributes, parameters, and constraints, represent application semantics, collectively, and provide an interface by which a feature object can communicate with both knowledge-based and geometric entities. On the other hand, feature properties, whether geometric or nongeometric, can be associated. This allows for a better information consistency control than with geometric relations alone.

Third, including the conceptual design stage into a multiple-view feature modeling scheme provides a solution space for downstream application optimizations; they are hence not rigidly confined to the existing design. Alternative solutions can be more conveniently explored to improve the overall product in quality, performance, and cost.

In conclusion, this paper introduces two generic types of associations used in the unified feature modeling scheme, sharing and dependency. These generic association types provide a basis for

1. embedding knowledge-based systems into CAx systems
2. integrating different CAx systems

The implementation of these association types is also discussed. Maintaining and using these associations enable feature-based collaborative and concurrent engineering. The unified feature definition and the two proposed association types establish an efficient intermediate information association layer, which automizes the maintainence of product models' validity, integrity, and consistency.

In the future, the prototype system will be further improved to overcome the following two limitations:

1. direct communication between applications. A distributed system for collaborative modification propagation among applications with more effective communication method (such as an agent-based mechanism) will be explored.
2. potential interapplication dependency loops. A more generic and robust across-domain change evaluation algorithms need to be developed.

References

[1] Shah, J. J., and Mantyla, M., 1995, *Parametric and Feature-Based CAD/CAM: Concepts, Techniques, and Applications*, Wiley, New York.

[2] Laakko, T., and Mantyla, M., 1993, "Feature Modeling by Incremental Feature Recognition," Comput.-Aided Des., **25**(8), pp. 479–492.

[3] Vieira, A. S., 1995, "Consistency Management in Feature-Based Parametric Design," *Proc. of ASME 1995 Design Engineering Technical Conference, Boston*, ASME, New York, Vol. 2, pp. 977–987.

[4] Ma, Y.-S., and Tong, T., 2003, "Associative Feature Modeling for Concurrent Engineering Integration," Comput Ind., **51**(1), pp. 51–71.

[5] Ma, Y.-S., Tor, S. B., and Britton, G. A., 2003, "The Development of a Standard Component Library for Plastic Injection Mould Design Using an Object Oriented Approach," Int. J. Adv. Manuf. Technol., **22**(9–10), pp. 611–618.

[6] Ma, Y.-S., Britton, G. A., Tor, S. B., Jin, L.-Y., Chen, G., and Tang, S.-H., 2004, "Design of a Feature-Object-Based Mechanical Assembly Library," Comput. Aided Des. App., **1**(1–4), pp. 379–403.

[7] de Kraker, K. J., Dohmen, M., and Bronsvoort, W. F., 1997, "Maintaining Multiple Views in Feature Modeling," *Proc. of 4th ACM Symposium on Solid Modeling and Applications, Atlanta*, ACM, New York, pp. 123–130.

[8] Bronsvoort, W. F., and Noort, A., 2004, "Multiple-View Feature Modeling for Integral Product Development," Comput.-Aided Des., **36**(10), pp. 929–946.

[9] Han, J. H., 1996, "3D Geometric Reasoning Algorithms for Feature Recognition," Ph.D. dissertation, University of Southern California, Los Angeles.

[10] de Martino, T., Falcidieno, B., and Habinger, S., 1998, "Design and Engineering Process Integration Through a Multiple View Intermediate Modeler in a Distributed Object-Oriented System Environment," Comput.-Aided Des., **30**(6), pp. 437–452.

[11] Subramani, S., and Gurumoorthy, B., 2003, "Associativity Between Feature Models Across Domains," *Proc. of 8th ACM Symposium on Solid Modeling and Applications, Seattle*, ACM, New York, pp. 316–321.

[12] Hoffman, C. M., and Joan-Arinyo, R., 1998, "CAD and the Product Master Model," Comput.-Aided Des., **30**(11), pp. 905–918.

[13] Chen, G., Ma, Y.-S., Thimm, G., and Tang, S.-H., 2004, "Unified Feature Modeling Scheme for the Integration of CAD and CAx," Comput. Aided Des. Appl., **1**(1–4), pp. 595–602.

[14] Karinthi, R. R., and Nau, D., 1992, "An Algebraic Approach to Feature Interactions," IEEE Trans. Pattern Anal. Mach. Intell., **14**(4), pp. 469–484.

[15] Brunetti, G., de Martino, T., Falcidieno, B., and Habinger, S., 1995, "A Relational Model for Interactive Manipulation of Form Features Based on Algebraic Geometry," *Proc. of 3rd ACM Symposium on Solid Modeling and Applications*, Salt Lake City, ACM, New York, pp. 95–103.

[16] Mandorli, F., Cugini, U., Otto, H. E., and Kimura, F., 1997, "Modeling With Self Validation Features," *Proc. of 4th ACM Symposium on Solid Modeling and Applications*, Atlanta, ACM, New York, pp. 88–96.

[17] Bidarra, R., and Bronsvoort, W. F., 2000, "Semantic Feature Modeling," Comput.-Aided Des., **32**(3), pp. 201–225.

[18] Regli, W. C., and Pratt, M. J., 1996, "What are Feature Interactions?" *Proc. of ASME Design Engineering Technical Conference and International Computers in Engineering Conference*, Irvine, CA, ASME, New York.

[19] Bidarra, R., de Kraker, K. J., and Bronsvoort, W. F., 1998, "Representation and Management of Feature Information in a Cellular Model," Comput.-Aided Des., **30**(4), pp. 301–313.

[20] Roy, U., and Bharadwaj, B., 2002, "Design With Part Behaviors: Behavior Model, Representation and Applications," Comput.-Aided Des., **34**(9), pp. 613–636.

[21] Xue, D., Yadav, S., and Norrie, D. H., 1999, "Knowledge Base and Database Representation for Intelligent Concurrent Design," Comput.-Aided Des., **31**(2), pp. 131–145.

[22] Xue, D., and Yang, H., 2004, "A Concurrent Engineering-Oriented Design database Representation Model," Comput.-Aided Des., **36**(10), pp. 947–965.

[23] Welch, R. V., and Dixon, J. R., 1992, "Representing Function, Behavior and Structure During Conceptual Design," *Design Theory and Methodology - DTM'92*, ASME, Scottsdale, AZ, pp. 11–18.

[24] Umeda, Y., Ishii, M., Yoshioka, M., Shimomura, Y., and Tomiyama, T., 1996, "Supporting Conceptual Design Based on the Function-Behavior-State Modeler," Artif. Intell. Eng. Des. Anal. Manuf., **10**(4), pp. 275–288.

[25] Qian, L., and Gero, J. S., 1996, "Function-Behavior-Structure Paths and Their Role in Analogy-Based Design," Artif. Intell. Eng. Des. Anal. Manuf., **10**(4), pp. 289–312.

[26] Ranta, M., Mantyla, M., Umeda, Y., and Tomiyama, T., 1996, "Integration of Functional and Feature-Based Product Modeling—The IMS/GNOSIS Experience," Comput.-Aided Des., **28**(5), pp. 371–381.

[27] Brunetti, G., and Golob, B., 2000, "A Feature-Based Approach Towards an Integrated Product Model Including Conceptual Design information," Comput.-Aided Des., **32**(14), pp. 877–887.

[28] Brunetti, G., and Grimm, S., 2005, "Feature Ontologies for the Explicit Representation of Shape Semantics," Int. J. Comput. Appl. Technol., **23**(2/3/4), pp. 192–202.

[29] Pratt, M. J., and Srinivasan, V., 2005, "Towards a Neutral Specification of Geometric Features," Int. J. Comput. Appl. Technol., **23**(2/3/4), pp. 203–218.

[30] Booch, G., Rumbaugh, J., and Jacobson, I., 1999, *The Unified Modeling Language User Guide*, Addison-Wesley, Reading, MA.

[31] Rumbaugh, J., Jacobson, I., and Booch, G., 2005, *The Unified Modeling Language Reference Manual*, 2nd ed., Addison-Wesley, Reading, MA.

[32] Thimm, G., Lee, S. G., and Ma, Y.-S., 2006, "Towards Unified Modeling of Product Life-Cycles," Comput Ind. **57**, pp. 331–341.

[33] Riley, G., and Giarratano, J., 2004, *Expert Systems: Principles and Programming*, Cambridge, Mass, Thomson Course Technology.

[34] Pahl, G., and Beitz, W., 1996, *Engineering Design: A Systematic Approach*, 2nd ed., Springer, New York.

[35] Guan, X., Duffy, A. H. B., and MacCallum, K. J., 1997, "Prototype System for Supporting the Incremental Modeling of Vague Geometric Configurations," Int. J. Comput. Appl. Technol., **11**(4), pp. 287–310.

[36] Chen, G., Ma, Y.-S., Thimm, G., and Tang, S.-H., 2005, "Knowledge-Based Reasoning in a Unified Feature Modeling Scheme," Comput. Aided Des. Appl., **2**(1–4), pp. 173–182.

[37] Forbus, K. D., and de Kleer, J., 1993, *Building Problem Solvers*, MIT Press, Cambridge, MA.

[38] MySQL, 2006, *MySQL 5.1 Reference Manual*, http://dev.mysql.com/doc/refman/5.1/en/, MySQL Inc., Cupertino, CA 95014.

[39] Pye, R. G. W., 1989, *Injection Mould Design: A Textbook for the Novice and a Design Manual for the Thermoplastics Industry*, 4th ed., Longman Scientific & Technical, New York.